Projektgruppe 446

Endbericht

Entwurf und Implementierung eines wissensbasierten Decision Support Systems zur personalisierten Unterstützung von Anfragen im Kunden Service

Projektgruppe 446 – Decision Support System

Teilnehmer:

Adiuware.Data

Martin Saternus
Suny Kaya
Yuan Zhang

Adiuware.Logic

Richard Süselbeck
Nils Müller
Daniel Müller
Frank von der Höh
Alexander Ullbrich

Adiuware.Windows

Ute Kersting
Christian Topnik

Adiuware.Web

Laith Raed
David Mittag

Projektgruppenleiter:

Stefan Berlik
Alexander Holland

TEIL I: Einleitendes und Zielsetzung

TEIL II: Die Vorstellung der einzelnen Komponenten

TEIL III: Abschliessendes

TEIL IV: Handbücher

Teil I
Einleitung; Entwicklungs-umgebung; Pflichtenheft; Erklärung von Grundbegriffen

1 Einleitung

„Die Bedeutung von Decision Support Systemen (DSS) hat in den letzten Jahren stark zugenommen.“ (*Quelle:* [SPR/WAT86])

Verantwortliche in den unterschiedlichsten Anwendungsbereichen sollten im Entscheidungsprozeß über ihre unterschiedlichen Entscheidungsmöglichkeiten und die jeweils daraus resultierenden Konsequenzen informiert sein.
Decision Support Systeme stellen hier ein wichtiges computerbasiertes Hilfsmittel in Planungs- und Entscheidungsprozessen dar.
Als Anwendungsbereich haben heute gerade Call Center oder Support Center im IT-Umfeld ein großes Interesse daran, durch die Möglichkeiten des Internet und durch den Einsatz von wissensbasierten Komponenten den Kontakt zu Ihren Kunden intensiv zu pflegen und zielgerichtet auf die Bedürfnisse des jeweiligen Kunden ausgerichtet individuelle Dienstleistungen als so genannte Support Services anbieten zu können.
Kunden können heute über vielfältige Zugangswege Kontakt zu einem Call Center aufnehmen und dort Beratungsdienste wie die Beantwortung von Kunden-Anfragen in Anspruch nehmen.
Hier lassen sich neben den etablierten Möglichkeiten wie Telefon oder Fax immer häufiger Kundenzugänge über das Internet oder per Email nennen.

1.1 Problemstellung

1. Ein Endkunde nimmt via Webbrowser Kontakt zu einem Call Center auf.

Er hat über Support Webseiten Zugang zu technischen Wissensdatenbanken und kann seine Anfrage selbst lösen.

- Der Kunde soll dabei Dokumente in der Wissensdatenbank über Stichworte suchen können
- oder sich durch Entscheidungsbäume zu empfohlenen Dokumenten leiten lassen.

Statische Entscheidungsbäume als Softwaretool stellen dabei Frage-Antwortpaare in einer gegebenen Baumstruktur dar. Die Beispiel-Antwort ist dabei direkt über einen Link mit dem empfohlenen Dokument verknüpft. Führt dies allerdings nicht zur Lösung, kann der Endkunde durch weiter verzweigte Frage- Antwortpaare in der Baumstruktur nach der Lösung suchen.

2. Ein Endkunde nimmt via Webbrowser Kontakt zu einem Call Center auf. Findet er jedoch in der Wissensdatenbank keine hilfreichen Lösungsdokumente und kann im Entscheidungsbaum keine entsprechenden Informationen bekommen, kann er einen Support- Mitarbeiter via Email oder Telefon kontaktieren.

   - Der Support-Mitarbeiter nimmt Kontakt mit dem Endkunden auf und kann im Lösungsprozess auf weitere technische Wissensdatenbanken und bereits dokumentierte Lösungen früherer Anfragen zurückgreifen,
   - sowie den Endkunden mit einen Mitarbeiter verbinden, der spezialisiert auf die Anfrage des Endkunden ist.

3. Ein Endkunde nimmt direkt telefonischen Kontakt zu einem Call Center Mitarbeiter auf.

Es wird allerdings angestrebt, die Anzahl selbst gelöster Anfragen immer weiter zu steigern (self solve rate).

1.2 PG- Aufgabe

Im Rahmen dieser Projektgruppe wird ein *webbasiertes Decision Support System* entwickelt, welches insbesondere basierend auf modernen Methoden der Entscheidungstheorie eine Entscheidungs-unterstützung bei der Beantwortung von Kunden-Anfragen erlaubt.

Kunden-Anfragen im Support können bei technischen Problemen des Kunden sehr vielfältig sein und unvollständiges Wissen beinhalten.
Auftretende Probleme wie sporadische Systemausfälle eines Servers lassen sich nur schwer zuordnen.
Der Kunde kann die auftretende Problematik entweder nur teilweise einordnen oder nennt bei der Problembeschreibung lediglich einzelne Stichworte.

Support Center Mitarbeiter haben Zugriff auf technische Wissensdatenbanken, in denen auch das Wissen bereits bearbeiteter und dokumentierter früherer Anfragen gespeichert sein kann.
Sie sind häufig auf bestimmte Hardware- und Softwareprodukte spezialisiert und müssen bei der Lösungsfindung an bestimmten Stellen mit anderen technischen Experten zusammenarbeiten.
Bei der Beantwortung von Anfragen gilt es nun, neben technischen Experten auch bereits bearbeitete
und dokumentierte Anfragen in Form gespeicherter Dokumente aus technischen Wissensdatenbanken
in den Lösungsprozess einzubeziehen. Für ein besseres Case Handling und zur Erreichung eines qualitativ höheren Serviceniveaus im Help Desk ist der Einsatz entscheidungsunterstützender Elemente dabei sehr hilfreich.

Das Projekt beinhaltet neben der praktischen Komponente (Realisierung des Decision Support Systems) auch Fragestellungen theoretischer Herkunft (mathematische entscheidungstheoretische Aspekte und graphbasierte Methoden), die im Rahmen dieser PG zu lösen sind.

1.3 Aufgabenumsetzung und Ziel

Das Ziel der Projektgruppe ist es, die Bearbeitung der im Support vorkommenden Kunden-Anfragen durch den Einsatz von effizienten graphbasierten Entscheidungsmodellen durch ein webbasiertes Decision Support System zu optimieren und diese praxisnah in Wissensdatenbanken eines Call Centers einzusetzen.

Entscheidungstheoretische Methoden stellen nun Konzepte zur Formalisierung von Entscheidungsprozessen und darauf aufbauend zur Auswahl geeigneter Alternativen bereit. Dabei kommt der Modellierung des Wissens des Entscheidungsträgers eine besondere Bedeutung zu. Das Wissen bei Help Desk Anfragen ist im Allgemeinen unvollständig und ungenau (wie beispielsweise sporadische Serverausfälle).
Bei der Umsetzung wurde vor allem Wert darauf gelegt, neue entscheidungstheoretische und graphbasierte Methoden kennenzulernen und diese praxisnah zur Lösung von Kunden-Anfragen im Decision Support System einsetzen. Für die Entwicklung

der graphbasierten Methode zur Entscheidungsunterstützung werden Bayes'sche Netze eingesetzt (**s. → 4. Grundlegende Begriffe**).
Ein Bayes'sches Netz stellt einen gerichteten azyklischen Graphen dar, wobei jeder Knoten des Graphen mit einer bedingten Wahrscheinlichkeitsverteilung attributiert.
Technische Informationen lassen sich nun in statischen Entscheidungsbäumen als Frage- Antwortpaare ablegen.
Knoteninformationen können dabei mit Kunden- und Mitarbeiterinformationen verknüpft werden.
Bei komplexeren Anfragen kann es außerdem vorkommen, das der Lösungsweg einer Kunden-Anfrage über mehrere Frage-Antwortpaare in der Baumstruktur führt.
An dieser Stelle bieten Entscheidungsunterstützende Tools wie *Dezision Works* von *Dezide* die Möglichkeit, mit Bayes'schen Netzen als Entscheidungsnetzen zu arbeiten und diese mit der Wissensdarstellung von Supportdaten durch statische Entscheidungsbäume als Frage-Antwortpaare zu kombinieren.
Durch die Angabe von bedingten Wahrscheinlichkeiten kann hierbei eine effiziente Lösung in der Wissensbasis ermittelt werden. Die zu beschaffende Information zur Entscheidungsfindung wie bspw. Angaben über früher gelöste Anfragen zur gleichen oder ähnlichen Problematik wird in den Lösungsprozess integriert, Informationswerte werden bestimmt und Wahrscheinlichkeiten neu ermittelt.

