INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Diplomarbeit

Konzeption und prototypische Implementierung eines Werkzeugs für die dienstorientierte Tariferstellung

Erika Kalix

Aufgabensteller: Prof. Dr. Heinz-Gerd Hegering

Betreuer: Igor Radisic
Abgabetermin: 14. September 2002

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Diplomarbeit

Konzeption und prototypische Implementierung eines Werkzeugs für die dienstorientierte Tariferstellung

Erika Kalix

Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Igor Radisic
Abgabetermin:	14. September 2002

Hiermit versichere ich, daß ich die vorliegende Diplomarbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 14. September 2002

. . . .
(Unterschrift des Kandidaten)

Zusammenfassung

Viele Unternehmen verlagern heutzutage die Verwaltung ihrer Kommunikationsdienste auf externe Provider. Da solche Großkunden über viele Nutzer verfügen, die vielseitige Aufgaben mit Hilfe von Individualdiensten im Intranet und in externen Netzen, z.B. dem Internet, erfüllen sollen, ist dies eine sehr komplexe Aufgabe. Provider von Kommunikationsdiensten brauchen daher Unterstützung bei der Aufgabe, geeignete Tarife für Verträge mit Kunden zu finden, die verschiedene Unternehmensstrukturen haben und individuelle Dienste mit verbrauchsorientierter Abrechnung wünschen. Daher soll ein Tariftool erstellt werden, das aus bewerteten Altverträgen geeignete Tarifparameter für bestimmte Anforderungen des Kunden ermittelt. Zur persistenten Speicherung der Altverträge wird ein Wissenbasiertes System gewählt, dessen Wissensbasis in einer Datenbank abgelegt ist. Das Tariftool soll auf einem Tarifmodell beruhen, das einen Vergleich der Verträge ermöglicht.

Zur Ermittlung eines geeigneten Tarifmodells wurde eine Literaturrecherche durchgeführt und ein allgemeines Tarifmodell abgeleitet. Ferner wurden Wissensbasierte Systeme analysiert. Als zweckmäßige Lösung erwies sich ein Wissensbasiertes System mit Objektorientierung und Regeln zur Modifikation des Programmablaufs, bei dem die Basisdaten in einer relationalen Datenbank gehalten werden.

Das Tariftool wurde als Prototyp objektorientiert entwickelt und implementiert. Im Tarifmodell, das dem Tariftool zugrunde liegt, werden die wichtigsten Faktoren, wie z.B. Bereitstellung von Ressourcen, Dauer, Dienstgüte und Volumen der Datenübertragung und gegebenfalls die Tageszeit berücksichtigt. Die Datenbank ist als relationale PostgreSQL-Datenbank vorgegeben, das Anwendungsprogramm wurde in der Programmiersprache Java implementiert. Die Basisdaten (Vertrag, Dienst, Kunde, Tarifparameter usw.) sind als objektorientierte Daten dargestellt und als Tabellen in der Datenbank gespeichert. Regeln können dazu beitragen, die Suche nach einem besten Vertrag zu optimieren. Für die Tariftoolfunktion *Finden des besten Vertrags* wurden geeignete Suchkriterien entwickelt. Weitere Funktionen des Anwendungsprogramms sind *Aktualisieren* der Wissensbasis, sowie *Schreiben und Lesen von Datensätzen.* Da der Prototyp *Tariftool* nicht den gesamten Umfang an Tarifdaten bewältigen kann, wurden repräsentative Daten für Dienste und Tarifparameter ausgewählt.

Das Ergebnis der Implementierung ist ein erweiterbarer Prototyp *Tariftool*, der auf andere Datenbanken portierbar ist, zu denen Javaprogramme über JDBC eine Verbindung herstellen können. Tests ergaben, dass das Tariftool im Rahmen seiner Möglichkeiten ein brauchbares Werkzeug zum Auffinden geeigneter Tarifparameter bei bestimmten Vorgaben über gewünschte Dienste und Kundenprofile ist.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einführung

1.1 Motivation

Großkunden neigen heute aus finanziellen Gründen und/oder wegen Mangel an geeigneten Personal dazu, den Betrieb und die Verwaltung von Kommunikationsnetzen und diversen Diensten - wie z.B. Multimedia-Telekonferenz oder Videoübertragung - im Rahmen von Outsourcing an externe Provider zu übertragen ([HAN 99]).

Für Provider ist eine solche Aufgabe sehr komplex, da Großkunden über viele Nutzer verfügen, die vielseitige Aufgaben mit Hilfe von Individualdiensten im Intranet und in externen Netzen, z.B. dem Internet, erfüllen sollen. Dazu müssen sie auch eine Vielzahl von Diensten in Anspruch nehmen, die größere Anforderungen an die Qualität der Datenübertragung stellen, als das Internet bisher üblicherweise bietet. Für diese wünschen sie eine nutzungsorientierte Abrechnung.

Mit zunehmender kommerzieller Nutzung von aufwendigen Diensten, die bestimmte bessere Dienstgüten benötigen, wird es ferner für die Provider interessant, qualifiziertere Dienste verbrauchsorientiert abzurechnen. Sie hoffen auch, mit qualifizierten Diensten Gewinne zu machen ([ckst99]).

Bei seiner Preisgestaltung muss der Provider einerseits zumindest so viel verlangen, dass seine Kosten gedeckt sind, andererseits muss er mit anderen Providern auf dem Markt um Kunden konkurrieren. Dazu wird ein geeignetes Tarifmodell benötigt.

Für eine derartig komplexere Tarifierung gibt es bisher noch wenig Erfahrungswerte, da Flatrate bisher meist der Standard bei der Abrechnung von Internetnutzung war. Dadurch wird die Tariferstellung schwierig ([Radi 02]). Bei der Auswahl von verbrauchsorientierten Tarifen ist zu bedenken, dass Messeinrichtungen installiert und gewartet werden müssen. Auch die kompliziertere Abrechnung verursacht zusätzliche Kosten.

1.2 Aufgabenstellung

Es soll ein Konzept für ein Tool entwickelt werden, das den Provider bei der Tarifgestaltung unterstützt. Der Provider soll mit Hilfe des Tools aus früheren Erfahrungen mit gewinnbringenden bzw. nicht kostendeckenden Abschlüssen lernen.

Um geeignete Tarife für die Vielfalt von Diensten mit verschiedenen Dienstgüten zu finden, sollen die Daten von Tarifabschlüssen gespeichert und bewertet werden, so dass bei Abschluss eines neuen Vertrags auf Erfahrungswerte zurückgegriffen werden kann. Dazu soll ein Wisssensbasiertes System als Basis dienen.

1.3 Vorgehensmodell

Aus einem typischen Szenario sollen Rollen und Interaktionen identifiziert werden, welche die Grundlage eines Modells für das Tool bilden. Dazu wird ein Dienstmodell aufgestellt. Ferner sollen mittels Literaturrecherche bisher untersuchte Tarifmodelle gefunden und ausgewertet werden. Aufgrund dieser Daten wird ein Tarifmodell vorgeschlagen, das die relevanten Faktoren, wie z.B. Bereitstellung von Ressourcen, übertragene Datenmengen und QoS berücksichtigt.

Tarifverträge, die nach diesem Modell abgeschlossen wurden, werden nach Ertrag/Kosten beurteilt und durch einen entsprechenden Providernutzen-/Providerkosten-Faktor ergänzt in einer Wissensbasis gespeichert. Aus diesen Daten soll dem Provider als Hilfe bei Vertragsverhandlungen ein geeigneter Tarif angeboten werden, der dem Nutzerprofil (gewünschte Dienste) entspricht.

Dazu soll ein wissensbasiertes System erstellt werden, das dem Provider nach Eingabe des Nutzerprofils die geeignetsten Parameter (bester Nutzen-/Kostenfaktor) für die Tarifierung aus den vorhandenen Verträgen und ihren Nutzwerten ermittelt und ausgibt. Ein Prototyp, der auf einem Java-Anwendungsprogramm und Tabellen mit Vertragsdaten in einer relationalen Datenbank beruht, wird implementiert und getestet.

1.4 Gliederung der Arbeit

In Kapitel 2 wird das Szenario der Situation vorgestellt und erläutert. Ein Anforderungskatalog wird daraus abgeleitet.

In Kapitel 3 werden mögliche Modelle für Wissensbasierte Systeme in einer Literaturrecherche ermittelt und beschrieben. Die Modellansätze werden klassifiziert, um Vergleiche zu ermöglichen.

In Kapitel 4 werden Modelle für die Tarifierung als State of the Art mittels einer Literaturrecherche ermittelt und beschrieben. Die wichtigsten Tarifmodelle werden im Hinblick auf die vorgesehene Anwendung beurteilt.

In Kapitel 5 wird aufgrund der Analyseergebnisse in Kapitel 3 und 4 eine Entscheidung getroffen, welches wissenbasierte System dem Tariftool zugrunde gelegt wird. Aus dem Szenario wird ein Aktivitätsdiagramm für das Tool abgeleitet. Ein Dienstmodell und eine allgemeine Tarifformel werden vorgestellt. Ein statisches und ein dynamisches Modell des Tariftools werden entworfen.

Auf Grund dieser Entscheidung wird in Kapitel 6 der Entwurf des Prototyps *Tariftool* beschrieben. Für die Wissensbasis wird ein Entity-Relationship-Diagramm und ein Datenbankschema angegeben. Das Wissensnutzungsprogramm wird mit Hilfe von Objektdiagrammen und weiteren Diagrammen nach UML erläutert. Die Suchkriterien werden spezifiziert. Zu implementierende Dienste und Tarifparameter werden ausgewählt.

In Kapitel 7 wird die prototypische Implementierung des Tools beschrieben. Die Architektur wird anhand eines Diagramms vorgestellt. Die Wissensbasis wird in einer relationalen Datenbank vom Typ PostgreSQL implementiert, das Programm zur Wissensverarbeitung wird in Java codiert. Die Benutzerschnittstelle wird beschrieben.

Kapitel 8 enthält eine Zusammenfassung mit einem Ausblick auf mögliche Erweiterungen des Tariftools.

Kapitel 2

Problemanalyse

In diesem Kapitel wird zunächst ein Szenario des Problems und die Anwendung eines Tariftools erklärt. Ein Anforderungskatalog wird aus dem Szenario abgeleitet.

2.1 Szenario

Allgemein liegt hier die Situation vor, dass ein Provider (Dienstleister) einem Großkunden (mit vielen Nutzern) einen Dienst liefern will, den der Kunde nutzen kann. In diesem speziellen Fall geht es um individuelle Kommunikationsdienste (s. Abbildung 2.1).

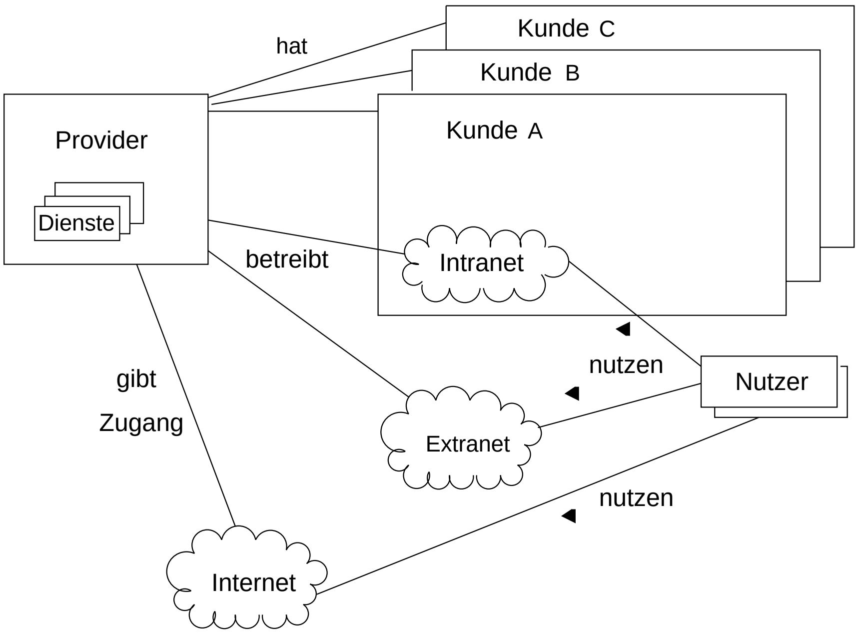

Abbildung 2.1: Modell der Situation

Die Preise für solche Dienste sind noch nicht standardisiert, so dass ein Verhandlungsspielraum besteht. Dieser wird eingeschränkt durch die Konkurrenz auf dem Markt.

Der Großkunde will verschiedene qualifizierte Netzdienste zu angemessenen möglichst niedrigen, individuellen Preisen durch seine User nutzen. Dies sind z.B.:

- Multimedia-Telekonferenz
- Video- oder Audioübertragung
- Nutzung von Anwendungen im Intranet
- Filetransfer
- Recherchieren im Internet
- Bestellen im Internet (e-commerce)
- Fax
- Email

Die Unternehmensstruktur des Kunden bedingt, dass jeder User entsprechend seiner Funktion beim Kunden spezielle Anforderungen an bestimmte Dienste hat, die er nutzen will. Die Gesamtheit dieser Anforderungen wird im Folgenden Nutzer-Dienstcharakteristik genannt. Da der Kunde seine Anforderungen genau kennt, wird er sie in einer Ausschreibung spezifizieren und Provider auffordern, Angebote abzugeben. Vor der Nutzung der Dienste soll ein entsprechender Vertrag geschlossen werden.

Der Provider weiß, dass der Großkunde auch von anderen Providern Angebote erhalten wird. Entsprechend muss er seine Preise kalkulieren: es soll kein Verlust herauskommen, aber die anderen Provider sollen möglichst unterboten werden. Er muss also einschätzen können, welche Preise marktgerecht sind. Bisher hat er in dieser Hinsicht wenig Erfahrungen. Außerdem ist es günstig, wenn er sein Angebot schnell einreichen kann; zu spät eingehende Angebote werden vielleicht vom Kunden gar nicht mehr berücksichtigt.

Bei der Vielzahl von individuellen Diensten kann der Provider nicht alle Preise und deren Angemessenheit im Kopf haben. Bei der Festlegung der Preise für einen Großkunden soll ihn das Tariftool performant, einfach und benutzerfreundlich unterstützen.

Das Tariftool soll

- Tarifvorschläge aufgrund bewerteter Altverträge und der Nutzer-Dienstcharakteristiken liefern und
- als Wissensbasiertes System realisiert werden.

Außerdem soll das Tariftool erweiterbar sein, sodass der Provider z.B. neue Dienste einbringen kann. Die globale Struktur des Tariftools als wissensbasiertes System mit Tarifierungsdaten und Schnittstellen zur Außenwelt ist in Abbildung 2.2 dargestellt.

Damit Verträge verglichen werden können, müssen sie gleiche Struktur und gleiche Nutzer-Dienstcharakteristiken sowie analoge Tarife haben. Das erfordert ein gemeinsames Tarifierungsmodell. Ferner soll das Tariftool auf verschiedene Systeme und Datenbanken portierbar sein.

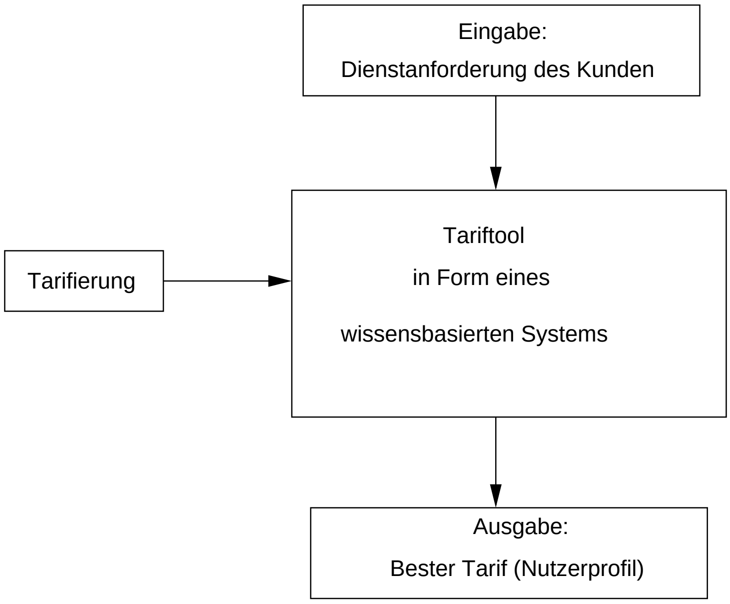

Abbildung 2.2: Globale Struktur des Tariftools

2.2 Anforderungskatalog

Aus dem zuvor beschriebenen Szenario wird der folgende Anforderungskatalog abgeleitet:

1. Verträge oder Teile von Verträgen können so strukturiert abgespeichert werden, dass sie vergleichbar sind.

2. Der Abschluss von Verträgen, die über einen bestimmten Zeitraum gelten, soll unterstützt werden.

3. Verträge können nach einer gewissen Laufzeit - z.B einem Monat - bezüglich des Gewinns bzw. Verlusts, den sie bringen, bewertet werden.

4. Eine allgemeine Tarifformel muss festzulegen sein, sodass Tarife und Verträge leicht verglichen werden können und die Abrechnung einfach wird.

5. Unterschiedliche Dienste sollen unterstützt werden.

6. Kunden mit unterschiedlichen Unternehmenstrukturen sollen unterstützt werden.

7. Kunden mit unterschiedlichen Dienstanforderungen sollen unterstützt werden.

8. Das Tariftool soll skalierbar (erweiterbar) sein, z.B. bezüglich der Aufnahme neuer Dienste und Tarifparameter.

9. Die für die Verträge mit den Kunden relevanten Daten sollen persistent gespeichert werden.

10. Bezüglich der gespeicherten Datenmenge soll es keine Begrenzung geben.

11. Die Suche nach Vergleichsverträgen soll performant sein.
12. Das Tariftool soll portierbar sein.
13. Das Tariftool soll benutzerfreundlich sein.
14. Zur Bedienung des Tariftools sollen keine besonderen Vorkenntnisse nötig sein.

Die Erfüllbarkeit dieser Anforderungen ist Voraussetzung dafür, dass ein Tariftool realisiert werden kann, welches den Provider in der vorgesehenen Weise unterstützt. Dies wird in den folgenden Kapiteln 3, 4 und 5 untersucht.

Kapitel 3

Wissensbasierte Systeme

Da das Tariftool als Wissensbasiertes System (WBS) realisiert werden soll, werden in diesem Kapitel mittels einer Recherche WBS-Modelle analysiert und bezüglich ihrer Eignung für dieses Projekt beurteilt. Dazu wird die Entwicklung eines Wissensbasierten Systems untersucht, bei der zunächst die der Anwendbarkeit eines WBS für das zu lösende Problem geprüft wird. Danach wird eine geignete Darstellung der Daten und ein passendes Verfahren für die Problemlösung gesucht. Zuletzt werden einige Anwendungen von WBS vorgestellt.

3.1 Einführung

Wissensbasierte Systeme (WBS) sind Anwendungen von Ergebnissen der KI-Forschung (KI: künstliche Intelligenz, engl. AI von Artificial Intelligence). KI ist die Wissenschaft davon, Maschinen Tätigkeiten ausüben zu lassen, die beim Menschen Intelligenz erfordern. In der Theorie wird versucht, Prinzipien zu verstehen, die Intelligenz möglich machen ([schn86]), und diese Prinzipien einzusetzen, um Computer nützlicher zu machen ([schn86], [joha90]).
Grundlage eines wissensbasierten Systems ist eine Wissensbasis; daneben enthält es als gleichwertigen zweiten Teil die Wissensverarbeitung (Problemlösungskomponente) (s. Abbildung 3.1 nach [Pupp 91]). Letztere wird auch als Inferenzmaschine bezeichnet.
Nach Kurbel ([kur92]) wird ein wissensbasiertes System wie folgt definiert:

`Ein wissensbasiertes System ist ein Softwaresystem, bei dem Fachwissen über ein Anwendungsgebiet (Domain Knowledge) explizit und unabhängig vom allgemeinen Problemwissen dargestellt wird.`

Ein wissensbasiertes System dient zur Bewältigung der Komplexität von großen Datenmengen und komplizierten Beziehungen zwischen den Daten in akzeptabler Zeit. Bei zu großen Datenmengen können sinnvolle Lösungen nicht in akzeptabler Zeit erbracht werden ([Reim 91]).
Das Ziel der Entwicklung von Wissensbasierten Systemen ("Knowledge Engineering") ist die Entscheidungsunterstützung von Nicht-Experten und Entlastung von Experten ([schn86]).
Expertensysteme sind spezielle wissensbasierte Systeme, bei denen Spezialwissen und Schlussfolgerungsfähigkeit qualifizierter Fachleute aus einem eng begrenzten Aufgaben- oder Sachgebiet nachgebildet wird. Die spezifische Problemlösungsfähigkeit eines menschlichen Experten soll zumindest annähernd erreicht oder übertroffen werden.

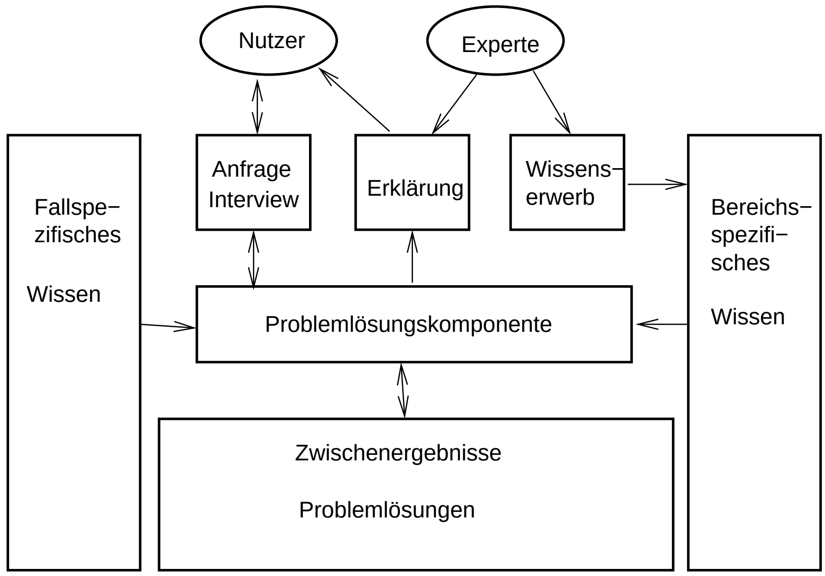

Abbildung 3.1: Modell eines Wissensbasierten Systems

Unter Wissen versteht man dabei Aussagen, die Experten für wahr halten. Unter Wissensrepräsentation versteht man die Struktur, in der die Daten und Interpretationsvorschriften gespeichert sind. Implizite Sachverhalte ergeben sich aus den expliziten durch Schlussfolgerungen (Inferenzen). Basisdaten und Auswertungsprogramm sind getrennte Teile, aber eng aufeinander bezogen.

Das Datenkonzept kann als Netz dargestellt werden mit Knoten, die Eigenschaften implizieren, und Kanten, die Beziehungen anzeigen. Unvollständiges Wissen kann durch Defaultwerte ergänzt werden. Sinnvoll ist eine Unterteilung der Wissensobjekte in Ober- und Unterklassen (Vererbungsprinzip).

Man unterscheidet im Wesentlichen zwischen regelbasierten und objektbasierten Systemen: Das sind zwei Sichtweisen derselben Situation: die Regelbasierung ist kanten-, d.h. beziehungsorientiert, die Objektbasierung ist knotenorientiert. Bei regelbasierten Systemen sind Aussagen wie:

Wenn Aussage A gilt, dann gilt auch Aussage B oder
IF Bedingung 1 THEN Aktion A

typisch; bei objektbasierten Systemen werden Ausprägungen von Objekten gleicher Klasse mit verschiedenen Attributwerten miteinander verglichen, um eine Lösung zu erhalten. Oft ist es sinnvoll, einen Teil des Wissens objektorientiert, einen anderen regelbasiert darzustellen. Solche Systeme werden als hybrid bezeichnet.

Die Daten der Wissensbasis werden heute in der Regel in einer Datenbank gespeichert. Die Wissensverarbeitung erfolgt durch ein Computerprogramm. Für die Effizienz des System ist wichtig, dass das Wissensnutzungsprogramm und die Wissensrepräsentation gut aufeinander abgestimmt sind.

Anforderungen an ein Expertensystem sind:

- Transparenz: Die Lösung bzw. der Lösungsweg wird erklärt
- Flexibilität: Elemente der Wissensbasis können hinzugefügt, geändert oder entfernt werden
- Benutzerfreundlichkeit: Der Benutzer benötigt keine besonderen Kenntnisse
- Kompetenz: Es zeigt eine hohe Problemlösungsfähigkeit

Die Entwicklung eines Expertensystems wird als "Knowledge Engineering" bezeichnet und als angewandte künstliche Intelligenz angesehen ([jack87]).

3.2 Entwicklung eines Wissensbasierten Systems

Im Folgenden wird analysiert, wie ein Wissensbasiertes System entwickelt wird; dies soll dazu dienen, die Struktur des Kapitels zu definieren. Ferner wird das Vorgehen bei der Entwicklung des Tariftools davon abgeleitet.

3.2.1 Überblick

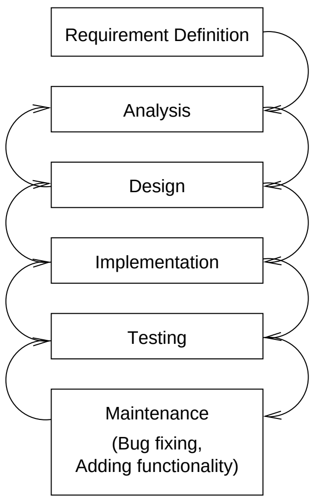

Abbildung 3.2: Lebenszyklus nach Edwards

Im Gegensatz zur Entwicklung herkömmlicher Programmsysteme wird hier nicht ein endgültiges Produkt angestrebt, sondern ein entwicklungsfähiges, das ständig durch neue Wissensdomänen und Regeln ergänzt wird. Der Lebenszyklus eines Wissensbasierten Systems ist in Abbildung 3.2 nach Edwards ([edw91]) abgebildet. Analoge Lebenszyklen finden sich bei Kurbel ([kur92]), Harmon ([haki86]) und Winstanley ([win87]).

Entsprechend soll auch bei der Entwicklung des Tariftools vorgegangen werden. Die einzelnen Phasen der Lebenszyklen von Wissensbasierten Systemen werden in den folgenden Abschnitten erläutert.

3.2.2 Problemanalyse (Requirement Definition)

In der Problemanalyse muss geklärt werden, ob sich ein Expertensystem für das Problem eignet. Das ist z.B. dann der Fall, wenn es nicht durch einen Algorithmus gelöst werden kann oder wenn es mehrere zulässige Lösungen gibt ([kra86]). Ferner sind **Kriterien für die Nützlichkeit eines Expertensystems** zu betrachten:

Nach Gröling ([groe96]) sollte das Wissen so stabil sein, dass die Pflege der Wissensbasis in einem akzeptablen Rahmen bleibt. Nach Schnupp ([schn86]) sind Voraussetzungen für den Einsatz von Expertensystemen:

- großer Nutzen,
- Experte nötig und vorhanden,
- mehrere zulässige Lösungen möglich und
- Änderung der Wissensbasis während der Einsatzzeit.

Diese Voraussetzungen sind für das Tariftool gegeben.

Die Eignung eines Problems zur Lösung durch ein Expertensystem ist am besten gesichert, wenn ein ähnliches Problem bereits erfolgreich so gelöst wurde. Beim vorliegenden Problem ist das das Assessmentsystem (s. Abschnitt 2.4.6, [weis90]). In der Wirtschaft sind dies Probleme der Unterstützung von:

- Forschung und Entwicklung, Fertigung,
- Beschaffung und Lagerhaltung,
- Vertrieb,
- Personal und Ausbildung,
- Finanzen / Rechnungswesen,
- Unternehmensplanung.

Das Tariftool-Problem fällt in den Bereich Vertrieb.

Grenzen der Anwendbarkeit eines Expertensystems sind nach Krallmann ([kra86]):

- Verarbeitung natürlich-sprachiger Texte,
- Lernendes System,
- Zuviel Alltagswissen benötigt,
- Erklärungskomponente muss zuviel leisten,
- Zu großes Anwendungsgebiet.

Diese Grenzen werden vom Tariftool-Problem nicht überschritten.

3.2.3 Anforderungsanalyse (Analysis)

Hier muss analysiert und entschieden werden, ob ein WBS für das vorliegende Problem eine geeignete Lösung ist.

3.2.4 Entwurf (Design)

Wenn man sich für ein Expertensystem entschieden hat, ist

- die **Wissensrepräsentation** festzulegen (s. Abschnitt 3.3)
- die **Problemlösungsstrategie** festzulegen (s. Abschnitt 3.4)

Wissenserwerb
Nachdem die Wissensrepräsentation festgelegt ist, kann man an die **Wissensakquisition** (Wissenserwerb) gehen. Diese hängt von der Aufgabenstellung ab; oft gibt es schon alte Datenbestände, die einbezogen werden müssen. Ein wichtiger Teil wird aber immer vom Experten einzuholen sein. Nach Bullinger u.a. ([buwa89]) sind folgende Aufgaben zu bewältigen:

- Erfassung: am besten durch Interviews mit mehreren Experten,
- Interpretation,
- Strukturierung: Deklaratives Wissen → Objekte; Prozedurales Wissen → Regeln,
- Repräsentation,
- Generierung der Wissensbasis.

Wird ein Expertensystem-Entwicklungssystem verwendet, muss die Wissenstruktur gegebenenfalls noch entsprechend angepasst werden. Diese eignen sich aber nur bei sehr ähnlich gelagerten Problemen.

In den älteren Expertensystemen lagen alle Daten im Arbeitsspeicher, inzwischen geht man bei größeren Systemen dazu über, die Wissensbasis zumindest teilweise in einer Datenbank zu halten. Dabei gibt es den:

- statischen Anschluss: alle Daten werden beim Laden des Systems aus der Datenbank ausgelesen, und den
- dynamischen Anschluss: erst zur Laufzeit werden die gerade benötigten Daten aus der Datenbank geholt.

Attribute von instantiierten Objekten lassen sich leicht in einer Relation abspeichern; schwieriger ist es mit Regeln: aber enthaltene Konstante können als Attributwerte in die Datenbank gebracht werden. In Objektrelationalen Datenbanken lassen sich auch Methoden speichern ([gfh90]). Datenbanken haben gegenüber Dateien den Vorteil größerer Robustheit und Recovery bei Absturz.

Für das Tariftool ist eine Datenhaltung in einer relationalen Datenbank vorgesehen; ein dynamischer Anschluss ist sinnvoll, da nicht immer alle Daten im Arbeitsspeicher gehalten werden müssen.

3.2.5 Implementierung (Implementation)

Benutzeroberfläche
Da die **Benutzeroberfläche** anwendungsspezifisch ist, kann hier nur darauf hingewiesen werden, dass die Akzeptanz des Wissensbasierten Systems durch die Benutzer zum größten Teil von der Benutzerfreundlichkeit der Oberfläche abhängt ([edw91], [pgpb96]). Es sind die Kenntnisse und Fähigkeiten der Benutzerzielgruppe zu berücksichtigen ([gfh90]):

- Hat der Benutzer allgemeine Fachkenntnisse?
- Ist der Benutzer mit der Bedienung von Computern vertraut?

Implementierung
Bei der **Implementierung** von wissensbasierten Systemen, die kommerziell genutzt werden sollen, ist zu bedenken, dass Kunden ungern ein KI-System auf einer sonst nicht verwendeten Workstation haben wollen, sondern lieber passend zu ihrer Computerumgebung. Deshalb, und weil die Performanz von Systemen, die in PROLOG oder LISP implementiert sind, oft nicht ausreichend ist, werden Expertensysteme in der letzten Stufe oft in übliche Programmiersprachen wie C, Pascal, Forth oder FORTRAN umgeschrieben ([kur92], [weis90]) Objektorientierte Systeme werden meist in C++ programmiert ([Wink 91], [Brau 89]).

3.2.6 Testphase (Testing)

Wenn ein Prototyp getestet wird, zeigt sich, ob die Ergebnisse schnell genug gefunden werden, und ob eine Optimierung erforderlich ist.

3.2.7 Wartung (Maintenance)

Bei der **Wartung** sind unabhängig voneinander die Inferenz zu verbessern und die Wissensbasis zu aktualisieren. Besonders bei Hinzufügen von Regeln muss darauf geachtet werden, dass keine Inkonsistenzen entstehen. Korrektheit wird nur von einigen logikbasierten Systemen nachgewiesen. Vollständigkeit der Lösungen kann insbesondere bei großen Systemen nicht garantiert werden; aber es gibt oft ähnliche oder bessere Lösungen, als ein menschlicher Experte findet ([gfh90]).

3.3 Techniken der Wissensrepräsentation

Wissensrepräsentation und Problemlösungsverfahren sind für die Performanz und damit die Akzeptanz durch den Benutzer wesentliche Grundlagen. Sie werden daher im Folgenden vertiefend betrachtet.

3.3.1 Einführung

Prinzipiell sind mindestens zwei Arten von Wissen zu repräsentieren:

- Daten, die die Objekte beschreiben, aus denen z.B. eine Diagnose oder ein Gesamtsystem abgeleitet werden soll, und
- Regeln, die diese Daten miteinander verknüpfen.

Meist ist es sinnvoll, sie auf verschiedene Weise darzustellen. Zur performanten Auswertung sollten Objekte in Objektbäume oder Sachgebiete aufgegliedert sein; entsprechend sollten auch die Regeln in Regelbäumen oder an Kontexte, die die Anwendung spezifizieren, angeordnet sein ([joha90]).

Das Expertenwissen lässt sich in Fachgebietswissen und Erfahrungswissen (heuristisches Wissen) aufteilen. Letzteres wird als Metawissen oder Kontrollwissen abgespeichert. Dazu kommt allgemeines Wissen aus der realen Welt, von dem aber nur ein begrenzter Teil aufgenommen werden kann ([gfh90]). Zur Bearbeitung einer bestimmten Aufgabe wird fallspezifisches Wissen eingegeben. Lernfähige Systeme können dieses zusammen mit dem Resultat in die Wissensbasis aufnehmen.

Die Art der Wissensrepräsentation richtet sich nach der Basisauswertung der Information. Die wichtigsten Arten sind regelbasiert, objektbasiert oder logikbasiert ([lufa91]). Diese werden im Folgenden kurz erklärt. Daneben gibt es Constraints, Probabilistic Reasoning, Non-monotonic Reasoning und Temporal Reasoning.

3.3.2 Regelbasierte Repräsentation

Alles Wissen ist in Regeln (wie z.B. IF A&B THEN C) niedergelegt, die Regelbäume bilden können. Daneben ist Objektwissen in Tabellen (Objekte, Attribute) dargestellt.
Ein inkrementelles Erweitern des Systems ist möglich. Neben solchen starren Produktionsregeln gibt es probabilistische Regeln, die folgende Form haben ([groe96], [pupp93]):

```
Wenn Praemisse 1 mit Wahrscheinlichkeit P1
und  Praemisse 2 mit Wahrscheinlichkeit P2
dann Konklusion mit Wahrscheinlichkeit X
```

Der Grad der Unsicherheit wird von Experten geschätzt oder aus repräsentativen Statistiken abgeleitet.

Neben den Regeln gibt es eine Menge Domänendeklarationen, in denen die Objekte beschrieben werden, auf die sich die Regeln beziehen. Beides zusammen bildet die Wissensbasis. Die Domänen enthalten Objekte, diese Attribute und deren Werte. Tripel (Objekt - Attribut - Wert) werden betrachtet. An Daten sind noch Variablen (Fakten) nötig, um Eingabewerte und Zwischenergebnisse festzuhalten.

3.3.3 Objektbasierte Repräsentation

Die objektorientierten Darstellungen (Objektklassen) werden im Zusammenhang mit WBS auch oft als Frames oder Schemata bezeichnet, Attribute als Slots. Die Objekte haben Methoden (Behaviors, Aktionen - bei Frames als Scripts bezeichnet), deren Ausführung durch Versenden von Nachrichten veranlasst wird. Daneben gibt es Get- und Set-Aktionen auf die Werte der Attribute.

Die objektbasierte Darstellung leitet sich von sog. semantischen Netzen ab, in denen es Vererbung (ist ein) und Asssoziation bzw. Komposition (hat, ist Teil von) gibt (Abbildung 3.3).

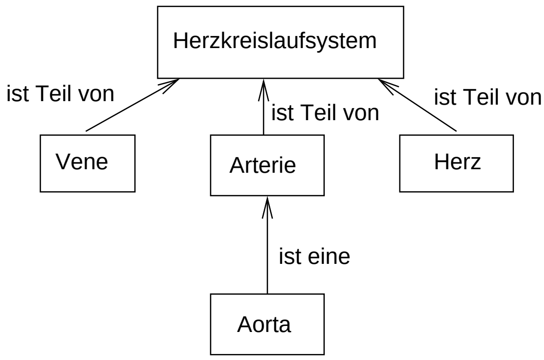

Abbildung 3.3: Semantisches Netz

Nach Jackson und Puppe ([jack87], [pupp93]) entsprechen strukturierte Objekte Zustandsknoten in der Graphentheorie, Beziehungen (Kanten) können gewichtet und gerichtet sein: "Semantische Netze" = assoziative Netze ([schn86], [lufa91]); Wälder (Menge von Bäumen) entsprechen Objektklassen.

3.3.4 Logikbasierte Repräsentation

Einfache Regel-Faktensysteme sind die einfachsten Systeme, die zur logischen Wissensdarstellung verwendet werden können. Beim System einer Schaltkreistechnik ist ein Fakt z.B. gegeben durch:

```
and_gate(a)    (Bedeutung: a ist ein AND-Gatter)
```

und eine Regel z.B. durch:

$$\forall X \quad and\_gate(X) \wedge val(in1, X, 1) \quad \wedge val(in2, X, 1) \Rightarrow val(out, X, 1)$$

(Wenn an beiden Eingängen eines AND-Gatters die Spannung 1 liegt, dann liegt sie auch am Ausgang.) Aus solchen Fakten und Regeln kann die Funktion des Schaltkreises durch logische Folgerungen ermittelt werden ([gfh90]).

Nach G.Brewka (in [sch87]), Puppe ([pupp93]) und Jackson ([jack87]) werden in logikbasierten Systemen Hornklauseln ausgewertet, die spezielle Formeln der Prädikatenlogik sind, nämlich

Klauseln mit genau einem positiven Literal (nicht negiertes Atom) ([klei92]):

$$P \vee \neg Q_1 \vee ... \vee \neg Q_n$$

Hornklauseln haben in der Syntax von Prolog, das zur Problemlösung von Expertensystemen mit logikbasiertem Wissen häufig benutzt wird, die allgemeine Form:

$$P : -Q_1, ...Q_n.$$

haben. P ist dabei das Ziel ([kra86]). Prolog impliziert Backtracking, was aber nicht bei jeder Anwendung sinnvoll ist.