1.4 Literatur

[SPR/WAT86] Sprague, R.H.J. und Watson, H.J. (Ed.): Decision Support Systems: Putting Theory into Practice. London, 1986

2 Die Entwicklungsumgebung

2.1 Visual Studio.NET

Visual Studio.NET ist die integrierte Entwicklungsumgebung (IDE - Integrated Development Environment) für die Implementierung von Lösungen auf der .NET Plattform. Im Gegensatz zu Vorgängerversionen sind hier alle Programmiersprachen (C#, C++, VB, JScript) und ASP.NET in einer einheitlichen Umgebung integriert. Dies bietet dem Entwickler ein einheitliches Interface zur Entwicklung verteilter Systeme auf der Microsoft Plattform. Speziell die Integration der XML Web Services erlaubt neben der Verwendung lokaler Ressourcen und Klassenbibliotheken die Verwendung von Services im Internet, auf die per SOAP zugegriffen werden kann. Darüber hinaus unterstützt die IDE den Lookup in UDDI Verzeichnissen und Disco Files zum Auffinden von Web Services. WSDL Files werden bei der Erstellung von Web Services ebenso automatisch erstellt wie Client-seitige Proxies zum Aufruf entfernter Web Services bzw. NET Komponenten.

Die einheitliche Entwicklungsumgebung und die Verwendung der .NET Framework Class Libraries erleichtert dem Entwickler das nahtlose Debugging zwischen .NET Komponenten, die in verschiedenen Programmiersprachen (z.B. C# und Visual Basic.NET) implementiert sind. Remote Debugging auf anderen Servern wird ebenso unterstützt wie das Debugging von Web Services.

Die folgenden Kapitel gehen auf einige spezifische Features der Visual Studio.NET Entwicklungsumgebung ein.

2.1.1 Die integrierte Entwicklungsumgebung

Im Folgenden werden einige Features der Visual Studio .NET IDE aufgeführt:

- **IntelliSense**: IntelliSense ist ein Feature, das den Entwickler bei der Programmierung unterstützt, indem es bei der Verwendung von Objekten z.B. alle verfügbaren Methoden auflistet, wenn der Objektname, folgend von einem Punkt eingegeben wird. Beim Aufruf von Methoden werden automatisch alle benötigten Parameter inkl. Datentyp und sogar alle überladenen Signaturen angezeigt. Dies erspart dem Entwickler langwieriges Suchen in Dokumentationen. Die so genannte 'Code Completion' erlaubt das automatische Vervollständigen von Identifiern (durch die Tab-Taste), wenn genügend Zeichen eingegeben wurden um das entsprechende Wort eindeutig zu identifizieren.
- **Debugging**: Der integrierte Debugger erlaubt das nahtlose Debugging in Projekten über Sprachgrenzen hinweg. Es kann also z.B. mit dem Debugger aus VB Code direkt in eine C# Komponente gesprungen werden. Ebenso kann aus Managed Code in Unmanaged Code (z.B. COM+ Komponenten oder native Win32 Anwendungen) hineingesprungen werden. Der Debugger kann auch über Rechnergrenzen hinweg zum Debuggen von Remote Komponenten oder Web Services eingesetzt werden.
- **Integrierter Web Browser**: Webseiten können über einen in die IDE eingebetteten Browser direkt gestartet werden. Ebenso können beliebige URLs aus dem Web in der Entwicklungsumgebung angezeigt werden, z.B. die MSDN Online Library Seiten.
- **Tabbed Documents**: Sind mehrere Fenster gleichzeitig in der IDE geöffnet, können diese über so genannte 'Reiter' direkt geöffnet werden.
- **Auto Hide**: Bestimmte Fenster (Toolbox, Solution Explorer, Server Explorer) verschwinden automatisch, wenn der Entwickler sie nicht benötigt und werden durch Überfahren des verbliebenen Icons mit der Maus geöffnet.
- **Command Window**: Im Command Window können IDE Kommandos oder Code Statements (Variablen setzen, Ausdrücke evaluieren, etc.) direkt ausgeführt werden.
- **Start Page**: Über die Startseite kann der Entwickler seine persönlichen Einstellungen in der DIE konfigurieren. Darunter fallen Keyboard-Shortcuts, Window Layouts, Help-Filter, Anzeige existierender Projekte, etc.
- **Bearbeiten von HTML**: HTML Seiten lassen sich neben der Code View auch in einer Design View bearbeiten. Hier wird das Layout der Seite angezeigt und kann bearbeitet werden. Über ein TargetSchema Property kann der Ziel-Browser (z.B. Internet Explorer oder Netscape Navigator) angegeben werden.
- **Bearbeiten von Cascading Style Sheets**: VS.NET bietet einen Editor zur komfortablen Erstellung von CSS Styles.
- **Bearbeiten von XML Files**: VS.NET bietet einen Editor zum Erstellen von XML Files inkl. Code Colouring, hierarchischer Darstellung der XML Struktur, etc.
- **Code Comment Web Reports**: .NET Sourcecode kann mit speziell formatierten Kommentaren versehen werden, die als HTML Files aufbereitet und als Dokumentation des Codes verwendet werden können.
- **Object Browser**: Der Object Browser bietet dem Entwickler eine Übersicht über die Klassen und deren Member (Properties, Methoden, Events), die in einem Projekt verwendet bzw. referenziert werden.

- **Macro Recording**: In der IDE können Macros aufgezeichnet und abgespielt werden um typische Arbeitsabläufe zu automatisieren.
- **IDE Automation Model**: Fast jedes Element der IDE kann über ein spezielles Objektmodell programmatisch bearbeitet und gesteuert werden.

2.1.2 Projekte und Solutions

Für die Entwicklung einer Anwendung, einer Komponente, eines Web Services, etc. bietet Visual Studio. NET so genannte 'Projekte' an. Ein Projekt umfasst dabei die Sourcefiles, Ressourcen, Referenzen und Data Connections (zu Datenbanken) und wird in eine DLL, ein Executable oder ein .NET Modul kompiliert. Für Projekte stellt VS.NET Templates zur Verfügung, die bei der Neuerstellung eines Projekts ausgewählt werden können und einen Code-Rahmen für die Anwendung zur Verfügung stellen. Beispiele für Templates sind: Web Service, Windows Applikation, Konsolen Applikation, Windows Service, etc.

Projekte können in so genannte 'Solutions' zusammengefasst werden. Eine Solution kann dazu genutzt werden, Settings für mehrere Projekte einheitlich zu setzen. Build-Abhängigkeiten zwischen einzelnen Komponenten, die sich referenzieren werden durch Solutions automatisch verwaltet und die einzelnen Bestandteile der Solutions in der richtigen Reihenfolge kompiliert (MAKE Mechanismus). Zur Verwaltung der Solutions ist in der IDE der so genannte 'Solution Explorer' integriert. Dieser bietet das Management der Solution bzw. Projekte in einer hierarchischen Sicht an. In einer Solution können Projekte verschiedener Programmiersprachen verwaltet werden.

2.1.3 Server Explorer

Der Server Explorer erlaubt das Management verschiedener Systemressourcen in der Visual Studio. NET IDE. Dazu zählen Datenbankverbindungen, System-Ereignisse (Event Viewer), Windows Dienste, Performance Counter und MSMQ Message Queues.

Der Server Explorer erlaubt auch das Management anderer Server, wenn die entsprechenden Zugriffsrechte vorhanden sind. Außer dem SQL Server können alle Datenbankserver verwaltet werden, die über einen OLE DB Treiber verfügen (z.B. Oracle, DB2 auf dem Host, etc.). Die Data Connections ermöglichen z.B. die Verwaltung von Tabellen, Stored Procedures, Views, etc.

2.1.4 Entwicklung von Web Services

Visual Studio.NET bietet dem Entwickler eine komfortable Umgebung zur Implementierung von Web Services bzw. Web Service Clients.

2.2 Sourcecodeverwaltung

Visual SourceSafe (VSS) ist die integrierte Sourcecodeverwaltung in Visual Studio.NET und bietet dem Entwickler die Verwendung der klassischen Befehle (Check-In, Check-Out) in Solutions und Projekten, ohne die Umgebung verlassen zu müssen. Dies fördert die Entwicklung von Software im Team, da durch das Aufsetzen identischer Arbeitsplätze und die Verwendung einheitlicher Umgebungen die Komplexität des Prozesses reduziert wird.

2.3 Virtuelles Privates Netzwerk

Das Netzwerk der PG 446 verfügt über eine Außenanbindung durch eine Virtuelles Privates Netzwerk (VPN), das über die folgenden Einstellungen erreichbar ist:

VPN Server	pg446.cs.uni-dortmund.de
IP-Adresse	Automatisch beziehen
DNS Serveradresse	Automatisch beziehen
Standardgateway für Remotezugang verwenden	Deaktiviert
DNS Suffix	pg446.cs.uni-dortmund,de
Benutzername	<Vorname> <Nachname>

2.4 Literatur

[ARCH02] Tom Archerm, Andrew Whitechapel: Inside C# - 2nd ed., Microsoft Press, Redmond,Washington 2002

[RI02] Jeffrey Richter: Applied Microsoft :NET Framework Programming, Microsoft Press, Redmond,Washington 2002

[PETZ02] Charles Petzold: Programming Microsoft Windows with C#, Microsoft Press, Redmond,Washington 2002

[SCE0] David Sceppa: Microsoft ADO.NET, Microsoft Press, Redmond,Washington 200

3 Pflichtenheft

In diesem Kapitel werden die Ziele aufgelistet und näher erläutert, die sich die PG am Anfang in der Designphase gestellt hat. Dabei wird auf den erörterten Hintergrund zu Decision Support Systemen eingegangen, sowie auf die Funktionalität des von uns zu entwickelnden Systems. Dabei werden alle Ziele genannt unabhängig davon ob sie im existierenden System umgesetzt wurden. Ein Vergleich zwischen den hier definierten Zielen und dem tatsächlichen system wird in einem späteren Kapitel gamacht.

3.1 Theoretischer Hintergrund

Wir haben für die Erstellung des Pflichtenhefts lediglich angenommen, dass folgende Strukturen vorliegen:

- Fragen
- Antworten
- Lösungen

Wobei wir davon ausgehen, dass die Lösungen in Form von „überarbeiteten“ Beispielfällen vorliegen. Wir unterscheiden bei den Lösungsdokumenten zwei Arten:

- **Primärdokumente**
  Primärdokumente sind solche, die durch Fragen und Antworten erreicht werden können. Sie zeichnen sich dadurch aus, dass zu dem detaillierten Lösungsweg auch noch Zusatzinformationen gespeichert werden (wie

Schlagwortliste, letzter Aufruf, Anzahl der Aufrufe, ...). Sie decken zusammen einen Großteil der Anfragen ab.

- **Sekundärdokumente**
  Sekundärdokumente enthalten die Zusatzinformationen nicht. Sie werden separat gespeichert. Es handelt sich um nicht unbedingt überarbeitete Dokumente. Außer dem Lösungsweg ist lediglich noch ein Berechtigungsflag enthalten. Sekundärdokumente sind alle Probleme, die je von einem Mitarbeiter bearbeitet wurden.

Warum die Zweiteilung der Dokumente?
Aufgrund von Recherchen von einzelnen Mitgliedern in der PG stellte sich heraus, dass diese zweigeteilte Struktur ist in der Wirtschaft so anzutreffen ist. Der Grund für Ihre Existenz ist einfach. Sehr spezielle Probleme, die selten auftreten, würden ein DSS langsam machen, da jede Möglichkeit ausgeschlossen werden muss. Es gibt viele solcher Probleme, man denke nur mal an Softwarekonflikte unterschiedlicher Versionen und Programme.

Am Beispiel: HP unterstützt zurzeit drei Betriebssysteme.

- HP UX 10.20
- HP UX 11.00
- HP UX 11.11

Jede dieser Versionen verfügt über zahlreiche mögliche Patches, die installiert oder nicht installiert sein können, sowie verschiedenste Kundensoftware, die installiert ist. Alle Rechnertypen arbeiten in einem heterogenen Netzwerk. Die möglichen Kombinationen sind unüberschaubar.

(3 BS * 25 PATCHES * 20 Softwarepakete) ^ 100 # der Rechner = 1500^100

Es können also nicht alle Kombinationen abgebildet werden. Jeden Fehler der auftaucht ins Netz zu übernehmen wäre auch schon allein deswegen falsch, weil der Fehler evtl. nur bei diesem Kunden auftreten kann, da er der einzige mit dieser Konfiguration war. Es kann auch sein, dass diese Konfiguration nicht ganz so selten ist, aber immer noch nicht häufig. Würde dieser Fall in ein DSS übernommen, müssten alle häufigen Suchanfragen diese Möglichkeit ausschalten, was die durchschnittliche Rechenzeit erhöhen würde. Zudem ist es mit einem hohen Aufwand verbunden, sämtliche Dokumente so zu überarbeiten, dass sie ohne weiteres dem Kunden präsentiert werden können. Möglicherweise gibt es auch Probleme, die nur von einem Support Mitarbeiter gelöst werden können und die der Kunde nicht direkt sehen soll. Solche Lösungen möchte man trotzdem speichern. Zu bemerken ist noch, dass alle Sekundärdokumente zusammen nur einen kleinen Prozentsatz der Gesamtanfragen ausmachen, ihre Menge jedoch groß gegenüber der der Primärdokumente ist. Die Größenordnung liegt bei ungefähr 100000 Sekundärdokumente zu 100 Primardokumenten. Um die Rechenzeit im Durchschnitt gering zu halten empfiehlt sich demnach die Zweiteilung. Dagegen spräche, dass ein solches DSS nicht abgeschlossen wäre. Allerdings kann man dem Benutzer eine Suche über die Sekundärdokumente anbieten. Ein weiterer Einwand ist, das manche Dokumente immer nur über die Suche zu finden wären. Die von der PG gefundene Lösung dazu sieht vor, dass häufig referenzierte Sekundärdokumente zur Aufnahme in die Primärdokumente vorgeschlagen werden können. Damit ist gibt es die Möglichkeit Probleme von ausgebildeten Mitar-

beitern zu Primardokumenten umwandeln zu lassen. Die Folge dass irgendwann alle Sekundärdokumente in Primärdokumenten umgewandelt worden sind trifft mir hoher Wahrscheinlichkeit nicht ein, da die Bedingung, dass die Sekundärdokumente nur einen kleinen Prozentsatz der angefragten Probleme ausmachen, greift. Schließlich ist die die Idee ja nur die häufig angefragten Dokumenten umzuwandeln.

3.2 Funktionalität der Oberflächen

In diesem Abschnitt werden alle Benutzer Interfaces erläutert die das System ursprünglich haben sollte. Dabei wird auf die unterschiedliche Nutzung durch die verschiedenen Benutzergruppen, also dem Endkunden und den Call – Center Mitarbeiter eingegangen.

3.2.1 Endkunden-Ansicht

Die Endkunden-Ansicht gliedert sich in 7 Bereiche.

- Modellbaum
- Eine Suchfunktion
- Fragenauswahl
- Detailbeschreibung für die ausgewählte Frage
- Gefundene Lösungen
- Angezeigte Lösung
- Kontakt

3.2.1.1 Modellbaum

Der Modellbaum dient der Gliederung des Problembereiches in disjunkte Teilbereiche. Die Funktionalität ähnelt der eines Dateibrowsers. Wird ein Blatt ausgewählt, erscheinen die zugehörigen Fragen im Frage-Auswahlbereich. Bei Anklicken eines inneren Knotens wird dessen Kinder angezeigt bzw. verborgen.

3.2.1.2 Suchfunktion

Der Benutzer kann über alle gespeicherten Fälle suchen. Dazu gibt er einen String ein und drückt auf den Button „Suchen“. Es gibt einen Ausgabebereich in dem alle gefundenen Lösungen der Suche angezeigt werden. Es gibt drei Kategorien von Suchtreffern:

- **Knoten des Modellbaums**

  Wird von der Suche ein Knoten des Modellbaumes gefunden, so wird dieser im Lösungsbereich angeben. Der Benutzer kann den knoten auswählen und das System schränkt sich im Modellbaum direkt auf den entsprechenden Knoten ein. Dem Benutzer wird freigestellt, dennoch alle Dokumente nach dem Suchstring zu durchsuchen.
- **Primärdokumente**

  Falls die Suche Primärdokumente findet, werden diese im Lösungsbereich angezeigt. Der zugehörige Knoten im Baum wird ebenfalls mit ausgeben. Falls der Benutzer mit den Suchergebnissen unzufrieden ist, kann er über die Sekundärdokumente suchen.
- **Sekundärdokumente**

  Wenn die Suche keine Primärdokumente findet, werden die

Sekundärdokumente durchsucht (eventuell nur auf Wunsch des Benutzers), und die entsprechenden Ergebnisse im Lösungsbereich angezeigt.

Bei allen Ausgaben wird der Modellbaum nicht verändert. Erst wenn der Benutzer im Lösungsbereich eine gefundene Lösung ausgewählt hat, kann sich der der Modellbaum verändern.

Darüber hinaus werden möglicherweise noch folgende Funktionen bereitgestellt:

- Häufig gesucht (häufigste Suchanfragen werden dem Benutzer präsentiert)
- Meinten Sie…
- Expertensuche (Suchfelder spezifizieren)

3.2.1.3 Fragenauswahl

In diesem Bereich werden die aktuell vom System favorisierten Fragen geordnet nach ihrer Relevanz oder, falls diese zu lang sind, deren Zusammenfassung angezeigt. Antworten sind hier noch nicht zu sehen.

Wird eine Frage markiert, wird sie in der Detailbeschreibung angezeigt.