Eine Spezialform logikbasierter Systeme verwendet **Fuzzy Logik**. Dabei geht es um die Behandlung unsicherer Daten oder Regeln. Es wird ein Certaintyfaktor angezeigt, der die Zugehörigkeit zu einer Menge mit 0 (keine) bis 1 (vollständig) angibt. Beispiel: Eine Bakterie hat eine ovale Form, dann gehört sie etwa mit der Certainty 0.4 zu den kugelförmigen und mit 0.6 zu den stäbchenförmigen Bakterien.

3.3.5 Hybride Repräsentation

Häufig werden mehrere Darstellungen nebeneinander genutzt (hybride Systeme); z.B. Regeln neben Objekten ([pupp93], [jack87], [sch87]). Dabei kann eine Wissensbasis mit strukturierten Objekten objektorientiert dargestellt und behandelt werden. Regeln dienen zur Optimierung von Suchverfahren.

3.3.6 Charakteristik der Repräsentationen

Bei regel- und logikbasierten Systemen liegt das Wissen in Form von Regeln oder logischen Aussagen vor, die meist verkettet sind ([krfr89]). Bei dem WBS *XCON* zur Konfiguration von Computern sind es z.B. etwa 3000 Regeln ([jack87]). Bei juristischen Systemen kommt es vor, dass Regeln (Gesetzestexte) verändert werden, so dass aus einem Tatbestand ein anderes Urteil folgt. Eine feste Codierung dieser Regeln ist daher nicht möglich, die regelbasierte Wissensbasis muss aktualisiert werden können.

Bei objektorientierten WBS dagegen liegt das Wissen in Form strukturierter Objekte vor. Das zu lösende Problem ist die Auswahl von Objekten durch Muster- und Größenvergleich ihrer Attribute. Diese Regeln können in einer objektorientierten Sprache fest codiert werden, da die anzuwendenden mathematischen Regeln keiner Revision unterliegen.

Ein solches Problem ist auch das des Tariftools. Da bei diesem auch noch einige Regeln nötig sein könnten, die eine Optimierung der Suche erlauben, ist ein hybrides System mit Objektorientierung und Regeln für das Tariftool die beste Lösung.

3.4 Techniken der Problemlösung

3.4.1 Inferenzmaschine

Die Inferenzmaschine (das Problemlösungsprogramm) steuert die Verarbeitung der statischen Daten, die teils tabellenartig (z.B. als Objekte), teils als bedingte Aktionen (Regeln, Methoden) vorliegen. Zwischenergebnisse oder Trace werden in Variablen abgelegt ([edw91]). Soweit Objekt- oder Regelbäume vorliegen, verläuft die Steuerung meist nach dem Prinzip "Teile und herrsche" ([joha90]). Im Prinzip führt die Inferenzmaschine immer eine Suche durch, z.B. nach der als nächste anzuwendenden Regel.

3.4.2 Suchmethoden

Beim Suchen in einer relationalen Datenbank kann über eine DB-Schnittstelle SQL verwendet werden.

Für das Suchen im Arbeitsspeicher gibt es zwei Basistypen für die Inferenz, die bei baumartigen Strukturen von Regeln oder Objekten anwendbar sind:

- **Top-down:** Ein Baum wird von der Wurzel aus durchsucht.
- **Bottom-up:** Ein Baum wird von den Blättern aus durchsucht.

In beiden Fällen kann die Suche nach Finden des ersten Ergebnisses abgebrochen oder erschöpfend durchgeführt werden. Letzteres wird meist durch heuristische Verfahren vermieden. Bei beiden Basistypen kann auch Tiefensuche oder Breitensuche angewendet werden ([gfh90]).

3.4.3 Regelbasierte Methoden

Nach Brewka (in [sch87]) erfolgt die Problemlösung über:

- **Rückwärtsauswertung oder zielgerichtete Auswertung:**
  Dies ist eine Topdown-Methode, die bei Regeln angewendet wird, deren Aktion darin besteht, neue Fakten abzuleiten (Hypothesen - Regel - Prämisse).

  Ein Ziel oder eine Menge möglicher Ziele wird vorgegeben. Aus den Regeln wird rückwärts auf die Bedingung geschlossen, bis man bei den Eingabefakten angekommen ist (Rückwärtsverkettung). Dies wird meist in der Diagnostik bei nicht so großer Datenmenge angewandt. Die Auswertungsreihenfolge ist nicht vorgegeben, Backtracking wird angewandt, wenn die Regeln nicht gleich zum Ziel führen. Die Anwendung von PROLOG ist meist möglich.

  Bei der Fehlerdiagnose wird ausgehend vom beobachteten Fehlerbild (evidence) nach der Fehlerursache (cause) gesucht ([edw91]).
  Die Regeln haben die Form:

  IF cause THEN evidence

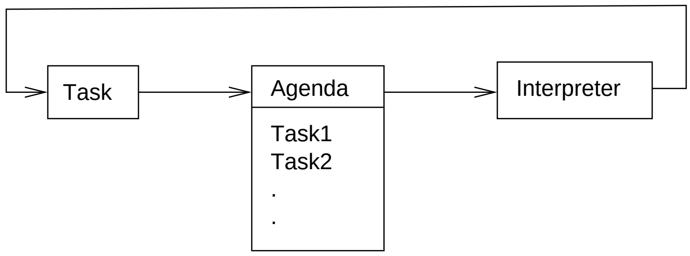

Abbildung 3.4: Fallbearbeitung bei CENTAUR

Die geprüften Regeln werden in eine Agenda(liste) eingetragen und vom Inferenzprogramm abgearbeitet. Als Resultat wird eine Liste von möglichen Ergebnissen und ihre Bewertung ausgegeben (Abbildung 3.4).

- **Vorwärtsauswertung oder data-driven:** Dies ist eine Bottom-up-Methode, möglich bei Regeltypen wie
  Fakten → Regel → Ableitung
  Die Auswertung der Regeln erfolgt vorwärts, angefangen mit den Eingabefakten (Vorwärtsverkettung). Das Verfahren wird meist bei großer Datenmenge angewandt. Zur Vereinfachung der Suche wird Patternmatching angewandt. Dabei wird die Eingabesymptomliste mit den Symptomlisten der einzelnen Objekte verglichen.

Regeln werden zunächst in heuristischer Form angegeben und dann in Produktionsformeln umgewandelt, z.B. wird aus der heuristischen Regel:

```
if
  the patient has a pain in the calf when walking,
  which disappears gradually in rest
then
  a stenosis of one of the arteries in the leg, possibly due to
  atherosclerosis, is conceivable
```

die Produktionsformel:

```
if
  same(complaint, calf-pain) and
  same(presence, walking) and
  same(absence, rest)
then
  add(cause, arterial-stenosis) also
  add(disorder, atherosclerosis)
fi
```

Die Funktion same(...) hat als Ergebnis true oder false; die Funktion add fügt die Konklusion der Ergebnismenge als Fakt zu. Daneben gibt es noch Funktionen wie less(a,b) u. ä., die durch das Inferenzprogramm ausgewertet werden können. Ist das Ergebnis der Produktionsformel "true", dann heißt es: Die Regel feuert (d.h. setzt Aktion in Gang).

Während der Abarbeitung der Regeln durch die Inferenzmaschine werden Fakten zur Faktenmenge (Ergebnis) zugefügt oder gelöscht. Das Vorgehensprinzip ist in Abbildung 3.5 dargestellt.

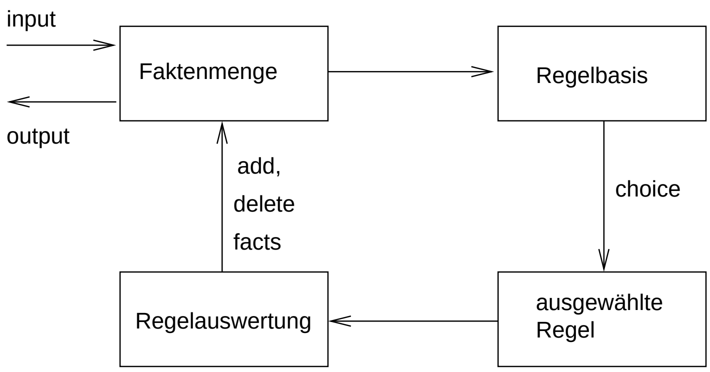

Abbildung 3.5: Inferenz in regelbasiertem Expertensystem

3.4.4 Objektbasierte Methoden

Bei objektbasierten Expertensystemen werden nacheinander die Methoden der Objekte ausgeführt. Diese können entweder Methoden anderer Objekte aufrufen oder Nachrichten an sie schicken. Backtracking ist hier nicht möglich. In vielen Fällen wird auch einfach eine Tiefen- oder Breitensuche durchgeführt. Die Suche kann durch "hill-climbing" beschleunigt werden, dabei muss darauf geachtet werden, dass man nicht nur ein lokales Maximum findet ([edw91]). Unter "hill-climbing" versteht man den Versuch, immer nur Suchwege einzuschlagen, die den aktuellen Zustand verbessern ([klei92]).

3.4.5 Logikbasierte Methoden

Bei logikbasierten Expertensystemen werden neue Regeln aus den in der Wissensbasis vorhandenen Regeln logisch abgeleitet (Reasoning oder Inferenz: [haki86], [lufa91]). Dies dient meist zum Beweis mathematischer Theoreme mittels Prädikatenlogik erster Stufe. Es werden Hornklauseln ausgewertet, meist mit PROLOG, das darauf abgestimmt ist. Die grundlegende Problemlösungsmethode heißt Resolutionswiderlegung. Damit kann man vereinfachte Formeln ableiten oder Widersprüche zwischen Klauseln zeigen. Backtracking ist hier möglich, aber nicht für jede Anwendung sinnvoll. Inkonsistenz ist auch hier möglich ([jack87]). Abgeleitete Informationen müssen von menschlichen Experten überprüft werden, da der Computer nicht über das allgemeine Wissen eines Menschen verfügt und die Grenzen seines Wissens nicht erkennt.

3.4.6 Sonstige Lösungsmethoden

Haupttypen von Problemlösungsmethoden sind nach Puppe u.a. ([pupp93], [pgpb96]):

- **Klassifikation**: Die Lösung wird aus einer Menge von Alternativen ausgewählt. Dabei werden z.T. heuristische Methoden, die auf unsicheren Regeln beruhen, angewandt. Nach dem Theorem von Bayes kann auf die Wahrscheinlichkeit der Lösung geschlossen werden. Lösungen werden auch mit modellbasierten, statistischen oder fallbasierten Methoden gewonnen.
- **Konstruktion**: Die Lösung wird aus einer Menge von Komponenten zusammengesetzt.
- **Simulation**: Die Lösung gibt die Reaktion eines Systems auf bestimmte Eingaben ([joha90]).

Prinzipiell können diese Methoden auf alle Arten von Wissensrepräsentationen angewandt werden; je nach Anwendungsfall kann sich eine als besonders geeignet erweisen.

3.4.7 Für das Tariftool geeignete Techniken

Für das Tariftool kommen aufgrund der erforderlichen objektorientierten Datenrepräsentation nur Suchmethoden und objektbasierte Methoden in Frage.

3.5 Architekturen hybrider Systeme

In hybriden Systemen wird oft der regelbasierte Teil der Problemlösung in einer anderen Programmiersprache implementiert als der objektorientierte und der logische Teil des WBS. Für die Zusammenarbeit dieser Komponenten gibt es folgende Architekturen ([kra86]):

- **Kompilierende Architektur:** Alle Komponenten werden in eine gemeinsame niedrigere Implementierungssprache übersetzt.
- **Heterarchische Architektur:** Die Komponenten kommunizieren, werden auf Anfrage aktiv und tauschen Ergebnisse aus.
- **Hierarchische Architektur:** Die Komponenten kommunizieren über eine Metakomponente, die Anfragen an die zuständige Basiskomponente weitergibt und Ergebnisse an die richtige Stelle leitet. Ein Beispiel dafür ist das Entwicklungssystem BABYLON mit Regeln, PROLOG, Objektorientierung, erweiterbar auch mit LISP-Prozeduren.

Für das Tariftool ist eine kompilierende Architektur mit Implementierung in Java vorgesehen.

3.6 Bestehende Implementierungen und Produkte

Nach Helten ([helt92]) gibt es für WBS die in Abbildung 3.6 dargestellten Anwendungsdomänen. Unser Tariftool ist in der äußeren Schale anzusiedeln.

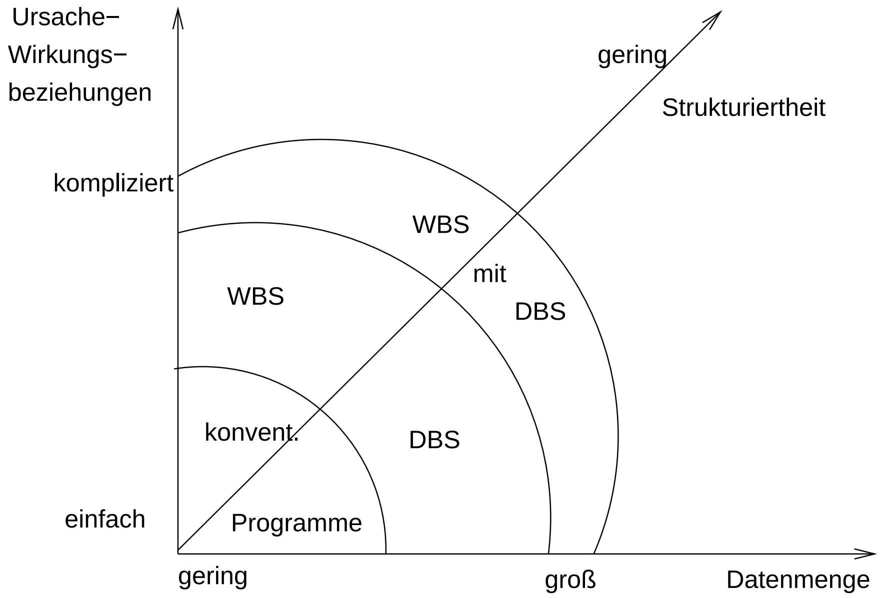

Abbildung 3.6: Anwendungsdomänen für WBS

Im Folgenden werden beispielhaft einige Anwendungen vorgestellt. Bei Anwendungen unterscheidet man nach Puppe u.a. ([pupp93], [pgpb96]), Schnupp u.a. ([schn86], [gfh90]),[kur92], [joha90]):

- **Diagnosesysteme für:**
  - Medizinische Systeme: Aus Symptomen wird auf Erkrankungen bestimmter Organe, wie Blut, Herz und Leber geschlossen und gegebenenfalls auch eine bestimmte Behandlung vorgeschlagen; Beispiele: MYCIN ([kra86], [jack87], [lufa91], [gfh90]), CENTAUR ([jack87],[lufa91]) und INTERNIST ([jack87], [lufa91]).
  - Technische Systeme: Diese werden zur Fehlersuche bei komplexen Maschinen und technischen Systemen sowie zur Strukturbestimmung chemischer Stoffe (Beispiel: HEURISTIC DENDRAL, [lufa91], [jack87], [gfh90], [haki86]) eingesetzt.
- **Assessmentsysteme (Schätzung, Vorhersage):** Hier wird anhand der Basisdaten geprüft, ob z.B. ein Projekt zu bestimmten Kosten und in einer vorgegebenen Zeit durchführbar ist oder ein Kredit- oder Versicherungsvertrag mit einem Kunden geschlossen werden kann (Beispiel TAREX [weis90]).
- **Präzedenz-Systeme:** Diese Systeme dienen dazu, kundenspezifisches Investment zu ermitteln, zur Produktauswahl z.B. aus einem Katalog oder zur Zusammensstellung von Zusatzteilen entsprechend der Kundenwünsche z.B. bei einem Auto. Im hybriden WBS BELI ([krem89]), einem Bestellsystem, wird die Wissensbasis aus strukturierten Objekten (Artikeln) gebildet, die objektorientiert dargestellt und verwaltet werden. Durch Regeln, die Preise, Liefertreue ect. berücksichtigen, wird ermittelt, an welchen Lieferanten die Bestellung geht.

- **Objekt-Identifikation:** Dabei wird nach der Klassifikation bestimmter Objekte aus der Botanik oder der Geologie (PROSPEKTOR, [jack87]) gesucht.
- **Beratung (Rechts- , Wirtschafts- , Finanzfragen):** Die logische Struktur der Ableitung von rechtlichen Entscheidungen kommt der Darstellung in einem regelbasierten oder logischen WBS entgegen. Problematisch ist die Umsetzung natürlicher Sprache, Einbindung von Allgemeinwissen und Fallvergleiche ([fitr86], Beispiel: LEX ([efh86], [fitr86])), EVA ([weis90]).
- **Überwachung, Steuerung, Qualitätskontrolle:** MEBADOK ([thur95]) überprüft die Konsistenz von Patientendaten aus drei verschiedenen Quellen, einer Patientenbasis, einer Diagnostikdatei und einer Pflegedatei durch Abgleich und Plausibilitätsprüfungen. Diese Dateien bilden zusammen mit Regeln, die Relationen zwischen drei Datentypen (Diagnose, Operation, Histologie) abbilden, die Wissensbasis in einer Datenbank. Das Inferenzprogramm wurde in FORTRAN77 implementiert.
- **Wissensweitergabe (Instruktion):** Beispiel IBUS (Intelligentes BenutzerUnterstützendes System [mali91]) ist ein WBS, das Benutzer auf den Einsatz eines komplexen Softwaresystems vorbereiten und bei der Bedienung durch flexible, kontextsensitive und adaptive Hilfemeldungen unterstützen soll. Die Wissensrepräsentation ist hybrid, Fachwissen und das für die Bedienung eines Computers erforderliche "Weltwissen" sind als Objekte dargestellt, die von einer Oberklasse erben. Ferner werden in einem Benutzermodell Lernfortschritte des Benutzers gespeichert und der Dialog dessen Wissen laufend angepasst. Daneben gibt es Produktions- und Metaregeln. Die Wissensbasis lässt sich durch andere Fachinformationen austauschen. Die Inferenzmaschine wurde in C als Prototyp implementiert.
- **Planung (Design) unter globalen Randbedingungen (Constraints):** Beispiel ELDAR ([pop92]) ist ein hybrides WBS, in das die schon bestehende relationale Datenbank mit 470000 Einträgen von Faktenwissen über Elektrolyte als Teil der Wissensbasis einbezogen wurde. Ferner gehören zur Wissensbasis eine Regelbank und eine Methodenbank. Das Problemlösungssystem besteht aus einem Kommunikationsmanager (in C), der Aufgaben durch Anfragen an die entsprechenden Teile der Wissensbasis löst. Der Informationsaustausch im WBS erfolgt über Kommunikationsbereiche. Die Regelbank enthält neben Regeln und Metaregeln PROLOG, die Methodenbank Module, die Berechnungen mit in Fortran, Pascal oder Assembler codierten Methoden durchführen. Das System läuft auf MS/DOS- oder SCO-UNIX-Computern. Die Serviceleistungen von ELDAR sind Faktenretrieval, Inter- und Extrapolation von Messergebnissen, Berechnung von thermodynamischen und Transporteigenschaften sowie Verifikation von Prognosen. Es kann z.B. bei der Entwicklung von Batterien eingesetzt werden.

Von diesen Beispielen sind 6 regelbasiert, 1 logikbasiert (INTERNIST), 3 hybrid mit Objektorientierung und Regeln (BELI, EVA, IBUS). Die WBS ELDAR und MEBADOK verwenden Datenbanken zur Speicherung der Wissensbasis. Regelbasierte Systeme haben Objektdaten meist in Frames deklariert, die man als Vorstufe von objektorientierter Darstellung ansehen kann.

3.7 Zusammenfassung

Das Tarifproblem ähnelt einem Beratungsproblem, z.B. EVA (Expertensystem zur Vermögensanlageberatung), das als hybrides WBS implementiert ist. Ein WBS mit Datenbank erfüllt Anforderung 9 (Abschnitt 2.2) nach persistenter Speicherung der Basisdaten und Anforderung 10 nach Speicherung beliebig großer Datenmengen. Auch die Erfüllung von Anforderung 8 nach Skalierbarkeit wird durch ein solches WBS gewährleitet. Daneben kann ein hybrides WBS mit Objektorientierung die Anforderung 1 nach strukturierter Datenspeicherung erfüllen. Ein hybrides WBS mit Regeln kann die Performanz optimieren (Anforderung 11). Aus der Analyse der in der Literatur beschriebenen WBS ergibt sich demnach, dass das Tariftool als hybrides WBS mit Datenbank als Wissensbasis realisierbar ist.

Kapitel 4

Analyse der Tarifierung

Zu den Anforderungen an das Tariftool gehört auch eine geeignete Tarifformel. Daher wird in diesem Kapitel analysiert, wie Tarifierung bisher realisiert wird. Zunächst wird der Tarifierungsvorgang beschrieben, zu dem Haushaltsplanung, Kostenanalyse und Untersuchung der Kundenwünsche gehört. Eine Diskussion der möglichen Tarifparameter schließt sich an. Danach werden die Verfahren zur Aushandlung eines Tarifs erläutert und Tarifformeln sowie Architekturen von Abrechnungssystemen vorgestellt. Zuletzt werden Erfahrungen mit pauschaler und nutzungsorientierter Abrechnung beschrieben.

4.1 Einleitung

Um dem Leser das Verständnis der folgenden Abschnitte zu erleichtern, werden zunächst relevante Begriffsdefinitionen aufgeführt. Danach wird der Tarifierungsvorgang beschrieben. Mit dem Tarif sollen Gebühren so ermittelt werden, dass sie die Kosten des Providers decken. Diese werden dem Kunden in Form von verbrauchten Ressourcen in Rechnung gestellt. Die Ressourcennutzung kann durch ein stochastisches Modell ermittelt werden oder über Dienstgüteklassen als maßgebende Kostenfaktoren.

Der Tarif kann mit dem Kunden vor der Nutzung ausgehandelt werden und dann während der Sitzung fest bleiben (statisches Modell). Es kann aber auch während der Sitzung Nachforderungen geben, besonders, wenn Überlastung des Netzes zu erwarten ist (dynamisches Modell). Tarifformeln dienen zur Berechnung der Gebühren. Sie enthalten Abrechnungseinheiten, die auf Nutzungsmessungen beruhen, und Tarifparameter, die mit dem entsprechenden Abrechnungseinheiten multipliziert einen Preis ergeben.

4.2 Begriffsdefinitionen

In der Telekommunikationsliteratur kommen oft Begriffe mit unterschiedlichen Bedeutungen vor. Im Folgenden werden die Begriffe so definiert, wie sie in dieser Arbeit verwendet werden.

- *Dienst:* Eine vom Dienstleister bereitgestellte und betriebene Funktionalität mit einer gewissen Güte, die vom Dienstnehmer an einer Schnittstelle genutzt werden kann ([schm01]). Ein Dienst kann eine Menge von Subdiensten enthalten.

- *Subdienst:* Dienst, der in einem Dienst enthalten ist ([schm01]).
- *Dienstmodell:* Für ein Problem aus der Telekommunikation relevante Objekte, ihre Rollen und Beziehungen zueinander.
- *Dienstvereinbarung:* Vertrag zwischen einem Kunden (Dienstnehmer) einem Dienstleister (Provider) über einen Dienst mit geeigneter Beschreibung der Funktionalität des Dienstes. Synonym dazu ist SLA (Service Level Agreement).
- *Service Level:* Für den Dienst (Service) benötigte Dienstgüte.
- *Dienstgüte:* (QoS: Quality of Service) Qualität des Dienstes.
- *QoS-Parameter:* Parameter zur Kennzeichnung der Dienstgüte, wie z.B. maximale Fehlerrate, maximaler Jitter, maximale Verzögerung, aber nicht Bandbreite.
- *Tarifmodell:* Die Policy, nach der die Tariffestsetzung erfolgt.
- *Tarifformel:* gibt funktionalen Zusammenhang zwischen Gesamtpreis und Einzelpreisen, die Konstante oder Produkte von Tarifparametern und zugehörigen Abrechnungseinheiten sind.
- *Tarifparameter:* gibt Preis pro Einheit einer Ressource (z.B. Volumen in Mbit) an (tariff parameter [haca99]).
- *Abrechnungseinheit:* gibt Einheit an, in der ein Tarifparameter diskret abzählbar abgerechnet wird.
- *Abrechnung:* Ermittlung der Gebühren aus Tarif und Nutzung.
- *Tarifierung :* Tariffestsetzung.
- *Provider: kurz für Internet Service Provider (ISP):* Stellt Dienst und/oder Konnektivität zur Verfügung.
- *Kunde:* schließt Vertrag mit Provider über Dienstnutzung ab.
- *Nutzer (User):* nutzt die Dienste des Providers.
- *Outsourcing:* Auslagerung einer Dienstleistung an eine andere Organisation, die die Verantwortung für die Dienstleistung übernimmt.

4.3 Allgemeine Beschreibung des Tarifierungsvorgangs

Bei der Tarifierung werden Kostenfaktoren für die Nutzung von Diensten festgelegt. Diese können pauschal sein oder pro Zeiteinheit bzw. pro Nutzungsvolumen angesetzt werden. Im letzteren Fall muss die Nutzung gemessen werden. In dieser Arbeit wird davon ausgegangen, dass dies für die betrachteten Tarifgrößen möglich ist. Eine Beschreibung der Messungen und Messapparate liegt nicht im Rahmen dieser Diplomarbeit. Für Zeiteinheiten und Volumeneinheiten müssen Dimensionen angegeben werden, z.B ms für Zeiteinheiten, Bytes für Datenvolumen.

Im Folgenden wird zunächst die Kostenermittlung und die Zusammenstellung des Tarifs untersucht.

4.3.1 Haushaltsplanung

Der Provider muss ermitteln, welche Kosten bei ihm anfallen und welche Einkünfte er von den Kunden erzielt. Er muss sein Ziel bestimmen: mindestens Kostendeckung oder Verlust (um Marktanteile zu gewinnen). Sein Budget sollte sich auf einem bestimmten Zeitraum, z.B. 1 Jahr, beziehen, und dann jeweils überprüft werden. Dieses Gewinnen von Feedback ist in Abbildung 4.1 nach [OGC 01] dargestellt.

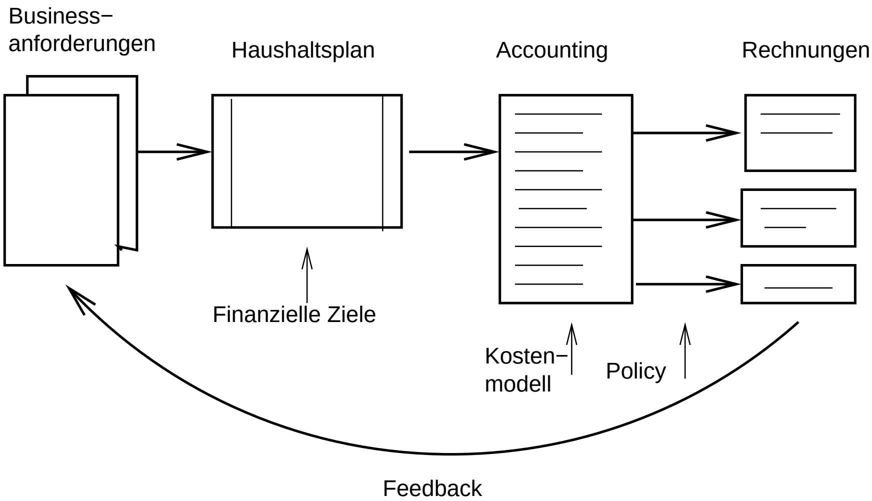

Abbildung 4.1: Modell des IT-Accountings (nach [OGC 01])

4.3.2 Kostenanalyse

Zur Ermittlung der Kosten, die für einen Provider anfallen, ist eine Kostenanalyse erforderlich. Im Einzelnen sind zu betrachten:

- Kosten für Bereitstellung des Netzes mit den benötigten Ressourcen wie z.B. Router und Leitungen,
- Kosten für Betrieb und Verwaltung: Personal, Computer, Miete,
- Kosten für Lizenzen, Investitionen,
- Kosten für Haushaltsplanung, Nutzungsermittlung, Abrechnungen,
- Kosten für die Bereitstellung von “value-added“- Diensten.

4.3.3 Bedürfnisse der Kunden

Um den Bedürfnissen der Kunden gerecht zu werden, sollten bei der Gebührenerhebung u.a. die folgenden Prinzipien berücksichtigt werden. Nutzungsermittlung und Rechnungserstellung sollen nicht zu einem großen Overhead führen, d.h. die Kosten für die Ermittlung der Abrechnungseinheiten müssen in einem angemessenen Verhältnis zu den übrigen Kosten stehen. Die Gebühren sollen nutzungsorientiert sein. Die Rechnung soll für den Kunden verständlich und nachvollziehbar sein. Die Tarife sollen einfach sein. Die Nutzung muss gemessen werden können.

Da Gebührenermittlung und -erhebung erhebliche Zusatzkosten verursachen, ist es für die Provider sinnvoll, sich auf die wesentlichen Nutzungen zu beschränken; Kosten entsprechend der Distanz z.B. machen heute keinen Sinn mehr. Die Gebühren sollen für den Kunden auch verständlich und nachvollziehbar sein.

4.3.4 Zusammenstellung des Tarifs

Übersicht
Die Tarifformel dient zur Berechnung der Gebühren. Sie enthält als Tarifparameter z.B. die Dauer der Nutzung, das Datenvolumen und QoS-Parameter. Tarifparameter können Funktionen anderer Einflussgrößen wie der Tageszeit sein. Zu jedem Tarifparameter gehört eine Abrechnungseinheit, in der die Nutzung erfasst wird: Milliskunden, Pakete, Megabits, Bytes u.a.. Die Nutzung kann in einem Vertrag festgelegt werden (reservierte Bandbreite) oder durch Messgeräte ermittelt werden. Man unterscheidet zwischen:

- **Pauschaler Tarifierung:**
  Hierbei bezahlt der Kunde eine i. allg. monatliche Zugangsgebühr, mit der alles, evt. bis zu einer bestimmten Nutzungsdauer, abgedeckt ist, und

- **Nutzungsorientierter Tarifierung:**
  Bei nutzungsorientierte Tarifierung wird außer der Zugangsgebühr eine Gebühr für jedes `login` verlangt, außerdem, je nach Tarif, Gebühren pro Nutzungsdauer, Größe der Transaktionen und/oder QoS-Parameter.

4.4 Analyse bestehender Kostenmodelle für die Tarifierung

4.4.1 Einleitung

Kosten entstehen den Providern zum einen durch Stausituationen, die eine Einhaltung vereinbarter Dienstgüte verhindern und so zu Regressforderungen führen. Zum anderen entstehen Kosten durch den Aufwand in Technik und Verwaltung zur Gewährleistung von zugesagten Dienstgüteklassen.

4.4.2 Kostenmodelle basierend auf stochastischen Modellen

Allgemeines
Eine Stausituation bildet einen Kostenfaktor, wenn mit Kunden vereinbarte Qualitätsgarantien vom Provider bei Stau nicht mehr eingehalten werden können und er dem Kunden Regress zahlen muss. Problematisch ist, dass Staus im Internet nicht so einfach gesteuert werden können; ein Accessprovider hat keinen unmittelbaren Einfluss auf den Zustand des Internets.
Das Internet ist nämlich auf verbindungslose Kommunikation (IP-Protokoll) und “best effort“ Qualität ausgelegt. Da unvorhersehbar ist, wann Nutzer sich ein- und ausloggen und die Bandbreite jeder Leitung durch statistischen Multiplex ausgelastet wird, ist prinzipiell eine

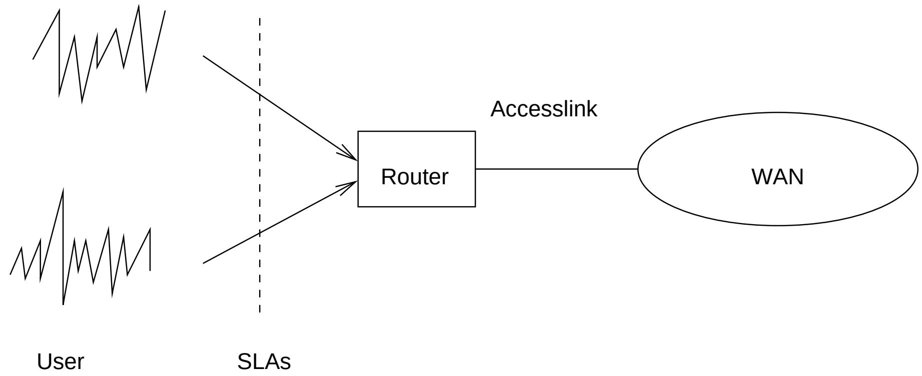

Abbildung 4.2: Modell des Accesslinks

Gewährleistung der bei höheren Diensten erforderlichen QoS nicht möglich. ISP müssen darauf achten, dass das Netz nicht voll ausgelastet ist und Belastungsspitzen abgefangen werden können, sodass kein Datenverlust und keine merkliche Verzögerung auftritt. Dies lässt sich erfahrungsgemäß durch entsprechende Techniken in IP- oder ATM-basierten Netzen erreichen.

Ein Maß für die Kostenerhebung soll die Ressourcennutzung sein.

Stauvermeidung durch stochastische Auswertung von SLAs

Access-Provider können Staus in ihrem Access-Link (s. Abbildung 4.2) vermeiden, wenn sie mittels eines stochastischen Programms ermitteln, ob sie noch einen SLA (Service Level Agreement))zulassen können oder nicht ([coke00], [CoSi99]).
Voraussetzungen dafür sind, dass der Access-Link und das IP-Netzwerk die Protokolle DiffServ (für differenzierte Dienste) und IntServ (für integrierte Dienste) sowie Router enthalten, die Priority Queuing, Class Based Queuing (CBQ) und Weighted Fair Queuing (WFQ) anbieten.

Im SLA muss der Nutzer seine mittlere und seine maximale Bandbreite angeben. Aus diesen wird mittels eines stochastischen Modells, das token buckets zur Glättung von Spitzen des Datenverkehrs und eine große Menge Nutzer auf dem Access-Link annimmt, eine sog. effektiv genutzte Bandbreite (EB) ermittelt (Einzelheiten s. Anhang A.2). Die maximale Bandbreite des Access-Links abzüglich der Summe der EBs aller Nutzer auf dem Link ergibt die noch verfügbare Bandbreite des Access-Links.

Nutzungsabhängige Kosten sollen die Nutzer veranlassen, nicht mehr Volumen zu belegen, als sie brauchen. Der Preis, den Nutzer entrichten müssen, wächst mit ihrer effektiv genutzten Bandbreite. Es wird angenommen, dass sie sich so preisbewusst verhalten. Feldforschungen unterstützen die Richtigkeit dieser Annahme. Die Abrechnung erfolgt nach der ABC-Formel ch= aT + bV + c ([song99]).

Prioritätsklassen als Maß für die Ressourcennutzung
Nach Gupta ([gswp95]) wird mit einer stochastischen Methode die Auswirkung eines dynamischen Priority-Modells auf die Kosten für Provider und Nutzer ermittelt. Durch Simulation eines Modells mit 4 bzw. 2 Prioritätsklassen wird der Nutzen als Funktion der Bandbreite berechnet. Danach ist der Durchsatz bei Flatrate niedriger als bei nutzungsbasierter Abrechnung. Wenn die Auslastung an die Grenze der verfügbaren Bandbreite geht, hilft auch die beste nutzungsbasierte Abrechnung nicht mehr; eine Erweiterung ist dann erforderlich.

Analoge stochastische Berechnungen werden von Stiller ([srl 00]), Reichl ([reic99]), Kelly ([kell95]), Kenyan ([kech01]), MacKie-Mason ([mava94]) und Villasis ([vill96]) durchgeführt.

4.4.3 Dienstgüteklassen als maßgebende Kostenfaktoren

Einleitung
Die Gewährleistung von höheren Dienstgüten (QoS) stellt technische Anforderungen an den Provider wegen der Wahrscheinlichkeit von Paketverlusten, Schwankungen der Datentransferrate und Inkonsistenz der Verzögerung in einem Datenstrom. Dienste stellen unterschiedliche Anforderungen an QoS (z.B minimale Verzögerungsschwankungen oder hohe Transfergeschwindigkeit). Durch Preis pro QoS sollen Nutzer veranlasst werden, nur soviel QoS zu fordern, wie sie wirklich brauchen.

Bei Diensten mit bestimmten QoS-Anforderungen muss der Provider deren Einhaltung überwachen. Neben der Installation von Messgeräten sind dazu laufende Messungen und Auswertung der Messergebnisse, gegebenfalls auch QoS-erhaltende Maßnahmen nötig, die der Provider erbringen muss, wenn er Verluste durch Regressforderungen vermeiden will ([rls99]).

QoS-angereicherte Dienste werden erreicht mit:

- **RSVP/IntServ:** Die benötigten Ressourcen werden auf der ganzen Strecke vom Sender zum Empfänger reserviert. Kosten fallen für die Dauer der Reservierung einer bestimmten Bandbreite an ([cazs99], [gswp95], [mava94], [gswt95]).

- **DiffServ:** Hier erfolgt keine Reservierung. Die Nutzer können bestimmte Serviceklassen (z.B. Premium Service, Assured Service) wählen und bekommen eine Abrechnung entsprechend der beanspruchten Dienstgüte. Diese kann nicht garantiert werden; bei Überlastung der Ressourcen sinkt die Dienstgüte ([kssw00], [dfsb99]).

Preisfaktoren im Internet
Nach Karsten u.a. ([kssw00]) gibt es folgende Serviceklassen im Internet:

- **Overprovisioned Best Effort:** Datenverkehr wie bisher im Internet, aber Nutzer haben überreichlich Netzkapazität zur Verfügung.
- **Price-controlled Best Effort:** Der per-packet-Preis wird in einem Signal übergeben. Er kann von der Stausituation im Netz abhängen.

Type	Flat fee	fixed price	variable price
overprovis. best effort	+	-	-
price-contr.-best effort	-	-	+
DiffServ	-	+	+
IntServ	-	+	+

Tabelle 4.1: Preisfaktoren für QoS

Die Preisfaktoren für diese QoS-angereicherten Dienste sind in Tabelle 4.1 aufgeführt.

4.5 Verfahren zur Aushandlung des Tarifs

Grundsätzlich ist zwischen statischen und dynamischen Tarifmodellen zu unterscheiden. Bei den statischen Modellen ist der Tarif zu Beginn der Sitzung vereinbart und ändert sich bis zum Ende nicht. Dagegen kann bei den dynamischen Modellen während einer Sitzung ein höherer Tarif gefordert werden, wenn eine Überlastung des Netzes eintritt. Der Nutzer hat dann die Möglichkeit, den neuen Vertrag zu akzeptieren oder schlechtere Qualität hinzunehmen.