3.2.1.4 Detailansicht der Fragen

In der Detailansicht wird die markierte Frage angezeigt. Dem Benutzer wird eine detaillierte Anleitung (falls nötig) zur Bearbeitung dieser Frage gegeben. Zudem werden die Antwortmöglichkeiten aufgelistet. Der Benutzer kann nun eine der Antwortmöglichkeiten auswählen.

3.2.1.5 Gefundene Lösungen

Hier werden Kurzbeschreibungen der vom System berechneten wahrscheinlichsten Lösungen angezeigt. Die markierte Lösung wird in der Detailansicht der Lösungen angezeigt.

3.2.1.6 Detailansicht der Lösungen

Die komplette Lösung wird angezeigt. Auch ihre Zusatzinformationen wie ID oder Schlagwörter werden aufgelistet. Der Benutzer kann die Lösung als hilfreich oder nicht hilfreich bewerten.

3.2.1.7 Kontakt

Dem Benutzer werden Kontaktmöglichkeiten aufgelistet. Er hat die Möglichkeit, einen Supportmitarbeiter anzurufen, eine Email zu schreiben oder einen „ContactMe“ Button zu drücken. Entscheidet er sich für letztere Möglichkeit, kann er die Kontaktmodalitäten bestimmen (Kontaktart, Zeit, …). Er wird dann von einem Support Mitarbeiter kontaktiert.

Die Kontaktinformationen hängen vom zuvor ausgewählten Modell ab, d.h. sie gehören zu einer Gruppe von Agenten, die vermutlich besonders geeignet sind, das Problem zu lösen.

3.2.1.8 Sonstiges

Das Programm soll zudem noch folgende Funktionen bereitstellen

- **Vor/Zurück (alternativ: History-Funktion)**
  Der Benutzer kann durch die gemachten Aktionen navigieren und Änderungen an diesen vornehmen. Alternativ werden alle gemachten Aktionen untereinander aufgelistet. Eine Einzelne kann ausgewählt und verändert werden.
- **Drucken**
  Dem Benutzer werden mehrere Druckmöglichkeiten präsentiert, von denen er eine auswählen und ausführen kann.
- **Speichern/Öffnen**
  Der Benutzer kann seine Anfrage speichern. Dazu bekommt seine Anfrage eine Fallkennung. Der Anfrageverlauf wird unter der Fallkennung auf dem Server gespeichert. Beim Öffnen kann der benutzer unter Angabe der Fall-ID einen gespeicherten Fall öffnen.
- **Fallkennung**
  Eine eindeutige Kennung, die vom System vergeben und auch angezeigt wird.

3.2.2 Call-Center-Ansicht

Die Call-Center-Ansicht gliedert sich in 4 Bereiche, wobei von den Benutzerrechten des Agenten abhängt, welche Bereiche angezeigt werden. Die Benutzerrechte werden über die Windows-Anmeldung des Agenten verwaltet. Die verfügbaren Bereiche sind:

- Case Explorer
- Call Editor
- Case Creator
- Super Admin

3.2.2.1 Case Explorer

Entspricht der Endkunden-Ansicht, nur dass der Kontakt nicht aufgeführt ist. Zudem sind Einschränkungen bei der Suchfunktion aufgehoben.

3.2.2.2 Call Editor

Im Call Editor kann der Agent einen Fall, den er bearbeitet, protokollieren. Der Editor gliedert sich in 3 Bereiche:

- **Baumstruktur**
  In hierarchisch organisierten Mailboxen werden die Fälle, die von den Agenten bearbeitet werden sollen, angezeigt. Es gibt globale, private und Gruppen- Ordner, wie Inbox, Important, Pending und Call-Back, in denen die Fälle einsortiert sind. Es gibt die Möglichkeit, private Boxen anzulegen, zu löschen oder umzubenennen. Autorisierte Benutzer können auch die globalen Boxen bearbeiten. Agenten, die die Aufgabe haben, auf Anfrage Fälle in das DSS aufzunehmen, haben eine zusätzliche Box für Fälle, die von anderen Agenten zur Weiterverarbeitung vorgeschlagen wurden (siehe unten).

- **Informationsbereich**
  In diesem Bereich sind grundlegende Informationen zum Fall zu finden.
  Der Informationsbereich Case ID ermöglicht das Anzeigen und Eingeben der Kennung eines Falles. Der zur eingegebenen Kennung gehörende Fall erscheint, wenn ein Button „Gehe Zu“ gedrückt wird. Im Bereich Customer Data werden die Benutzerdaten, wie Name, Telefonnummer, Adresse und Firmenname, angezeigt. Der Bereich Call Information informiert den Agenten über die bereits vom DSS erhobenen Informationen.

Protokollbereich

Alle den Fall betreffenden Vorgänge werden in folgenden Kategorien protokolliert:

- **Notes**
  Unter diesem Punkt kann der Agent sich die weitere Vorgehensweise notieren, zum Beispiel, wann er den Kunden wieder anrufen soll.
- **Problem**
  Zunächst wird eine Kurzbeschreibung des Problems eingegeben. Anschließend folgt eine detaillierte Beschreibung des Problems, die sich durch weitere Gespräche mit dem Kunden verändern kann. Zur Unterstützung des Prozesses kann ein autorisierter Agent diesem Bereich vordefinierte Eingabe-Objekte (Radio Buttons, Checkboxes und Textfelder) hinzufügen. Beispielsweise wird beim Support für Software die Frage nach dem Betriebssystem sehr häufig auftauchen. Diese Frage kann dann mit Ihren Antwortmöglichkeiten im Problembeschreibungs-Bereich angezeigt werden.

- **Action**
  Der Agent kann hier bereits durchgeführte Versuche zur Problemlösung beschreiben.
- **Link**
  Es können Verknüpfungen zu Primärdokumenten angegeben werden, die bei der Lösung des Problems geholfen haben. Dabei kann zu jeder Verknüpfung ein kurzer Text eingegeben werden, der beschreibt, inwiefern das Dokument hilfreich war.
- **Report**
  Der Endbericht des Agenten zu einem abgeschlossenen Fall kann hier eingegeben werden.

Zudem stellt diese Ansicht folgende Funktionen zur Verfügung:

- **New Case**
  Es öffnet sich ein Dialog, in dem der Agent die relevanten Daten eingibt. Eine Case ID wird automatisch generiert. Außer dem Namen des Kunden und einem Titel sollten auch die eventuell vorhandenen DSS-Informationen (über Angabe der eindeutigen ID des DSS-Vorgangs) angebunden werden.
- **Save Case**
  Fälle können gespeichert werden, so dass über die Boxen oder die Case ID auf sie zugegriffen werden kann.
- **Publish Case**
  Es erscheint ein Dialog, um den Case zur Übernahme in das DSS vorzuschlagen. Eventuell wären hier Prioritäten anzugeben. Der Fall wird in eine spezielle Box verschoben, auf die nur autorisierte Benutzer Zugriff haben.
- **Grab Case**
  Fälle, die in den Boxen anderer Agenten liegen, können unter Angabe der Case ID zum aktuellen Benutzer transferiert werden. Dies wird nötig wenn ein Agent aus irgendwelchen Gründen, wie Krankheit oder Urlaub, nicht zugegen ist, und der Fall trotzdem bearbeitet werden musss.

3.2.2.3 Case Creator

Diese Ansicht dient dazu, Fälle für die Übernahme in die Menge der Primärdokumente umzuwandeln. Die Ansicht ist horizontal geteilt in eine Modellauswahl und Möglichkeiten zur Bearbeitung des ausgewählten Modells.

- **Modellauswahl**
  In einem Baum, der identisch zu dem der Endkunden-Ansicht ist, können Modelle angelegt, verschoben, kopiert, gelöscht und zur Bearbeitung ausgewählt werden.
- **Modellbearbeitung**
  Auf der linken Seite sind alle Fragen und Probleme (sortiert nach eben diesen Kategorien) aufgeführt, die im ausgewählten Modell vorkommen. Es können Fragen und Probleme angelegt, gelöscht oder zur Bearbeitung ausgewählt werden. Der rechte Bereich dient der Anzeige und Bearbeitung von Detailinformationen zum links ausgewählten Objekt.

Wenn links eine der beiden Kategorien ausgewählt ist, werden rechts alle Objekte der ausgewählten Kategorie aufgelistet.
Wurde links eine einzelne Frage gewählt, finden sich rechts folgende Details. Der Fragetext, sowie nähere Informationen dazu, wie die Frage zu beantworten ist und die Antwortmöglichkeiten die die Frage hat. Zudem werden alle Lösungen die von dieser Frage beeinflusst werden, mit dem dazugehörenden Grad der Beeinflussung, der vorgegebene Stärken annehmen kann, angezeigt.
Wurde links ein Problem bzw. eine Lösung ausgewählt, werden rechts folgende Details dargestellt. Eine Kurze und Ausführliche Problembeschreibung, analog zu oben alle Fragen die mit dem Problem verknüpft sind und die dazu passende Beeinflussung.

3.2.2.4 Super Admin

Diese Ansicht ermöglicht es einem speziell autorisierten Administrator des Systems verschiedene Einstellungen festzulegen, die den Umgang mit dem System für die anderen Agenten erleichtern soll. So kann der Administrator z.B. vordefinierte Eingabemasken für Betriebssysteme in dem oben erläuterten Protokollbereich einbetten.

4 Grundlegende Begriffe

4.1 Decision Support System (DSS)

Decision Support System heißt übersetzt entscheidungsunterstützendes System. Ein DSS soll dem Menschen beim Treffen von Entscheidungen helfen. Um dies zu erreichen werden viele Ansätze verfolgt. Die meisten Ansätze versuchen dabei dem Benutzer für seine Entscheidung relevantes Wissen in aufgearbeiteter Form zur Verfügung zu stellen. Dafür erarbeiten sie Modelle, in die Wissen eingepflegt wird. Dabei soll das Modell möglichst einfach sein und sich für spätere Berechnungen eignen, aber gleichzeitig das ganze Wissen erfassen können. Ein weiteres Problem von DSSen ist das enthaltene Wissen. Ein DSS kann nur dann gut arbeiten, wenn das enthaltene Wissen vollständig und aktuell ist. Das setzt aber voraus, dass das DSS gepflegt wird. Hier ergibt sich das Problem durch die für die Pflege benötigte Zeit. Viele Anwender sehen es als nicht so wichtig an das System zu pflegen und der sich dadurch ergebende schlechtere Informationsstand im System führt dazu, dass weniger Leute das System benutzen.

4.2 Case Based Reasoning (CBR)

CBR ist eine Technik für DSS. Jedes Problem wird als Fall betrachtet, indem ein Benutzer nach und nach Indizien oder Symptome angibt. Die einzelnen Anwendungsfälle (Cases) werden gespeichert und mit einem späteren Fall verglichen. Ähnelt der aktuelle Fall einem älteren in der Datenbank, so wird der alte Fall als potentielle Lösung dem Anwender zur Verfügung gestellt. Ähnelt der Fall keinem in der Datenbank, so wird mithilfe eines Experten eine Lösung bestimmt und anschließend diese Lösung dem Benutzer zur Verfügung gestellt und als neuer Fall zur Basis hinzugefügt.

4.3 Bayes'sche Netze

Bayes'sche Netze sind graphische Modelle die Graphentheorie und Wahrscheinlichkeitstheorie miteinander kombinieren. Sie sind eine einfache, graphische Notation für bedingte Unabhängigkeit und damit für die kompakte Spezifikation vollständiger gemeinsamer Verteilung.
Der beste Weg Bayes'sche Netze zu verstehen ist, sie sich als Modell einer Situation vorzustellen, in der Kausalität eine Rolle spielt in der aber unsere Vorstellung der Vorgänge unvollständig ist und wir deshalb eine probabilistische Art brauchen diese zu beschreiben. Es folgt eine exakte Definition
.
Bayes'sche Netze sind Graphen für die gilt:

Die Knoten des Graphen sind eine Menge von Zufallsvariablen
Die Knoten sind paarweise durch eine Menge von gerichteten Kanten verbunden
Jeder Knoten hat eine Tabelle bedingter Wahrscheinlichkeiten (conditional probability table (CPT)), die den Effekt der Eltern auf den Knoten wiedergibt
Der Graph ist gerichtet und azyklisch

Die intuitive Bedeutung einer Kante von X nach Y ist, das Knoten X direkten Einfluss auf Knoten Y hat.

4.4 Inferenz

Inferenz ist die Auswertung eines Bayes'schen Netzes. In die Berechnung gehen bekannte Ereignisse als Evidenz ein. Die Inferenz berechnet für ein Ereignis die Wahrscheinlichkeit, dass es eintritt unter Berücksichtigung der gegebenen Evidenz. Wenn wir in unserem Programm Inferenz betreiben meint das, dass wir für jedes noch nicht evidente Ereignis einmal den Inferenzalgorithmus aufrufen.

4.5 Literaturangaben

Decision Support Systems & Case Based Reasoning:

[POW01]D. J. Power, Supportting Decision Makers: An Expanded Framework, University of Northern Iowa, 2001

http://dssresources.com/

Bayes'sche Netze + Inferenz:

[CHAR91] E. Charniak, *Bayesian Networks without Tears*, American Association for Artificial Intelligence, Menlo Park, 1991

[PEARL88] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, 1988

[RUSS/NOR95] S. Russell, P. Norvig, *Artificial Intelligence – A Modern Approach*, Prentice Hall, Upper Saddle River, 1995

[RIED03] M. Riedmiller, *Vorlesungsskript „Einführung in die künstliche Intelligenz"*, Dortmund, 2003

[JENS01] F. V. Jensen, Bayesian *Networks and Decision Graphs*, Springer, 2001

Teil II
Das Projekt und seine Komponenten; Dokumentationen; Tests

5 Adiuware- Die Architektur

Die Architektur des Adiuware-Systems wurde im Hinblick auf Modularität und Skalierbarkeit entworfen. Die Modularität wird durch eine Aufteilung des Systems in vier Kernmodule erreicht. Diese sind die Logik, die Datenbank, die Windows-Applikation und die Weboberfläche. Jedes Modul ist völlig unabhängig von den anderen und kann jederzeit ausgetauscht werden, ohne dass die anderen Module verändert werden müssen.

Der Kunde greift über seinen Internet Browser auf den Webserver zu und stellt über die Weboberfläche seine Anfragen an Adiuware. Das Logikmodul holt sich die nötigen Daten aus der Datenbank, führt alle Berechnungen durch und leitet die Daten zur Ausgabe an die Weboberfläche weiter.

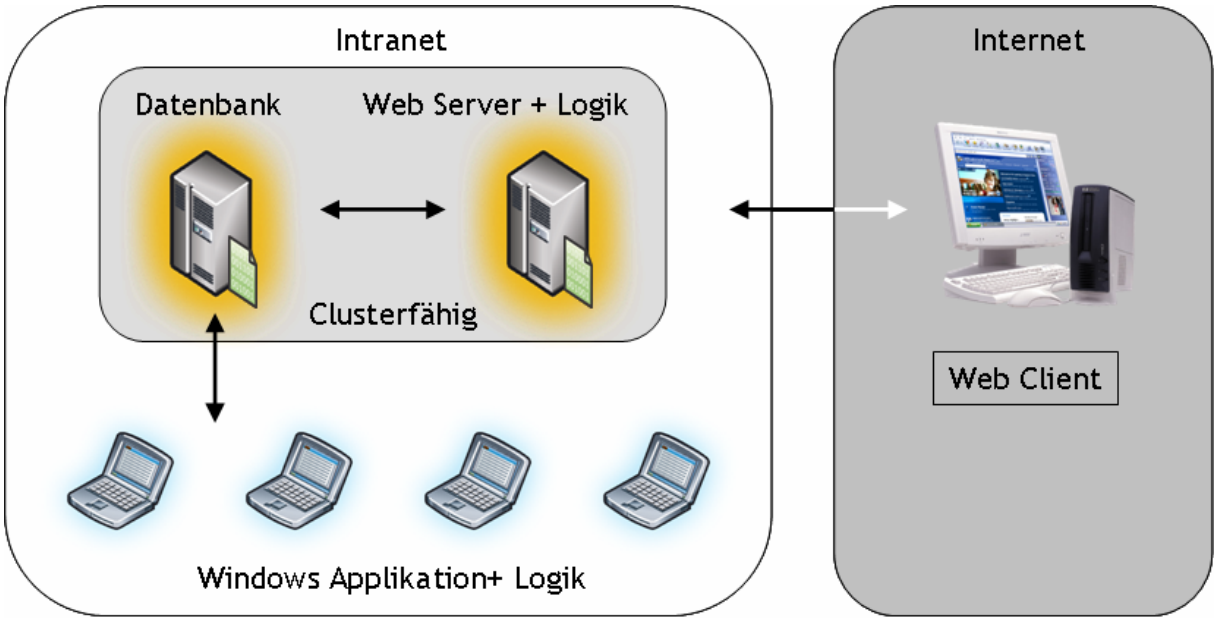


Abbildung 1 – Architektur

Der Supportmitarbeiter nutzt zum Zugriff auf Adiuware eine eigene Windowsapplikation, die auf seinem Arbeitsplatzrechner läuft. Diese Applikation leitet alle Anfragen wiederum an die Logik weiter. Diese läuft allerdings ebenfalls auf dem Rechner des Mitarbeiters und greift über das Intranet auf den Datenbankserver zu um an die nötigen Daten zu kommen.