4.5.1 Statische Verfahren

- **Paris Metro Pricing (PMP):**
  Nach Odlycko ([odly98], [odla98],[odly01]) soll es beim Internet zwei Preisklassen geben, eine für Basisdienste, eine für höhere Dienste, analog zum Zweiklassensystem bei der Pariser Metro. Beim Paris Metro Pricing gibt es zwei Klassen (1 und 2), die verschiedene Qualität und Preise haben: Das Publikum regelt den Zugang: wenn Klasse 2 voll ist, geht es in die Klasse 1, die mehr kostet. Wenn diese voll ist, gehen wieder mehr in Klasse 2.
  Entsprechend wählen Internetnutzer Flatrate, wenn sie nicht mehr Bandbreite und QoS brauchen, und gehen von dem Mehrwertdienst ab, wenn dort wegen Stau die gewünschte Qualität nicht mehr geboten wird. Im Modell PMP werden verschiedene Kanäle zur Verfügung gestellt, die unterschiedliche QoS bieten und entsprechend verschiedene Preise haben: Flatrate für niedere QoS. Diffserv ermittelt den QoS-Eintrag im IP-Paket.

- **Profilmodell:**
  Hierbei wird das gewünschte Service-Profil (QoS) oder (beim Expected Capacity Allocation Model) die erwartete Bandbreite (Fankenhauser [fsp97]) vom Nutzer im Paket angegeben. Keine Angabe bedeutet "best effort". Danach richtet sich der Preis und ggf. auch danach, ob bei Stau Pakete verworfen werden. Das Profil wird an den Netzgrenzen ermittelt (z.B. liest ein im Nutzerclient installierter Modul oder ein Zugangsadapter die Zugangsdaten ([hart99], ([kzs00], [fobi99])).

  Nach [rls99] kann man Nutzern Nutzerprofile zuteilen, die ihren Anforderungen bezüglich Dienst und QoS entsprechen und entsprechend kostenpflichtig sind. Die IP-Pakete erhalten entsprechende Tags ("In" oder "Out"), die in den Schaltstellen vorzugsweise an den Netzgrenzen durch Profilmeter überprüft werden. Bei Stau werden Pakete derjenigen Nutzer verworfen, die ihre Profile überziehen (auch [clar95]).

Auch die Dienste können klassifiziert werden. Nutzer der gleichen Klasse werden gleich behandelt. Bei Bedarf (Stau) werden Pakete aus niedrigen Serviceklassen verworfen ([rls99]).

- **Prioritätsmodell:**
  Die gewünschte Priorität des Dienstes wird im Header des IPv4-Pakets angegeben. Bisher sind 4 Prioritätsklassen möglich. Durch Ausnutzen reservierter Bits oder IPv6-Protokoll können weitere Prioritätsklassen vereinbart werden. Die Auswertung erfolgt in den Routern mittels Tags oder priorisierten Warteschlangen. Bei Stau werden niederpriore Pakete verworfen ([lsow99], [mava94], [gswp95], [gswt95], [haca99], [fsp97], [cazs99])

4.5.2 Dynamische Verfahren

- **Auktionen:**
  Der Nutzer meldet am Connection Admission Control Point, wieviel er für einen bestimmten Dienst mit definierter QoS zu zahlen bereit ist. Bei hoher Auslastung des Netzes (Staugefahr) gibt es möglicherweise Nutzer, die mehr bieten. Die Nutzer werden in der Reihenfolge höchster Bieter, nächsthöchster Bieter usw. zugelassen, soweit Kapazität frei ist. Sie müssen aber meist nicht den gebotenen Preis zahlen, sondern nur den "Market-clearing Preis". Das ist ein Großhandelspreis. Provider können ein Netzwerk mieten und Nutzer können Verträge über Nutzung verschiedener Dienstklassen mit ihnen abschließen, z.B.

  - Premium: sofort senden zu jedem Preis;
  - Economy: senden nur bei Preis unter bestimmten Limit.

  Das Netzwerk sollte fähig sein, die Gebühren für die Nutzer dynamisch der Belastung des Netzes anzupassen, sodass der gewünschte Nutzungsverhalten erreicht wird ([karo97]).
  Auctioning lässt sich mit einer Spieltheorie als stabil nachweisen, d.h. es läuft auf einen stabilen "Operation point" hinaus (keine Oscillationen)([slcl99]).

  Dieses (smart market-)Verfahren ([mava94], [slcl99], [kzs00], [karo97], [fobi99], [sark95], [rls99], [srl 00],[vill96]) ist recht komplex, da Verhandlungen mit allen Nutzern geführt werden müssen. Außerdem haben Provider durch Senden großer Datenmengen die Möglichkeit, den Preis hochzutreiben ([ckst99]). Das Verfahren ist daher nur von theoretischem Interesse. Nach Fankenhauser u.a. ist Smart market eine Belastung für die Router, die bei geänderten Prioritäten ihre Warteschlangen umsortieren müssen.

Bei Smart-market-pricing kann die Verbindung unterbrochen werden, wenn der Nutzer zuwenig anbietet und verliert. Hier ist ein Second-Chance System nützlich, die Verbindung bleibt erst noch erhalten, und dem Nutzer wird die erforderliche Preis mitgeteilt, den er dann bieten kann. Neue Nutzer haben diese Möglichkeit nicht. Der Nutzer muss seine Verbindung beobachten und auf Angebote schnell reagieren (CATI :[srl 00]).

Eine solche Verbesserung bietet die `Delta Auction` ([fbrs99], [fsvp98], [sfp98]). Dabei darf ein Bieter, der zuwenig geboten hat, nochmal sein Gebot erhöhen, wenn er schon eine Ressource belegt hat. Neubieter kommen nicht nochmals zum Zug. CHiPS (Connection-Holder is Preferred) von ([srl 00]) ist ein ähnliches Verfahren.
MacKie-Mason u.a.([mmm95]) schlagen vor, dass der Nutzer einen Defaultwert für den Preis angibt, den er zu zahlen bereit ist. Im Gateway wird dann über seine Pakete entschieden.

- **Dynamic (Volume) Pricing Modell:**
  Bei Mangel an Ressourcen, d.h. Gefahr von Stau (congestion) legt der Provider einen Preis fest, den Nutzer akzeptieren oder nicht ([fbrs99], [fsvp98], [sfp98], [haca99]). In dem Modell von Gupta u.a. ([gswp95]) wird der Preis für eine bestimmte, vom Nutzer gewünschte (dienstabhängige) Priorität laufend neu berechnet aufgrund der bei der Auslastung zu erwartenden Verzögerungen. Beim Login erfährt der Nutzer diesen Preis und kann akzeptieren und weitermachen oder aussteigen. Bei höherer Priorität hat er keine Verluste zu erwarten, da das TCP-Protokoll bei Stau nur verzögert.

  Etwas anders wird dieser Begriff von Songhurst ([song99]) verstanden: Er bezieht sich auf Dynamic Pricing for ABR (available bit rate bei ATM) Services: Grundlage ist ABR und keine strikten Anforderungen an Verzögerung und Verzögerungsvarianz. Anwender können Minimalrate angeben und bei verfügbarem Volumen auch mehr senden. Informationen über die Auslastung des Netzes erhalten sie über Kontrollnachrichten des Netzwerks. Jede Quelle sendet Forward-RM-Kontrollzellen (RM: Resource management). Das Ziel sendet Backward-RM-Zellen. Switches setzen gegebenenfalls ein Congestionbit (oder löschen es). Die Quelle reduziert darauhin ihren Datenstrom (bzw. erhöht ihn wieder). Es handelt sich offensichtlich um elastische Datenströme.

- **Feedbackmodell:**
  Dieses Modell wird auch "Responsive Pricing" genannt ([ckst99]). Hier wird vom System lastabhängig ein Preis ermittelt, der steigt, wenn ein Stau auftritt. Dies wird dem Nutzer mitgeteilt, der den Preis entweder akzeptiert oder Verlust seiner Daten in Kauf nimmt ([haca99]). Karsten u.a ([kssw00]) sehen Feedback mit proportional angemessenen Preisen für ein Multiservice-Kommunikationsnetz mit verschiedenen Dienstklassen vor.

4.6 Analyse von Tarifformeln

4.6.1 Tarifparameter

In der Regel wird zwischen ISP (Internet Service Provider) und Nutzer ein SLA (Service Level Agreement) geschlossen bzgl. Service und Service Level ([CoSi99]). Der Preis für einen Dienst ergibt sich als Summe über Produkte von vereinbarten Tarifparametern und gemessenen Abrechnungseinheiten. Dies muss auch beim Tariftool berücksichtigt werden.

Aus der Analyse ergeben sich folgende Tarifparameter beispielhaft als geeignet:

1. **Grundgebühr:**
   Meist wird für die Möglichkeit, den Netzservice zu nutzen, eine Grundgebühr erhoben.

2. **Sitzungszugang:**
   Vor Zugang zur Anwendung einer Netzapplikation muss ein `login` gemacht werden, am Ende ein `logout`. Was dazwischen liegt, wird als Sitzung (Session) bezeichnet. Für diesen Zugang wird im Allgemeinen eine Gebühr erhoben Songhurst [song99]).

3. **Value-added Services:**
   Wenn der Dienst von einem Provider des Providers (3^{rd} party) kommt, wird meist ein fester Betrag abgerechnet ([pina01]).

4. **Dauer der Netznutzung:**
   Analog zu POT-Diensten wird insbesondere bei Nutzung geringer Bandbreiten und geringer Anforderung an Qualität ein von der Nutzungsdauer abhängiger Preis verlangt. Bei Diensten, die höhere Bandbreite benötigen, ist der Tarifparameter (Abrechnungstyp Dauer) von der Bandbreite abhängig ([alva99]).

5. **Bandbreite:**
   Wenn hohe Bandbreiten gebraucht werden - oft auch als Volumen bezeichnet - , dann wird ein von der Anzahl der übertragenen Bytes abhängiger Preis verlangt ([alva99], [brow95]).
   Reserviertes Volumen wird als solches abgerechnet, genutztes Volumen dagegen nach der Anzahl von priorisierten Paketen (priority based packet pricing). (Priorisierung ist bei ATM-basierten Netzen über eine Angabe im Header einer Nachricht möglich.) Statisches und dynamisches Pricing ist anwendbar; bei dynamischem Pricing steigt der Preis mit den Anforderungen und dem Stau. Die Abrechnung nach Volumen ist nach [gswp95], [mava94], [gswt95] am kompliziertesten.

6. **Transaktion:**
   Bei einer Transaktion hängt der Preis von der Charakteristik (QoS, Volumen, mean rate, peak rate) ab ([mcba95]).

7. **QoS-Parameter:**
   Bei QoS-Parametern geht es um geringe Restfehlerrate, Verfügbarkeit. Priorität und Robustheit gegenüber Bursts. Letzteres wird durch "Leaky Buckets" erreicht. Einige einfache Dienste kommen mit "best Effort" aus, komplexe brauchen bestimmte QoS. Wie Qualität sich auf den Preis auswirken kann, wird im Quasimodo-Projekt untersucht ([smir00]). QoS-Charging im Internet hat noch keinen Standard, aber es gibt verschiedene Näherungen (Ansätze). Pricing schemes werden durch Charging policies spezifiziert. Accounting ist möglich bei IP VPN, die über ATM oder Frame Relay realisiert sind. Das Quasimodo Projekt wurde gegründet, um QoS-charging als Dienst im QoS-bewussten (aware) Internet zu betrachten.
   Bei [alva99] kann ein QoS-Level durch "priority pricing" ausgewählt werden.

   Meist geht es um die für den Service benötigte Qualität. Aufgrund des dem Internet zugrundeliegenden IP-Protokolls gibt es nur verbindungslose Kommunikation. Mehrere Datenübertragungen werden statistisch gemischt transportiert. Eine Garantie strikt einzuhalten ist unmöglich. Nur die Wahrscheinlichkeit dafür zu erhöhen ist machbar: Dies geschieht durch Reservierung von mehr Bandbreite oder Markierung von IP-Paketen als priorisiert bzw. tagged. Nicht priorisierte oder out-tagged Pakete werden bei Stau verworfen. Abrechnen lässt sich zur Zeit nur die Peakbandwidth als Zusatzparameter beim Volumen-Tarifparameter.

8. **Tageszeit:**
   Diese wird nur selten erwähnt, Carle u.a. ([chsz99]) verwenden eine Businesszeit und eine Nachtzeit zur Preisfindung. Praktische Anwendung findet die Anwendung nach Brownlee ([brow95]) in Neuseeland. Meist werden zwei tageszeitabhängige Tarife berücksichtigt.

9. **Sonstige:**
   Weitere mögliche Parameter für IP-basierte Dienste sind in Tabelle 4.2 nach [pina01] aufgelistet.

Hauptparameter von Tarifformeln
Bei den in der Literatur aufgeführten Tarifformeln sind drei Tarifparameter maßgebend, die wesentliche Ressourcen repräsentieren, nämlich Dauer, Volumen und QoS-Parameter. Die Tarifparameter können konstant oder Funktionen der Ressource oder anderer Parameter sein. Diese Tarifparameter gehen in die Formeln der folgenden Abschnitte ein.

4.6.2 ABC-Formeln

- Nach [haca99], [nia397], [nia297] setzt sich die Tarifformel zur Berechnung des Gesamtpreises aus Tarifparametern und Ressourceeinheiten zusammen

$$P = C + \sum_i p_i(R_i) * R_i$$

service	flat.	packet	disc	cpu	sess.	serv.	content	transaction	skill
websv.	x								
email	x								
telefon					x	x			
chat	x								
news	x								
Pers.Web			x	x					
Cont.hotel			x				x	x	
Dir.Serv.	x				x		x	x	
Game					x		x	x	x
Finance	x					x	x	x	x
Info	x		x	x	x	x	x	x	
Fax		x			x	x		x	
Audio					x	x	x		
Video					x	x	x		
Conferenc.					x	x	x		
training				x	x	x	x	x	
Website Host.	x		x	x		x		x	
VPN	x				x	x			
Multimed Tel.	x		x	x	x	x	x	x	
unif. messag.	x		x	x	x	x	x	x	

Tabelle 4.2: IP-basierte Dienste nach [pina01]

mit C: fester Preis für Einchecken, R_i: Anzahl Ressource-Abrechnungseinheiten - im Wesentlichen Dauer und Volumen -, p_i: Tarifparameter, P: Kosten (Gebühr).

- Dieses Tarifschema entspricht dem ABC-Schema von [song99] u.a. (s.o.).
  Die allgemeine Formel lautet dort

$$P = a * T + b * V + c$$

c sind Kosten für das Setup der Verbindung; a und b sind von PCR (peak cell rate) gewählter mittlerer Bandbreite abhängig, T und V sind gemessene Dauer bzw. Volumen (Mbits). Hierbei gibt es nur zwei Messungen.

Die Nutzer sollen beim Aufbau der Verbindung ihre maximale (PCR) und möglichst genau die benötigte mittlere Bandbreite (mean rate) angeben. Der Netzprovider muss die erforderlichen Ressourcen reservieren. Der Tarif wird so angesetzt, dass Nutzer sowohl bei Unterschreitung der belegten Bandbreite als auch bei Überschreitung mehr zahlen, als für die genutzte Bandbreite nötig wäre. Dies wird dadurch begründet, dass sie einmal mehr belegen als sie brauchen und anderen Nutzer keinen Zugang gewähren und zum anderen, dass sie Staus verursachen können, wenn sie die vereinbarte mittlere

Bandbreite öfters überschreiten ([nia197]).

Der Nutzer kann wählen zwischen niedrigen Kosten für die Übertragungszeit (Faktor a) bei höheren Kosten für die Bandbreite (Faktor b) und höheren Kosten für die Übertragungszeit bei niedrigeren Kosten für die Bandbreite. Diese Faktoren werden aus einer Funktion der Effektiven Bandbreite von der Mittleren Bandbreite ermittelt (s. Abbildung 4.3).

Die effektive Bandbreite wird mittels stochastischer Berechnungen aus einem Modell ermittelt, das statistisches Multiplexing von vielen Datenübertragungen voraussetzt sowie Ausgleichen von Belastungsspitzen (peak rates) durch leaky buckets. Sie ist die Bandbreite, die von einem Nutzer gerade genutzt wird; dabei liegt sie zwischen der mittleren Bandbreite und der maximalen Bandbreite (peak rate) und umso näher an der maximalen Bandbreite, je näher diese an der verfügbaren Bandbreite liegt ([kell95]). Zur Ermittlung der Parameter a und b siehe Anhang A.

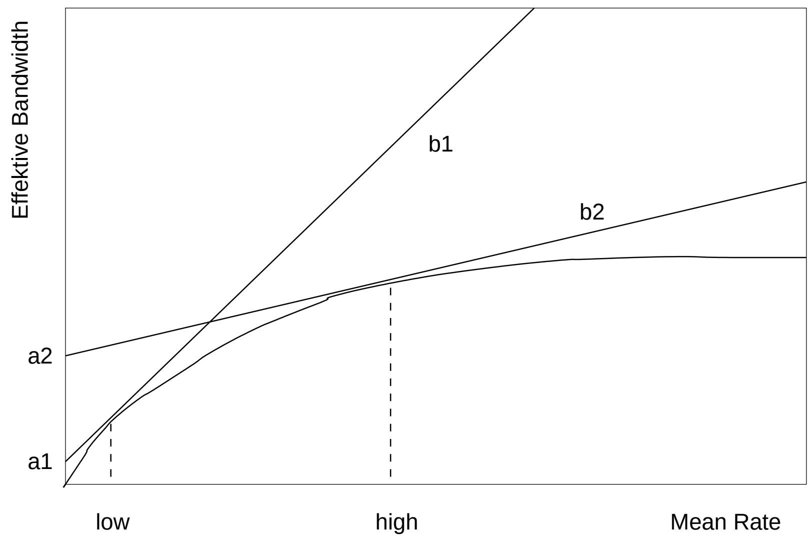

Abbildung 4.3: Tarife für garantierte Dienste

- Nach [hart99] gilt für den Tarif das Schema:

$$P = f(f_d(B) * D, f_v(p) * V)$$

mit B Bandbreite, D Dauer, p Dienstprofil (entspricht QoS) und V Datenvolumen.

- Nach [alva99] gilt für die Gebühr c:

$$c = \sum_j t_j * p'_j(b_j) + v_j p''_j(b_j)$$

mit $\sum_j$: Summe über alle QoS-Level, t_j: Dauer, b: Spitzenbandbreite, p': Preisfaktor für die Dauer, p": Preisfaktor für Bandbreite b_j, v: die genutzte Bandbreite.

4.6.3 Taxband-Formel

Wenn die Zahl der Messungen verdoppelt wird, kann der angemessene Preis genauer ermittelt werden. Dies geschieht in der Taxband-Formel. Man betrachtet Intervalle bezüglich Volumensnutzung (daher Name: entsprechendes Verfahren bei Steuer: Taxes). Stochastische Rechnungen führen zu einer Näherung der EB als exponentielle Funktion eines Volumenschwellwerts h (s. Abbildung 4.4 nach [song99]).

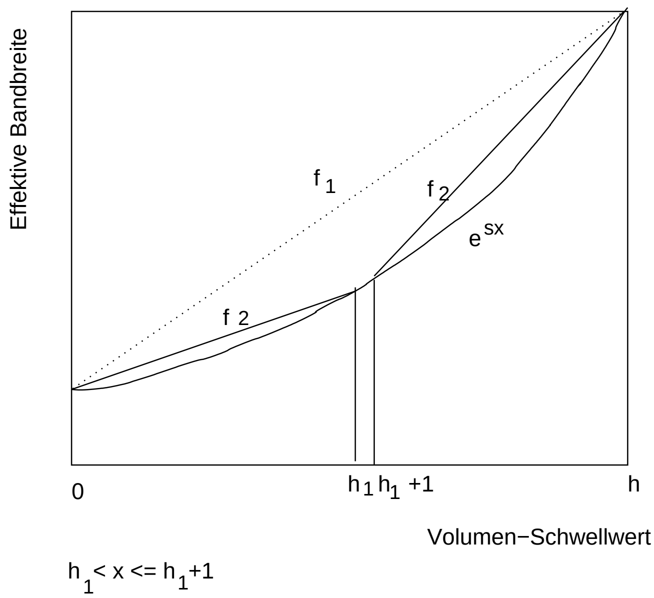

Abbildung 4.4: Taxband-Formel

Diese monoton wachsende konvexe Funktion hat obere Schranken durch die lineare Funktion f1, die der Tangente in Abbildung 4.3 entspricht und der aus 2 Strecken zusammengesetzten Funktion f2, die im Taxband Charging Scheme als Näherung für die Charging Funktion verwendet wird.

$$charge = a_1 T_1 + a_2 T_2 + b_1 V_1 + b_2 V_2$$

Diese Abrechnung ist fairer als die nach einfacher Linearität, die Kosten für die Erstellung der Abrechnung ist aber entsprechend höher. Eine Quelle, die weniger gleichmäßig sendet (bursty), muss mehr zahlen (gewünschter Effekt).

4.6.4 Tarife für garantierte Dienste

Nach Carle und Zseby ([cazs99]) gibt es für Dienste, die über mehrere Router im Internet laufen, folgende Tarifformel:

$$ch = L \cdot \sum_{i=1}^{h} b * P(B) + c * P(C) + d * P(D) + K \cdot h \cdot V$$

mit ch: Charge, L: Dauer, h: Anzahl Hops (Router), a, b, c Anteile der Ressourcen in den Routern, B: Bandbreite, C: Rechnerkapazität, D: Verzögerung (Delay Ressource) P: Preisfaktoren für reservierte Ressourcen, K: Preisfaktor für genutzte Ressourcen, V: Volumen.

Für garantierte Dienste gilt die Tarifformel: ch = a*r + b*(R-r) mit r,R IntServ-rate-Faktoren.

Für kontrollierte Last gilt: ch = a*r + c*e (e: statistisch verfügbare Rate).

Für garantierte Raten gilt: ch = c*r.

Dabei sind a, b, c Preisfaktoren.

Ein Tarif kann dargestellt werden durch eine Menge von Formeln und Regeln, die angeben, unter welchen Bedingungen die Formeln gültig sind. Zur Repräsentation von Tarifen haben die Autoren die Tariff Formula Language TFL entwickelt. Mit ihr können auch komplexe Tarife dargestellt werden. Einfaches Beispiel für garantierten Dienst:

```
# parameter a
a = 0.5
# parameter b
b = 0.2
# tariff formula
p = a*tr + b * (sr-tr)
```

tr und sr sind dabei die reservierte token bucket rate und die service rate.
Ein weiteres Beispiel für tageszeitabhängigen Tarif:

```
# setup charge
sc = 0.5
# volume units in bytes
vu = 800000
# price per volume unit
pv = 0.5
# time units in secs
tu = 100
# price per time unit
pt = IF(AND(td>TIME("00:00:00"),
```

```
td < TIME("05:00:00")), 0.5,
IF(AND(td>=TIME("05:00:00"),
td < TIME("21:00:00"), 0.8, 0.5))
# tariff formula
p = sc + (v/vu)*pv + (t/tu) * pt
```

4.6.5 Generische Tarifformel

Die folgende generische Tarifformel wird von Radisic ([rada00]) vorgestellt:

$$P = a_1(x)A_1 + a_1(x)A_1 + ... + a_1(x)A_1 + c(x)$$

mit P: Gesamtpreis, c(x) pauschale Gebühr für die Bereitstellung von Diensten,
die Summanden $a_i(x)A_i$ sind nutzungsbezogen:
A_i: Anzahl Abrechnungseinheiten für eine gemessene Dienstnutzung (z.B in Bytes bzw. ms),
$a_i(x)$: Kostenfunktion für eine Abrechnungseinheit A_i,
x: Einflussgrößen für die Kostenfunktion (z.B. Dienstgüteklasse, Tageszeit,...).

Für das Beispiel eines IP-Connectivity-Dienstes ergibt sich:

$$P(IPC) = a_1(x) \cdot B_s + a_2(x) \cdot B_e + a_3(x) \cdot P_s + a_4(x) \cdot P_e + a_5(x) \cdot D + c(x)$$

mit B_s: Anzahl gesendete Bytes, B_e: Anzahl empfangene Bytes, P_s: Anzahl gesendete Pakete, P_e: Anzahl empfangene Pakete, D: Dauer der Übertragung.

x kann als Bitmuster definiert werden, sein Wert kann als Tarifklasse aufgefasst werden, z.B. können 2 Bits 4 Tageszeiten anzeigen. Die Kostenfunktionen können in Wertetabellen implementiert werden.

4.6.6 Eignung der Tarifformeln für das Tariftool

Abgesehen vom Tarif für garantierte Dienste erfüllen alle Tarifformeln die Anforderung 5 des Anforderungskatalogs (Abschnitt 2.2) nach Unterstützung unterschiedlicher Dienste. Diese müssen nur entsprechende Tarifparameter haben. Die ABC-Formel und die Taxbandformel kommen für das Tariftool nicht in Frage, da Verträge für bestimmte Zeitintervalle abgeschlossen werden sollen und keine Tarifänderung während der Nutzung vorgesehen ist (Anforderung 2). Die generische Tarifformel kann als Grundlage einer Tarifformel für das Tariftool dienen, denn sie erfüllt Anforderung 2, 4 und 5 und widerspricht keiner Anforderung.

4.7 Architekturen von Abrechnungssystemen

Im Folgenden werden Architekturen von Abrechnungssystemen untersucht, um mögliche Strukturen für das Tariftool zu ermitteln. Die in diesem Abschnitt beschriebenen Architekturen stellen QoS-angereicherte Dienste mit entsprechender Abrechnung zur Verfügung. Sie unterscheiden sich in der Architekturbasis, den angewandten Methoden und den angebotenen Service Levels.

4.7.1 TINA-basierte Architekturen

Billing Service für das TINA-Business-Modell
Basis ist das TINA-Business-Modell mit verteiltem Accounting ([camh00]). Es gibt 5 Basisrollen: User/Customer, Retailer, Broker, Connectivity Provider und 3^{rd} Party-Provider. Das Modell ist in Abbildung 4.5 dargestellt.

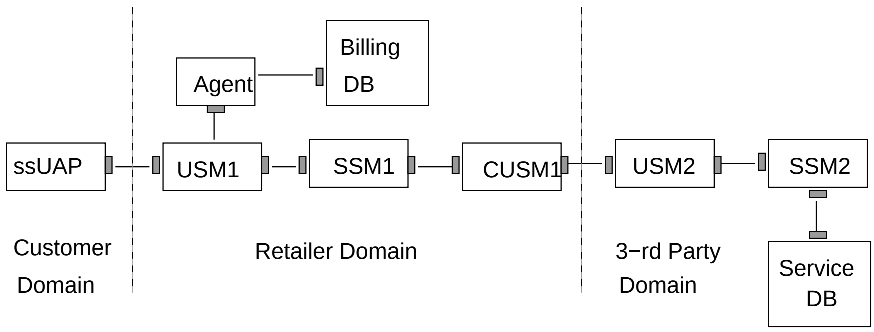

Abbildung 4.5: Charging Modell für Produktkauf (nach [camh00])

Beim Kauf von Produkten oder Diensten wird der Preis des Dienstes in die Billing-DB der Domäne eingetragen. Der Kunde baut eine trusted Relationship zu einem Billing Service Provider (BSP) auf. Die Accounting-Information (Dauer, Ressourcenmenge) wird von Retails gesammelt (Abbildung 4.6). Die Informationen werden als CORBA-Objekte durch XML

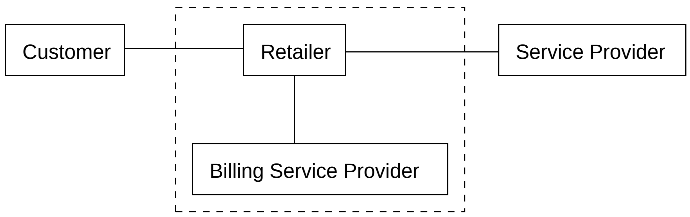

Abbildung 4.6: Billing Service Provider (nach [camh00])

zwischen den Objekten übertragen.

Ca$hman: (Charging and Accounting Schemes in Multiservice ATM Networks)
Die Dienstbereitstellung und die Gebührenermittlung arbeiten nach Songhurst ([song99]) in-

teraktiv. Nachdem der Nutzer authentisiert und autorisiert ist, kann er einen Service wählen und einen QoS bzw. einen Tarif aushandeln. Für die Abrechnung gibt es eine integrierte Infrastruktur. Das Management ist ein End-to-End-Management mit Fokus auf Dienstanforderungen. Die Messung der Nutzung (Anzahl ATM-Zellen) erfolgt Ressource-verknüpft, z.B. im Switch. Die verwendete TINA-Architektur enthält:

- Ressource Management Schicht (NM & NEM aus TMN)
- Service Management Schicht
- Distributed Processing Environment
- Network-Ressourcen gruppiert in Subnetz

ACTS-SUSIE-Projekt
Das ACTS-SUSIE-Projekt beschäftigt sich mit QoS-angereicherten IP-Diensten (Premium IP-Diensten) über ATM. Ein Referenzmodell für einen Abrechnungsdienst wird definiert, das den Anforderungen von IntServ und DiffServ gerecht wird, da sowohl reservierte als auch genutzte Ressourcen berücksichtigt werden. Es wird als VIPCAS (Value Added IP Charging and Accounting Service) implementiert. Die nutzungsbasierte Abrechnung berücksichtigt reservierte und genutzte Ressourcen.

Beim Dienstzugang wird vom Dienstmanager das Nutzerprofil geprüft, das auf dem Dienstvertrag beruht. Das Dynamic Host Configuration Protocol (DHCP) und GEM-PTP (Generic Extensible and Modular Policy Transfer Protocol) werden für die Implementierung verwendet. Die von den NetTraMet-Metern gemessene Accounting-Information wird in einem PIP-NAR-Datenstruktur (Premium IP Network Accounting Record) abgelegt:

Measurement point identification
Record Description
Flow Description
Reserved Resources
Used Resources
Data Extension

Tabelle 4.3: Datenstruktur PIP-NAR

Die Datenstruktur PIP-NAR wird an die Charging-Schicht weitergeleitet, die mit dem TINA-Accounting-System verbunden ist. In den used Resources sind Datenvolumen und Anzahl der übertragenen Pakete enthalten. Mit PIP-NAR können auch Nutzungsinformationen zwischen Providern übertragen werden.

Die Tarif-Policy wird mittels TFL (tariff formula language, S. 39) dargestellt. Das Charging Information Protocol (CIP) verteilt die PIP-NARs. Grundlagen der Architektur sind TINA und CORBA. Die Messungen werden an den Edge-Routern nach dem IETF Real Time Flow Measurement (RTFM) durchgeführt. Die Konfiguration erfolgt mittels SNMP und einer speziellen MIB. Business- und Nachstunden werden unterschiedlich bewertet. Bei garantierten Diensten sind die IntServ-Parameter *bucket rate* und *service rate* einzugeben. Die Kosten werden unter die Teilnehmer des Multicast aufgeteilt ([haca99], [chsz99]).

4.7.2 Internet-basierte Architekturen

Da im Internet verbindungslos übertragen wird, muss hier die Einhaltung vereinbarter QoS gesichert werden. Dies wird über spezielle Protokolle erreicht.

CATI: (Charging and Accounting Technologies for the Internet)
VPNs in CATI bestehen aus logischen und sicheren Tunneln durch das Internet. Die ISPs (Internet Service Provider) müssen QoS anbieten (über ATM, RSVP und DiffServ). Die Architektur von CATI ist in Abbildung 4.7 nach ([fobi99]) dargestellt:

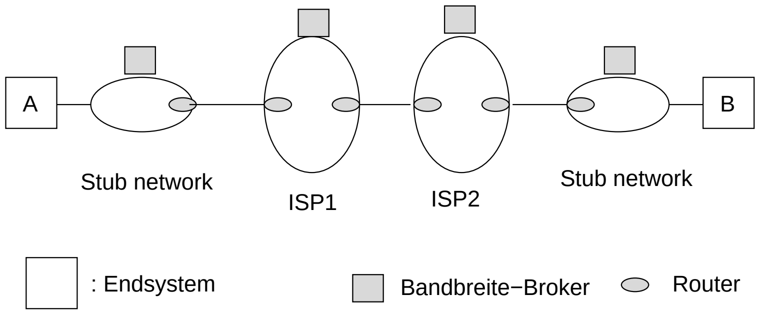

Abbildung 4.7: Allgemeine CATI-Architektur (nach [fobi99])

Der Bandwidth Broker kontrolliert den DiffServ-Verkehr. RSVP wird um Preisinformationen erweitert. Es gibt zwei DiffServ-Klassen:

- Assured Service Soft Guaranty: Pakete mit hohem Priority Tagging werden mit hoher Wahrscheinlichkeit übertragen,
- Premium Service: für zeitkritische Anwendungen, die Garantien für explizite Bandbreite und minimale Verzögerung verlangen.

QoS-angereicherte Dienste
Hartanto ([hart99]) verwendet für das Management und die Abrechnung von höheren qualitätsangereicherten Diensten im Internet das Dynamic Host Configuration Protocol (DHCP). Für die Implementierung wird GEM-PTP (Generic Extensible and Modular Policy Transfer Protocol) entwickelt. U.a. werden die in Abbildung 4.8 dargestellten Objekte implementiert.

Beim Dienstzugang wird vom Dienstmanager das Nutzerprofil geprüft, das auf den Dienstvertrag verweist.

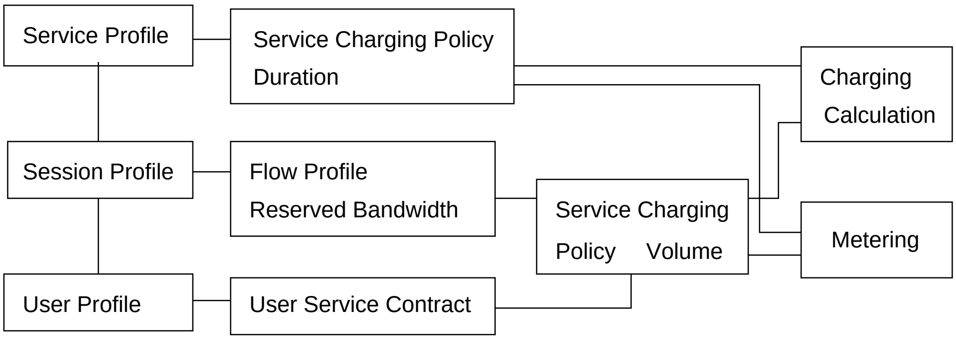

Abbildung 4.8: Implementierung von Accounting (nach [hart99])

Abrechnungsarchitektur ARROW
Fankhauser u.a. ([fsp97]) schlagen ein Konzept vor, das mit dem Protokoll IPv6 Dienstklassen mit Reservierung ermöglicht und mit RSVP allen Datagrammen ein Flow-label erteilt. Letzteres vereinfacht die Abrechnung; bei best-effort (ohne Label) müssen nur Pakete gezählt werden. Charging und Accounting wird in Routern oder Switches durchgeführt.

Ein Anwendungsprogramm vermittelt Teile dieser Informationen an Nutzer (Feedback), Server (für Reverse Charging) und Router (Kontrolle). Das Netwerkmodell erlaubt deterministische (hard) und statistische (soft) Garantien. Das Metering beruht auf Überwachen von Paketen und Datagrammen. Die Pakete werden gefiltert auf Prioritybits, Preisangaben und Authenticy-Angaben.
Bei garantierten Diensten reicht es aus, in den Accesspunkten (beim ISP) zu messen, da der Pfad durch RSVP festgelegt ist. Es gibt 4 Dienstklassen:

- best-effort mit niederer und hoher Priorität (LP, HP) und
- statistisch oder deterministisch garantiert (SG, DG).

Ein dynamisches Pricing wird vorgeschlagen:

price = trafficfactor * baseprice * Throughput * connectiontime
mit trafficfactor = $(1+x)^2$ bei Congestion, mit $x > 0$, und = 1 sonst.

Priority Pricing, dynamisches Pricing und Expected Capacity Allocation Model konnten einfach implementiert werden. Das Smart Market Modell benötigt einen größeren Overhead. Am besten schneiden die garantierten Dienste (mit Reservierung) ab.

Ein Kontrollprogramm, das auf dem Reservation-Protokoll RSVP basiert, wurde von Fankhauser, Stiller u.a. ([fsvp98], [sfp98], [sfpw98]) entwickelt. Die Pricing-Modelle Delta Auction und dynamisches Pricing ([fsp97]) wurden getestet. Die Payment-Information wurde in das RSVP eingefügt. Mit IPv4 und UDP ergab sich ohne Billing (nur Accounting und Charging) ein Overhead von 27 - 30 %. Flow-based Charging (Reservierung) war am günstigsten.

4.7.3 Eignung der Abrechnungsstrukturen für das Tariftool

Für den Prototyp Tariftool sind die vorgestellten Architekturen unnötig komplex. Bei einer Erweiterung des Tariftools auf verteilte Systeme sollte auf die oben angeführten Strukturideen zurückgegriffen werden.

4.8 Erfahrungen mit pauschaler und nutzungsorientierter Abrechnung

Erfahrungen mit pauschaler und nutzungsorientierter Abrechnung wurden untersucht, um festzustellen, was für das Tariftool geeignet sein könnte.

4.8.1 INDEX: (INternet Demand EXperiment)

- **Empirische Daten: Nutzerverhalten**
  Mit dem Projekt INDEX sollte die Nachfrage nach Internet-Zugang in Abhängigkeit von den erhobenen Gebühren empirisch ermittelt werden. Als Szenario dient ein zukünftiger Telecommunikationsmarkt mit Internet Service Providern (ISPs). Faktoren sind die nutzbare Bandbreite als Maß für QoS und die Preisgestaltung. Es gibt:

  - per Minute-Preise für Bandbreiten über 8 kbps
  - per Byte-Preise für Übertragungen über 8 kbps
  - Mischung von Dauer- und Volumen-Preisen
  - Flatrate 8 kbps, mit Option für zeitweise höhere Bandbreiten

  Nach Altmann u.a. ([alch01], [alva99]) lag die Nutzung bei Flatrate erheblich höher als bei den anderen Preissystemen. Die empirische Daten aus Proj. INDEX zeigen, dass Nutzer den Preisplan vorziehen, der Flatrate für einen Basisdienst und nutzungsbasierte Preise für gelegentliche Dienste mit höherem QoS vorsieht. Die höhere QoS wird dann auch voll ausgenutzt (Preference: need better service oder want to save money.). Das ist auch für den Provider günstig, der Dienste mit höherer Qualität on Demand anbieten will und erwartet, dass die Spitzenlast so reduziert wird.

- **Empirische Daten aus Providersicht**
  Nach Odlycko ([odly01]) ermutigt Flatrate Nutzer zu Verschwendung. Provider wünschen, dass Nutzer ihre Dienste möglichst täglich und oft nutzen, das sicher ihnen einen Marktanteil, sie wollen aber davon auch profitieren. Ein Problem ist auch die Konkurrenz zwischen Providern: wenn ein Provider Flatrate einführt, müssen die anderen nachziehen, denn die Nutzer bevorzugen Flatrate, auch dann, wenn dauerberechneter Dienst für sie günstiger wäre. *Sie scheuen die tickende Uhr* stellt [odly01] bei Auswertung des INDEX-Projekts fest.