Jede Ausgabe von Daten erfolgt immer über die Logik, weder die Windowsapplikation, noch die Weboberfläche haben direkten Zugriff auf die Datenbank.

Um die Skalierbarkeit zu gewährleisten sind die Datenbank, das Logikmodul und der Webserver clusterfähig, sie können also auf mehreren Rechnern verteilt laufen. Somit kann auch eine große Anzahl von Anfragen gleichzeitig bearbeitet werden.

Da die Logik auf dem Arbeitsplatzrechner des Supportmitarbeiters jeweils nur eine Personbedienen muss, ist Clustering hier nicht vorgesehen.

5.1 Architekturbeschreibung zu Adiuware.Data

Eine Datenbank ist ein elektronisches System zum Speichern, Verwalten und Auslesen von Daten und kann mit einer Datendatei verglichen werden.
Die Daten selbst können entweder “alphanumerisch“, also aus Buchstaben und Sonderzeichen bestehen, und somit Daten und Textteile darstellen oder aber „numerisch“ sein, sodass sich die Daten ausschließlich aus Zahlen zusammensetzten.

Um überhaupt mit Datenbanken arbeiten zu können, wird ein Datenbankmanagementsystem (DBMS) benötigt, mit dessen Hilfe Datenbanken aufgebaut, gepflegt, verändert und abgerufen werden können.
Mit einem DBMS wird zudem die Sicherheit, Integrität, Konsistenz und der Zugriff auf die Daten gewährleistet, sodass auch mehrere Benutzer gleichzeitig auf die Daten zugreifen können.

Die Abfrage von Informationen aus einer Datenbank erfordert des Weiteren eine besondere Abfragesprache, wie z.B. SQL (Structured Query Language).
Genauer gesagt dient SQL als eine Schnittstelle zwischen Anwender und System, mit dem der Anwender dem System mitteilen kann, was er gerne aus dem Datenbestand erfahren möchte.

Einige Beispiele für Datenbanken sind:

- Personaldatenbanken in Firmen, Patientenkarteien in Arztpraxen, Versichertenkarteien,
- Telefonbuch, Adressverzeichnisse,…
- Speicherung wissenschaftlicher Forschungsergebnisse wie z. B. Gendatenbank

Die Benutzung einer Datenbank ist also vor allem dann sinnvoll, wenn in einem Projekt bzw. Unternehmen eine Menge von Daten gespeichert und verwaltet werden müssen, so dass
eine Datenbank nicht nur für klein angelegte OLTP- Anwendungen (Online Transaction Processing), sondern auch für groß ausgerichtete OLTP-, Data Warehousing- und E-Commerce Anwendungen sehr nützlich ist.

Nachdem in diesem Abschnitt die Definition und der Aufbau einer Datenbank sowie weitere wichtige Begriffe erläutert wurden, werden im nächsten Abschnitt die zentralen Aufgaben und Anforderungen, die von Datenbanksystemen erfüllt werden müssen beschrieben.
Im zweiten Teil dieses Kapitels wird ein Überblick über die Datenbank gegeben, die von der Projektgruppe 446 verwendet wurde und die Gründe dazu erläutert.
Und im letzten Teil des Kapitels wird beispielhaft anhand der Klasse Database_Model die Struktur, Funktion und der Aufbau der Datenbankklassen verdeutlicht.

5.2 Architekturbeschreibung zu Adiuware.Logic

Die Logik, also der Teil von Adiuware in dem es um die Verarbeitung von Anfragen geht, kann auf zwei unterschiedliche Weisen von Benutzern angesprochen werden.

Dazu greift die Logik auf die in der Datenbank gelagerten Daten direkt zu und verarbeitet diese. Zum einen werden die Anfragen von Kunden an die Logik weitergeleitet. Das kann zum einen eine beantwortete Frage mit der entsprechend gegebenen Antwort sein, oder ein oder mehrere, logisch miteinander verknüpfte, Suchbegriffe sein. Im ersten Fall wird in dem zugrunde liegenden Adiuware Netz, welches an anderer Stelle noch genauer beschrieben wird, die Wahrscheinlichkeit der so beantwortet Frage auf 1 gesetzt. Damit nun der Benutzer neue Fragen und Lösungen bekommen kann, wird daraufhin der Inferenzalgorithmus angeworfen. Er errechnet für die anderen Fragen und Lösungen im Netz neue Wahrscheinlichkeiten, die sich aus der Beeinflussung der beantworteten Frage auf die anderen Knoten im Netz ergeben. Dazu wird, wie später auch noch näher erläutert wird, die Methode der exakten Inferenz betrieben. Nachdem der Algorithmus nun für alle Knoten im Netz die neuen Wahrscheinlichkeiten bestimmt hat, werden sowohl die drei wahrscheinlichsten Fragen und Lösungen an den Benutzer, sprich an die GUI, von der Logik zurückgegeben. Darauf hin kann der Benutzer das Spiel wieder von vorne beginnen, indem er eine weitere Frage beantwortet. Gibt er einen Suchbegriff ein, wird eine eigens entwickelte Suchmaschine angeworfen, die über vorher generierte Stichwortlisten die Dokumente heraussucht, in deren Stichwortliste der oder die Suchbegriffe drin vorkommen. Die Dokumente werden sortiert nach der Häufigkeit der darin enthaltenen Suchbegriffe an die GUI zurückgegeben. Wie nun die Stichwortlisten und die Netze in das Logikmodul kommen wird im anschließenden Teil erläutert, in dem es um den Zugriff der anderen Benutzergruppe auf die Logik geht.
Nachdem nun grob erläutert wurde wie die Kommunikation der Kunden über die GUI mit der Logik läuft, wird jetzt erklärt wie die Kommunikation der Agenten mit der Logik abläuft. Über entsprechende Eingabemasken werden von den Agenten sowohl neue Fälle in Form von Textuellen Dokumenten, und auch neuen Fragen, mit den dazu gehörenden Wahrscheinlichkeiten und den Beeinflussungen auf andere Fragen, in die DB gestellt, und somit auch der Logik zugänglich gemacht. Damit nun die oben beschriebenen Mechanismen und Methoden auch zur Anwendung kommen können, muss nach jeder Aktualisierung eine Neuberechnung gemacht werden. Dabei werden die Stichwortlisten der einzelnen Dokumente für die Suche erstellt und die CPT Einträge der Fragen und Lösungen in dem veränderten Adiuware Netz neu erstellt. Für letzteres wird ein Generator verwendet, der Simulationen von Benutzerdurchläufen macht, um somit die einzelnen Einträge zu generieren. Näheres zu beiden Verfahren findet sich in der weiteren Beschreibung der Logik. Nachdem nun die Neuberechnung in dieser Weise geschehen ist, kann der Benutzer mittels der oben genannten Methoden die Logik wieder benutzen.

5.3 Architekurbeschreibung von Adiuware.Windows

Die Quellcodearchitektur des Windows-Frontends setzt sich im Wesentlichen aus vier großen Klassen bzw. Sammlungen von Klassen zusammen:

- Die Hauptklasse „AdiWinGui“
- Die Interface-Klasse „Preprocessing“
- Mehrere Klassen zur Kapselung einzelner Elemente der Programmlogik
- Verschiedene kleinere Klassen, die der Benutzerinteraktion dienen

In den folgenden Kapiteln werden diese Klassen bzw. Quellcodedateien näher beschrieben und ihre Funktionsweise z.T. anhand einiger beispielhaft aufgeführter Methoden erläutert.

5.3.1 Die Hauptklasse „AdiWinGui“

Die Haupt- bzw. Startklasse des Windows-Frontends dient nicht nur als Einsprungpunkt bei Programmstart, sondern ist auch für die Darstellung des Hauptfensters der Anwendung verantwortlich.
Ein Großteil der Klassenattribute besteht demzufolge aus den Winforms-Elementen, die für die optische Darstellung der Bedienoberfläche benötigt werden (hierzu zählen z.B. das Pulldown-Menü, die Karteireiter usw.).
Alle Methoden der Klasse dienen einzig dem Zugriff auf diese Anzeige-Elemente, um ihre Darstellung den Nutzereingaben entsprechend anzupassen. Eine direkte Anbindung an die Programmlogik besteht nicht, sondern wird über die im nächsten Kapitel vorgestellte Interface-Klasse realisiert.
Sämtliche Abfragen von Nutzerinteraktion werden über so genannte „Event-Handler“ realisiert. Diese Methoden werden automatisch bei der Betätigung von Steuerelementen der grafischen Oberfläche aufgerufen und veranlassen alle zur Verarbeitung des somit gegebenen Befehls nötigen Aktionen.