  Die Umstellung von Flatrate auf nutzungsorientierte Preise bewirkt dramatische Effekte: Bei Erhebung von 1 cent/Mbyte statt vorher 0 sank die Nutzung innerhalb eines Jahres auf 1/3. Betreiber wollen differenzierte Dienste einführen und dafür differenzierte Gebühren verlangen. Die Nutzer wünschen eine "Grundversorgung" Flatrate und nur auf Verlangen mehr QoS mit z.B. Blockpreisen. Für den Nutzer sind einfache, vorhersehbare und risikofreie Abrechnungen wichtig.

4.8.2 Erfahrungen in Neuseeland

Nach Brownlee ([brow95]) wird in Neuseeland seit 1990 nach Volumen abgerechnet; Universitäten teilen sich die Kosten. Der Charging-Overhead ist bedeutend (signifikant). Die Kosten pro Mbyte liegen zwischen 4 und 2.30 $ bei Volumina zwischen 100 und 7000 Mbytes/Monat. Bei niederpriorem FTP und Mail gibt es einen Discount von 30 %. Zwischen 20:00 und 9:00 Uhr gibt es einen 80 %-igen Discount.

Nach OGC ([OGC 01], Kap 5) zeigen Fallstudien, dass Kunden z.B. auf Service Desk verzichten, wenn sie meinen, die Kosten seien nicht gerechtfertigt, obwohl es nützlich für alle wäre, den Service zu nutzen.

4.8.3 Erfahrungen mit Internet-Charging

Nach Reichl ([rls99]) wird üblicherweise eine monatliche Gebühr erhoben und eine Gebühr für die Verbindung pro Stunde. Von Volumencharging ist man abgekommen, außer bei Standverbindungen (fixed connections), bei denen pro Megabyte abgerechnet wird (meist mit Discount). Eine weitere Ausnahme ist die Flaschenhalsverbindung zwischen den Universitäten von Großbritannien und Neuseeland. Es wird auch nach "bursty rate" abgerechnet: Dabei misst der ISP etwa jede Stunde das Volumen; die 5 höchsten Prozent werden weggelassen, der höchste verbleibende Wert wird zur Bestimmung der Bandbreite genommen, die der Gebühr zugrunde liegt.

4.8.4 Erfahrungen mit SWITCH

SWITCH ist der Internet Provider für Forschung und höhere Bildung der Schweiz, gegründet als non-profit-Unternehmen. Jedes Jahr wird festgelegt, wie die Gebühren zu berechnen sind. Diese werden nur Kunden, nicht den Bildungsanstalten berechnet. Es gibt:

- Access charge (Kapazität)
- Charge pro Gigabyte Datenübertragung

Vor 2000 entfiel 1/3 der Kosten auf Access, 2/3 auf Übertragung. Im Jahr 2000 soll das umgekehrt sein. Es wird mehr nach außen als nach innen übertragen. Staus treten beim eingehenden Verkehr nicht auf. Man hatte USENET News von push-Mode auf pull-Mode umgestellt: das erwies sich als schlecht, denn die Belastung der Computer nahm zu. Innerhalb des Schweizer Forschungsnetzes mit nationalem Backbone werden keine Gebühren erhoben. Zwischen den Forschungsanstalten gibt es Peer Agreements: d.h. Kosten werden gegeneinander aufgerechnet (Nach Reichl, Stiller u.a. [rls99], [srl 00]).

4.9 Zusammenfassung

In diesem Kapitel wurde der Tarifierungsvorgang untersucht. Es zeigte sich, dass Stausituationen und vereinbarte QoS die Kosten stark beeinflussen können. Ein Tarif kann statisch oder dynamisch (on-line) vereinbart werden.

Ferner wurde untersucht, welche nutzungsunabhängigen und welche nutzungsorientierten Tarifparameter in eine allgemeine Tarifformel eingehen sollten. Großkunden und Provider sind daran interessiert, dass nutzungsgerecht abgerechnet wird; in die Tarifformel für das Tariftool gehen also auch nutzungsorientierte Elemente ein. Architekturen von Abrechnungssystemen und Erfahrungen mit solchen wurden beschrieben.

Da der Kunde und der Provider einen Vertrag über eine bestimmte Zeit abschließen, gilt für das Tariftool das statische Verfahren zur Aushandlung des Tarifs: Nutzer können während einer Sitzung keine SLAs abschließen.

Nach den obigen Ausführungen ist es sinnvoll, neben Dauer und Volumen der Datenübertragung auch QoS-Parameter und eventuell auch die Tageszeit (tags, nachts) zu berücksichtigen. Daneben ist eine feste, z.B. monatliche, Gebühr für die Bereitstellung der Netzdienste angebracht. Einzelheiten werden im folgenden Kapitel untersucht.

Kapitel 5

Systemanalyse

In der Systemanalyse wird ermittelt, **was** *das System leisten soll, d.h. die Funktionen des Tariftools werden spezifiziert. An verschiedenen Szenarien wird das Verhalten des Tariftools aufgezeigt. Eine Anforderungsanalyse ergibt Anwendungsfall-, Interaktions- und Aktivitätsdiagramme nach UML. Eine Tarifformel und relevante Tarifparameter werden danach vorgestellt. Danach wird das MNM-Dienstmodell mit dem Dienstmodell des Tariftools verglichen, das die wichtigsten Entitäten und ihre Abhängigkeiten voneinander enthält. Die Struktur des WBS für das Tariftool wird festgelegt. Im statischen Modell werden Klassendiagramme entwickelt, im dynamischen Modell wird das Verhalten des Inferenzprogramms analysiert.*

5.1 Einleitung

Zunächst werden die Szenarien von Vertragsverhandlungen beschrieben, die vom Tariftool unterstützt werden sollen. Danach wird eine Anforderungsanalyse durchgeführt, bei der die aktiven Komponenten des Systems ermittelt werden. Damit Verträge bezüglich ihres Tarifs verglichen werden können, wird eine allgemeine Tarifformel aufgestellt,. Zur Klärung von Rollen und Entitäten im Tariftool wird ein Dienstmodell aufgestellt. Die Verwendung einer Datenbank für das Tariftool wird erläutert. Danach wird ein statisches und ein dynamisches Modell für das Tariftool als Grundlage für den Systementwurf entwickelt.

5.2 Szenarien des Tariftools

In Kapitel 2 wurde das Szenario einer Vertragsverhandlung zwischen Großkunde und Provider global erläutert. Im Folgenden wird analysiert, wie Verhandlungen mit Hilfe des Tariftools im Einzelnen ablaufen.

Für die Reaktion des Tariftools auf eine Suchanforderung gibt es folgende Szenarien:

Hauptszenario:

```
Der Provider gibt eine Nutzer-Dienstcharakteristik ein
Tariftool findet mindestens einen passenden Vertrag
Tariftool ermittelt besten Vertrag
Tariftool gibt besten Vertrag mit Bewertung aus
```

```
Bester Vertrag bringt keinen Verlust
```

Der Kunde akzeptiert nun die Tarife des Vertrags oder er fordert Änderungen. Wenn der Kunde für den Provider interessant erscheint, wird er sich vielleicht darauf einlassen, sofern die Bewertung gut genug ist. Wenn die Bewertung den Altvertrag aber als nicht gut ausweist, kann der Provider versuchen, einen ähnlichen Vertrag mit etwas höheren Tarifen abzuschließen (s. auch Abbildung 5.3).

Nebenszenario

```
Der Provider gibt eine Nutzer-Dienstcharakteristik (n Dienste) ein
Tariftool findet keinen passenden Vertrag
Tariftool sucht Vertrag mit einem Dienst weniger (n-1 Dienste)
Tariftool ermittelt besten Vertrag
Tariftool gibt besten Vertrag mit Bewertung aus
Bester Vertrag bringt keinen Verlust

Provider schlägt Vertrag mit Ergänzung für fehlenden Diensttarif
vor
Kunde akzeptiert Tarife des Vertrags
Provider schließt Vertrag mit Kunde
```

Das Suchen nach Teilübereinstimmungen kann natürlich fortgeführt werden, bis nur noch ein Diensttarif zu vergleichen ist. Zu überlegen ist, ob eine Prioritätsreihenfolge berücksichtigt werden sollte und wie gegebenfalls die Prioritäten gesetzt werden sollten. Im folgenden Abschnitt werden die Szenarien durch UML-Diagramme veranschaulicht.

5.3 UML-Diagramme für die Tarifverhandlung

In der Anforderungsanalyse ist herauszuarbeiten, was das System (hier Tariftool) zu leisten hat. Die Hauptfunktion des Tariftools ist:

- einen Vertrag mit bestimmten Nutzer-Dienstcharakteristiken und/oder bestimmter Kundencharakteristik sowie bester Bewertung zu finden.

Nutzer-Dienstcharakteristik bezeichnet die Menge Dienste, die ein User nutzt. Kundencharakteristik bezeichnet bestimmte Eigenschaften eines Kunden, die für den Provider und einen zwischen Kunde und Provider abgeschlossenen Vertrag wichtig sind (Näheres s. Abschnitt 5.7).

Im Folgenden bezeichnet `Nutzerprofil` eine geeignete Codierung für die Nutzer-Dienstcharakteristik und `Kundenprofil` eine geeignete Codierung für die Kundencharakteristik. Aus dem Szenario lassen sich zwei Hauptanwendungsfälle (use cases) ableiten:

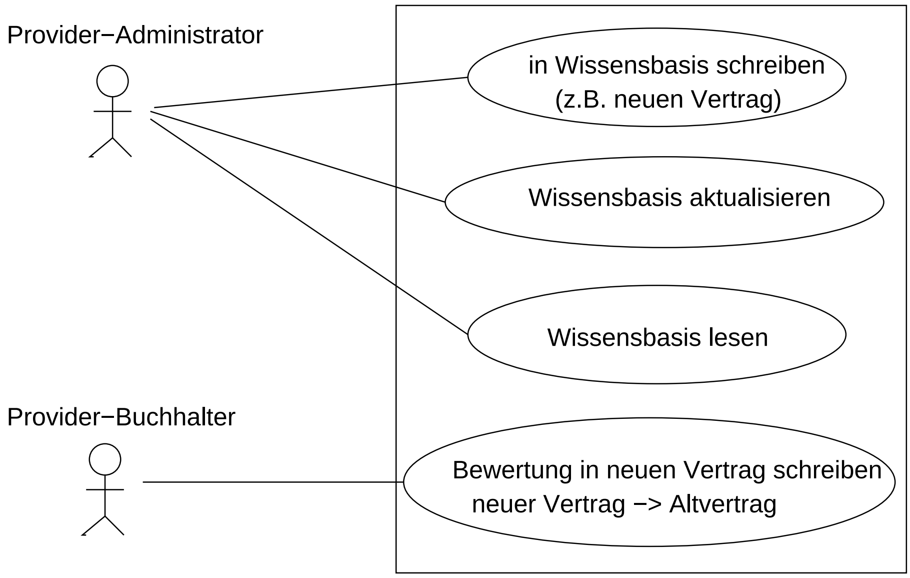

Abbildung 5.1: Anwendungsfall-Diagramm: Datenaktualisierung

1. den für den Aufbau und die Aktualisierung der relevanten Dateien, worunter die Bewertung der Verträge fällt, und

2. die Verhandlung über den Abschluss eines Vertrages, den ein Provider-Vertreter mit dem Kunden unter Bezugnahme auf bewertete Altverträge führt; dazu muss ein passender Altvertrag gesucht und gefunden werden.

Diese Anwendungsfälle sind in Abbildung 5.1 bzw. 5.2 als UML-Anwendungsfall-Diagramme dargestellt ([brj99], [fosc00], [gom00], [sewo00]). In Abbildung 5.1 gibt es die Akteure Provider-Administrator und Provider-Buchhalter, die verschiedene Funktionen des Providers wahrnehmen. (Unter *Wissenbasis aktualisieren* ist das Ändern von Sätzen zu verstehen.) Von ihnen werden in Abschnitt 5.7.5 Klassen für das Verwalten der Wissensbasis (Wissenserwerb) abgeleitet. In Abbildung 5.2 ist der Akteur Provider-Vertreter dargestellt, aus dem die Klasse Vertreter für die Wissensnutzung, nämlich die Suche nach einem besten Vertrag abgeleitet wird.

Das Interaktionsdiagramm (Abbildung 5.3) beschreibt das Hauptszenario, in dem ein durch Nutzerprofil und Kundenprofil gekennzeichneter Vertrag mit bester Bewertung gefunden wird. Das Aktivitätsdiagramm für die Unterstützung des Providers bei der Aushandlung eines Tarifvertrags (Abbildung 5.4) zeigt mehrere Fälle, die beim Suchen nach einem geeigneten Vertrag auftreten können. Ein solcher ist in Abschnitt 5.2, weitere in Abschnitt 5.8 genauer beschrieben.
Für das Suchen nach einem passenden Vertrag sind geeignete Strukturen der Suchkriterien Nutzerprofil, Kundenprofil und Bewertung festzulegen. Außerdem ist ein geeignetes Tarifmodell erforderlich, das den Vergleich von Tarifen ermöglicht. Suchkriterien und Tarifmodell werden in den folgenden Abschnitten analysiert.

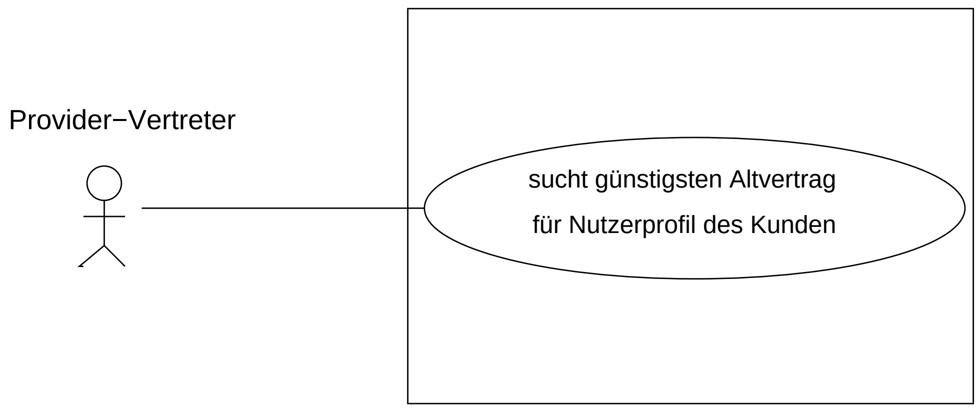

Abbildung 5.2: Anwendungsfall-Diagramm: Vertragsverhandlung

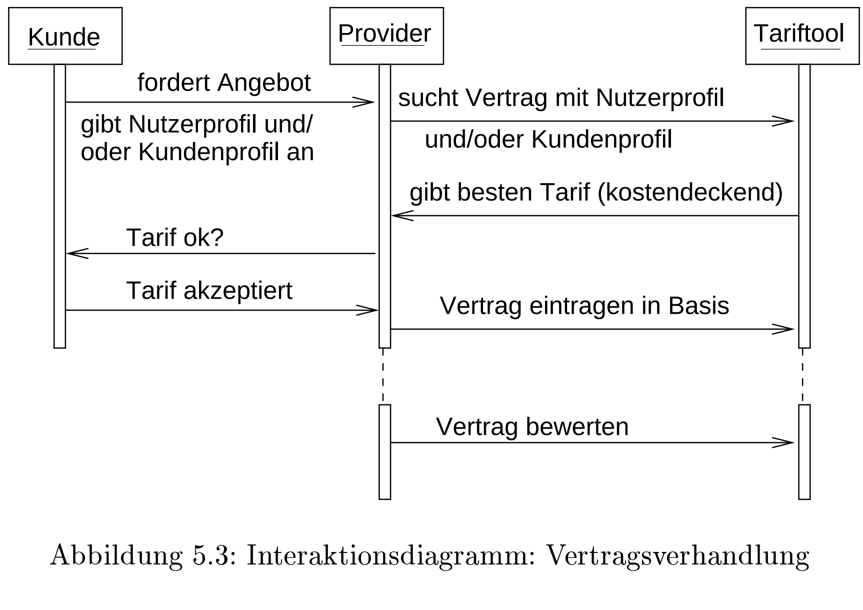

Abbildung 5.3: Interaktionsdiagramm: Vertragsverhandlung

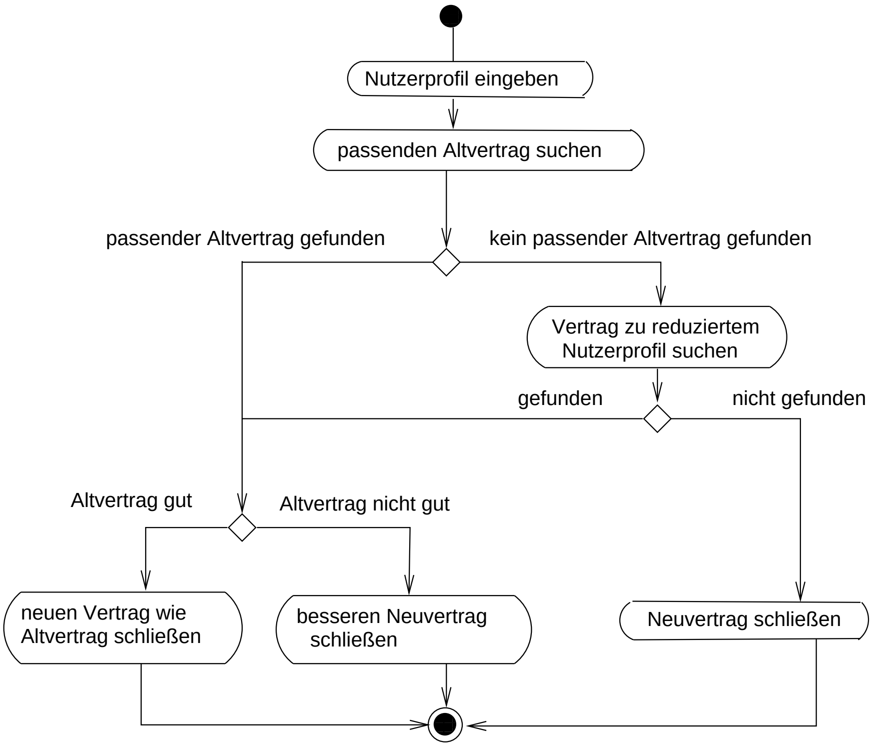

Abbildung 5.4: Aktivitätsdiagramm: Besten Vertrag suchen

5.4 Auswahl der Tarifformel

5.4.1 Einführung

In der Analyse der Tarifierung (Kapitel 4) wurde die prinzipielle Gebührenermittlung sowie Tarifformeln zur Berechnung von Gebühren diskutiert. Eine ähnliche Tarifformel, wie in Abschnitt 4.6 beschrieben, wird im Folgenden für das Tariftool aufgestellt. Sie soll dazu dienen, Verträge zu vergleichen, und unterschiedliche Dienstanforderungen unterstützen (Anforderungen 1, 4 und 7 in Abschnitt 2.2).

5.4.2 Auswahlprinzipien

Bei der vorliegenden Aufgabe des Tariftools handelt es sich um den Abschluss und die Verwaltung von Verträgen zwischen einem Provider und einem Unternehmen, die vor Beginn der Nutzung über bestimmte Zeiträume abgeschlossen werden. In den Verträgen werden Bandbreite und QoS-Parameter der Dienste festgelegt. Daher kommen die in der Literatur ausgiebig behandelten Methoden mit während der Nutzung wechselnden effektiven Bandbreiten für das Tariftool nicht in Frage.

- Das Tarifmodell, das dem Tariftool zugrunde liegt, ist also ein **statisches Modell**.
- In der Tarifformel sollen konkrete Tarifparameter berücksichtigt werden.
- Die Tarifformel soll einfach sein.
- Stausituationen und ihr Einfluss auf die Preise können in diesem Prototyp nicht berücksichtigt werden.
- Regressvereinbarungen werden pauschal, d.h. dienstunabhängig, getroffen.

Regressbedingungen sind in diesem Modell vertrags- d.h. kundenbezogen, während die in die Formel eingehenden Tarifparameter sich auf Dienstvereinbarung, Dienst und Tarifparameterart beziehen. Regress geht in die Tarifformel nicht ein, da diese für die regelmäßige Abrechnung für Nutzung von Diensten vorgesehen ist. Regressforderungen dagegen sollten nur in Ausnahmefällen kommen.

5.4.3 Auswahl der relevanten Tarifparameter

Unter Tarifparameter werden Faktoren verstanden, mit denen gemessene Abrechnungseinheiten multipliziert werden, um Preise zu erhalten (z.B. wird eine Nutzungsdauer in sec gemessen, so hat der zugehörige Tarifparameter die Dimension Preis/sec, evtl. Euro/sec). Wie im vorigen Kapitel 4 ausgeführt wurde, teilt man die Tarifparameter ein in

- **Dienstunabhängige Tarifparameter:**
  Dazu gehören Abonnement und Sitzungszugang. Die Verfügbarkeit von Diensten wird meist mit einer monatlichen festen Gebühr (evt. bandbreiteabhängig) abgegolten, der Zugang zu einer Sitzung (Session Access) mit einer Gebühr pro Sitzungszugang, die noch von der Tageszeit abhängen kann.
- **Dienstabhängige Tarifparameter:**
  Hierbei spielen Nutzungsdauer, -kapazität und -qualität eine bedeutende Rolle. Sie werden in der Umgebung des Nutzerterminals gemessen und von da an den Provider weitergeleitet. Dies und die Zuordnung der Messwerte zu Diensten ist technisch möglich, auf Details kann im Rahmen dieser Arbeit nicht eingegangen werden. Zu jedem dieser Tarifparameter ist eine Abrechnungseinheit definiert, z.B. sec bei Dauer. Diese Tarifparameter können noch von der Tageszeit, der genutzten oder reservierten Bandbreite bei Dauer u.a. abhängen. Für Spezialdienste (meist von Third Parties) wird meist ein Stückpreis erhoben.

Im Tariftool sollen beide Arten berücksichtigt werden. Aus der Literaturrecherche (4.6.1) ergeben sich folgende Tarifparameter als relevant:

- Volumen/ Kapazität (Anzahl Bytes oder MBits),
- Nutzungsdauer,
- QoS-Parameter (z.B. maximale Verzögerung),
- Tageszeit (tags, nachts),
- Access (Sitzungszugang),

- Spezialdienste (z.B. Video on Demand),
- Verfügbarkeit von Diensten (Abonnement).

Von diesen Tarifparametern sind Volumen-, Dauer-, Access- und Abonnement-Tarifparameter direkte Multiplikatoren von Abrechnungseinheiten, QoS- und Tageszeit-Tarifparameter treten als Argumente der übrigen auf, d.h. jene sind Funktionen von QoS- und Tageszeit, gegebenfalls auch voneinander.

5.4.4 Auswahl des Diensttarifs

Da es sich beim Tarifmodell um ein statisches Modell handelt, bei dem stochastische Parameter keine Rolle spielen, kann die ABC-Formel (Abschnitt 4.6.2) nicht unverändert übernommen werden. Die Abhängigkeit von den wichtigsten Messdaten wie Volumen, Dauer und Setup/Access soll aber auch in diese Tarifformel eingehen. Bei den Tarifparametern wird der allgemeine Ansatz von Radisic (s. Abschnitt 4.6.5) berücksichtigt, bei dem die Tarifparameter Funktionen von verschiedenen Parametern sind (s.u.).

Zu jedem Dienst gehört mindestens ein Tarifparameter. Allgemein gilt für einen Diensttarif die Tarifformel:

$$T_D = A(X) \cdot v + B(Y) \cdot d + G$$

mit:

- A,B: Tarifparameter für die Abrechnungseinheiten v bzw. d,
- X, Y: Menge von Parametern, von denen A bzw. B abhängen, z.B. QoS-Parameter, Tageszeit, Dauer, Bandbreite,
- v: Anzahl Abrechnungseinheiten für Volumen (Tarifparameter A),
- d: Anzahl Abrechnungseinheiten für Dauer der Nutzung (Tarifparameter B),
- G: Stückpreis für Spezialdienste.

Setzt sich der Dienst aus Subdiensten zusammen, so gilt:

$$T_D = \sum_{SD} T_{SD}$$

mit SD: Subdienste, für die als Dienste die obige Tarifformel anwendbar ist.

5.4.5 Auswahl des Gesamttarifs

Der Gesamttarif für einen bestimmten Zeitraum, z.B. einen Monat, ergibt sich als Summe über alle Diensttarife:

$$T_G = \sum_D T_D + n \cdot T_A + \sum_S T_S + K$$

mit

- T_A : Tarifparameter für Zugang zu einer Sitzung (Access/Session),
- n: Anzahl Sessions im Abrechnungszeitraum,
- T_S: Tarifparameter für Sonderdienste,
- K: Grundtarif pro Abrechnungszeitraum (Abonnementgebühren).

Mit Hilfe dieser Tarifformel ist es möglich, Verträge/Dienstvereinbarungen zu vergleichen und Anforderung 4 (Abschnitt 2.2) zu erfüllen. Dabei können auch Dienste zu einer Dienstklasse (service class) zusammengefasst und mit gemeinsamen Tarifparametern versehen werden.

5.5 Dienstmodell

Um aufzuzeigen, welche Rollen und Entitäten bei einem Outsourcing-Projekt im IT-Kommunikationsbereich auftreten können, wird zunächst das MNM-Dienstmodell vorgestellt. Die verschiedenen Rollen der Entitäten sind in Abbildung 5.5 (nach [ghk01], deutsche Bezeichnungen nach [schm01]) dargestellt.

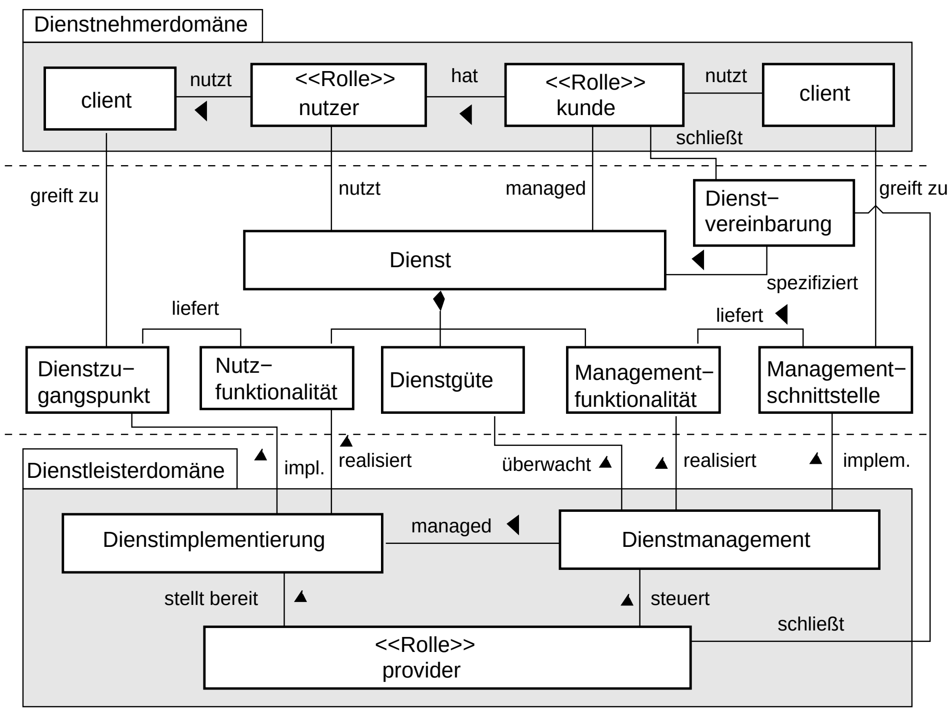

Abbildung 5.5: MNM-Dienstmodell (Service View nach [ghk01])

Das MNM-Dienstmodell zeigt die zur Diensterbringung und -nutzung erforderlichen Funktionalitäten in voller Allgemeinheit, sodass spezielle Dienstmodelle daraus abgeleitet werden können. Es gibt Rollen der Kundenseite (Dienstnehmerdomäne), Rollen der Providerseite (Dienstleisterseite) und seitenunabhängige Funktionalitäten. Das MNM-Dienstmodell kann als Referenzmodell für dienstorientierte Anwendungen dienen und somit sicherstellen, dass alle wichtigen Aspekte berücksichtigt werden.

Beim Tariftool-Modell geht es um Verhandlungen zwischen zwei Parteien - Kunde und Provider - , die verschiedene Ziele haben. Die verschiedenen Entitäten und Rollen sind in Abbildung 5.6 dargestellt. Ein Dienst, wie in Abbildung 5.5 dargestellt, ist Gegenstand der Verhandlung.

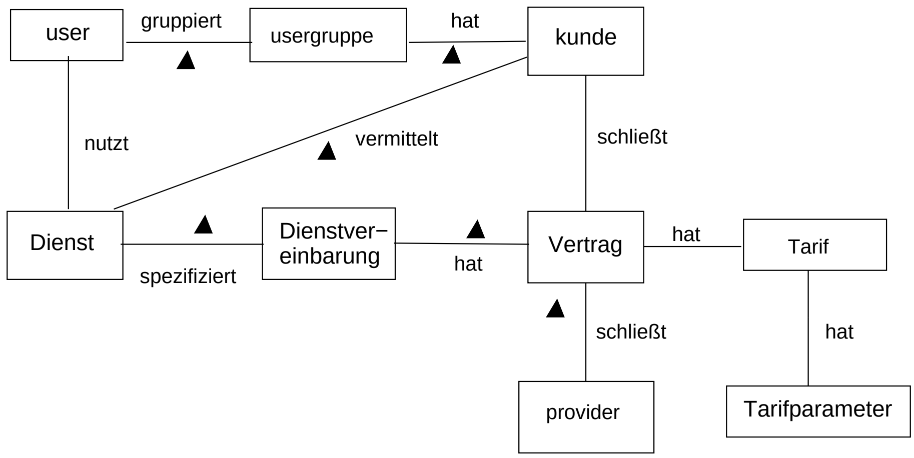

Abbildung 5.6: Dienstmodell: Wichtigste Entitäten des Tariftools

Im Dienstmodell des Tariftools (Abbildung 5.6) werden nur diejenigen Entitäten aufgezeigt, die bei der Aufgabe, den günstigsten Vertrag zu finden, eine Rolle spielen. So fallen z.B. Managementaktivitäten und die Dienstimplementierung fort. Vertrag und Dienst sind beim Tariftool-Modell etwas genauer spezifiziert, weil diese hier eine Hauptrolle spielen. Außerdem sind die User mit gleichen Diensten zu Usergruppen zusammengefasst, welche jeweils einer Dienstvereinbarung zugeordnet sind. Bei Beziehungen, die mit *hat* gekennzeichnet sind, handelt es sich um Kompositionen.

Im Mittelpunkt dieses Modells steht der Vertrag, der Beziehungen zu Kunde und Provider als Vertragspartner sowie zu Diensten und Tarifen als Vertragsgegenstände hat. Die Dienstvereinbarungen sind Teil des Vertrags, sie beziehen sich jeweils auf eine Usergruppe des Kunden. Durch diese Einbeziehung der Dienstvereinbarungen erhält der Vertrag eine klare Struktur analog zur Kunden-Usergruppen-Struktur. Es muss also nicht für jeden User ein Tarif abgeschlossen werden.

Den vereinbarten Diensten wird ein Tarif zugewiesen, dessen Struktur durch die Tarifformel (s. Abschnitt 5.4.4) festgelegt ist. Den Tarifparametern werden Werte gegeben, die eine Dimension Preis pro Abrechnungseinheit haben.

Im Folgenden wird zunächst das Wissensbasierte System erläutert, welches Grundlage des Tariftools werden soll.

5.6 Auswahl des Wissensbasierten Systems

5.6.1 Einführung

Wissensbasierte Systeme (WBS) wurden in Kapitel 3 aufgrund von Literaturrecherchen untersucht und bewertet. Es zeigte sich, dass für das Tarifmodell Modelle mit Objektorientierung in Frage kommen. Bei den Vorschlägen, Wissensbasierte Systeme objektorientiert zu implementieren, wird allerdings übersehen, dass es sich um widersprüchliche Konzepte handelt:

- Bei WBS sind Basisdaten und Problemlösung säuberlich getrennt.
- Bei objektorientierten Systemen ist die Problemlösung eng an die behandelten Daten gekoppelt (Methoden).

Zur Lösung dieses Konflikts wird wie folgt verfahren:

- Die instantiierten Objekte der Wissensbasis enthalten nur Attribute mit Basisdaten und deren Zugriffsroutinen.
- Die Problemlösung wird in Objekten realisiert, die keine Basisdaten, nur Variable und Methoden enthalten.
- Regeln für die Steuerung des Ablaufs der Suche nach einem besten Vertrag werden als Tabellen von Zustands- und Aktionskonstanten, also Basisdaten, implementiert.

Die Analyse des statischen Modells in Kapitel 5 hat gezeigt, dass es im *Tariftool* viele Objekte mit Assoziationen untereinander gibt. Eine objektorientierte Darstellung dieser Objekte ist somit angezeigt. Das zu lösende Problem ist die Auswahl von Objekten durch Muster- und Größenvergleich ihrer Attribute mit Vorgaben. Das Problemlösen durch Suchen ist auch die Strategie bei objektorientierten WBS (vgl. 3.3.6).
Suchen kann in einigen Fällen durch Regeln optimiert werden, die eine Variation des Programmablaufs ermöglichen.

- Ein logikbasiertes WBS ist hier nicht geeignet, da eine logische Ableitung von Formeln nicht erforderlich ist.
- Ein rein regelbasiertes WBS genügt den Ansprüchen nicht, da es die objektorientierte Darstellung einer Vielzahl von Objekten nicht berücksichtigt.
- Ein reines objektorientiertes WBS wiederum lässt keine Regeln zu, die unabhängig von der Inferenz, der Problemlösungskomponente, verändert werden können.

Da es eine große Menge objektorientierter Daten für das Tariftool gibt (s. Abschnitt 5.7), wird also ein **hybrides WBS** mit Schwerpunkt auf objektorientierter Darstellung unter Berücksichtigung von Regeln für die Suchstrategie angestrebt. Es wird eine kompilierende Architektur (s. Kapitel 3) realisiert, d.h. alle Komponenten werden in der Programmiersprache Java implementiert. Durch die Einbeziehung einer Datenbank wird sichergestellt, dass es praktisch keine Begrenzung der gespeicherten Datenmenge gibt (Anforderung 10 in Abschnitt 2.2).

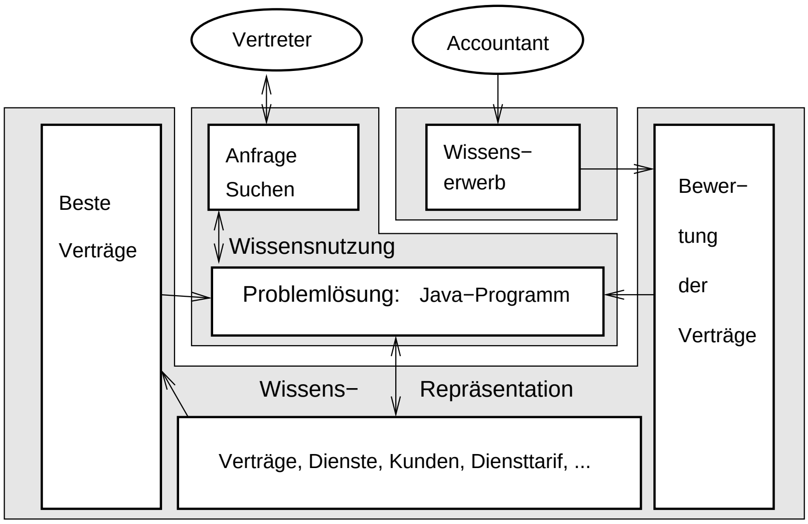

Abbildung 5.7: Anpassung des WBS-Grundmodells an das Tariftool

5.6.2 Aufbau des Wissensbasierten Systems

Die Anpassung des State-of-the-Art-Grundmodells eines WBS an die Anforderungen dieses Projekts findet sich in Abbildung 5.7. Die Hauptkomponenten des Wissensbasierten Systems sind in Abbildung 5.8 dargestellt. In der Wissensrepräsentation ist dort *Vertrag* als Beispiel für die enthaltenen Objekte eingetragen.

5.6.3 Wissensbasis und Datenbank

Die Persistenz einer Wissensbasis ist in einer Datenbank am besten gewährleistet (Anforderung 9 in Abschnitt 2.2). Da relationale Datenbanken am meisten verbreitet sind und eine solche auch für die Entwicklung des Tariftools zur Verfügung steht, wird die Wissensbasis in einer relationalen Datenbank abgespeichert.

Für die Nutzung einer Datenbank in einem wissensbasierten System gibt es zwei Modelle:

- Die Datenbank enthält die gesamte Wissensbasis.

- Die Datenbank enthält nur einen Teil der Wissensbasis, der Rest ist in Dateien abgelegt.

Für die erste Möglichkeit spricht, dass die Daten einheitlich abgelegt sind, für die zweite, dass die ständig verfügbaren Daten von Dateien schneller eingelesen werden können, das Tariftool also eher einsatzbereit ist. Beim Tariftool müssen nicht alle Daten ständig verfügbar sein. Sie würden auch einen großen Teil des Arbeitspeichers belegen. Die gesamte Wissensbasis wird daher in der relationalen Datenbank abgelegt.

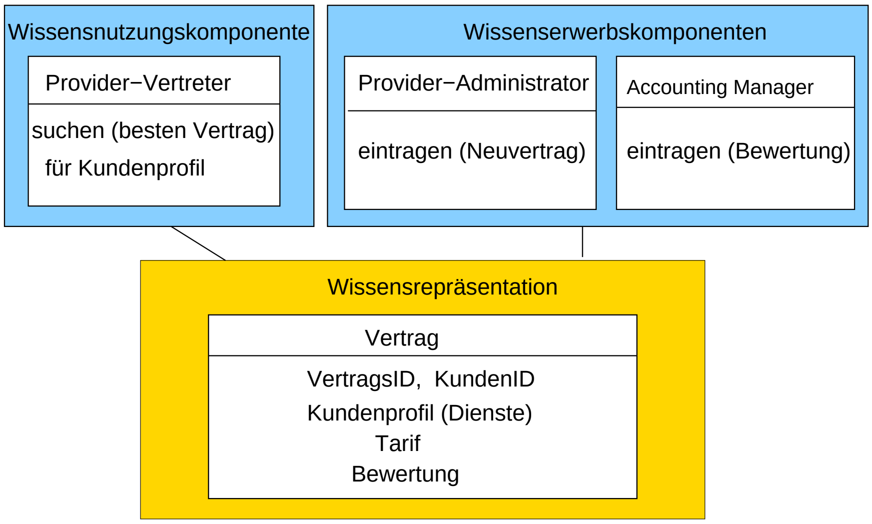

Abbildung 5.8: Hauptkomponenten des Wissensbasierten Systems

Für den Zugriff auf die Datenbank gibt es drei Möglichkeiten:

- Beim Start des Tariftools werden alle Daten in den Arbeitsspeicher eingelesen.
- Daten aus der Datenbank werden nur bei Bedarf eingelesen.
- Ständig benötigte Daten werden beim Start eingelesen, die übrigen nur bei Bedarf.