5.3.2 Die Interface-Klasse „Preprocessing“

Die Kommunikation mit der dem Programm zugrunde liegenden Verarbeitungslogik und dem Datenbestand, auf dem diese operiert, wird durch die Klasse „Preprocessing“ bzw. die in ihr enthaltenen, statisch deklarierten Methoden ermöglicht.
Dies beginnt bereits mit dem Programmstart, bei welchem zunächst mittels der Methode „public static void init(string dbString)“ eine Anbindung zur Datenbank hergestellt wird. Danach kann durch Aufruf der Methode „public static void loginCurrentUser()“ die Authentifizierung des Benutzers erfolgen.
Erst nachdem diese Methodenaufrufe erfolgreich beendet wurden, können alle weiteren Anfragen an die Datenbank und die Programmlogik durch die Interface-Klasse erfolgen.

5.3.3 Klassen zur Kapselung einzelner Elemente der Programmlogik

Verschiedene Ansichten des Hauptfensters stellen einen Teil des in der Datenbank gespeicherten Datenbestandes dar und eröffnen dem Nutzer verschiedene Interaktionsmöglichkeiten. Exemplarisch seien hierfür die Baumansichten der Karteireiter „Case Explorer“ und „Case Creator“ genannt, welche die Baumstruktur der Anfrage-Modelle grafisch aufbereiten.
Ein derartiges Baumelement bzw. „Treeview“ enthält Instanzen der Standardklasse „TreeNode“, welche jedoch neben einer Zeichenkette für die Darstellung keine weiteren Informationen aufnehmen kann. Da eine Auswahl eines solchen „TreeNodes“ jedoch eine Anfrage an ein Objekt der in der Programmlogik vereinbarten Klasse „ModelTreeNode“ nach sich ziehen kann, muss eine Verbindung zwischen diesen beiden Klassen ermöglicht werden.
Dies geschieht durch die für das Windows-Frontend entworfene Klasse „WinformsTreeNode“, welche alle Eigenschaften eines Standard-„TreeNodes“ erbt (und somit in einem „TreeView“ angezeigt werden kann), darüber hinaus aber auch ein Attribut der Klasse „ModelTreeNode“ beinhaltet und dadurch die Verbindung zwischen grafischer Oberfläche und Programmlogik herstellt.
Alle anderen eingesetzten Kapselungsklassen Arbeiten nach demselben Prinzip und unterscheiden sich nur in der jeweils gekapselten Logik-Klasse bzw. der erweiterten Winforms-Klasse.

5.3.4 Verschiedene Klassen zur Benutzerinteraktion

Ein typischer Programmablauf beinhaltet nicht nur Interaktionen des Benutzers mit dem Hauptfenster. Um ein effizientes und intuitives Arbeiten zu ermöglich, gibt es vielmehr eine große Anzahl von Dialogfenstern, die in bestimmten Situationen erscheinen, um zusätzliche Informationen vom Nutzer zu erfragen – so lässt z.B. ein Betätigen der „Rename“-Schaltfläche des „Case Creators“ einen Dialog erscheinen, der um die Angabe eines neuen Namens für das aktuell gewählte Element in der Baumansicht bittet.
Da diese Klassen jedoch ausschließlich durch einfache Vererbung von Klassen der Standardklassen von „VS.NET“ entstanden sind, wird hier auf eine ausführliche Erläuterung ihres Aufbaus verzichtet.

5.4 Adiuware.Web

Mit Adiuware soll dem Benutzer ein System zur Verfügung gestellt werden, welches Problemanfragen schnell und präzise löst. Für den Benutzer soll sich gegenüber bekannter Standardhilfen nicht viel ändern: Stichworte oder ganze Fragen in ein dafür vorgesehenes Feld eingeben und Lösung angeboten bekommen. Oft ist das Problem, dass zu viele unterschiedliche Lösungen angeboten werden, die ein schnelles Vorgehen fast unmöglich machen. Adiuware versucht von Anfang an, das Problem und seine Lösungen möglichst einzugrenzen (→ Navigationsbaum). So kann man z.B. den Druckertyp angeben, um den es sich bei der Suchanfrage handelt und so den Bereich der möglichen Lösungen verkleinern. Eine grundlegende Neuerung gegenüber herkömmlichen Suchanfragen ist, dass das System auf eine Anfrage des Benutzers seinerseits Fragen anbietet, die auf mögliche Problembereich abzielen. Gibt der Benutzer beispielsweise ein „Drucker druckt nicht“ könnte eine möglich Frage von Seiten des Systems sein „Netzstecker eingesteckt?“. Auf diese Art ist es möglich, Lösungsansätze aus anderen Bereichen möglichst aussen vor zu lassen, um die Anzahl gefundener Lösungsdokumente klein zu halten. Dennoch ist auch eine Suche auf scheinbar unrelevanten Dokumenten möglich und zwar dann, wenn der Benutzer keine Lösung auf den wahrscheinlichsten Dokumenten finde konnte. Ziel des Systems soll sein, möglichst immer eine Lösung anbieten zu können.

Im Folgenden werden die einzelnen Menüs und Masken mit ihren Funktionen im Detail aufgeführt. Dabei wurde im Design darauf geachtet, Adiuware vom Aufbau einer herkömmlichen Internetseite anzupassen, um eine intuitive Bedienung zu erleichtern. Dies beinhaltet weitesgehend Standardfunktionen, wie man sie von herkömmlichen Webanwendungen kennt.

Ingesamt greift Adiuware.Web auf 3 Seiten zu:

- Login Seite
- Interne Start Seite
- Suchseite

Diese Architektur wurde so gewählt, da die drei Seiten vom Inhalt und den Funktionen her sehr unterschiedlich sind. Der Grundaufbau der drei Seiten dagegen ist der Gleiche. Dieser gliedert sich immer in 3 Teile:

- Titelleiste
- Menübereich

- Informations- bzw. Arbeitsfeld

Die Suchseite stellt den zentralen Arbeitsbereich dar. Da hier die meiste Programmierarbeit steckt, soll die Architektur kurz beschrieben werden:

Um die Produkte in einem Baum darstellen zu können, wurden die IEWebControls von Microsoft verwendet, die in späteren VisualStudio.Net Versionen als Standard Control mit eingearbeitet werden sollen. Diese Controls sind frei verfügbar und beinhalten den sogenannten TreeView.
Um die Fragen und Lösungen anzeigen zu können, wurden eigene UserControls geschrieben, die einen hierarchischen Aufbau entsprechen, QuestionContainerControl und SolutionContainerControl. Beide folgen der gleichen Idee.

Kurzbeschreibung des QuestionContainerControls aus dem Quellcode:

```
Der QuestionContainerControl ist in der Hierachie

“QuestionContainerControl",
“SelectionControl” &
"QuestionControl"

auf der obersten Stufe.

Die Fragenfelder die in "QuestionControl" erzeugt werden, werden hier zu-
sammengefasst!
Zudem greift die SelectionControl auf diese Klasse zu:
Es wird ein Fragenobjekt an diese übergeben und die Antwortmöglichkeiten
werden in einer RadioButtonlist aufgelistet!
Damit umgehen wir das Problem, dass ein Fragenobjekt innerhalb der Questi-
onContainerControls "vergessen" werden. Das Problem vorheriger Architektu-
ren ist gewesen, dass man so den Integer-Wert der Antworten nicht mehr ü-
bermitteln konnte!

Das QuestionControl hat folgende Funktion:
Da die Anzahl der Fragen dynamisch ist, werden die Fragen werden diese in
eine Array List geladen. Der Vorteil der ArrayList ist, dass Elemente immer
wieder hinten angehängt werden.
In dem UserControl wird eine Methode benutzt um die Behandlung der Detail-
beschreibungen zu kontrollieren, welche im SelectionControl angezeigt wird.
Der Ablauf soll wie folgt beschrieben sein:
Wird eine entsprechende Frage angeklickt erscheint ein Feld mit der ent-
sprechenden Detailbeschreibung.
Wird eine 2. Frage angeklickt um eine Detailbeschreibung zu erhalten, wird
das gleiche Feld benutzt. Um das Detailfeld auszublenden wird auf die ent-
sprechende Frage geklickt die gerade geöffnet ist.
```

Dieser Lösungsansatz ist zwar sehr aufwendig, dennoch wirkt das Ergebnis optisch integriert in die Architektur der gesamten Seite.

6 Dokumentation zu Adiuware.Data

6.1 Grundlagen einer Datenbank

In diesem Abschnitt werden die allgemeinen Anforderungen und zentralen Aufgaben von Datenbanken erläutert, die erfüllt werden müssen um höchste Daten- Konsistenz und Sicherheit zu gewähren.

6.1.1 Aufgaben einer Datenbank

Die zentralen Aufgaben, die Datenbanken erfüllen müssen sind:

1. Benutzerverwaltung und routinemäßige Verwaltungsaufgaben
2. Sicherheit
3. Sicherung und Wiederherstellung
4. Dienste
5. Optimieren und Überwachung der Leistung und Dokumentation

Aus diesem Grund gibt es für (fast) jede Datenbank ein Datenbankadministrator, der für die Erfüllung der genannten Aufgaben verantwortlich ist.