Wegen der großen Menge der vom Tariftool benötigten Daten erscheint es sinnvoll, die Datenbank dynamisch anzuschließen, d.h. außer einigen ständig benötigten Daten Basisdaten nur dann auszulesen, wenn sie benötigt werden.

5.7 Statisches Modell

5.7.1 Überblick

Im statischen Modell werden entsprechend der objektorientierten Softwareentwicklung die benötigten Klassen von den Rollen und Entitäten des Tariftool-Dienstmodells abgeleitet, so z.B. die Klasse *Kunde* von der Rolle *kunde*. Sie dienen als Grundlage für die Implementierung des Tariftools in der Programmlösungskomponente und in der Wissenbasis, die in einer Datenbank persistent niedergelegt ist (vgl. Abschnitte 3.2.4 und 5.6).

Aus dem Dienstmodell für das Tariftool (Abbildung 5.6) wird das Klassendiagramm (Abbildung 5.9) abgeleitet, in das die zahlenmäßigen Beziehungen eingetragen sind. *hat*-Beziehungen bezeichnen Kompositionen. Die noch benötigten Entitäten wurden ergänzt. Dieses Diagramm zeigt mehr Details als das Dienstmodell (Abbildung 5.6):

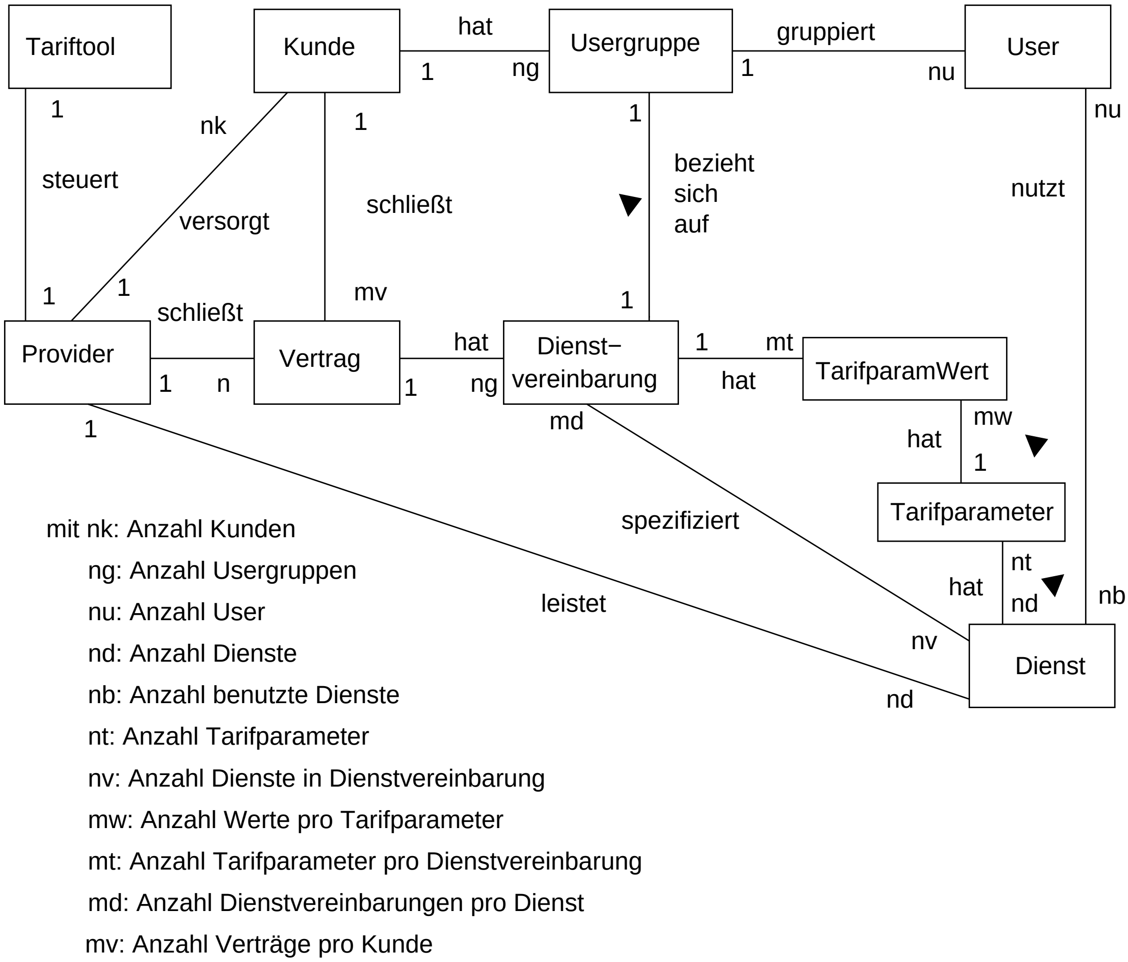

Abbildung 5.9: Klassendiagramm für das Tariftool

- 1:1 - Beziehungen zwischen Dienstvereinbarung und Usergruppe, Tariftool und Provider,

- 1:n - Beziehungen zwischen Provider und Vertrag, Provider und Kunde, Tarifparameter und TarifparamWert, Kunde und Usergruppe, Usergruppe und User sowie Kunde und Vertrag, Vertrag und Dienstvereinbarung, Dienstvereinbarung und TarifparamWert,

- n:m - Beziehungen zwischen User und Dienst, Dienst und Tarifparameter sowie Dienst und Dienstvereinbarung.

Für die n:m - Beziehungen müssen Assoziationsklassen spezifiziert werden (s. Abschnitt 5.7.3). Aus diesem Tariftool-Entitäten-Modell werden die Klassendiagramme in den Abschnitten 5.7.3 und 5.7.5 abgeleitet.

5.7.2 Aufteilung der Klassen

Entsprechend der Aufteilung eines wissensbasierten Systems (Abschnitt 3.1) in einerseits eine Wissensbasis, die für das Tariftool in einer relationalen Datenbank liegt, andererseits Wissenserwerb- und Problemlösungskomponenten werden die Klassen aufgeteilt in

- **Domänenklassen**, die in Attributen das Basiswissen (Fakten) sowie Zugriffsroutinen enthalten. Jede Domänenklasse wird in eine Relation der relationalen Datenbank abgebildet, dabei ist:

  Klassenname = Relationenname (Tabellenname)
  Attributname = Attributname (Spaltenname) in Relation
  Eine Zeile in einer Relation (Datensatz) entspricht der Instanz einer Klasse (Objekt), sie enthält die Attributwerte des Objekts.

  Die Zugriffsroutinen sind nur in der Klassenstruktur deklariert.

- **Klassen für Wissenserwerb und Wissensnutzung**, die Methoden und Variable für die Problemlösung und den Wissenserwerb enthalten. Wegen ihrer Ableitung aus den Akteuren in den Abbildungen 5.2 und 5.1 werden sie im Folgenden als Akteurenklassen bezeichnet. Sie haben kein Abbild in der Datenbank.

- **Regelklassen**, die Wissen, d.h. Attributswerte, in der Datenbank enthalten, welche zur Steuerung des Suchprozesses dienen.

Einzelheiten sind in den folgenden Abschnitten dargestellt.

5.7.3 Domänenklassen

Im Folgenden werden die wichtigsten Domänenklassen mit ihren wichtigsten Daten vorgestellt, die die Wissensbasis im Tariftool bilden. Die Assoziationen zwischen den wichtigsten Domänenklassen und die verschiedenen Rollen sind in Abbildung 5.10 dargestellt. *hat*-Beziehungen bezeichnen Kompositionen.

Für die n:m - Beziehungen müssen Assoziationsklassen spezifiziert werden, dies sind hier:

- DienstVertrag für Dienst und Dienstvereinbarung,
- UserDienstprofil für User und Dienst,
- DienstTarif für Dienst und Tarifparameter.

Alle Klassen sind in mindestens einer der Abbildungen 5.6, 5.9 oder 5.10 enthalten.

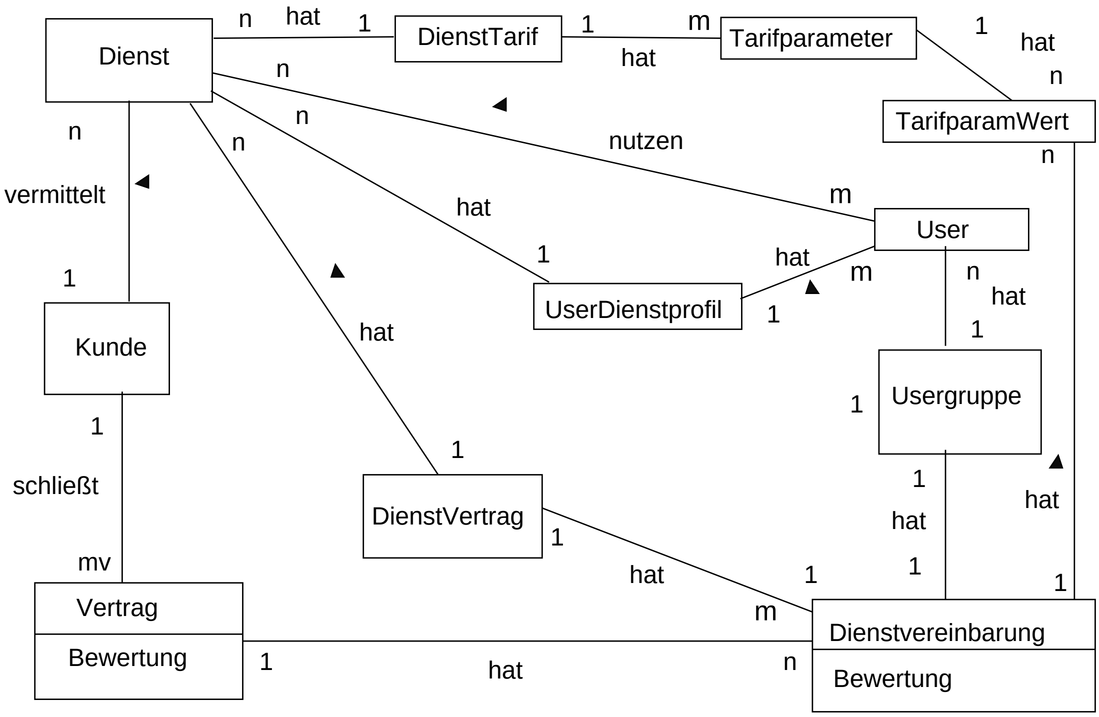

Abbildung 5.10: Klassendiagramm der Domänenklassen

- **Klasse Kunde**
  Die Klasse *Kunde* repräsentiert den Kunden, der die Verantwortung für den Vertrag bezüglich der Dienste für seine User hat. Die User nutzen die Dienste und sind in den Usergruppen zusammengefasst, die jeweils User mit gleichen Diensten enthalten. Jeder Kunde schließt einen Vertrag mit dem Provider. Durch die Einbeziehung von kundenspezifischen Daten, die Einfluss auf Vertragsbedingungen haben können, wird Anforderung 6 (Abschnitt 2.2) nach Unterstützung von Kunden mit verschiedenen Unternehmensstrukturen berücksichtigt. Die Assoziationen der Klasse sind in Abbildung 5.6, 5.9 und 5.10 dargestellt.

Kunde
+kdID : eindeutiges Identifikationsattribut des Kunden -vtID : eindeutiges Identifikationsattribut des Vertrags mit dem Provider -kName : Name des Kunden -kAdresse : Adresse des Kunden -kTelefonnr : Telefonnummer des Kunden -usergrp : Referenz zu den Usergruppen des Kunden -nutzerklassenwert : Zahl, die angibt, zu welcher Nutzerklasse der Kunde gehört, d.h. ob er z.B. bis 5000, über 5000, über 10000 oder über 50000 Nutzer hat -nNiederlgIn : Anzahl Niederlassungen im Inland -nNiederlgOut: Anzahl Niederlassungen im Ausland -bonitaet: Wert, den der Kunde für den Provider hat

- **Klasse Usergruppe:**
Die Klasse *Usergruppe* fasst alle User mit gleicher Dienstnutzung zusammen. Für jede Usergruppe gibt es eine Dienstvereinbarung, die Teil des Vertrages ist (s. u.). Die Assoziationen der Klasse sind in Abbildung 5.6, 5.9 und 5.10 dargestellt.

Usergruppe
+grID : eindeutiges Identifikationsattribut der Usergruppe -kdID : eindeutiges Identifikationsattribut des Kunden -gName : Name der Usergruppe

- **Klasse User:**
Die Klasse *User* repräsentiert die eigentlichen Nutzer der Dienste. Am Ort der Nutzer können die aufgelaufenen Kosten erfasst, d.h. die Nutzung der Dienste periodisch gemessen werden. Nach Aufsummieren der Messwerte pro Usergruppe kann das Ergebnis bei den Daten des TarifparamWerts, der zur den User betreffenden Dienstvereinbarung, Dienst und Tarifparameter gehört, in anzEinheiten abgelegt werden. Die Ergebnisse der Berechnung von Gebühren aus Tarifparametern und anzEinheiten können in einem erweiterten Tariftool zur automatischen Bewertung der Verträge dienen (s. Abschnitt 6.5.6). Die Assoziationen der Klasse sind in Abbildung 5.6, 5.9 und 5.10 dargestellt.

User
+usID: eindeutiges Identifikationsattribut des Users -grID : eindeutiges Identifikationsattribut der Usergruppe -uName : Name des Users -aussendienst : Angabe, ob User im Aussendienst beschäftigt ist oder nicht -standort : Standort des Users -terminalID : Kennung des vom User verwendeten Clients -terminaltype : Typ des vom User verwendeten Clients

- **Klasse UserDienstprofil:**
Diese Assoziationsklasse bildet das Bindeglied zwischen User und Dienst, die eine n:m-Beziehung haben. Diese Klasse repräsentiert die für den User bzw. seine Usergruppe vereinbarten Dienste. Sie enthält die Dienste, die der User nutzt. Die Assoziationen der Klasse sind in Abbildung 5.10 dargestellt.

UserDienstprofil
+udpID : eindeutiges Identifikationsattribut des UserDienstprofils -usID : eindeutiges Identifikationsattribut des Users -diID : eindeutiges Identifikationsattribut des Dienstes (s.u.)

- **Klasse Vertrag:**
Der Vertrag setzt sich zusammen aus kundenspezifischen Daten und einer Anzahl Dienstvereinbarungen über Mengen von Diensten, die Usergruppen zugeordnet werden. In ihm sind die Regressbedingungen aufgeführt, da diese dem Kunden, nicht den einzelnen Dienstvereinbarungen zugeordnet werden. Die Struktur des Vertrags ist für

alle Kunden gleich, so dass Anforderung 1 (Abschnitt 2.2) nach Vergleichbarkeit der Verträge erfüllt ist. Die weiteren Assoziationen der Klasse sind in Abbildung 5.6, 5.9 und 5.10 dargestellt.

Vertrag
+vtID : eindeutiges Identifikationsattribut des Vertrags -kdID : Identifikationsattribut des Kunden, mit dem der Vertrag geschlossen wurde -datumVertragsabschluss : Datum des Vertragsabschlusses -datumKuendigung : Datum der Kündigung des Vertrags - auch nach der Kündigung kann die Bewertung eines Vertrags geprüft werden -datumBeginnLeistungen : Datum des Beginns der vereinbarten Leistungen -datumEndeLeistungen : Datum der Beendigung der vereinbarten Leistungen -regressbedingungen : Code für die Regressbedingungen -wert : Bewertung des Vertrags/Dienstvereinbarung nach Nutzen für den Provider -datumBewertung : Datum der Bewertung (z.B. eine monatliche Bewertung)

- **Klasse Dienstvereinbarung:**
  In dieser Klasse sind die für eine Usergruppe vereinbarten Dienste aufgeführt. Sie ist Teil des Vertrags (1:n-Beziehung). Die Struktur der Dienstvereinbarung ist für alle Kunden gleich, so dass Anforderung 1 (Abschnitt 2.2) nach Vergleichbarkeit der Verträge erfüllt ist. Durch die Zuordnung von Dienstvereinbarungen zu Verträgen können unterschiedliche Dienstanforderungen der Kunden berücksichtigt werden (Anforderung 7 aus Abschnitt 2.2). Die der Dienstvereinbarung zugeordneten Dienste finden sich in der Assoziationsklasse *DienstVertrag*. Die Assoziationen der Klasse sind in Abbildung 5.6, 5.9, und 5.10 dargestellt.

Dienstvereinbarung
+dvID : eindeutiges Identifikationsattribut der Dienstvereinbarung -vtID : eindeutiges Identifikationsattribut des Vertrags -grID : eindeutiges Identifikationsattribut der Usergruppe, die die vereinbarten Dienste nutzt -wert : Bewertung des Vertrags/Dienstvereinbarung nach Nutzen f. Provider -datumBewertung : Datum der Bewertung (z.B. eine monatliche Bewertung)

- **Klasse DienstVertrag:**
  Die Klasse *DienstVertrag* ordnet Dienstvereinbarungen Dienste zu. Sie ist eine Assoziationsklasse zwischen den Klassen *Dienst* und *Dienstvereinbarung*, welche eine n:m-Beziehung zueinander haben. Da zwischen Dienstvereinbarung und Usergruppe eine 1:1-Beziehung besteht, werden hiermit auch den Usergruppen Dienste zugeordnet. Die Assoziationen der Klasse sind in Abbildung 5.10 dargestellt.

DienstVertrag
+vdID : eindeutiges Identifikationsattribut des DienstVertrags -dvID : eindeutiges Identifikationsattribut der Dienstvereinbarung -diID : eindeutiges Identifikationsattribut des Dienstes

- **Klasse Dienst:**
Die Klasse *Dienst* beschreibt einen Dienst und gibt gegebenfalls seine Subdienste an. Die Assoziationen der Klasse sind in Abbildung 5.6, 5.9 und 5.10 dargestellt.

Dienst
+diID : eindeutiges Identifikationsattribut des Dienstes -dienstname : Bezeichnung des Dienstes -subdienste : Subdienste, die zum Dienst gehören -dienstbeschreibung : Beschreibung des Dienstes

- **Klasse DienstTarif:**
Die Klasse *DienstTarif* ordnet Diensten Tarife zu. Sie ist eine Assoziationsklasse zwischen den Klassen *Dienst* und *Tarifparameter*, die eine n:m-Beziehung haben, und stellt somit eine formelhafte Tarif-Beziehung dar. Die Assoziation der Klassen Dienst und Tarifparameter ist in Abbildung 5.10 dargestellt.

DienstTarif
+dtID : eindeutiges Identifikationsattribut des DienstTarifs -diID : eindeutiges Identifikationsattribut des Dienstes -tpID : eindeutiges Identifikationsattribut des Tarifparameters

- **Klasse Tarifparameter:**
In der Klasse *Tarifparameter* werden für die Tarifformel Abrechnungseinheiten festgelegt. Sie enthält auch eine Beschreibung des Tarifparameters. Die Assoziationen der Klasse sind in Abbildung 5.10 und 5.9 dargestellt.

Tarifparameter
+tpID : eindeutiges Identifikationsattribut des Tarifparameters -parName: Name des Tarifparameters -tBeschreibung : textuelle Beschreibung des Tarifparameters -abrechnungseinheit: Dimension (z.B. sec, Bytes, Mbits, u.a.) -abrechnungszeitraum: Zeitraum für periodische Abrechnung

- **Klasse TarifparamWert:**
  In der Klasse *TarifparamWert* werden die in der Dienstvereinbarung festgelegten Preise pro Tarifparameter und Dienst abgespeichert. In anzEinheiten können die pro Usergruppe, Dienst und Dienstvereinbarung im Abrechnungszeitraum aufgelaufenen Abrechnungseinheiten gespeichert werden. In einer Erweiterung des Tariftools kann daraus automatisch die Bewertung einer Dienstvereinbarung berechnet werden (Anforderung 3 in Abschnitt 2.2 sowie Abschnitt 6.5.6). Die Assoziationen der Klasse sind in Abbildung 5.9 und 5.10 dargestellt.

TarifparamWert
+twID : eindeutiges Identifikationsattribut des TarifparamWerts -dvID : Referenz zur Dienstvereinbarung -tpID : Referenz zum Tarifparameter -diID : Referenz zum Dienst -preisfaktor: Preis pro Abrechnungseinheit (AE) -anzEinheiten: Ermittelte Anzahl AE pro Abrechnungszeitraum -kosten : Gesamtpreis im Abrechnungszeitraum

5.7.4 Regelklassen

Übersicht

Wie bereits in Kapitel 3 erläutert wurde, dienen Regeln dazu, den Ablauf von Wissensnutzungsprogrammen zu steuern, d.h. dafür zu sorgen, dass bestimmte Programmzweige durchlaufen und andere ausgelassen werden. Beim Tariftool z.B. kann so die Suche nach bestimmten Diensten der Suche nach Kundenmerkmalen vorgezogen werden und umgekehrt.

Die Wirkung von Regeln kann man mit Zustandsdiagrammen modellieren, wie es in einem Beispiel in Abbildung 5.11 dargestellt ist.

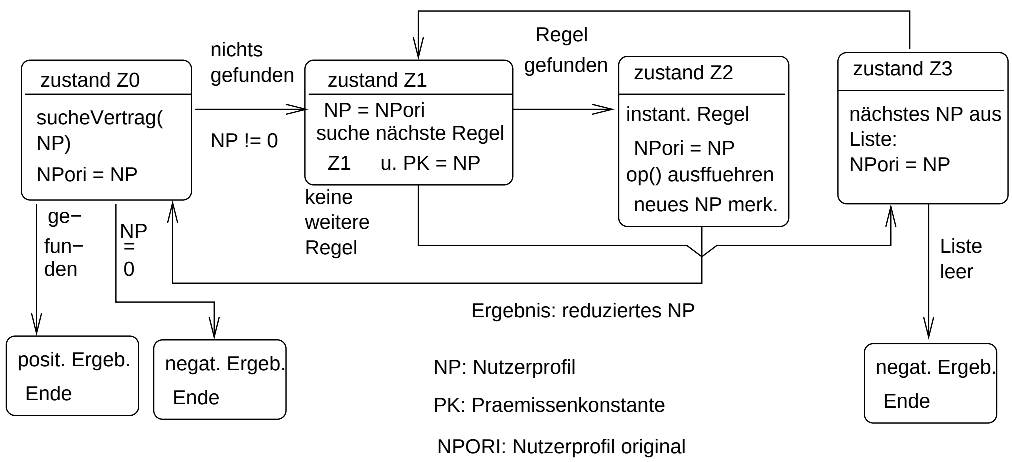

Abbildung 5.11: Zustandsdiagramm mit Regelverwendung

Im Rahmen des Tariftool-Prototyps ist es nicht möglich, eine syntaktische und semantische Sprachanalyse von Regeltexten durchzuführen. Zudem sollen die Regeln so abgelegt sein, dass der Kunde sie leicht ändern kann, wenn er einen anderen Programmablauf wünscht als defaultmäßig eingestellt ist.

Es gibt nun folgende Realisierungsmöglichkeiten:

1. **Einbettung der Regeln** als Produktionsregeln in ausführbare Java-Objekte, die als File oder Blob (großes Objekt in der Datenbank) vorliegen und bei Bedarf in den Arbeitsspeicher geladen werden.
2. **Strukturierung von Regeln** so, dass Daten/Werte auf auszuführende Operationen verweisen, die Teil der Inferenz sind. Dabei kann auch eine Konstante in der Regelklasse bestimmen, welche Funktion die Inferenz in einem bestimmten Zustand ausführen soll. Die Funktion wird von einer Methode der Inferenz ausgeführt.

Eine Änderung von Java-Objekten ist nur durch Fachpersonal möglich und kann nicht von jedem Kunden erwartet werden. Daher wird die Lösung durch mögliche Änderung von Konstanten in der Regel realisiert. Wie eine solche Regel aussieht, wird im Folgenden erklärt.

Strukturen von Regelklassen

1. **Regeln mit Masken**

   Eine typische Regel im Tariftool lautet:

   ```
    IF fuer nutzervektor nichts gefunden
    THEN setze  nutzervektor =  nutzervektor & Maske
   ```

   Dabei ist die Maske so gesetzt, dass ein Tarifparameter herausfällt. In der abgespeicherten Regel muss für den Nutzervektor eine entsprechend strukturierte Konstante bzw. ein String (`praemissenkonst`) stehen, so dass die Maske dazu passt. D.h. für die obige Regelstruktur gibt es mehrere Regeln mit Werten für alle möglichen Nutzervektoren bzw. Kundenvektoren und für alle Masken, die einen Nutzervektor um einen Parameter reduzieren.

   ```
   IF NP == praemissenkonst
   THEN NP := NP & aktion
   ```

   praemissenkonst entspricht dem gerade relevanten Suchkriterium. Der Klartext (Formulierung der Regel) könnte zum Test ausgegeben werden.

Dabei sieht die Methode op() etwa so aus:

```
case typ
 M: NP := NP & aktion;
 ...
```

Vor Ausführen der Methode wird geprüft, ob die Praemisse erfüllt ist, d.h.

```
mit NP = Suchkriterium (Kundenvektor oder UserDienstvektor):
 NP == praemissenkonst gilt.
```

Danach wird weiter nach einem passenden Vertrag mit reduziertem NP gesucht, bis einer gefunden wird, oder NP auf 0 Parameter reduziert (s. Abbildung 5.11).

Die Regelbasis ist erweiterbar, ohne dass die Inferenzmaschine verändert werden muss, denn es werden immer alle Regeln abgesucht. Die Reihenfolge der Speicherung in der Datenbank bestimmt die Reihenfolge der Abarbeitung der Regeln zu einem Zustand.

2. **Regeln durch Konstante repräsentiert.**
Hierbei gibt es zu jedem Zustand, von dem aus mehrere Verzweigungen möglich sind, eine Regel, in der die der Zustandsvariablen zugeordnete Konstante angibt, welche Aktion auszuführen ist.

Klasse Regel:

Regel
regID : eindeutiges Identifikationsattribut der Regel regelName : Bezeichnung der Regel zustand : Zustand der Inferenz, in dem die Regel anwendbar ist regelKonstante : Konstante, die den weiteren Ablauf der Inferenz bestimmt klartext : Formulierung der Regel konst1 : 1. moeglicher Wert k1text: Aktion zu k1 konst2 : 2. moeglicher Wert k2text: Aktion zu k2 . .
op()

In einem bestimmten Zustand sucht das Inferenzprogramm in der Datenbank nach der Konstanten, die angibt, wie es weiter verfahren soll. Eine IF THEN ELSE, oder Case-Struktur führt dann zu der auszuführenden Methode im Inferenzprogramm.

Dabei sieht die Methode op() etwa so aus:

```
case zustand:
  Z1:
    case regelKonstante
      1: action1();
      2: action2();
     .
     .
```

Soll eine andere Aktion ausgeführt werden, muss nur die Regelkonstante ausgetauscht werden.

Die zweite Darstellung von Regeln durch Konstanten, die in case-Strukturen ausgewertet werden, hat den Vorteil, dass Änderungen leicht eingebracht werden können. Es lässt sich so einfach heuristisch erforschen, wie man am besten vorgeht, ohne dass am Inferenzprogramm etwas geändert werden muss. Es wird daher die Darstellung von Regeln durch (zustand, konstante)-Paare realisiert.

`aktion_1` regelt, nach welchem Suchkriterium zuerst gesucht wird (1: nach Vertrag, 2: nach Dienstvereinbarung (s. Abschnitt 5.8.2). `aktion_2` regelt, nach wieviel Diensten gesucht werden kann (0: nach den 7 ersten aus der Tabelle Dienstschema, 1: nach den 15 ersten, 2: nach insgesamt 30, sofern sie in der Tabelle enthalten sind, s. Abschnitt 7.7).

5.7.5 Akteurenklassen: Klassen für Wissensnutzung und -erwerb

Im Folgenden werden die wesentlichen Daten der Akteurenklassen

- Vertreter
  Klasse zum Suchen von Verträgen und Dienstvereinbarungen
- Account (Buchhalter in Abbildung 5.1)
  Klasse zur Bewertung von Verträgen und Dienstvereinbarungen
- Admin
  Klasse zum Lesen, Ändern und Schreiben von Datensätzen der *tarif*-Datenbank

sowie die der Klasse Tariftool (Hauptprogramm) dargestellt (Abbildung 5.12). Sie haben kein Abbild in der Datenbank.
Diese Klassen Vertreter, Account und Admin repräsentieren die Akteure, welche in den Usecase-Diagrammen (nach [gom00], [brj99], [fosc00]) in Abbildung 5.2 und 5.1 (in Abschnitt 5.3) dargestellt sind.

Die Klassen der Akteure enthalten somit die Inferenz, d.h. die Problemlösungskomponente, und den Wissenserwerb. Sie alle stellen den Provider in speziellen Funktionen dar.

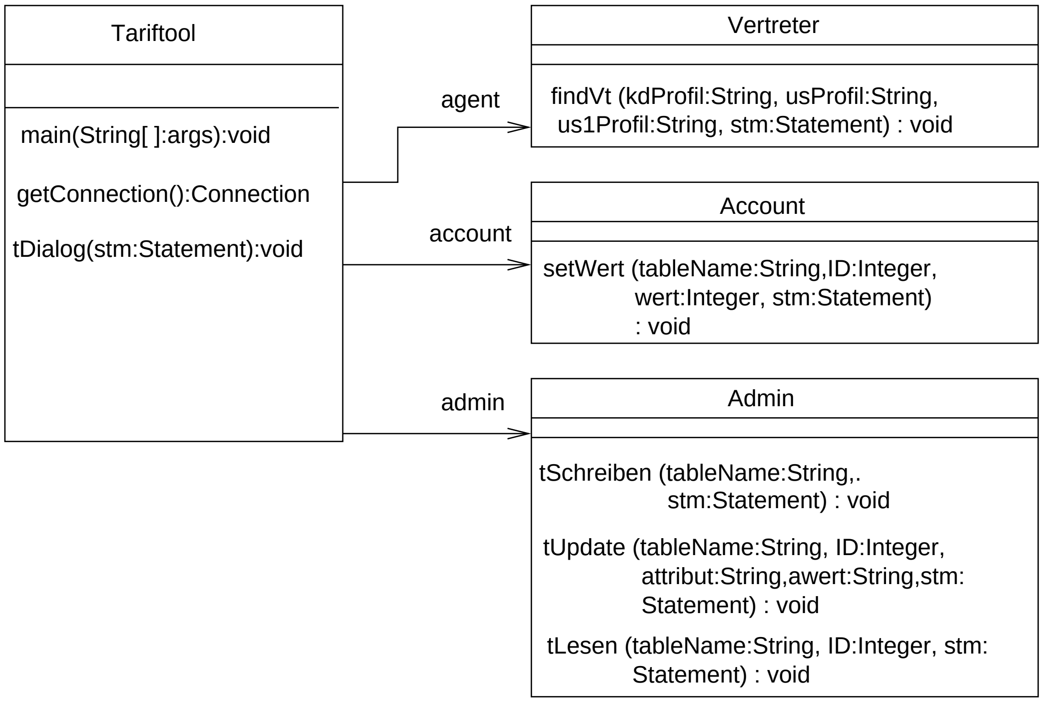

Abbildung 5.12: Klassendiagramm der Akteure

Klasse für das Hauptprogramm

Klasse Tariftool:

Tariftool
+con : Assoziation zur Klasse Connection der Datenbank ...
main(String[ ]:args) : void getConnection(): Connection tdialog(Statement stmt) : void

Klassen für den Wissenserwerb

Die Methoden der Klasse *Admin* (s. Abbildung 5.12) greifen auf alle Tabellen der Wissensbasis lesend und schreibend zu. Die Methode der Klasse *Account* schreibt nur Bewertung und Datum in die Tabellen *Vertrag* und *Dienstvereinbarung*.

Klasse für die Wissensnutzung

Die Wissensnutzung wird von der Klasse Vertreter für die Inferenz ausgeführt. Die Methode *findVt* liest in den Tabellen *Regel*, *Vertrag* und *Dienstvereinbarung*.

Klasse Vertreter:

Vertreter
-vwert : Bewertung des gefundenen Vertrags -dwert : Bewertung der gefundenen Dienstvereinbarung -vstr: Kundenvektor zur Reduktion -dstr: Nutzervektor zur Reduktion -tabName: Tabellenname -subTable : zweiter Tabellenname -vtId: Identifikation des gefundenen Vertrags -dvId: Identifikation der gefundenen Dienstvereinbarung -vfound : Kennung, ob Vertrag gefunden wurde -dfound : Kennung, ob Dienstvereinbarung gefunden wurde -vreduz : Kennung, ob Kundenvektor reduziert wurde -dreduz : Kennung, ob Nutzervektor reduziert wurde -passend : Kennung, ob gefundener Vertrag zu Dienstvereinbarung passt -both : Kennung, ob nach Dienstvereinbarung und Vertrag gesucht wird -vwert : Bewertung des Vertrags -dwert : Bewertung der Dienstvereinbarung ...
findVt(kdProfil:String, usProfil:String, us1Profil:String, stm:Statement):void reduV1(j:Integer, zustand:Integer, kdProfil:String): Integer reduD1(j:Integer, zustand:Integer, usProfil:String): Integer

Zum Inhalt der Felder s. Abschnitt 7.7.4, zu den Funktionen Abschnitt 5.8.2.

5.8 Dynamisches Modell

5.8.1 Einführung

Im dynamischen Modell wird analysiert, welche Aufgaben die Programme ausführen, was sie dazu benötigen und wie ihr zeitlicher Ablauf aussieht. Eine Übersicht wurde schon in Abschnitt 5.3 (Abbildung 5.3 und 5.4) gegeben. Im Fall des Tariftools spielen Zugriffe auf die relationale Datenbank zur Wissensnutzung und zum Wissenserwerb eine besondere Rolle. Im Fall der Wissensnutzung wird in der Datenbank nach geeigneten Verträgen gesucht, im Fall des Wissenserwerbs werden Daten aktualisiert und neue Datensätze geschrieben. Zur Kontrolle können Datensätze auch gelesen werden.

In den folgenden Abschnitten wird die Suche in der Datenbank und das Einbringen neuer Wissensinhalte genauer beschrieben.

5.8.2 Programmablauf des Tariftool-Suchprogramms

Der Ablauf dieses Inferenzprogramms ist global in Abbildung 5.13 dargestellt. Abhängig davon, was gesucht und (nicht) gefunden wurde, durchläuft das Programm verschiedene Zustände. In jedem Zustand ist eine Auswahl möglich, wie weiter vorzugehen ist, d.h. eine Verzweigung zu verschiedenen Methoden. Welche auszuwählen ist, steht in der Relation `Regel` in der Datenbank beim Attributswert `zustand` als `Aktionskonstante`. Durch Eingabe verschiedener Werte in die Tabelle Regel kann der Anwender das Programm tunen, d.h. es so steuern, dass es beim Ablauf einen anderen Programmzweig durchläuft.

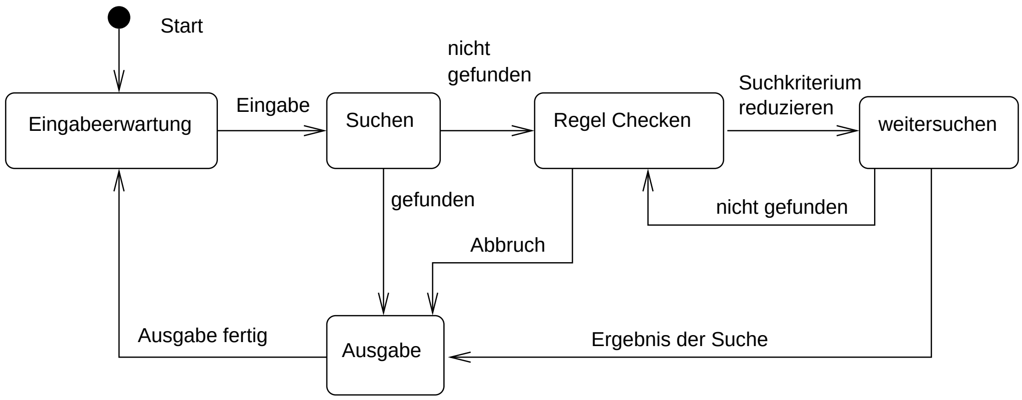

Abbildung 5.13: Globales Zustandsdiagramm der Inferenz

Die Klasse Vertreter enthält als Suchprogramm (Inferenzprogramm im WBS-Fachjargon) die Subroutinen, die den besten Vertrag für bestimmte Suchkriterien wie Kundenvektor und Nutzervektor ermitteln. Diese sind in der Vertrags- und der Dienstvereinbarungsklasse als Strings mit Einsen für vorhandene bzw. Nullen für nicht vorhandene Items eingetragen. Die Bedeutung der Items ist den Klassen Kundenschema und Dienstschema zu entnehmen (Abschnitt 6.3). Durch die Verwendung von Suchkriterien wird die Suche effizient gestaltet (Anforderung 11 in Abschnitt 2.2).

Werte für das Suchkriterium `Kundenvektor` sind auch in jeden Vertrag in der Datenbank eingetragen, sowie Werte für das Suchkriterium `Nutzervektor` in jede Dienstvereinbarung. Mittels `query` wird nach den gewünschten Verträgen und/oder Dienstvereinbarungen gesucht. Aus den gefundenen Verträgen/Dienstvereinbarungen werden diejenigen mit der besten Bewertung ermittelt.

Danach werden die zugehörigen Tarifparameter gesucht, die auch in der Datenbank eingetragen sind und die Kennung der zugehörigen Dienstvereinbarung enthalten. Die Dienstvereinbarung enthält das eindeutige Identifikationsattribut des Vertrages.

Im Detail sind verschiedene Zustände in Abhängigkeit von der Eingabe zu berücksichtigen (s. Abbildung 5.13). Wird kein Vertrag bzw. keine Dienstvereinbarung gefunden, die den eingegebene Kundenvektor bzw den Nutzervektor enthält, so wird eine Zustandvariable auf eine neuen Wert gesetzt.

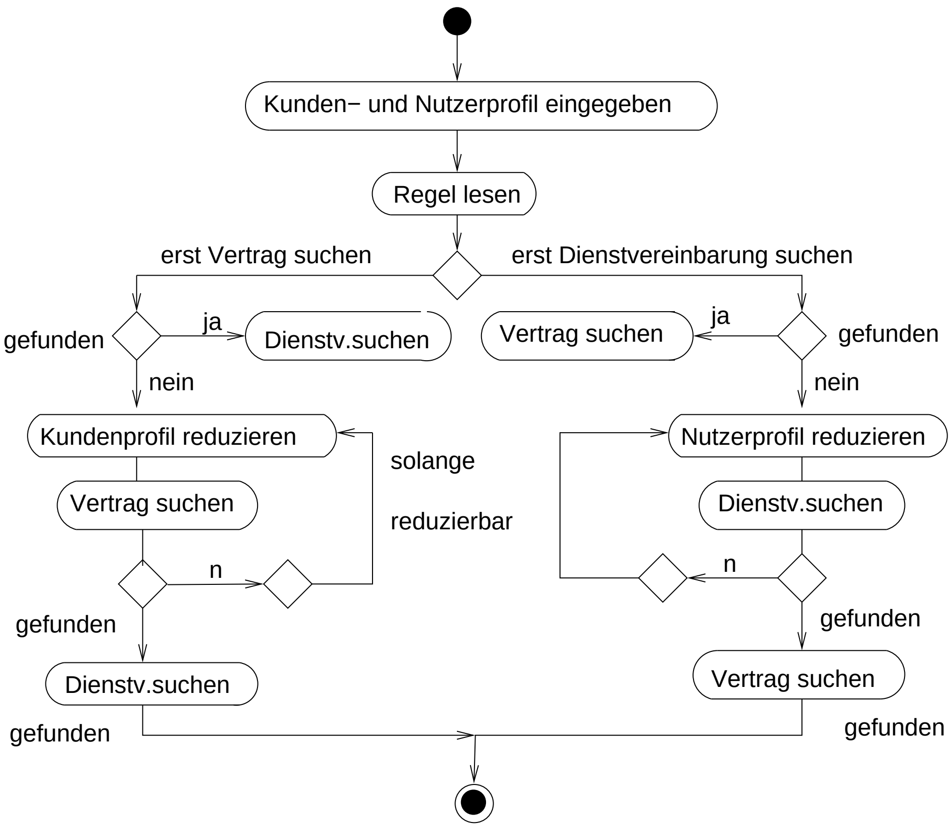

Abbildung 5.14: Aktivitätsdiagramm der Suchroutine (Klasse Vertreter)

Wird bei Suchen nach dem Nutzervektor kein Vertrag gefunden, geht es weiter wie in Abbildung 5.11 (Abschnitt 5.7.4); bei Kundenvektor analog. Von den Anforderungen des Kunden wird eine weggelassen und versucht, für das so reduzierte Suchkriterium einen Vertrag zu finden. Technisch wird zunächst das erste Item aus dem Nutzer- bzw. Kundenvektor weggelassen (d.h. die kennzeichnende 1 im Nutzer- bzw. Kundenvektor auf 0 gesetzt) und versucht, dazu passende Dienstvereinbarungen /Verträge zu finden. Ist auch dies ohne Erfolg, wird das folgende Item auf 0 gesetzt und das vorige wieder auf 1, sodass immer die gleiche Anzahl Einsen im String ist (1 weniger als zu Beginn der Suche). Bei Suchen nach zwei Kriterien wird durch eine Regel festgelegt, nach welchem zuerst gesucht wird, und gegebenfalls, welches Suchkriterium zuerst reduziert wird.

Wird mit den Anfangssuchkriterien kein Vertrag gefunden, geben weitere Regeln an, wie die Kriterien zu reduzieren sind, und auch, welches Kriterium zuerst reduziert wird. Regeln können auch festlegen, dass wechselweise reduziert wird, oder erst ein Kriterium ganz. Dies ist in Abbildung 5.14 global dargestellt.

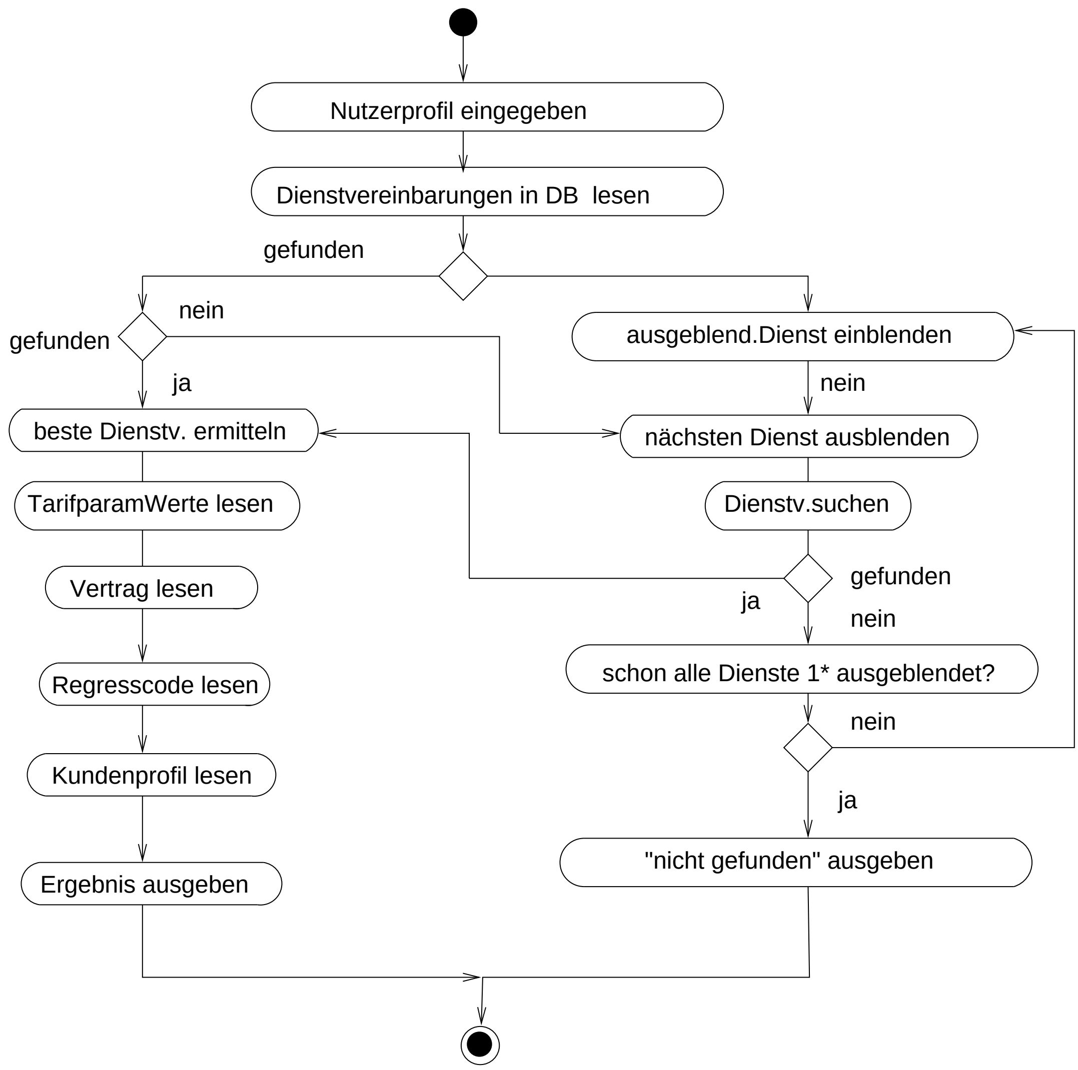

Abbildung 5.15: Aktivitätsdiagramm: Suchen nach bester Dienstvereinbarung

In Abbildung 5.15 ist das Suchen nach einer Dienstvereinbarung mit bester Bewertung und bestimmtem Nutzervektor, der die vereinbarten Dienste kennzeichnet, sowie die Ermittlung der Ergebnisse dargestellt. Abbildung 5.16 stellt das Suchen nach einem Vertrag mit bester Bewertung und bestimmtem Kundenvektor dar, der das Kundenprofil kennzeichnet, sowie die Ermittlung der Ergebnisse dargestellt.

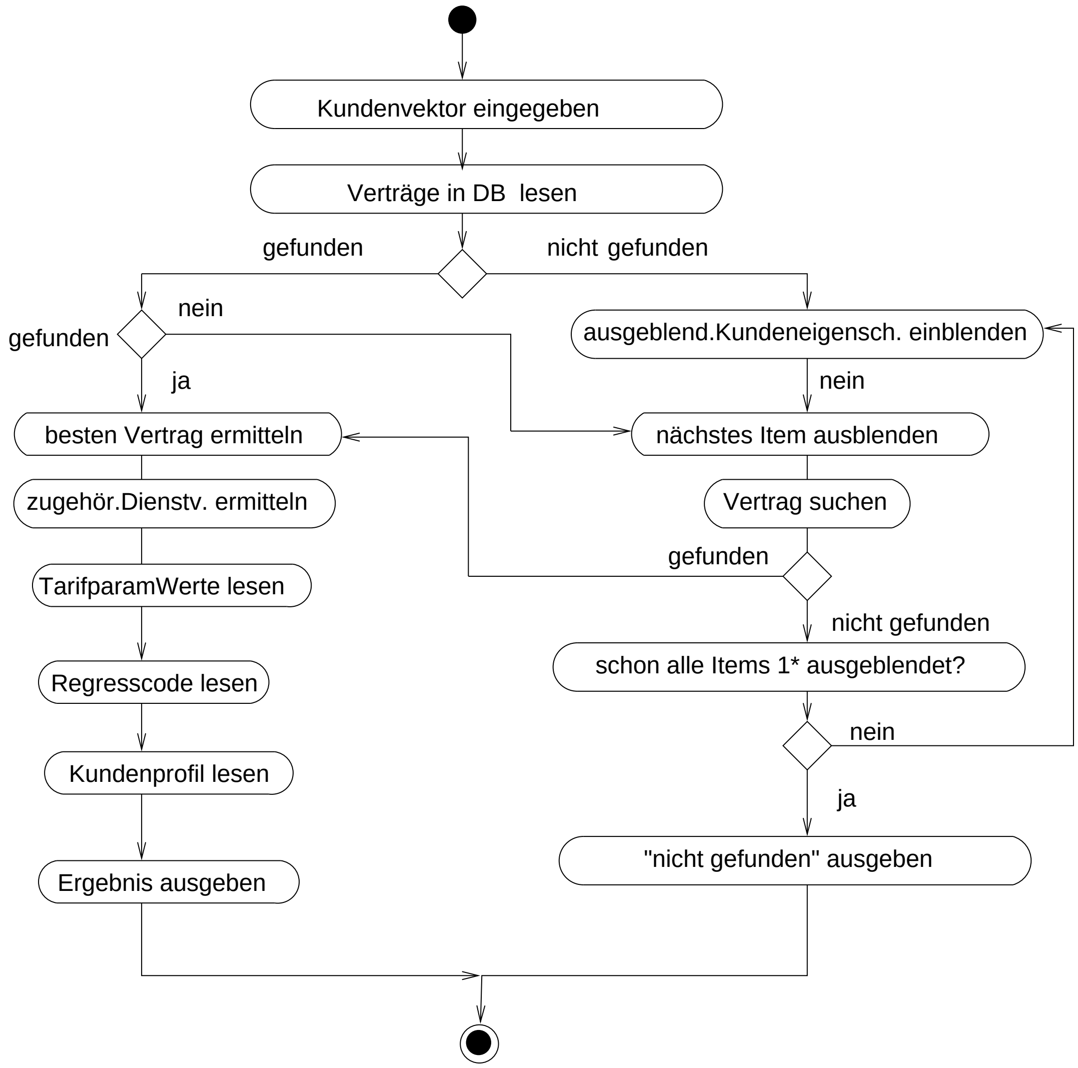

Abbildung 5.16: Aktivitätsdiagramm: Suchen nach bestem Vertrag

5.8.3 Verwaltung von Domänen- und Regelobjekten

Hierfür gibt es die Funktionen:

- schreiben,
- ändern,
- lesen,

die mit SQL-Kommandos (s. z.B. [hart01]) ausgeführt werden.

Nach Auswahl der Funktion ist der Klassenname (Relationsname) zu wählen, danach ist einzugeben:

- bei Schreiben: die Daten des neuen Objekts, diese werden mittels `insert` in die Datenbank eingetragen;
- bei Ändern: die zu ändernden Attribute und Werte, diese werden mittels `update` in die Datenbank eingetragen;
- bei Lesen: die Objekt-ID, die gespeicherten Daten werden mittels `select` aus der DB gelesen und ausgegeben.

Einzelheiten siehe Abschnitt 7.7.

5.8.4 Fehlerbehandlung

Die SQL-, IO- und sonstigen Exceptions werden abgefangen und entsprechende Fehlermeldungen werden ausgegeben.

5.9 Zusammenfassung

Zunächst wurden die Szenarien für die Aufgaben des Tariftools analysiert. Danach wurde für das Tariftool eine allgemeine Tarifformel entwickelt, die den Vergleich von Tarifen ermöglicht. Ein hybrides Wissensbasiertes System wurde als Grundlage des Tariftools ausgewählt. Aus dem Dienstmodell des Tariftools wurden Entitäten und Klassen für die Implementierung abgeleitet. Ein statisches Modell des Tariftools mit Domänenklassen für die Basisdaten der Wissenbasis sowie einer Regelklasse zum Tunen der Suchfunktion des Tariftools wurde entwickelt. Aus den Anwendungsfällen der Anforderungsanalyse wurden Akteurenklassen abgeleitet, in denen die Funktionen des Tariftools ablaufen. Der Ablauf der Suchfunktion wurde im dynamischen Modell beschrieben, ebenso die Verwaltungsfunktionen.

Kapitel 6

Entwurf eines geeigneten Toolmodells

In diesem Kapitel werden geeignete Werte für den Entwurf eines Tariftools aus den in Kapitel 5 analysierten möglichen Werten ausgewählt. Zur Optimierung der Suche nach einem geeigneten Vertrag werden Suchkriterien analysiert und festgelegt. Die Wissensrepräsentation wird entworfen und sinnvolle Werte für die Basisdaten werden ermittelt. Ein Datenbankschema wird anhand eines Entity-Relationship-Diagramms entwickelt. Die Umsetzung von Datentypen des Programms in Datenbanktypen wird analysiert. Die Package-Struktur des Tariftools wird festgelegt.

6.1 Einleitung

Da das Tariftool als Prototyp nicht die Gesamtheit aller Tarifanforderungen bewältigen kann, werden einige Einschränkungen getroffen, die die Anzahl der Verträge, Dienste und Tarifparameter betrifft. Für den Prototyp wird eine Auswahl repräsentativer Dienste und Tarifparameter getroffen, die die Basis des Wissens in der Datenbank bilden. Die Struktur der Wissensrepräsentation für das Tariftool wird analysiert und festgelegt. Zur Optimierung der Suche nach passenden Verträgen werden Suchkriterien festgelegt. Die Package-Struktur des Tariftools wird entworfen.

6.2 Design-Entscheidungen

Geschäftsgrundlage für die folgende Analyse des Systems sind:

1. Der Provider verwaltet seine Daten wie Verträge, Kunden, Dienste usw. selbst.

2. Jeder Kunde hat nur einen Vertrag mit dem Provider. Diese Einschränkung dient zur Vereinfachung des komplexen Systems. Ein Vertrag enthält dabei mindestens eine Dienstvereinbarung; es können weitere Dienstvereinbarungen zugefügt werden.

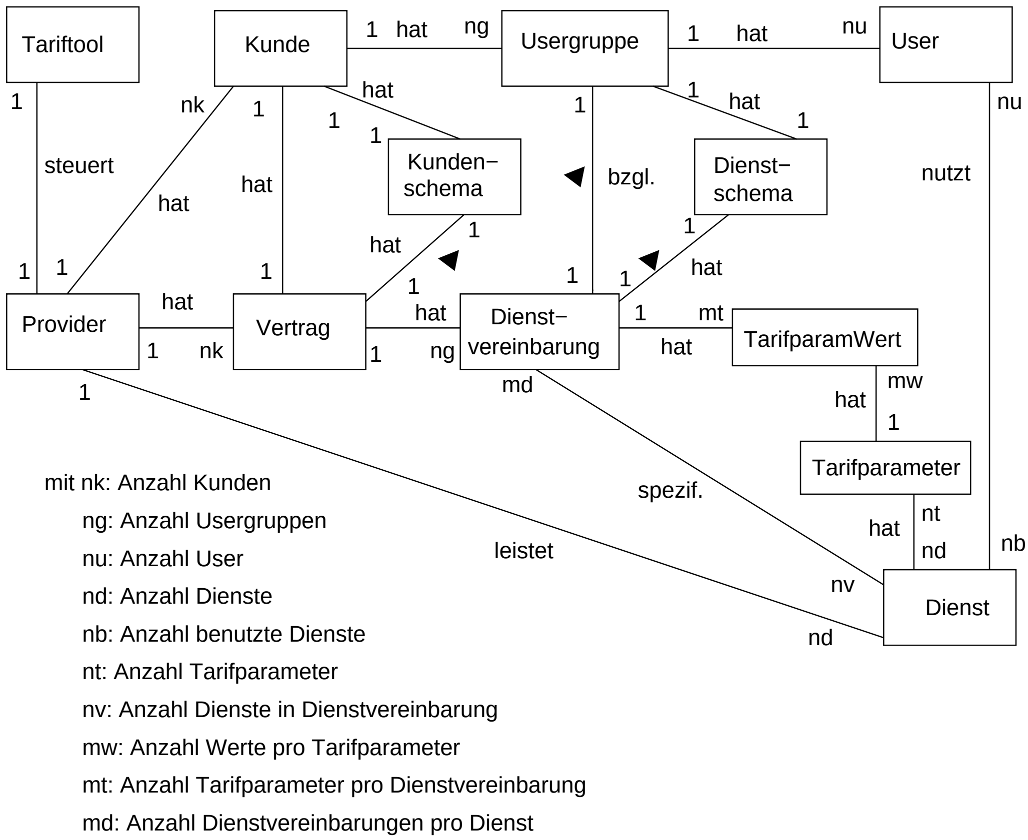

Abbildung 6.1: Klassendiagramm des Tariftool-Entwurfs mit Hilfsklassen

3. Zur Optimierung der Suche durch Vermeidung aufwändiger Joins zwischen den Tabellen der Datenbank werden Hilfsgrößen, nämlich die Klassen *Kundenschema* und *Dienstschema* eingeführt. Von ihnen werden die Suchkriterien *Kundenvektor* und *Nutzervektor* abgeleitet. Einzelheiten werden in Abschnitt 6.3 beschrieben.

Das Klassendiagramm Abbildung 6.1 zeigt die Klassen des Tariftool-Entwurfs mit den Hilfsklassen. Abbildung 6.2 zeigt eine Übersicht über die Domänenklassen, die in die Datenbank abgebildet werden. In beiden Abbildungen bezeichnen *hat*-Beziehungen Kompositionen.

Zwischen den Klassen des Tariftool-Entwurfs bestehen folgende Beziehungen:

- 1:1 - Beziehung: Kunde - Vertrag, Kunde - Kundenschema, Vertrag - Kundenschema, Dienstvereinbarung - Dienstschema, Usergruppe - Dienstvereinbarung , Usergruppe - Dienstschema, Tariftool - Provider,
- 1:n - Beziehung: Kunde - Usergruppe, Usergruppe - User, Dienstvereinbarung - TarifparamWert, Kunde - Dienst, Tarifparameter - TarifparamWert, Vertrag - Dienstvereinbarung,
- n:m - Beziehung: Dienst - Dienstvereinbarung, Dienst - Tarifparameter, User - Dienst.

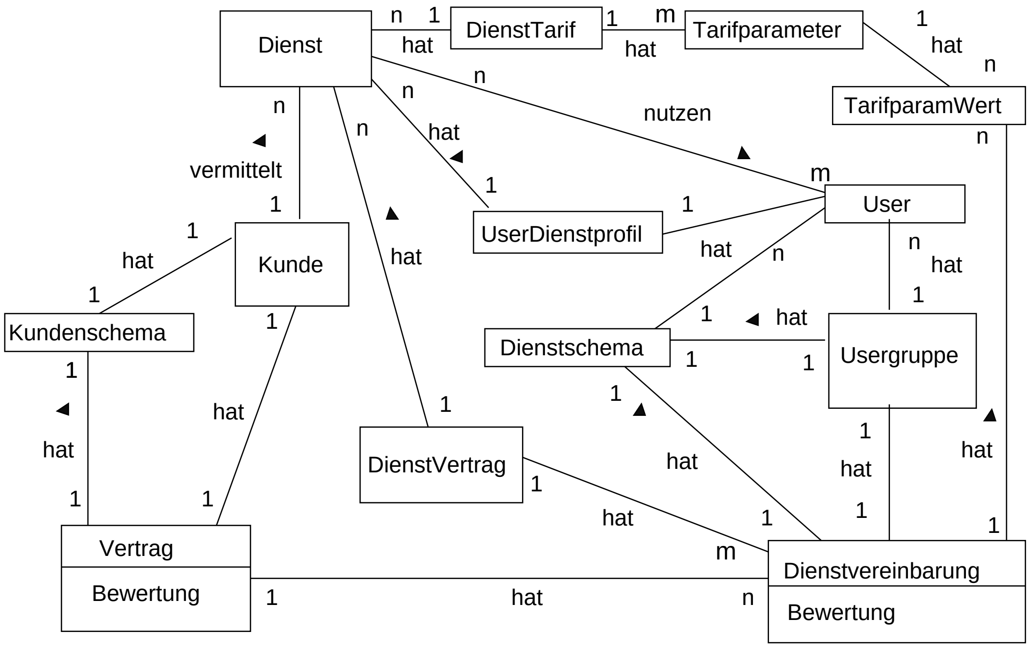

Abbildung 6.2: Klassendiagramm: Domänenklassen mit Hilfsklassen

Für die letzteren müssen Assoziationsklassen eingeführt werden, die die Verbindung bilden:

- DienstTarif für Dienst - Tarifparameter,
- DienstVertrag für Dienst - Dienstvereinbarung,
- UserDienstprofil für User - Dienst.

6.3 Suchkriterien

6.3.1 Einleitung

In Abschnitt 5.2 wurde analysiert, dass relevante Verträge durch Suchen nach NutzerDienstcharakteristik und Bewertung gefunden werden können. Nun soll ermittelt werden, wie diese Größen in geeignete Suchkriterien gefasst werden können. Um aufwändige Joins beim Suchen in der Datenbank zu vermeiden, werden Suchkriterien festgelegt, nach denen in den Tabellen *Vertrag* bzw. *Dienstvereinbarung* gesucht werden kann. Damit wird die Suche nach einem besten Vertrag optimiert und Anforderung 11 (Abschnitt 2.2) erfüllt.

Im Prototyp *Tariftool* können nicht alle möglichen Kundenprofile und Nutzerprofile berücksichtigt werden. Es wird eine Auswahl getroffen, die im Folgenden erläutert wird.

6.3.2 Bewertung

Zur Bewertung eignet sich ein Attribut, dessen Werte sich in eine Folge ordnen lassen, so dass jeder Wert genau einen Vorgänger bzw. einen Nachfolger hat. Dazu lassen sich z.B. ganze Zahlen aus einem begrenzten Wertebereich verwenden. Gleiche Bewertungen sind zugelassen.

Es ist nicht Aufgabe dieser Arbeit, Kriterien für die Berwertung von Verträgen aufzustellen. Dies liegt im Ermessen des Providers. Hier werden beispielhaft Werte zwischen 0 und 100 verwendet. -100 bezeichnet einen noch nicht bewerteten Vertrag/Dienstvereinbarung.

6.3.3 Kundenvektor

Der Kundenvektor enthält Eigenschaften des Kunden, die sich aus Sicht des Providers auf den Tarif auswirken können, z.B.

- Anzahl der Nutzer,
- Anzahl Niederlassungen im Inland,
- Anzahl Niederlassungen im Ausland,
- Bonität des Kunden.

Aus möglichen Wertebereichen für diese Daten wird ein *Kundenschema* abgeleitet, das eine Struktur für den *Kundenvektor* bildet:

- **Klasse Kundenschema:**
  Das *Kundenschema* stellt eine Schablone zur Vektordarstellung des Kundenprofils dar. Die beispielhaften Daten sind von Unternehmensgrößen in der BRD abgeleitet (Süddeutsche Zeitung, 2002). Die Assoziationen der Klasse sind in Abbildung 6.1 dargestellt.

Kundenschema
+kvID : eindeutiges Identifikationsattribut des Kundenvektors -kdID : eindeutiges Identifikationsattribut des Kunden -nutzerklasseGT50T: Kunde hat mehr als 50000 Nutzer -nutzerklasseGT10T: Kunde hat mehr als 10000 Nutzer -nutzerklasseGT5T: Kunde hat mehr als 5000 Nutzer -nutzerklasseLE5T: Kunde hat nicht mehr als 5000 Nutzer -nNiedlInGT50: Kunde hat mehr als 50 Niederlassungen im Inland -nNiedlInLE50: Kunde hat nicht mehr als 5 Niederlassungen im Inland -nNiedlAuslGT0: Kunde hat Niederlassungen im Ausland -nNiedlAusl0: Kunde hat keine Niederlassungen im Ausland -bonitaet: Bonität -reserve

Der Kundenvektor ist ein Abbild einer Instanz des *Kundenschemas* auf einen String. Für jedes `true` ist im String eine 1 gesetzt, die restlichen Stellen sind mit Nullen gefüllt. Dieser 10

Charakter lange Kundenvektor ist in der Tabelle *Vertrag* eingetragen und dient als Suchkriterium. Durch Änderung der Tabelle *Vertrag* lässt sich der Kundenvektor auf 15 Charakter erweitern (bei größeren Feldern lässt sich der Kundenvektor nicht mehr als String vergleichen). Entsprechen müsste die Benutzerschnittstelle geändert werden (Anforderung 8 in Abschnitt 2.2).

6.3.4 Nutzervektor

Insbesondere bei der Vielzahl von möglichen Diensten kann im Prototyp *Tariftool* nur ein Teil berücksichtigt werden. Für das Suchen nach vereinbarten Diensten in der Tabelle *Dienstvereinbarung* werden in der Literatur häufig genannte Dienste (s. Abschnitt 4.6) in einem *Dienstschema* aufgeführt:

- **Klasse Dienstschema:**
  Das Dienstschema stellt eine Schablone zur Vektordarstellung des Nutzerprofils dar. Die Bezeichnungen sind eindeutig und bedürfen keiner weiteren Erklärung. Die Assoziationen der Klasse sind in Abbildung 6.1 dargestellt.

Dienstschema
+dsID: eindeutiges Identifikationsattribut des Dienstschemas -filetransfer -email -multimedia-konferenz -e-commerce -video-on-demand -abonnement -setup/access -print-website -applicationA -download -videotelefon -newsgroup -fax -websurfing -webhosting -telefon -mP3 -jopac -internet-recherche -sonderdienst -reserveA

Der Nutzervektor ist ein Abbild einer Instanz der Klasse *Dienstschema* auf einen 15 Charakter langen String. Dienste, die der Nutzer nutzt (`true` in einer Instanz des *Dienstschemas*), sind mit einer 1 gekennzeichnet, der Rest mit 0. Die folgenden 5 Items können in einem Folgevektor *nutzervektor_2* gespeichert werden. D.h. dass 30 Dienste berücksichtigt werden können ohne Änderung des Suchprogramms. Eine weitere Erweiterung ist durch Hinzufügen eines *nutzervektor_3* in die Tabelle *Dienstvereinbarung* möglich; dazu müsste auch das Wissensnutzungsprogramm ergänzt werden. Der Nutzervektor ist in der Tabelle *Dienstvereinbarung* eingetragen und dient als Suchkriterium.

Die Liste von Diensten in der Tabelle *Dienstschema* kann erweitert und an die Erfordernisse des Providers angepasst werden; in der aktuellen Version des Tariftools enthält sie 20 Items (Anforderung 8 in Abschnitt 2.2). Da ein Vergleich von Arrays der Datenbank mit Strings nur bis zu einer begrenzten Länge möglich ist, sind bei Erweiterungen über 30 Items hinaus weitere Nutzervektoren in der Tabelle *Dienstvereinbarung* einzuführen.

Die Aufnahme von Sammeldiensten wie Serviceklassen, bei denen einer Gruppe von Diensten gemeinsame Tarifparameter zugewiesen werden, ist möglich. Dazu gehören z.B.:

- *Premium:* Diese Serviceklasse hat die höchste QoS; die höchste erforderliche Bandbreite wird reserviert, es dürfen keine Verzögerungen auftreten; abgerechnet wird nach Bandbreite und Dauer der Dienstleistung, die z.B. in einer Mediakonferenz für 5 Personen bestehen kann.

- *Assured:* Hier wird eine mittlere Bandbreite reserviert; die Serviceklasse ist für Übertragungen von Videos oder Webseiten geeignet. Abgerechnet wird nach Bandbreite und Dauer der Übertragung.

- *Priority:* Hier wird keine bestimmte Bandbreite reserviert, die verfügbare Bandbreite wird jeweils genutzt; Verzögerungen sind zugelassen, aber es dürfen keine IP-Pakete verworfen werden. Abgerechnet wird nach Bandbreite-Intervallen und Dauer der Übertragung in der jeweiligen Bandbreite.

- *Best Effort:* Hier wird mit einer Mindestbandbreite oder der verfügbaren Bandbreite mit *best effort* (wie z.Zt. im Internet) übertragen. Verlust von IP-Paketen wird in Kauf genommen.

6.4 Wissensrepräsentation: Entwurf des Datenbankschemas

6.4.1 Überblick

Wie bereits in Kapitel 5 erläutert wurde, werden nur die Basisdaten, d.h. die Domänenklassen und die Regelklasse, in die Datenbank abgebildet. Das Datenbankschema kann daher aus dem Klassendiagramm Abbildung 5.9 abgeleitet werden. Nur die Klasse Provider entfällt. Die Regelklasse hat keine Beziehung zu den Domänenklassen.
Die entsprechenden Klassendefinitionen sind in Abschnitt 5.7 aufgeführt. Die Objektklassen der Wissensbasis werden als Relationen (Entities) in die relationale Datenbank abgebildet ([brat90], [sau92], [mau97]). Die Objekt-IDs werden dabei zu Primärschlüsseln. Das sich ergebende Entity-Relationship-Diagramm ist in Abbildung 6.3 dargestellt.

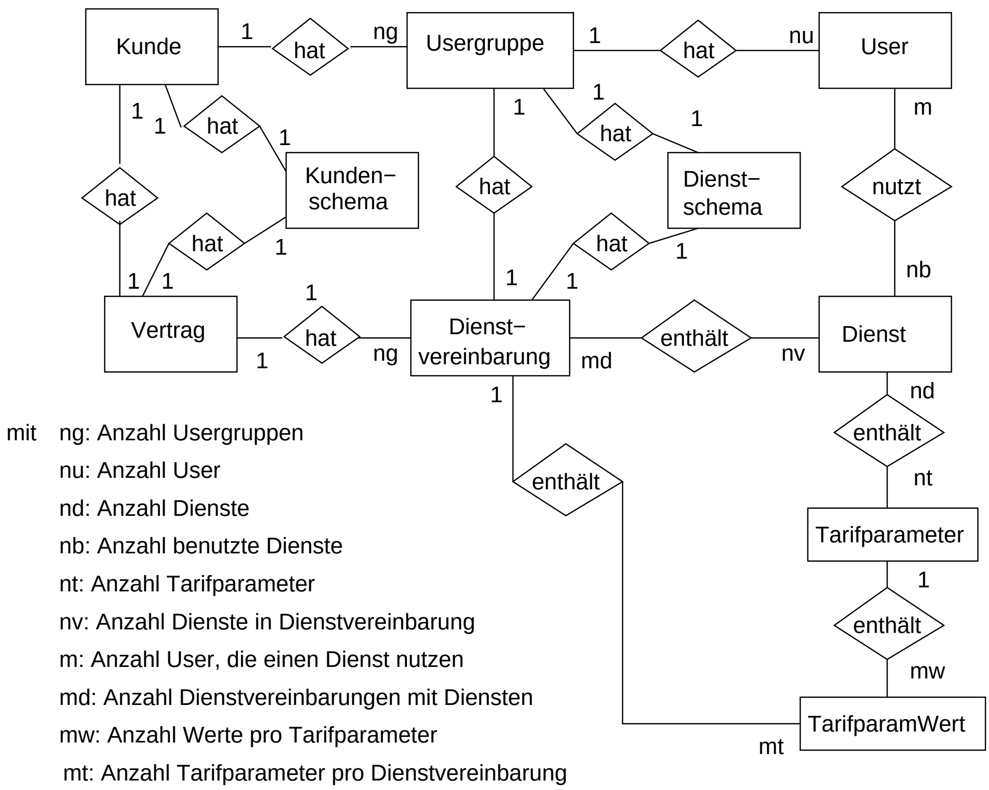

Abbildung 6.3: Entity-Relationship-Diagramm

Bei Relationen, die eine 1:1 - Beziehung zueinander haben, muss eine von beiden den Primärschlüssel der anderen als Fremdschlüssel enthalten. Möglich ist auch, dass beide den Primärschlüssel der anderen als Fremdschlüssel enthalten. Haben Relationen eine 1:n - Beziehung zueinander, dann muss die Relation der n-Seite den Primärschlüssel der anderen als Fremdschlüssel enthalten. Bei n:m- Beziehungen ist das Einrichten einer Zwischenrelation (analog zu Assoziationsklassen) erforderlich, hier z.B. bei der User-Dienst-, der Dienst-Dienstvereinbarung- und der Dienst-Tarifparameter-Assoziation (vgl. Abbildung 6.3).

Die einfachen Attribute dieser Relationen entsprechen den in Abschnitt 5.7 aufgeführten Attributen. Die Datentypen werden so umgesetzt, wie sie in Tabelle 6.1 (nach [brj99], [dehn99], [whfi00], [hart01]) aufgeführt sind.

UML	Java	JDBC	Datenbank (PostgreSQL)
String	String	CHAR, VARCHAR	char(x), varchar(x)
Integer	int	INTEGER	integer
Long	long	BIGINT	int8
Boolean	boolean	BIT	bool (1 Byte)
Date	java.sql.date	DATE	date (Ansi-SQL 4 Byte)
Time	java.sql.time	TIME	time

Tabelle 6.1: Datentypen in UML, Java, JDBC und Datenbank PostgreSQL

Zusammengesetzte Attribute (arrays) werden nicht in die Relation übernommen. Bei der Instantiierung von Klassen müssen gegebenenfalls weitere Tabellen ausgewertet werden, die zum Hauptobjekt Beziehungen haben, z.B. bei `Kunde` auch die zugehörigen Usergruppen. Da die Bezeichnung `User` in der PostgreSQL-Datenbank als Tabellenname nicht zulässig ist, wird die Klasse `User` auf die Relation `Nutzer` abgebildet, die Klasse `UserDienstprofil` auf `Nutzerprofil`. Da einige Tabellen Referenzen auf andere Tabellen benötigen, erhalten alle Tabellen einen expliziten Primärschlüssel. Die Relationen (Tabellen) der *tarif-* Datenbank und ihre Fremdschlüssel sind in der folgenden Tabelle 6.2 aufgelistet.

Relation	Fremdschlüssel	Fremdschlüssel aus Relation
Kunde	vt-id	Vertrag
Kundenschema	kd-id	Kunde
Usergruppe	kd-id, dv-id	Kunde, Dienstvereinbarung
Userdienstprofil	di-id, us-id	Dienst, Nutzer
Nutzer	gr-id	Usergruppe
Vertrag	kd-id	Kunde
Dienstvereinbarung	vt-id, gr-id	Vertrag, Usergruppe
Dienst	-	-
Dienstschema	gr-id	Usergruppe
DienstTarif	di-id, tp-id,	Dienst, Tarifparameter
DienstVertrag	di-id, dv-id, gr-id	Dienst, Dienstvereinbarung, Usergruppe
Tarifparameter	-	-
Tarifparamwert	dv-id, tp-id, di-id	Dienstvereinbarung, Tarifparameter, Dienst
Regel	-	-

Tabelle 6.2: Relationen und enthaltene Fremdschlüssel

6.4.2 Struktur der Relationen in der PostgreSQL-Datenbank

Die Relationen in der relationalen Datenbank PostgreSQL werden von den Klassen abgeleitet, die in Abschnitt 5.7.3 und 5.7.4 aufgeführt sind. Um aufwändige Joins bei Abfragen zu vermeiden, werden einige Fremdschlüssel aufgeführt, die an sich nicht unbedingt erforderlich wären.

- **Kunde:**

```
id integer primary key,
kName varchar(20),
adressse varchar(30),
telefon varchar(16),
nutzerklassenwert integer,
nNiederlassInland integer,
nNiederlassAusland integer,
kWert integer
```

- **Nutzer (Klasse User):**

```
id integer primary key,
gr_id integer references Usergruppe,
userName varchar(20),
terminalID varchar(20),
terminaltype varchar(20),
standort varchar(20),
aussendienst bool
```

- **Usergruppe:**

```
id integer primary key,
kd_id integer references Kunde,
gName varchar(20)
```

- **Nutzerprofil (Zwischenrelation UserDienstprofil):**

```
id integer primary key,
us_id integer references User,
di_id integer references Dienst,
gr_id integer references Usergruppe
```

- **Dienst:**

```
id integer primary key,
dienstName varchar(20),
dienstbeschreibung varchar(30),
subdienstvektor varchar(20)
```

- **DienstTarif (Zwischenrelation):**

```
id integer primary key,
di_id integer references Dienst,
tp_id integer references Tarifparameter
```

- **Tarifparameter**

```
id integer primary key,
tName varchar(20),
tpBeschreibung varchar(30),
abrechnungsEinheit varchar(10),
subdienst varchar(20),
ds_id integer references Dienstschema,
di_id integer references Dienst,
abrechnungszeitraum varchar[5]
```

- **Vertrag:**

```
id integer primary key,
kd_id integer references Kunde,
regressbed varchar(10),
vertragsabschluss date,
kuendigung date,
beginLeist date,
endeLeist date
kp_id integer references Kundenprofil,
ks_id integer references Kundenschema,
kd_profil char(10), //kundenvektor
wert integer,
w_datum date
```

- **Dienstvereinbarung:**

```
id integer primary key,
vt_id integer references Vertrag,
gr_id integer references Usergruppe,
ds_id integer references Dienstschema,
nutzervektor char(15),
nutzervektor_2 char(15),
wert integer,
w_datum date
```

- **DienstVertrag (Zwischenrelation):**

```
id integer primary key,
di_id integer references Dienst,
dv_id integer references Dienstvereinbarung,
gr_id integer references Usergruppe
```

- **TarifparamWert:**

```
id integer primary key,
dv_id integer references Dienstvereinbarung,
tp_id integer references Tarifparameter,
preisfaktor integer,
anzeinh integer,
kosten integer,
di_id integer references Dienst
```

- **Kundenschema:**

```
id integer primary key,
nutzerklasseGT50T bool,
nutzerklasseGT10T bool,
nutzerklasseGT5T bool,
nutzerklasseLE5T bool,
nNiedlInGT50 bool,
nNiedlInLE50 bool,
nNiedlAuslGT0 bool,
nNiedlAusl0 bool,
bonitaet bool
```

- **Dienstschema:**

```
id integer primary key,
filetransfer bool,
email bool,
multimediaConf bool,
eCommerce bool,
videoOnDemand  bool,
abonnement bool,
setup bool, (access)
printWebsite bool,
applicationA bool,
download bool,
videotelefon bool,
newsgroup bool,
fax bool,
websurfing bool,
```

```
webhosting bool,
telefon bool,
mP3 bool,
jopac bool,
internetRecherche bool,
sonderdienst bool,
reserveA bool
```

- **Regel:**

```
id integer primary key,
zustand integer,
aktion_1 integer,
text_1 varchar(15),
aktion_2 integer,
text_2 varchar(15)
```

6.5 Wissenserwerb

6.5.1 Einführung

Nachdem die Wissensrepräsentation nun festgelegt ist, kann nach dem Entwicklungs- und Lebenszyklus des Wissensbasierten Systems (s. 3.2) der Wissenserwerb untersucht werden. Mantz ([msu87]) schlägt vor, die Wissenserwerbkomponente nicht ins Problemlösungssystem zu integrieren, sondern als gesondertes Programm zu implementieren, da der Wissensnutzer meist nicht berechtigt ist, an der Wissensbasis etwas zu ändern. Dies dürfte auch bei dem vorliegenden Problem der Fall sein, wird aber beim Prototyp des Tariftools nicht berücksichtigt.

- **Akquisition des Basiswissens**
  Hier ist vor der Einführung des Tariftools eine große Menge von Daten für alle Relationen einzubringen. Im Fall des Prototyps geht das am einfachsten im Batchmode oder direkt über Datenbank-SQL-Kommandos (Abschnitt 7.5).

- **Bewertung eines neuen Vertrags**
  Dieser Teil kann in die Problemlösung integriert werden.

- **Aktualisieren von Basisdaten**
  Auch dieser Teil kann in die Problemlösung integriert werden.

Kunden- und Nutzerdaten sind für diesen Prototyp beliebige Testdaten, die aber einen gewissen Bezug zur realen Welt der Unternehmen haben sollen, die Kommunikationsdienste outsourcen. Wichtig ist es, geeignete Werte für die Tarifparameter zu finden.

Im Rahmen der vorliegenden Arbeit ist es nicht möglich, alle in der Systemanalyse aufgeführten Parameter zu berücksichtigen. Es muss eine angemessene Auswahl getroffen werden. Diese wird in den folgenden Abschnitten erläutert.

6.5.2 Auswahl repräsentativer Dienste für den Prototyp

In Abschnitt 4.6 wurden in der Literatur häufig erwähnte Dienste aufgelistet. Davon wird eine Auswahl getroffen, die Dienste mit mehr und mit weniger Dienstgüteanforderungen umfasst. (Für Regressionsbedingungen wird im Tariftool nur ein Dummy-Kürzel angegeben, hierfür müsste der Provider eine entsprechende Liste anlegen.)

1. Filetransfer
2. Email
3. Multimedia-Konferenz
4. E-Commerce
5. Video-on-demand (Ausbildung)
6. Abonnement
7. Access (Setup)

6.5.3 Auswahl repräsentativer Tarifparameter

In der Auswertung der Literaturrecherche wurden in den Abschnitten 4.6, 4.8 und 5.4.3 Tarifparameter beschrieben, die als wichtigste genannt waren. Von diesen werden folgende Tarifparameter für den Prototyp Tariftool ausgewählt:

```
1, 'Dauer bei lt 64kbit/s',  'cent/hr'
2, 'Dauer lt 1 Mbit/s ',     'cent/min'
3, 'Dauer ge 1 Mbit/s ',     'cent/sec'
4, 'Volumen le 100 Kbyte',   'cent/kByte'
5, 'Volumen le 1 Mbyte',     'cent/kByte'
6, 'Volumen gt 1 Mbyte',     'cent/Mbyte'
7, 'QoS_Faktor_1 best ',     '*4'
8, 'QoS_Faktor_2  ',         '*3'
9, 'QoS_Faktor_3  ',         '*2'
10, 'QoS_Faktor_4 b. eff',   '*1.5'
11, 'Tageszeit_Faktor ',     '*3'
12, 'Abonnement',            'euro/mt'
13, 'Access/Setup',          'euro/Ac'
```

6.5.4 Tarifparameter der ausgewählten Dienste

Welche Tarifparameter für die ausgewählten Dienste in Frage kommen, ist in den Abschnitten 4.6 und 5.4.3 bereits erörtert worden. Von diesen werden folgende für den Prototyp Tariftool ausgewählt (Tabelle 6.3, Zahlenwerte s. vorigen Abschnitt).

Dienst	Tarifparameter-1	Tarifparameter-2	Tarifparameter-3
FTP	5	1	
Email	1	11	
MultimediaKonferenz	7	11	6
Ecommerce	8	11	4
VideoOnCommand	9	11	2
Abonnement	12		
Access/Setup	13		

Tabelle 6.3: Tarifparameter der Tariftool-Dienste

Service	Charge (\$ per Mbits s^{-1})
rt-VBR	0.2
CBR	0.1
ABR/nrt-VBR	0.005
UBR	0.0002

Tabelle 6.4: Serviceklassen in ATM-basierten Netzen

6.5.5 Defaultwerte für Tarifparameter

Die Tarifparameterwerte sollen bei Vertragsverhandlungen zwischen Kunde und Provider abgestimmt werden. Für fiktive Verträge zum Programmtest werden Defaultwerte eingesetzt, die als plausibel betrachtet werden könnten. In der Literatur werden Werte für Tarifparameter angegeben, die von Caffrey ([caff99]) sowie Karsten u.a. ([karo97]) als angemessen betrachtet werden. Werte für Services in ATM-basierten Netzen zeigt Tabelle 6.4. Ferner wird ein allgemeines Beispiel für Kosten aufgeführt (Tabelle 6.5). Der Dienst Multimediakonferenz kann in den Service rt-VBR (real time, variable Bitrate) eingeordnet werden. Für die übrigen werden plausible Werte angesetzt. Für das Tariftool kann danach folgender Ansatz gemacht werden (s. Tabelle 6.6, * darin bedeutet multiplikativer Faktor).

Service	Charge (\$ per Mbits s^{-1})
Videokonf. 5 Personen, 64kbit s^{-1} (VBR)	0.2 (4 \$/minute)
Video-on-demand (UBR)	0.0002 (ca. 1 \$/hour)

Tabelle 6.5: Kostenbeispiel

Nr.	Tarifparameter	Wert	Dimension
1	Dauer bei lt 64kbit/s	100	cent/hr
2	Dauer lt 1 Mbit/s	300	cent/min
3	Dauer ge 1 Mbit/s	5	cent/sec
4	Volumen le 100 Kbyte	300	cent/kByte
5	Volumen le 1 Mbyte	400	euro/kByte
6	Volumen gt 1 Mbyte	40	cent/Mbyte
7	QoS-Faktor-1 best	4	*
8	QoS-Faktor-2	3	*
9	QoS-Faktor-3	2	*
10	QoS-Faktor-4 b. eff	1.5	*
11	Tageszeit-Faktor	3	*
12	Abonnement	50	euro/mt
13	Access/Setup	1	euro/Ac

Tabelle 6.6: Default-Tarifparameterwerte

6.5.6 Bewertung der Verträge/Dienstvereinbarungen

Um Verträge bewerten zu können (Anforderung 3 in Abschnitt 2.2), müssen zunächst die aus den vereinbarten Diensten und Tarifparameterwerten resultierenden Abrechnungen erstellt werden. Diese können in einer Erweiterung des Tariftools aus den in den entsprechenden Feldern gespeicherten Werten berechnet werden, wie im Folgenden kurz beschrieben wird.

Zur Ermittlung der für den Kunden anfallenden Kosten ist der Eintrag der Anzahl gemessener Abrechnungseinheiten pro Dienstvereinbarung, Dienst und Tarifparameter in der Tabelle *TarifparamWert* erforderlich. Der in der Tarifverhandlung zwischen Kunden und Provider-Vertreter ausgehandelte Tarifparameterwert tpw hängt in diskreter Weise von der Art des zugehörigen Tarifparameters tp, z.B. von dessen Dimension (z.B. sec oder hr), dem zugeordneten Dienst di und eben der geschlossenen Dienstvereinbarung dv ab. Die Dienstvereinbarung ist einer Usergruppe des Kunden zugeordnet und ein Teil des Vertrags:

$$tpw = tpw_{tp,di,dv}$$

Sei $A_{tp,di,dv}$ die Anzahl der für die Dienstvereinbarung dv, den Dienst di und den Tarifparameter tp gemessenen Abrechnungseinheiten, so berechnen sich die Kosten K des Kunden zu:

$$K = \sum_{dv} \sum_{di} \sum_{tp} tpw_{tp,di,dv} \times A_{tp,di,dv}$$

Aus der Differenz zum erwarteten Gewinn ergibt sich ein Maß für die Bewertung. Sie ist im Prototyp des Tariftools auf ein Intervall ganzer Zahlen zwischen -99 und 100 beschränkt. -100 bedeutet, dass noch keine Bewertung vorliegt.

Zum Vergleich der Ergebnisse mit den Kosten des Providers sollten noch die vereinbarten Regressbedingungen bewertet werden.

6.6 Struktur des Programmsystems Tariftool

6.6.1 Überblick

Nach dem WBS-Entwicklungsplan (3.2) ist als Nächstes die Problemlösungsstrategie festzulegen. Die Grundlagen wurden schon in Abschnitt 5.7.5 und 5.8 erörtert. Das Problemlösungssystem wird in der Programmiersprache *Java* implementiert. Schreiben, Lesen und Ändern der Basisklassen wie `Kunde`, `Vertrag` und `Dienstvereinbarung` werden in das Tariftool integriert. Die Klassendiagramme sind in Abschnitt 5.7.5 dargestellt. Benötigt werden:

- Komponente zum Start des Tariftools, zum Dialog mit dem Benutzer und zum Connect an die Datenbank
- Wissensnutzungs-Komponente zum Ermitteln des besten Vertrags für ein Nutzer-/Kundenprofil
- Wissenserwerb-Komponente zum Schreiben und Ändern von Tabellen der Datenbank.

6.6.2 Package-Struktur des Tariftools

Für das Tariftool wird folgende Package-Struktur gewählt:

- `package tarif`: Dieses Grundpackage enthält die Klasse Tariftool mit dem Hauptprogramm für den Start des Tariftools,
- `package tarif.suchen`: Dieses Package enthält die Klasse Vertreter zum Suchen des besten Vertrags,
- `package tarif.erwerb`: Dieses Package enthält die Klassen Account und Admin zur Verwaltung der Basisdaten,
- `package tarif.daten`: Dieses Package enthält die Klassen der Basisdaten (d.h die Domänenklassen und die Regelklasse s. 5.7.3 und 5.7.4).

Die Package-Struktur ist in Abbildung 6.4 dargestellt.

6.6.3 Package tarif

Das Package `tarif` enthält die Klasse `Tariftool`. Diese importiert:

`java.sql.*; java.io.*; java.util.*;`

Die Klasse `Tariftool` enthält:

- das Hauptprogramm `main`:

  ```
  public static main (String argv[ ])
  Funktion: Aufruf der Routine zur Datenbankanbindung (getConnection)
            und der Routine zum Dialog mit dem Benutzer (tdialog)
  ```

- die Routine zur Datenbankanbindung `getConnection`
- die Routine zum Dialog mit dem Benutzer `tdialog` :
  Beschreibung s. Abschnitt 7.7 Benutzerschnittstelle

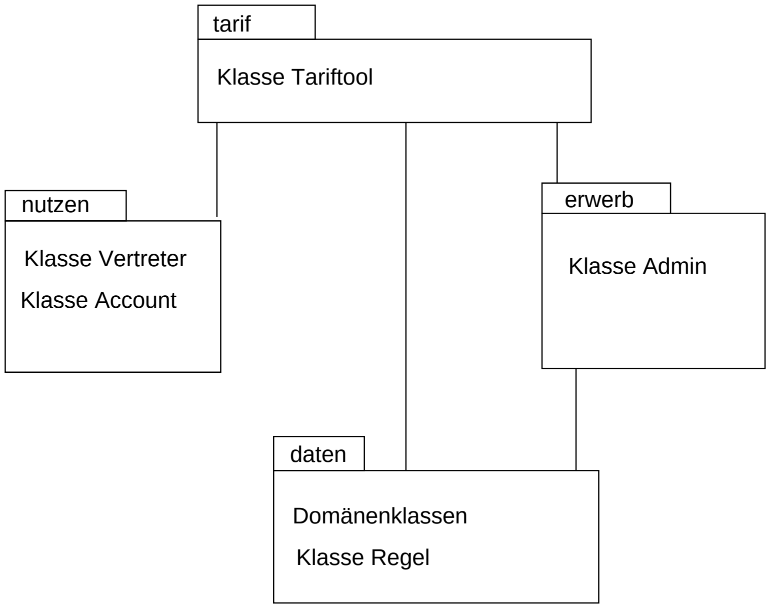

Abbildung 6.4: Package-Struktur des Tariftools

```
public static void tdialog (Statement stmt)
       throws IOException,SQLException
Parameter: stmt: Statement fuer die Datenbankaufrufe
Funktion: Dialog mit dem Benutzer
          Aufruf von Vertreter-, Account- und Adminroutinen
```

6.6.4 Package tarif.suchen

Das Package `tarif.suchen` enthält die Klasse `Vertreter`, die public sein muss.

- **Vertreter**:
  Die Klasse `Vertreter` importiert:

  `java.sql.*; java.io.*; java.util.*;`

  Die Klasse `Vertreter` enthält:

  - Routine `findVt`

```
public static void findVt(String kdProfil, String usProfil, String us1Profil,
       Statement stmt)
       throws IOException,SQLException
Parameter: kdProfil: Kundenvektor
           usProfil: Nutzervektor
           us1Profil: Nutzervektor_2 (Erweiterung)
           stmt: Statement fuer die Datenbank-Aufrufe
Funktion: Beste(n) Vertrag/Dienstvereinbarung finden
```

Beschreibung:
In dieser Prozedur werden aus den Relationen *Regel*, *Vertrag* und *Dienstvereinbarung* Daten gelesen. In der Relation *Vertrag* wird nach dem geforderten Kundenvektor gesucht, in *Dienstvereinbarung* nach dem Nutzervektor, d.h. nach den vereinbarten Diensten für die Usergruppe des Kunden. Soll sowohl nach Kunden- als auch nach Nutzervektor gesucht werden, wird aus *Regel* ermittelt, wonach zuerst gesucht wird (aktion_1). Wird der gesuchte Vektor nicht gefunden, so wird versucht, einen Vertrag oder eine Dienstvereinbarung zu finden, bei der ein Item weniger vorliegt.

– Routine `reduV1`:

```
public static int reduV1(int j, int zustand, String kdProfil)
Parameter: j: Itemposition
           zustand: Zustand des Find-Prozesses
           kdProfil: Kundenvektor
Funktion: Die Anzahl der Items in Kundenvektor wird um 1 reduziert;
          Das Ergebnisprofil wird in eine Klassenvariable vstr geschrieben.
returns:  Restanzahl der Items im Kundenvektor (j-1) oder -1 bei Ende
```

– Routine `reduD1`:

```
public static int reduD1(int j, int zustand, String usProfil, String
           us1Profil)
Parameter: j: Itemposition
           zustand: Zustand des Find-Prozesses
           usProfil: Nutzervektor
           us1Profil: Nutzervektor_2
Funktion: Die Anzahl der Items in Nutzervektor bzw. im Nutzervektor_2 wird
          um 1 reduziert; Das Ergebnisprofil wird in eine Klassenvariable
          dstr  bzw. ustr geschrieben.
returns:  Restanzahl der Items im Nutzervektor (j-1) oder -1 bei Ende
```

6.6.5 Package tarif.erwerb

Das Package `tarif.erwerb` enthält die Klassen Account und Admin, die public sein müssen.

- **Account:**
  Die Klasse `Account` importiert:

  `java.sql.*; java.io.*; java.util.*;`

  Die Klasse `Account` enthält:

  – Routine `setWert`:

```
public static void setWert(String tableName, int ID, int wert,
           Statement stmt)  throws SQLException, IOException
Parameter: tableName : vertrag oder dienstvereinbarung
           ID : Datensatz-ID
           wert: Bewertung des Vertrags bzw. der Dienstvereinbarung
```

```
             stmt: Statement fuer die Datenbank-Aufrufe
Funktion: Der Vertrag oder die  Dienstvereinbarung, die in tableName
          angegeben ist, wird mit "wert" bewertet. Das Tagesdatum
          wird ermittelt. Die Daten werden mittels
          UPDATE in die Datenbank geschrieben, das Datum in der Form
          mm/tt/jjjj.
```

- **Admin:**
  Die Klasse `Admin` importiert:

  `java.sql.*; java.io.*; java.util.*;`

  Die Klasse `Admin` enthält die Routinen:

  - `tLesen`·

    ```
    public static void tLesen(String tableName, int ID, Statement stmt)
         throws SQLException, IOException
    Parameter: tableName: Name der Tabelle (Relation) in der Datenbank
               ID: Identifikation des Datensatzes (primary key)
               stmt: Statement fuer die Datenbank-Aufrufe
    Funktion:  Datensatz aus der Datenbank lesen
    ```

  - `tUpdate`:

    ```
    public static void tUpdate(String tableName, int ID, String attribut,
             String awert, Statement stmt)
             throws SQLException, IOException
    Parameter: tableName: Name der Tabelle (Relation) in der Datenbank
               ID: Identifikation des Datensatzes (primary key)
               attribut: zu aenderndes Attribut (Spaltenname)
               awert: neuer Attributswert
               stmt: Statement fuer die Datenbank-Aufrufe
    Funktion: Datensatz in der Datenbank aktualisieren
    ```

  - `tSchreiben`:

    ```
    public static void tSchreiben(String tableName, Statement stmt)
             throws SQLException, IOException
    Parameter: tableName: Name der Tabelle (Relation) in der Datenbank
               stmt: Statement fuer die Datenbank-Aufrufe
    Funktion: Datensatz in die Datenbank schreiben
    ```

6.6.6 Package tarif.daten

Dieses Package enthält die in Abschnitt 5.7.3 und 5.7.4 beschriebenen Klassen der Wissensbasis. Die Klassen müssen public sein.

6.7 Zusammenfassung

Für den Prototyp Tariftool wurden einschränkende Design-Entscheidungen getroffen. Um die Suche nach besten Verträgen oder Dienstvereinbarungen effizient zu gestalten, wurden Suchkriterien eingeführt, die den Kunden und seine vereinbarten Dienste charakterisieren. Die Struktur der Wissensrepräsentation für das Tariftool wurde festgelegt und die einzelnen Tabellen entsprechend der Vorgabe durch die im Kapitel 5 definierten Objektklassen deklariert. Falls erforderlich, wurden den Tabellen Fremdschlüssel zugeordnet oder Assoziationsklassen abgeleitet. Der Zusammenhang wurde durch ein Entity-Relationship-Diagramm veranschaulicht. Da der Prototyp Tariftool nicht den gesamten Umfang an Tarifdaten bewältigen kann, wurden repräsentative Werte ausgewählt. Für das Tariftool wurde eine Package-Struktur entworfen.

Kapitel 7

Prototypische Implementierung und Test

Durch prototypische Implementierung wird die Funktionsfähigkeit des zuvor beschriebenen Modells des Tariftools nachgewiesen. Nach der Beschreibung von Architektur und Entwicklungsumgebung wird das Erzeugen von Tabellen (Relationen) in der PostgreSQL-Datenbank `tarif` *erläutert. Ferner wird die Schnittstelle zum Benutzer festgelegt. Zuletzt werden Erweiterungsmöglichkeiten und Maßnahmen zur Gewährleistung der Portabilität des Tariftools beschrieben.*

7.1 Einleitung

Nachdem in Kapitel 5 analysiert wurde, was das Tariftool leisten soll, und in Kapitel 6 herausgearbeitet wurde, in welcher Weise dies geschehen soll, wird nun die Implementierung des Tariftools durchgeführt. Da es sich um einen Prototyp handelt, können nicht alle Aspekte Berücksichtigung finden; es muss eine Auswahl der Parameter getroffen werden. Nach Beschreibung der Architektur des Tariftools wird angegeben, wie die Relationsstrukturen zu erzeugen sind. Anschließend wird die Benutzerschnittstelle festgelegt.

7.2 Architektur des Systems

Das Tariftool, das die zuvor in Kapitel 5 und 6 beschriebenen Funktionen hat, muss folgende Aufgaben erfüllen:

- Eingaben vom Benutzer annehmen und bearbeiten,
- Ausgaben an den Benutzer machen,
- Datensätze der Datenbank lesen, ändern oder schreiben.

Das Programmsystem `Tariftool` bewältigt die Aufgaben `Ein/Ausgabe`, `Inferenz`, `Wissenserwerb` und `Datenbankzugriff`. Es setzt auf der Java-Datenbankschnittstelle (JDBC-API) auf. Die Architektur wird nach UML ([Oest 99] u.a) entworfen. Sie ist in Abbildung 7.1 dargestellt.

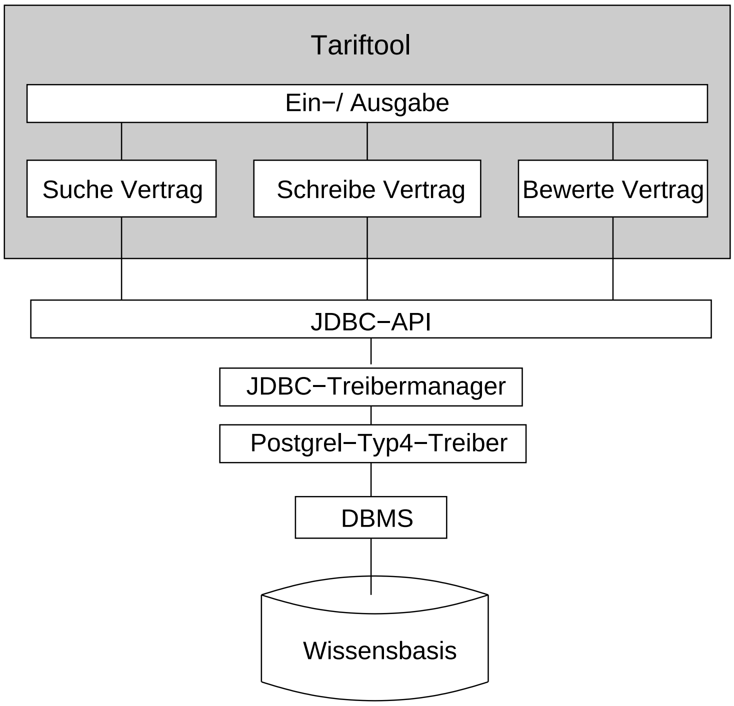

Abbildung 7.1: Architektur des Tariftools in der JDBC-Umgebung

7.3 Entwicklungsumgebung

Das System wird in Linux-Umgebung entwickelt. Als Programmiersprache wird Java verwendet ([argo96], [flan97]), [bloc02], [leca02]).

Als Datenbank steht PostgreSQL zur Verfügung ([pgrs98], [momj01], [wodr02], [hart01], [momb01]). Für die Entwicklung des Tariftools ist die Datenbank `tarif` für den Benutzer `kalixe` eingerichtet.
PostgreSQL stellt für JDBC einen Typ-4 -Driver (d.h. ganz in Java implementiert) zur Verfügung. Das Archiv postgresql.jar des PostgreSQL-drivers muss in den CLASSPATH aufgenommen werden. Die Verbindung mit dem Javaprogramm wird durch eine Property-Datei ermöglicht. Die Schnittstelle zu Java bildet JDBC. Anleitung zur Programmierung findet man in [dehn99], [hobb97], API [whfi00], Jepson [jeps97], [sasa00], [kueh99], [hoco00].

7.4 Erzeugen der Relationsstrukturen

Die in Kapitel 6 Abschnitt 6.4.2 aufgeführten Relationen (Tabellen) des Tariftools in der PostgreSQL-Datenbank werden mit dem SQL-Kommando `CREATE TABLE` eingerichtet:

```
CREATE TABLE <tabname>(
                  id integer primary key,
                  ...,
                  );
```

Fremdschlüssel werden eingetragen (z.B.) mit :

```
kd_id integer references Kunde,
```

Die zu referenzierende Tabelle muss schon eingerichtet sein. Daher sollten die Tabellen in der Reihenfolge

- `Kunde`,
- `Usergruppe`,
- `Nutzer`,
- `Kundenschema`,
- `Dienstschema`,
- `Vertrag`,
- `Dienstvereinbarung`,
- `Dienst`,
- `Tarifparameter`,
- `Tarifparamwert`,
- `DienstTarif`,
- `DienstVertrag`,
- `Userdienstprofil`,
- `Regel`

eingerichtet werden.

Es können auch nachträglich weitere Tabellenspalten (Attribute) eingerichtet werden mit (z.B.):

```
ALTER TABLE <tabname> ADD vt_id integer references vertrag;
```

Attribute können auch umbenannt werden. Es ist aber nicht möglich, Attribute (Spalten) zu entfernen. Gegebenenfalls muss die Tabelle gelöscht (`DROP TABLE`) und neu eingerichtet werden ([hart01]).

7.5 Schreiben der Testdaten

Beim Wissenserwerb vor Einsetzbarkeit des Tariftools und zum Testen ist es zweckmäßig, die bereits bekannten Daten in Dateien zu schreiben und mit dem Programm `InsTab` (mit der Property-Datei InsTab.prp) einzulesen.

Die Daten eines Satzes müssen in einer Zeile einer Eingabedatei stehen, durch Kommata getrennt, z.B. in einer Datei mit Namen `Kunde.dai` für Einträge in die Tabelle `Kunde`:

```
4, 'Kunz', 15, 4, 'Bonn', '554433', 25000, 20, 3, 'true'
5, 'Sommer', 25, 8, 'Mainz', '254433', 600, 5, 0, 'false'
```

Zum Schreiben von einzelnen Datensätzen verwendet man das SQL-Kommando `insert` z.B. wie folgt:

```
tarif=> insert into Kunde values (
tarif(> 4, 'Mair',15, 4, 'Berg' '4441016',5000,10,0,'true');
INSERT 43520 1
```

Dabei muss beachtet werden, dass die Attribute in der richtigen Reihenfolge eingegeben werden.

Aktualisieren der Basisdaten ist mit dem SQL-Kommando `update` möglich ([hart01]) z.B.:

```
tarif=> update vertrag set wert = 50;
```

7.6 Installation und Test

Das Tariftool setzt das Datumformat 'ISO, US' voraus, d.h., dass Eingaben nach US-Norm als 'mm/dd/yyyy' interpretiert werden und Ausgaben im ISO-Format 'YYYY-MM-DD' erfolgen. Dieses Format ist auf der PostgreSQL-Testdatenbank bereits eingestellt. Der Datenbank-Administrator kann den Benutzer berechtigen, das Datumformat mittels der Umgebungsvariablen DATESTYLE einzustellen, falls es in einer anderen Form vorliegt. Den eingestellten Wert erfährt man nach Hartwig ([hart01]) mit dem Kommando:

```
SHOW DATESTYLE
```

Als Ergebnis erhält man (wie hier z.B.):

```
tarif=> show datestyle;
NOTICE:  DateStyle is ISO with US (NonEuropean) conventions
SHOW VARIABLE
```

Die Umgebungsvariable wird passend gesetzt mit:

```
SET DATESTYLE = 'ISO, US';
```

Das Tariftool benötigt zum Aufbau einer Verbindung zur Datenbank Informationen über Driver, Datenbanknamen, Namen und Passwort des Benutzers. Diese Werte können über eine Datei `Tariftool.prp` eingelesen werden. Durch die Verwendung einer solchen Datei ist das Tariftool portierbar, ohne dass Eingriffe in den Programmcode gemacht werden müssen. Im vorliegenden Fall enthält die Datei für Driver, Datenbank und Benutzer die Daten:

```
org.postgresql.Driver
jdbc:postgresql:tarif
kalixe
erika
```

Der CLASSPATH für die Anbindung des PostgreSQL-Drivers wurde gesetzt:

```
setenv CLASSPATH
.:/usr/local/mmm/dist/pgsql/current/share/java/postgresql.jar
```

Die Tabellen in der Datenbank wurden wie in Abschnitt 6.4.2 beschrieben eingerichtet und die im Abschnitt 6.5 ausgewählten Daten eingebracht.

Die Sourcen und Klassen des Tariftools befinden sich in den Verzeichnissen:

- tarif
- tarif/erwerb
- tarif/suchen
- tarif/daten

Der Programmstart erfolgt aus dem Verzeichnis, das *tarif* übergeordnet ist, mit:

```
/usr/lib/SunJava2/bin/java tarif/Tariftool
```

Die Tests wurden am pcheger3-Computer des NM-Rechnerparks unter der Kennung *kalixe* durchgeführt. Es stand die Linux-Version SuSE 7.0, die Java-Version 1.2.2 und die PostgreSQL-Version 7.2.1 zur Verfügung.

Das Ergebnis der Tests zeigte, dass der Prototyp des Tariftools im Rahmen seiner Möglichkeiten ein brauchbares Werkzeug ist.

7.7 Benutzerschnittstelle

Nach dem Start des Tariftools kann der Benutzer die gewünschte Funktion auswählen. Durch die Menüauswahl und Einzelabfrage gewünschter Dienste und Kundenprofile ist die Eingabe benutzerfreundlich gestaltet und der Benutzer benötigt keine besonderen Fachkenntnisse (Anforderung 13 und 14 aus Abschnitt 2.2).

7.7.1 Funktionsauswahl

Dazu erscheint folgendes Menü:

```
1. Besten Vertrag suchen
2. Vertrag bewerten
3. Datensatz schreiben
4. Datensatz aktualisieren
5. Datensatz lesen
0. Tariftool beenden
```

Die Auswahl erfolgt durch Eingabe der Funktionsnummer.

7.7.2 Besten Vertrag suchen

Eingabe:
Die Items des Kunden- und des Dienstschemas werden ausgegeben, sodass der Benutzer sich den gewünschte Kundenvektor bzw. den Nutzervektor zusammenstellen kann. Ist in der

Tabelle Regel aktion_2 auf 1 gesetzt, können bis zu 15 Items für den Nutzervektor (Reihenfolge wie im Dienstschema in Abschnitt 6.3) ausgewählt werden. Ist aktion_2 auf 2 gesetzt, können auch die folgenden Items aus dem Dienstschema gewählt werden. Standardmäßig ist im Prototyp aktion_2 auf 0 gesetzt, d.h. die ersten 7 Items aus dem Dienstschema können gewählt werden, dies entspricht dem aktuellen Inhalt der Datenbank.

Ausgabe:
Die Identifikation (ID) des besten Vertrags und/oder der besten Dienstvereinbarung werden ausgegeben, gegebenenfalls auch, dass keine oder keine vollständige Übereinstimmung der Kunden- oder Nutzerprofile gefunden werden konnte. Eine Liste der vereinbarten Dienste sowie passenden Tarifparameterwerten wird ausgegeben.

7.7.3 Vertrag bewerten

Eingabe:
Der Benutzer wählt zwischen Vertrag und Dienstvereinbarung, gibt dessen Kennzeichnung (ID) ein und die einzutragende Bewertung. Das Tagesdatum wird vom Tariftool ergänzt.

Ausgabe:
Auftrag ausgeführt / SQL-Fehlermeldung (z.B. ID nicht vorhanden).

7.7.4 Datensatz auswählen

Eingabe:
Eine Liste der Tariftool-Tabellen wird ausgegeben:

```
Bitte Nummer der Tabelle eingeben:
1 : Kunde
2 : Vertrag
3 : Dienstvereinbarung
4 : Dienst
5 : Tarifparameter
6 : TarifparamWert
7 : Nutzer
8 : Usergruppe
9 : Kundenschema
10 : Dienstschema
11 : Userdienstprofil
12 : DienstTarif
13 : DienstVertrag
14 : Regel
```

Der Benutzer wählt die gewünschte Tabelle durch Angabe der angezeigten Nummer aus. Danach gibt er die ID des zu lesenden oder zu aktualisierenden Datensatzes ein. Die für Aktualisieren und Schreiben noch erforderlichen Eingaben sind in den folgenden Abschnitten beschrieben.

7.7.5 Datensatz lesen

Eingabe:
Nach der Auswahl des Datensatzes (s.o.) ist keine weitere Eingabe erforderlich.

Ausgabe:
Die angeforderten Daten werden ausgegeben, sofern der Datensatz in der Datenbank steht, andernfalls eine SQL-Fehlermeldung.

7.7.6 Datensatz aktualisieren

Eingabe:
Nach der Auswahl des Datensatzes (s.o.) ist der Name des zu ändernden Attributs (Spaltenname der gewählten Tabelle) einzugeben; danach wird der neue Wert des Attributs abgefragt (Eingabe bei Chars in ' eingeschlossen, Datum als 'mm/tt/jjjj'. Letzteres Format hängt von den Installationsdaten der Datenbank ab und kann bei anderen Datenbanken anders sein (s. Abschnitt 7.6).

Ausgabe:
Auftrag ausgeführt / SQL-Fehlermeldung bei fehlendem Datensatz oder falschem Eingabeformat.

7.7.7 Datensatz schreiben

Eingabe:
Nach der Auswahl des Datensatzes (s.o.) werden die Attribut- (Spalten-)namen der ausgewählten Tabelle nacheinander ausgegeben und Eingabe des Attributwerts verlangt. char-Daten sind mit Anführungszeichen einzugeben (z.B. als Adresse 'Bonn'), Datum als 'mm/tt/jjjj' (s.o.).

Ausgabe:
Datensatz eingetragen /SQL-Fehlermeldung, z.B. wenn Primary key ID schon vorhanden ist.

7.8 Erweiterungsmöglichkeiten

Bei Abgabe dieser Arbeit sind in der Testdatenbank *tarif* Tarifparameter für 7 Dienste eingetragen (s. Abschnitt 6.5), nach denen man suchen kann. Mit dem Wert aktion_2 =0 in der Tabelle *Regel* werden an der Benutzerschnittstelle für *besten Vertrag suchen* auch nur diese abgefragt.
Als Erweiterung bietet sich an, aktion_2 = 1 zu setzen, dann kann man weitere 8 Dienste aus dem in Abschnitt 6.3.4 vorgestellten *Dienstschema* auswählen, bei aktion_2 = 2 noch zusätzlich 6, die aber alle noch in die Datenbank eingebracht werden müssten. Dies ist ohne Änderung des Programms möglich. Mit Erweiterung der Schnittstelle entsprechend weiterer gewünschter Dienste können noch 9 zusätzliche Dienste abgefragt werden, insgesamt also 30 Dienste (Anforderung 5 und 8, Abschnitt 2.2).

7.9 Portabilität

Portabilität des Tariftools ist einmal durch die Implementierung in der Programmiersprache Java gegeben, zum anderen durch das Einlesen der datenbank- und benutzerspezifischen Daten aus einer Datei. Auf den Gebrauch von Gleitkommazahlen wurde verzichtet, da die Darstellung durch JDBC für die PostgreSQL-Datenbank sich von denen für andere Datenbanken unterscheidet. Vererbung wurde beim Entwurf der Objektklassen weggelassen, da sie sich zwar auf die PostgreSQL-Datenbank, nicht aber auf andere gebräuchliche Datenbanken abbilden lässt. (Das Format des Datums ist bei Datenbanken einstellbar.) Damit wird die Anforderung 12 (Abschnitt 2.2) nach Portabilität des Tariftools erfüllt.

7.10 Zusammenfassung

Die Entwicklungs- und Anwendungsumgebung sowie die Architektur des Tariftools wurden beschrieben. Die Erzeugung und das Beschreiben von Tabellen in der Datenbank mit Basisdaten für das Tariftool wurden erläutert. Die Benutzerschnittstelle wurde beschrieben. Aufgrund der Verwendung der Programmiersprache Java und der Vermeidung von Vererbung und Gleitkommazahlen ist das Tariftool auch auf andere Datenbanken portierbar, die über JDBC eine Verbindung zu Javaprogrammen herstellen können.

Kapitel 8

Zusammenfassung und Ausblick

8.1 Zusammenfassung

Viele Unternehmen verlagern heutzutage die Verwaltung ihrer Kommunikationsdienste auf externe Provider. Da solche Großkunden über viele Nutzer verfügen, die vielseitige Aufgaben mit Hilfe von Individualdiensten im Intranet und in externen Netzen, z.B. dem Internet, erfüllen sollen, ist dies eine sehr komplexe Aufgabe. Provider von Kommunikationsdiensten brauchen daher Unterstützung bei der Aufgabe, geeignete Tarife für Verträge mit Kunden zu finden, die verschiedene Unternehmensstrukturen haben und individuelle Dienste mit verbrauchsorientierter Abrechnung wünschen.

Für eine derartig komplexe Tarifierung gibt es noch wenig Erfahrungswerte. Daher werden die Daten früherer Tarifabschlüsse gespeichert und mit einer Bewertung versehen, die auf ihre Eignung als Vorbild schließen lässt. Damit man diese Tarifabschlüsse vergleichen kann, ist es erforderlich, dass ihre Struktur auf einem allgemein anwendbaren Tarifmodell beruht. Auf dieser Basis wurde ein *Tariftool* als Prototyp erstellt, das aus bewerteten Altverträgen geeignete Tarifparameter für bestimmte Anforderungen des Kunden ermittelt.

Ein allgemeines Tarifmodell wird aus Ergebnissen einer Literaturrecherche über Tarifierung abgeleitet. In ihm werden die wichtigsten Faktoren, wie z.B.

- Bereitstellung von Ressourcen (Abonnement),
- Anzahl Zugänge (access/setup)
- Dauer der Übertragung,
- Volumen der Datenübertragung,
- QoS-Parameter (Verzögerung, Jitter, u.a.) und
- zwei Tageszeitbereiche

berücksichtigt.

Eine Literaturrecherche bezüglich Wissensbasiertes Systeme ergab, dass für das Tariftool ein System mit

- objektorientierter Datendarstellung,
- Speicherung der Basisdaten in einer relationalen Datenbank und
- Anwendungsprogramm mit Regeln zur Modifikation des Programmablaufs

am geeignetsten ist.

Ein Prototyp *Tariftool*, der ein Wissensbasierten System darstellt, d.h. Basisdaten in einer Datenbank hält, die durch ein Anwendungsprogramm verwaltet und genutzt werden, wurde implementiert. Die Datenbank ist als relationale PostgreSQL-Datenbank vorgegeben, das Anwendungsprogramm wurde in der Programmiersprache *Java* objektorientiert geschrieben.

Die Basisdaten (Vertrag, Dienst, Kunde, Tarifparameter usw.) sind als objektorientierte Daten dargestellt und als Tabellen in der Datenbank gespeichert. Das Anwendungsprogramm des Tariftools enthält

- für die Wissensnutzung:
  die Klasse *Vertreter* mit Routinen zum Finden des besten Vertrags für die Wünsche des Kunden. Zur effizienten Suche wurden Suchkriterien für die Unternehmensstruktur des Kunden und die geforderten Dienste aufgestellt. Wird ein passender Vertrag nicht gefunden, so wird nach einem Vertrag gesucht, der ein Item weniger in den Suchkriterien enthält.
- für den Wissenserwerb:
  die Klassen *Account* zum Bewerten von Verträgen und *Admin* zum Lesen, Aktualisieren und Schreiben von Datensätzen.

Da der Prototyp *Tariftool* nicht den gesamten Umfang an Tarifdaten bewältigen kann, wurden repräsentative Daten für Dienste und Tarifparameter ausgewählt.

Durch Implementieren des *Tariftools* in der Programmiersprache *Java* und Vermeidung von speziellen Funktionen der PostgreSQL-Datenbank ist das *Tariftool* auch auf andere Datenbanken portierbar, zu denen Javaprogramme über JDBC eine Verbindung herstellen können.

8.2 Ausblick

Nach den Erfahrungen mit dem Prototyp *Tariftool* erscheint es als aussichtsreich, Tariftools zu entwickeln, die größere Bereiche der Tarifierung abdecken. In der vorliegenden Arbeit sind einem Dienst ein oder mehrere Tarifparameter fest zugeordnet, nur die Preise können frei gewählt werden. Bei einem erweiterten Modell könnten Variationen von einem Dienst berücksichtigt werden, z.B dass FTP nicht über *best effort* betrieben wird, sondern über eine feste Bandbreite. Bisher nicht berücksichtigte Dienste und Tarifparameter können eingebracht werden. Auch die Zusammenfassung von Diensten in Dienstgüteklassen mit gemeinsamen Tarifparametern kann eingeführt werden.

Bisher wurde auch davon ausgegangen, dass alle Gebühren von dem entrichtet werden, der sich einloggt. Bei einer Konferenzschaltung müssten aber die Kosten gerechterweise auf alle Teilnehmer verteilt werden. Auch kann es sein, dass derjenige, der zuerst empfängt, bereit ist, die Kosten zu übernehmen (Reverse Charging).

Regressbedingungen könnten, statt wie bisher nur dem Kunden, jedem Dienst zugeordnet werden; auch sind Regressbedingungen für bestimmte Stufen der Qualitätsminderung denkbar. Analog zu Tarifparametern könnten Regressparameter vereinbart werden.

Das Suchprogramm kann auch insoweit erweitert werden, dass nach Verträgen mit noch weniger Übereinstimmung mit den Kundendaten und -wünschen gesucht wird. Auch eine Erweiterung zur automatischen Erstellung von Abrechnungen und Bewertungen aufgrund von gespeicherten Daten ist machbar.

Anhang A

Ergebnisse stochastischer Berechnungen

A.1 Ermittlung der Parameter a,b beim ABC-Modell

Beispiele:

Die Abbildungen A.1 und A.2 zeigen Tarifparameter a und b für PCR 2.0 und 10 in Abhängigkeit von der mittleren Bandbreite.

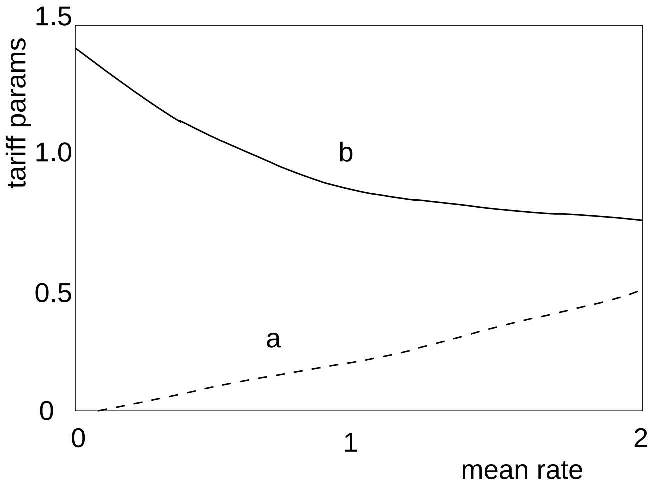

Abbildung A.1: Tarifparameter bei PCR 2

In den Tabellen A.1 und A.2 sind einige typische Beispiele für niedrige und hohe mittlere Bandbreiten ausgeführt (nach [song99]). Tabelle A.3 zeigt die Kosten für eine 100 sec lange Übertragung.

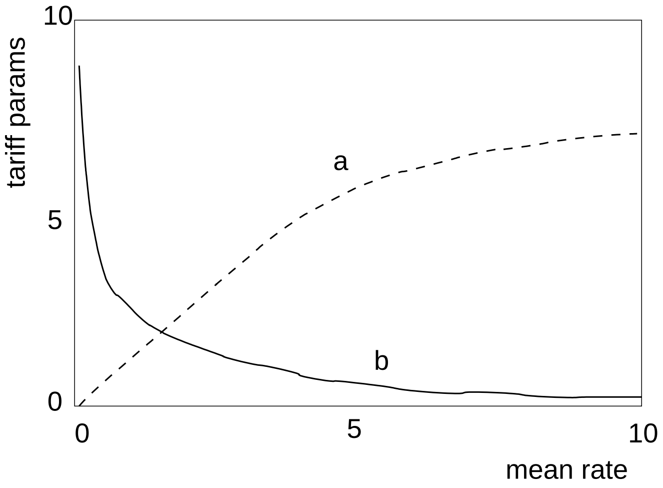

Abbildung A.2: Tarifparameter bei PCR 10

Connection Type	Peak (h)	Mean (m)	a(h,m)(s^{-1})	b(h,m)($(\text{Mbit})^{-1}$)
1	0.1	0.04	2.7×10^{-4}	1.0
2	2.0	0.02	1.3×10^{-4}	1.4
3	10.0	0.01	1.1×10^{-4}	7.9

Tabelle A.1: Preisfaktoren für niedrige mittlere Bandbreiten

Connection Type	Peak (h)	Mean (m)	a(h,m)(s^{-1})	b(h,m)($(\text{Mbit})^{-1}$)
4	2.0	1.0	0.2	1.0
5	10.0	1.0	1.7	2.2
6	10.0	2.0	3.0	1.3

Tabelle A.2: Preisfaktoren für hohe mittlere Bandbreiten

Connection Type	Peak (Mbit/s)	Mean (Mbit/s)	Charge (100 s)
1	0.1	0.04	4.0
2	2.0	0.02	2.8
3	10.0	0.01	8.0
4	2.0	1.0	116
5	10.0	1.0	392
6	10.0	2.0	557

Tabelle A.3: Kosten für 100 sec langen Datentransfer

A.2 Effektive Bandbreite (EB) als Maß für Ressourcennutzung

A.2.1 Einführung

Eine mögliche QoS-Garantie hängt mit der Pufferoverflow-Wahrscheinlichkeit zusammen. Die Puffer befinden sich als `Leaky Buckets` in den Routern. Der Operating Point des Links ist charakterisiert durch die Multiplexing- Parameter s (Space: abhg. v. Puffergröße β der `Leaky Buckets`) und t (Zeit: abhg. von Durchfluss ρ der `Leaky Buckets`). Es gibt drei Verfahren:

- Charging nach empirischem Wert:
  Dies bewirkt "eat-what-you-can"-Verhalten der Nutzer
- Nur EB berechnen:
  Damit gibt es keinen Ansporn auf Bescheidenheit bei Nutzung
- Ansatz hier:
  Näherung zwischen den Extremen: linear: beste Werte für Nutzer bei genauem Mittelwert (mean rate) m und Spitzenwert PCR. Statistische Parameter sind die leaky-bucket-Parameter β, ρ .

  f(X) = a0 + a1*g(X)
  mit f: Kosten/Zeiteinheit, Zustandsvariable X = (X_1,...,X_T): Last zu Zeitpunkten 1,..,T, g(X): gemessene mean rate :

  $$g(X) = 1/T \times \sum_{i=1}^{T} X_i.$$

  Die obere Grenze der EB als Funktion von m ist an jeder Stelle als Tangente gegeben mit dem Anstieg $\overline{\alpha}$(m,h). Für den Nutzer ist es am besten, wenn er seine mean rate m möglichst genau angibt, sonst muss er für mehr Ressourcen zahlen, als er tatsächlich in Anspruch nimmt.

A.2.2 Näherungen für die obere Grenze von EB

Zur Vermeidung von Staus dienen Leaky Buckets mit Parametern (β_k [Bytes], ρ_k[Mbps]]) . Die Ergebnislast ist:

$$\overline{X[0,t]} \leq H(t) := min_{k=1,K}\{\rho_k t + \beta_k\}$$

Für kleine Puffer erhält man für die EB α die Näherung:

$$\tilde{\alpha}_{on-off}(m,p) = 1/s \times log[1 + m/h(e^{sh} - 1)] \quad (p = policed \approx h, h = PCR)$$

Das Ergebnis : die obere Grenze der EB ist eine konvexe Funktion der mittleren Bandbreite (Abbildung A.3 nach [song99]).

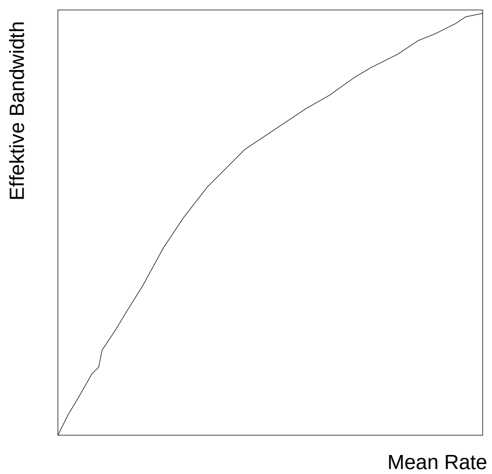

Abbildung A.3: Berechnete mittlere Bandbreite

A.2.3 Performance Evaluation

Der worst case traffic hat als Funktion der Zeit t oft die Form eines umgekehrten T (inverted-T-Pattern s. Abbildung A.4). Die Größe der Blöcke und Lücken (Gaps) hängt von s und t ab.

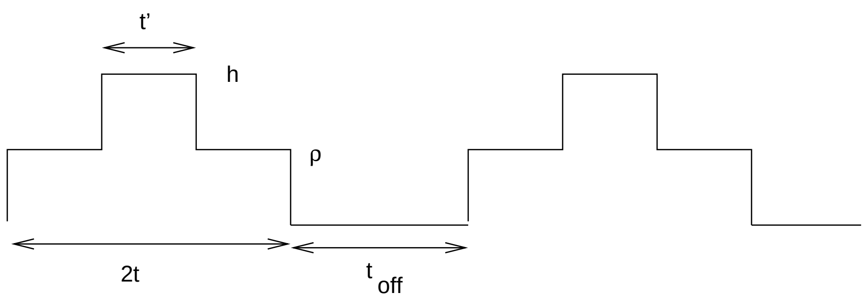

Abbildung A.4: Inverted T-Approximation

Fairness wird definiert als $\tilde{\alpha}$(m,h)/α(m,h). Experimente zeigen, dass bei kleineren Puffern die Fairness größer ist; inverted-T-Pattern ist relativ fair.

A.2.4 User-Netz-Interaktion

Nutzer können den anfänglichen Kontrakt ändern während die Verbindung besteht. Das Netz informiert über den Operation Point. Der Nutzer kann dann aus einer Tabelle Parameter a1, a2 wählen, die seiner aktuellen mean rate entsprechen. Im Folgenden wird gezeigt, dass ein Gleichgewicht existiert und durch den Kostenansatz die social welfare (Anzahl zugelassener Nutzer) maximiert wird.

Dazu wird ein Modell mit deterministischem Multiplex betrachtet, bei dem das Netz aus einem Shared Link mit Kapazität C und Puffer der Größe B besteht. Die Datenströme der Nutzer werden von einem Leaky Bucket geglättet (policed, d.h. die Spitzen werden eingeebnet). Es wird eine Indifferenzkurve G(ρ,β,h) betrachtet, bei der für festes h (peak rate, bzw. Verzögerung) Paare (ρ,β) aufgezeichnet sind (s. Abbildung A.5). Die Indifferenzfunktion G ist konvex, geht gegen unendlich für ρ gegen m, und ist =0 für ρ =h.

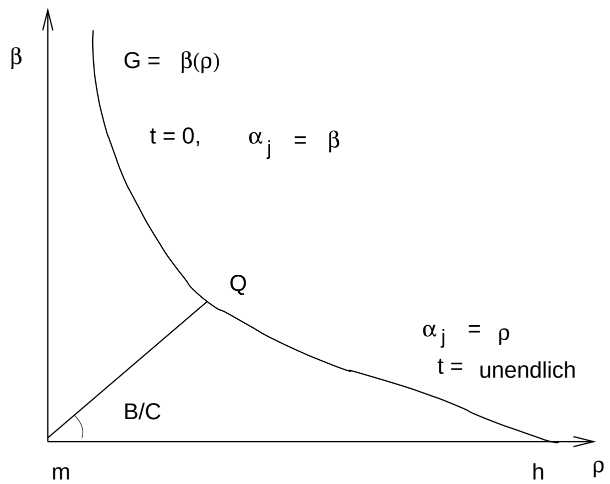

Abbildung A.5: Indifferenzkurve G

Die EB einer Verbindung, die mit (ρ, β) überwacht wird, ist:

$$\alpha_j(\infty, t) = \overline{X[0,t]}/t = \rho_j + \beta_j/t \quad für \quad t > 0 \quad \alpha_j(\infty, 0) = \beta_j.$$

Die Akzeptanzregion A ist durch die Randbedingungen (constraints) $\sum_j \rho_j \leq$ C und $\sum_j \beta_j \leq$ B definiert.

$$EB(t = \infty) = \rho_j \quad und \quad EB(t = 0) = \beta_j$$

Minimale Kosten entstehen dem User, wenn:

$$min_{\rho,\beta}\alpha(\infty, t; m, h) \quad und \quad (\rho, \beta) \in G \quad gilt$$

Optimal für den Netzbetreiber ist, dass er eine maximale Anzahl User n zulassen kann, so dass

$\sum_j \rho_j \leq$ C und $\sum_j \beta_j \leq$ B .

Nutzer und Netzwerk aktualisieren ihre Parameter in diskreten Schritten:

1. Netzwerk berechnet s,t aufgrund des aktuellen Verkehrs
2. Nutzer wählen leaky-bucket-Parameter (ρ, β)
3. Netzwerk berechnet s,t neu für neuen Operation Point

Annahme: alle n Nutzer haben gleiche Anforderungen:

$$n\rho \leq C, \quad n\beta \leq B$$

An Punkt Q in der Indifferenzkurve (Abbildung A.5) gelten beide Randbedingungen. Seien dort die Leaky-Bucket-Param ρ^*, β^*, dann gilt:

n* = C/ρ^* = B/β^*.

Q ist Gleichgewichtspunkt, der User hat dort minimale Kosten und der Provider optimale Bedingungen.

Anhang B

Abkürzungsverzeichnis

ABC-Formel	Abrechnungsformel mit Tarifparametern a,b, c
ABR	available bit rate (ATM)
AI	Artificial intelligence
API	Application Program Interface
ATM	Asynchronous Transfer Mode
BPS	Billing Service Provider
CATI	Charging and Accounting Technologies for the Internet
CA$HMAN	Charging and Accounting Schemes in Multiservice ATM Networks
CBQ	Class Based Queuing
CBR	Constant Bit Rate (ATM)
CIP	Charging Information Protocol
CORBA	Common Object Request Broker Architecture
DB	Datenbank
DBMS	Data Base Management System (Java)
DHCP	Dynamic Host Configuration Protocol
DiffServ	Differentiated Services
EB	Effektiv genutzte Bandbreite
FTP	File Transfer Protocol
GEM-PTP	Generic EXtensible and Modular Policy Transfer Protocol
IETF	Internet Engineering Task Force
INDEX	INternet Demand EXperiment
IntServ	Integrated Services
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISP	Internet Service provider
JDBC	Java Data Base Connectivity
KI	Künstliche Intelligenz
MIB	Management Information Base
MNM	Munich Network Management

NEM	Network Element Management (TMN)
NM	Network Management (TMN)
nrt-VBR	not real time Variable Bit Rate (ATM)
PCR	Peak cell Rate
PIP-NAR	Premium IP Network Accounting Record
PMP	Paris Metro Pricing
POT	Plain old telephony
QoS	Quality of Service
RM	Resource Management
RSVP	Resource Reservation Protocol
RTFM	Real Time Flow Measurement
rt-VBR	real time Variable Bit Rate (ATM)
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
TCP	Transmission Control Protocol
TFL	Tariff Formula Language
TINA	Telecommunications Information Network Architecture
TMN	Telecommunications Management Network
UBR	Unspecified Bit Rate (ATM)
UDP	User Datagram Protocol
UML	Unified Modeling Language
VBR	Variable Bit Rate (ATM)
VIPCAS	Value Added IP Charging and Accounting Service
VPN	Virtual Private Network
WBS	Wissensbasierte Systeme
WFQ	Weighted Fair Queuing
XML	EXtensible Markup Language

Literaturverzeichnis

[add85] ADDIS, T.R. (Herausgeber): *Designing knowledge-based systems.* Kogan Page Ltd, 1985.

[alch01] ALTMANN, J. und K. CHU: *How to charge for network services - flat-rate or usage-based.* Computer Networks, Seiten 519–531, 2001.

[alva99] ALTMANN, J. und P. VARAIYA: *Managing usage-based pricing in a future telecommunication market.* Technischer Bericht, University of California at Berkeley, 1999. eecs.berkeley report.

[argo96] ARNOLD, K. und J. GOSLING: *Java Die Programmiersprache.* Addison Wesley, 1996.

[bal99] BALZERT, H. (Herausgeber): *Lehrbuch Grundlagen der Informatik.* Spektrum Akademischer Verlag, Heidelberg, 1999.

[bawe99] BANNERT, G. und M. WEITZEL: *Objektorientierter Softwareentwurf mit UML.* Addison Wesley, 1999.

[bgbb00] BALMER, R., F. BAUMGARTER, M. GÜNTER und T BRAUN: *RSVP Interface for the differentiated service VPN.* Technischer Bericht, ETH Zürich, 2000. http://www.tik.ee.ethz.ch/ cati/.

[bloc02] J., BLOCH: *Effektiv Java programmieren.* Addison Wesley, 2002.

[brat90] BRATHWAITE, K. S.: *Datenbankentwurf - eine Einführung.* McGraw-Hill, Hamburg, 1990.

[Brau 89] W. BRAUER, C. FREHSE (Herausgeber): *Wissensbasierte Systeme.* Springer, 1989.

[breu01] BREU, R.: *Objektorientierter Softwareentwurf.* Springer, Berlin, 2001.

[brj99] BOOCH, G., J. RUMBAUGH und I JACOBSON: *Das UML-Benutzerhandbuch.* Addison Wesley, Boston, USA, 1999.

[brow95] BROWNLEE, N.: *New Zealand experiences with network traffic charging.* Technischer Bericht, Univ. Aukland, März 1995. http://www.press.umich.edu/jep/works/BrownNewZe.html.

[buwa89] BULLINGER, H.-J. und G. WASSERLOOS: *Die Entwicklung praxisgerechter Expertensysteme.* Verlag Moderne Industrie, 1989.

[caff99] CAFFREY, J. L.: *ATM services charging.* Computer Communications, Seiten 1652–1656, 1999.

[camh00] CHEN, C., R. ACHTERBERG, S. MOHAPI und H. HANRAHAN: *Billing Service for TINA Business model.* Technischer Bericht, University Whitwatersrand, Johannesburg, South Africa, 2000.

[cazs99] CARLE, G. und T. ZSEBY: *Tariff dependent error control for heterogeneous real-time multicast services.* Technischer Bericht, GMD FOKUS, Berlin, 1999. http://www.fokus.gmd.de/glone.

[chsz99] CARLE, G., F. HARTANTO, M SMIRNOV und T ZSEBY: *Charging and Accounting for QoS-enhanced IP Multicast.* In: *Proceedings of the IFIP 6-th Internat. Workshop for High-Speed Networks*, Seite 18, 1999.

[ckst99] CURRENCE, M., A. KURZON, D. SMUD und L. TRIAS: *A causal analysis of usage-based billing on IP networks.* Technischer Bericht, Colorado.edu, 1999.

[clar95] CLARK, D. D.: *A model for cost allocation and pricing in the Internet.* Technischer Bericht, Computer Sci Lab MIT, März 1995. http://www.press.umich.edu/jep/works/ClarkModel.html.

[coke00] COURCOUBETIS, C., F. KELLY und E.A.: *A study of simple usage-based charging schemes for broadband networks.* Telecommunication Systems, 2000.

[CoSi99] COURCOUBETIS, C. und V.A. SIRIS: *Managing and Pricing Service Level Agreements for Differentiated Services.* In: *Proceedings of the IFIP/IEEE IWQoS 99*, 1999.

[dehn99] DEHNHARDT, W.: *Anwendungsprogrammierung mit JDBC.* Carl Hanser Verlag, 1999.

[dfsb99] DERMLER, G., G. FANKHAUSER, B. STILLER, NT BRAUN und M. GÜNTER: *Integration of Charging and accounting into Internet models and VPN.* Technischer Bericht, ETH Zürich, 1999. http://www.tik.ee.ethz.ch/ cati/.

[edw91] EDWARDS, J.S. (Herausgeber): *Building knowledge-based systems.* Pitman Publ., 1991.

[efh86] ERDMANN, U., H. FIEDLER, F. HAFT und R. TRAUNMÜLLER (Herausgeber): *Computergestützte Juristische Expertensysteme.* Attempto Verlag Tübingen, 1986.

[fbrs99] FOUKIA, N., D. BILLARD, P. REICHL und B. STILLER: *User behavior for a Pricing scheme in a multi-provider scenario.* Technischer Bericht, ETH Zürich, 1999. http://www.tik.ee.ethz.ch/ cati/.

[fitr86] FIEDLER, H. und R. TRAUNMÜLLER (Herausgeber): *Formalisierung im Recht und Ansätze juristischer Expertensysteme.* J. Scheitzer Verlag München, 1986.

[flan97] FLANAGAN, D. (Herausgeber): *Java in a nutshell.* O'Reilly Inc, USA, 1997.

[fobi99] FOUKIA, N. und D. BILLARD: *Charging and accounting technology for the Internet: The CATI Project.* Technischer Bericht, TOS, Geneva University, Switzerland, 1999.

[fosc00] FOWLER, M. und K. SCOTT (Herausgeber): *UML konzentriert.* Addison Wesley, Boston, USA, 2000.

[fsp97] FANKHAUSER, G., B. STILLER und B. PLATTNER: *Arrow: A flexible architecture for an Accounting and Charging Infrastructure in the next generation Internet.* Technischer Bericht, ETH Zürich, 1997. http://www.tik.ee.ethz.ch/ cati/.

[fsvp98] FANKHAUSER, G., B. STILLER, C. VOEGTLI und B. PLATTNER: *Reservation-based charging in an integrated services network.* Technischer Bericht, ETH Zürich, 1998. http://www.tik.ee.ethz.ch/ cati/.

[gfh90] GOTTLOB, G., T. FRÜHWIRTH und W. HORN (Herausgeber): *Expertensysteme.* Springer Verlag Wien, New York, 1990.

[ghk01] GARSCHHAMMER, M., R. HAUCK, I. RADISIC, H. ROELLE und H. SCHMIDT: *The MNM Service Model - Refined Views on Generic Service Management.* Technischer Bericht, LMU, München, Juli 2001. Internes Papier.

[gjpsw97] GUPTA, A., B. JUKIC, M. PARAMESWARAN, D.O. STAHL und A.B. WHINSTON: *Streamlining the digital economy: How to avert a tragedy of the common.* Technischer Bericht, CREC, University of Texas at Austin, 1997. http://cism.bus.utexas.edu/works/articles/.

[gom00] GOMAA, H. (Herausgeber): *Designing concurrent, distributed, and real-time applications with UML.* Addison Wesley, Boston, USA, 2000.

[groe96] GROELING, J.: *EDV-Unterstützung durch verteilte Expertensysteme für die Anspruchsprüfung und Beratung in der Versicherungswirtschaft* . Dissertation, Univ. Göttingen, 1996.

[gswp95] GUPTA, A., D.O. STAHL und A.B. WHINSTON: *A priority pricing approach to manage multi-service class networks in real-time.* Technischer Bericht, University of Texas at Austin, März 1995. http://www.press.umich.edu/jep/works/GuptaPrior.html.

[gswt95] GUPTA, A., D.O. STAHL und A.B. WHINSTON: *The Internet: A future tragedy of the common?* Technischer Bericht, University of Texas at Austin, Juli 1995. http://cism.bus.utexas.edu/works/articles/.

[haca99] HARTANTO, F. und G. CARLE: *Policy-Based billing architecture for Internet differentiated services.* In: *Proceedings of the IFIP 5-th Internat. Conf. on broadband Communications*, 1999.

[haki86] HARMON, P. und D. KING: *Expertensysteme in der Praxis.* Oldenbourg Verlag München. Wien, 1986.

[HAN 99] Hegering, H.-G., S. Abeck und B. Neumair: *Integrated Management of Networked Systems – Concepts, Architecture.* Morgan Kaufmann Publishers, ISBN 1-55860-571-1, 1999. 651 p.

[HAN 99a] Hegering, H.-G., S. Abeck und B. Neumair: *Integriertes Management vernetzter Systeme — Konzepte, Architektur.* dpunkt-Verlag, ISBN 3-932588-16-9, 1999. 607 S.

[hara01] Hauck, R. und I. Radisic: *Service oriented application management - do current techniques meet the requirements?* Technischer Bericht, LMU, München, 2001. Internes Papier.

[hart01] Hartwig, J.: *PostgreSQL.* Addison Wesley, Boston, USA, 2001.

[hart99] Hartanto, F.: *Policy-based Architecture for Advanced and Dynamic Internet Service Creation and management.* Technischer Bericht, GMD Berlin, 1999. GMD Technical Report, Berlin.

[helt92] Helten, E.: *Wettbewerbsvorteile durch wissensbasierte Systeme.* Technical Report, Versicherungswissenschaft, München, 1992.

[hobb97] Hobbs, A. (Herausgeber): *Datenbank programmieren mit JDBC.* SAMS, Markt & Technik, 1997.

[hoco00] Horstman, C.S. und G Cornell: *Core Java Band 2 Expertenwissen.* Markt & Technik Verlag, 2000.

[jack87] Jackson, P.: *Expertensysteme.* Addison-Wesley, 1987.

[jeps97] Jepson, B: *Java Database Programming.* Wiley & Sons Inc., 1997.

[joha90] Johannes, P.: *Expertensysteme: Entscheidungskriterien für Manager.* Oldenbourg Verlag München. Wien, 1990.

[kahl98] Kahlbrandt, B.: *Softwareengineering: OOSE mit UML.* Springer, Berlin, 1998.

[karo97] Karsten, M., J. Roos und P.J. Steenekamp: *An intelligent agent implementation for integrated services billing.* Technical Report, Univ. Pretoria; Telkom, SA, 1997. http://www.citeseer.nj.nec.com/155877.html.

[kech01] Kenyon, C. und G. Cheliotis: *Stochastic models for telecom commodity prices.* Computer Networks, 2001.

[keei99] Kemper, A und A. Eicklier: *Datenbanksysteme: eine Einführung.* Oldenbourg Verlag, 1999.

[kell95] Kelly, F.P.: *Charging and Accounting for bursty connections.* Technischer Bericht, Univ. Cambridge, England, März 1995. http://www.press.umich.edu/jep/works/KellyCharg.html.

[klei92] Kleine-Büning, H.: *Wissensbasierte Systeme.* Addison Wesley, 1992.

[kra86] KRALLMANN, H. (Herausgeber): *Expertensysteme im Unternehmen.* Erich Schmidt Verlag Berlin, 1986.

[krem89] KREMS, J. (Herausgeber): *Expertensysteme im Einsatz.* R.Oldenbourg Verlag, München, Wien, 1989.

[krfr89] KRUSE, H.-G. und U. FRANK (Herausgeber): *Praxis der Expertensysteme.* Carl Hanser Verlag, München, Wien, 1989.

[kssw00] KARSTEN, M., J. SCHMITT, B. STILLER und L. WOLF: *Charging for packet-switched network communication - motivation and overview.* Computer Communications, Seiten 290–302, 2000.

[kueh99] KÜHNEL, R.: *Die Java 2 Fibel, 3. Auflage.* Addison Wesley, 1999.

[kur92] KURBEL, K. (Herausgeber): *Entwicklung und Einsatz von Expertenystemen.* Springer-Verlag Berlin, 1992.

[kzs00] KNEER, H., U. ZURFLUH und B. STILLER: *A role- and service-based business process for a value-added Internet.* Technischer Bericht, IFI, University of Zurich, 2000.

[leca02] LEMAY, L. und R. CADENHEAD (Herausgeber): *Java 2.* Markt & Technik Verlag, München, 2002.

[lsow99] LIN, ZH., D.O. STAHL, P.S. OW und A.B. WHINSTON: *Exploring traffic pricing for the Virtual private Network.* Technischer Bericht, University of Texas at Austin, 1999. WIST99 paper.

[lufa91] LUCAS, P. und L. VAN DER GAAG: *Principles of expert systems.* Addison-Wesley, 1991.

[mali91] MALINOWSKI, U.: *Wissensbasierte Benutzerunterstützung für Softwaresysteme entwickelt am Beispiel der Personaldisposition im Verkehr* . Dissertation, Techn. Univ. Braunschweig, 1991.

[mau97] MATHIESSEN, G. und M. UNTERSTEIN: *Relationale Datenbanken und SQL: Konzepte der Entwicklung und Anwendung.* Addison Wesley, Boston, USA, 1997.

[mava94] MACKIE-MASON, J. und H.R. VARIAN: *Pricing the Internet.* In: *Proceedings of the Public Access to the Internet*, 1994.

[mcba95] MCKNIGHT, L.W. und J.P. BAILEY: *An introduction to Internet economics.* Technischer Bericht, MIT, März 1995. http://www.press.umich.edu/jep/works/McKniIntro.html.

[mmm95] MACKIE-MASON, J.K., L. MURPHY und J. MURPHY: *The role of responsive pricing in the Internet.* Technischer Bericht, XX, März 1995. http://www.press.umich.edu/jep/works/MackieResp.html.

[mmva94] MACKIE-MASON, J. und H.R. VARIAN: *FAQs about usage-based pricing.* Technischer Bericht, Univ. of Michigan, September 1994. http://www.sims.berkeley.edu/ hal/papers.

[momb01] MOMJIAN, B.: *PostgreSQL Einführung und Konzepte.* Addison Wesley, Boston, USA, 2001.

[momj01] MOMJIAN, B.: *PostgreSQL: Introduction and Concepts.* Addison Wesley, 2001.

[msu87] MANTZ, R., M. SCHEER und T. UTHMANN: *Expertensysteme in Industrie und Wissenschaft.* Technical Report, Gesellschaft für Mathematik und Datenverarbeitung mBH, Sankt Augustin, 1987. GMD-Studien Nr. 136.

[nia197] TINIOS, G.: *Charging Models in ATM Networks* . Technical Report, INTRASOFT, 1997. NIA-G1, Draft A, GINA project.

[nia297] TINIOS, G. und M. RAFFALI: *ATM Connection Detail Records and Charging Parameters* . Technical Report, INTRASOFT, 1997. NIA-G2, Draft A, GINA project.

[nia397] MORRIS, D.: *Customer Requirements for ATM Charging* . Technical Report, CANCAN, 1997. NIA-G3, Draft A, GINA project.

[nia498] LAKE, J.: *ATM Charging Strategies.* Technical Report, CANCAN, 1998. NIA-G4, Draft A, GINA project.

[odla98] ODLYZKO, A.: *Paris Metro Pricing for the Internet.* Technischer Bericht, ATT Labs Research, 1998. http://www.research.att.com/ amo.

[odly01] ODLYZKO, A.: *Internet pricing and the history of Communications.* Computer Networks, Seiten 493–517, 2001.

[odly98] ODLYZKO, A.: *Paris Metro Pricing: The minimalist differentiated services solution.* Technischer Bericht, ATT Labs Research, 1998. http://www.research.att.com/ amo.

[Oest 99] OESTERREICH, B.: *Developing Software with UML.* Addison Wesley, 1999.

[oest01] OESTERREICH, B.: *Die UML-Kurzreferenz für die Praxis.* Oldenbourg Verlag München, Wien, 2001.

[OGC 01] GOVERNMENT COMMERCE (OGC), OFFICE OF (Herausgeber): *Service Delivery.* The Stationary Office, 2001.

[pfe96] PFEIFER, T. (Herausgeber): *Wissensbasierte Systeme in der Qualitätssicherung.* Springer Verlag Berlin, 1996.

[pgpb96] PUPPE, F., U. GAPPA, K. POECK und S. BAMBERGER: *Wissensbasierte Diagnose- und Informationssysteme.* Springer Verlag, 1996.

[pgrs98] POSTGRESQL: *PostgreSQL.* Technischer Bericht, xx, 1998. http://www.postgresql.org/.

[pina01] TELECOM, PINACL: *Billing system options for new generation IP networks.* Technischer Bericht, Pinacl Telecom, 2001. PL0303 Release Issue 1.

[pop92] Popp, H.E.: *Hybride wissensbasierte Systeme zur Datenanalyse und Eigenschaftsvorausberechnung physikalisch-chemischer Systeme dargestellt an Elektrolytlösungen.* Dissertation, Univ. Regensburg, 1992.

[Pupp 91] Puppe, F.: *Einführung in Expertensysteme.* Springer Verlag, 1991.

[pupp93] Puppe, F.: *Systomatic introduction to Expert systems.* Springer Verlag, 1993.

[rada00] Radisic, I.: *Abrechnungsmodelle für den IP- und ATM-Dienst.* Technischer Bericht, LMU, München, Oktober 2000. Internes Papier.

[radb00] Radisic, I.: *Ein Abrechnungsmodell für den Web-Hosting-Dienst.* Technischer Bericht, LMU, München, Dezember 2000. Internes Papier.

[radc01] Radisic, I.: *Ein Abrechnungsmodell für den Email-Hosting-Dienst.* Technischer Bericht, LMU, München, Februar 2001. Internes Papier.

[Radi 02] Radisic, I.: *Using Policy–Based Concepts to Provide Service Oriented Accounting Management.* In: *Proceedings of the 8th International IFIP/IEEE Network Operations and Management Symposium (NOMS 2002)*, April 2002.

[reed02] Reed, P.R. Jr. (Herausgeber): *Developing Applications with Java and UML.* Addison Wesley, Boston, USA, 2002.

[rees00] Reese, G.: *Data Base Programming with JDBC and JAVA.* Franzis', Poing, 200.

[reic99] Reichl, P.: *Approximated Price functions for dynamic volume-based Pricing of multiclass internet traffic.* Technischer Bericht, ETH Zürich, 1999. http://www.tik.ee.ethz.ch/ cati/.

[Reim 91] Reimer, U.: *Einführung in die Wissensrepräsentation.* Teubner, 1991.

[rls99] Reichl, P., S. Leinen und B. Stiller: *Pricing models for Internet services.* Technischer Bericht, ETH Zürich, März 1999. http://www.tik.ee.ethz.ch/ cati/.

[sark95] Sarkar, M.: *An assessment of pricing mechanisms for the Internet – A regulatory Imperative.* Technischer Bericht, Michigan State University, East Lansing, März 1995. http://www.press.umich.edu/jep/works/SarkAssess.html.

[sasa00] Saake, G. und K.U. Sattler: *Datenbanken und Java.* dpunkt Verlag, 2000.

[sau92] Sauer, H.: *Relationale Datenbanken: Theorie und Praxis.* Addison Wesley, Boston, USA, 2. Aufl., 1992.

[sch87] Schnupp, P. (Herausgeber): *Expertensysteme: Strategien und Erfahrungen.* Oldenbourg, 1987.

[schm01] Schmidt, H.: *Entwurf von Service Level Agreements auf der basis von Dienstprozessen .* Dissertation, Ludwig-Maximilians-Universität München, Juli 2001.

[schn86] Schnupp, P. (Herausgeber): *State of the Art: Expertensysteme.* Oldenbourg, 1986.

[sewo00] SEEMANN, J. und J. WOLFF VON GUDENBERG: *Software-Entwurf mit UML*. Springer Verlag, Berlin, 2000.

[sfjr99] STILLER, B., G. FANKHAUSER, G. JOLLER, P. REICHL und N WEILER: *Open Charging and QoS Interfaces for IP Telephonie*. Technischer Bericht, ETH Zürich, 1999. http://www.tik.ee.ethz.ch/ cati/.

[sfp98] STILLER, B., G. FANKHAUSER und B. PLATTNER: *Charging of multimedia flows in an integrated services network*. Technischer Bericht, ETH Zürich, 1998. http://www.tik.ee.ethz.ch/ cati/.

[sfpw98] STILLER, B., G. FANKHAUSER, B. PLATTNER und N. WEILER: *Charging and accounting for integrated Internet services*. Technischer Bericht, ETH Zürich, 1998. http://www.tik.ee.ethz.ch/ cati/.

[shen98] SHENKER, S.: *Service Models and Pricing Policies for an Integrated Services Internet*. Technical Report, Xerox Corp., 1998. 98??

[slcl99] SEMRET, N, R. LIAO, CAMPBELL. A.T. und A.A. LAZAR: *Market Pricing of Differentiated Internet Services*. In: *Proceedings of the IEEE/IFIP IWQOS 99*, Seite 10, 1999.

[smir00] SMIRNOV, M, T ZSEBY und AL: *Quality of Service Methodologies and solutions within the service framework: Measuring, Managing and Charging QoS*. Technischer Bericht, Deutsche Telecom, 2000. Eurescom Project EDIN 0060-0906.

[song99] SONGHURST, D. J. (Herausgeber): *Charging Communication networks - From Theory to Practise*. Elsevier Science B. V., 1999.

[srin95] SRINAGESH, P.: *Internet cost structures and interconnection agreements*. Technischer Bericht, Bell Com.Res. Inc, März 1995. http://www.press.umich.edu/jep/works/SrinCostSt.html.

[srl 00] STILLER, B., P. REICHL und S. LEINEN: *Pricing and Cost Recovery for Internet Services*. Netnomics, Januar 2000.

[thur95] THURMAYR, G.R.: *Ein wissensbasiertes System für die Qualitätssicherung in der medizinischen Basis- und GSG-Dokumentation - Methodologie und Darstellung der Evaluation* . Habilitation, Technische Universität München, Dezember 1995.

[vill96] VILLASIS, S.J.: *An optimal pricing mechanism for Internet's end-users* . Dissertation, Univ. of Idaho, Mai 1996.

[weis90] WEISSENFLUH, A.: *Expertensysteme im Einsatz zur Unterstützung betrieblicher Entscheidungen*. Verlag Paul Haupt, Bern, Stuttgart, 1990.

[whfi00] WHITE, S., M. FISHER, R. CATTEL, G. HAMILTON und M. HAPNER: *JDBC API Tutorial and Reference*. Addison-Wesley, 2000.

[wilm02] WILMS, R.: *Programmieren in FORTRAN, Perl, Java und C#*. Franzis', Poing, 2002.

[win87] WINSTANLEY, G. (Herausgeber): *Program design for knowledge-based systems.* Sigma Press, Wilmslow, UK, 1987.

[Wink 91] WINKELMANN, K. (Herausgeber): *Wissensbasierte Systeme in der Praxis.* Siemens AG, 1991.

[wodr02] WORSLEY, J.C. und J.D. DRAKE (Herausgeber): *Practical PostgreSQL.* O'Reilly Inc, USA, 2002.