Benutzerverwaltung und routinemäßige Verwaltungsaufgaben
Die Benutzerverwaltung besteht im Verwalten von Anmeldungen von Datenbankrollen, da alle Benutzer, die die Datenbank verwenden möchten über autorisierte Zugriffsrechte verfügen müssen.
Zu den routinemäßigen Aufgaben gehört das Überwachen der Speicherverwendung der Datenbank, das neue Erstellen von Indizes, das Überprüfen der Gültigkeit von Datenbankobjekten und das Überwachen des Allgemeinzustandes des Systems.

Sicherheit
Eine hohe Sicherheit ist durch die Überwachung des Systems und durch das Aufzeichnen aller auftretender Probleme gewährleistet.
Die Sicherheit eines Systems ist deshalb von besonderer Bedeutung, da durch das Eindringen Unbefugter, die möglicherweise Daten zerstören oder stehlen, sehr hohe Kosten für das Unternehmen bzw. für die Firma entstehen können.

Sicherung und Wiederherstellung
Diese Operationen stellen sicher, dass die Datenbank auch nach einem schweren Hardwarefehler weiterhin unbeschädigt verwendet- und die Datenbank wieder in einen betriebsbereiten Zustand gesetzt werden kann.

Dienste
Um eine möglichst hohe Leistung zu gewährleisten muss sichergestellt werden, dass das System zusätzlich bestimmte Dienste zur Verfügung stellt.

Optimieren und Überwachung der Leistung und Dokumentation
Die Leistung eines Systems muss kontinuierlich überwacht werden, wobei alle Änderungen aufgezeichnet werden sollten.

Mögliche Probleme können sich durch eine stärkere CPU- Auslastung, höhere Antwortzeiten,... ankündigen.

6.1.2 Anforderungen an eine Datenbank

An alle Datenbanken werden folgende Anforderungen gestellt:

1. Sicherheit / Authentifizierung
2. SQL Injections
   - Verhinderung von SQL- Injections
3. Skalierbarkeit
   - Clustering
   - Load Balancing
4. Performance

Sicherheit und Authentifizierung
Ein wichtiger Bestandteil vieler Webanwendungen ist die Fähigkeit, Benutzer zu identifizieren und den Zugriff auf Ressourcen steuern zu können.
Das Ermitteln der Identität einer anfordernden Entität wird als Authentifizierung bezeichnet.
Im Allgemeinen muss der Benutzer zur Authentifizierung Anmeldeinformationen wie Benutzername und Kennwort angeben und sobald eine authentifizierte Identität verfügbar ist, muss ermittelt werden, ob diese Identität auf eine angegebene Ressource zugreifen darf.
Diesen Prozess bezeichnet man als Autorisierung.

Einige Anwendungen passen außerdem Inhalte anhand der anfordernden Identität oder anhand einer Gruppe von Rollen, zu denen eine anfordernde Identität gehört, dynamisch an.
So können ASP.NET Framework- Anwendungen z. B. dynamisch überprüfen, ob die derzeit anfordernde Identität einer bestimmten Rolle angehört.
Dies ist vor allem dann sinnvoll, wenn beispielsweise in einer Anwendung geprüft werden soll, ob der aktuelle Benutzer zur Rolle "Manager" gehört, um in diesem Fall Inhalte für Manager zu erzeugen.

- Zusammen mit IIS stellt ASP.NET Authentifizierungs- und Autorisierungsdienste für Anwendungen bereit, die den Zugriff für Benutzerkonten verweigern, denen nicht ausdrücklich Zugriff auf eine Datenbank, eine Tabelle oder eine Ansicht gewährt wurde.

Hinweis:
ASP.NET- Anwendungen werden standardmäßig im Kontext des ASPNET- Benutzerkontos ausgeführt. Diese müssen den Zugriff auf das ASPNET- Benutzerkonto gewähren, damit überhaupt eine ASP.NET- Anwendung die Daten in einer Datenbank lesen und aktualisieren kann. Dazu muss dem ASPNET- Benutzerkonto nur minimale Berechtigungen zugewiesen werden. Dies hält den potentiellen Schaden an einer ASP.NET- Anwendung in Grenzen, falls diese von einem Angreifer manipuliert wird.

SQL- Injections

Bei einer SQL- Injection versucht ein Angreifer SQL- Anforderungen in eine Datenbank einzuschleusen oder die Abfragen so zu manipulieren, dass man zusätzliche Daten in der Datenbank erhält.
In den meisten Fällen geschieht dies über die Applikation, die den Zugriff auf die Datenbank bereitstellt, so dass der Hacker auf beliebige Tabellen und Daten zugreifen und deren Inhalte manipulieren kann.
(z.B. die Manipulation von Kreditkartennummern, die ein Webshop zu Abrechnungszwecken gespeichert hat.)

Folgende Punkte erhöhen die Gefahr von SQL- Injections:

- Überprüft eine Web-Applikation Benutzereingaben nicht ausreichend, ist damit jede Datenbank SQL- Injection anfällig.
- Die Gefahr einer SQL- Injection wird weiter erhöht, wenn mit der Anwendung Verbindungen über ein Konto hergestellt werden, welches zu viele Berechtigungen besitzt.
- Weiter muß darauf geachtet werden, dass solange wie injizierter SQL-Code syntaktisch richtig ist, keine Manipulationen (serverseitig) programmgesteuert entdeckt werden können.

Verhinderung von SQL- Injections

Zur Verhinderung von SQL- Injections müssen folgende Punkte berücksichtigt werden

- Alle Benutzereingaben sowie Typen müssen geprüft werden, indem parametrisierte *gespeicherte Prozeduren (Stored Procedures)* aufrufen werden. Die Überprüfung der Benutzereingaben erfolgt dabei immer durch Testen des Typs, der Länge, des Formats und des Bereichs. Jeder Wert außerhalb des zulässigen Bereichs löst dabei eine Ausnahme aus und nicht überprüfte Eingaben verursachen Programmfehler, die von Hackern als Eintrittspunkt in das System verwendet werden.
- Zu beachten ist außerdem, dass alle Programme, die für die Ausführung in einer sicheren Umgebung vorgesehen sind, in eine unsichere Umgebung kopiert werden können.
- Eine weitere Möglichkeit zum Schutz vor SQL- Injections ist das Filtern der Eingaben (Entfernen von Escapezeichen). Dies ist allerdings wegen der großen Anzahl an Zeichen, die Probleme bereiten können, aber kein zuverlässiger Schutz.

Die *Aufgaben und Eigenschaften von gespeicherte Prozeduren (Stored Procedures)* die zur Prüfung von Benutzereingaben und Typen benutzt werden um SQL- Injections zu verhindern werden im folgendem näher erläutert:

Die allgemeine Aufgabe von *Stored Procedures* ist, SQL- Statements beim Zugriff auf die Datenbank “zubündeln“, so dass die meisten vom Clienten an den Server übergebenen SQL- Statements durch eine einzige *Stored Procedure* ersetzt werden.

Sie erleichtern den Benutzern außerdem den Zugriff auf die Datenbank, so dass diese nicht mehr die architektonischen Details der Tabellen kennen müssen und auch nicht direkt auf die Tabellendaten zugreifen, sondern lediglich die Prozeduren der gewünschten Aufgaben ausführen.

Des Weiteren gilt:

- In einem Programm übergibt man die Benutzereingaben den *Stored Procedures*, erst hier wird eine SQL- Query erzeugt und ausgeführt.
- Die *Stored Procedures* sind deshalb besonders sicher, weil sie keine weiteren SQL- Befehle in Benutzereingaben akzeptieren.
- *Stored Procedures* sind Routinen, die auf dem Datenbankserver installiert sind und vom Client aus gestartet werden können, d.h. sie sind in kompilierter Form in der Datenbank gespeichert und ermöglichen prozedurale Programmierung. Diese müssen nicht zum Client übertragen werden sondern laufen direkt auf dem Server ab, wodurch die Abarbeitung gesteigert wird und die Netzbelastung sinkt.
- Auch SQL-Anweisungen werden in *Stored Procedures* im allgemeinen schneller ausgeführt, da diese schon beim Kompilieren analysiert und somit „fest verdrahtet" werden und nicht erst zur Zeit ihrer Ausführung.
- Die Erzeugung einer *Stored Procedure* erfolgt mit einer CREATE-Anweisung:

CREATE PROCEDURE prozedur_name [(<parameter>[<parameter>,...]) AS

[(lokale Deklarationen)]
BEGIN
<Anweisung>
EXCEPTION
<Fehlerbehandlung>
END [prozedur_name]

- *Stored Procedure* können Eingabeparameter annehmen, lokale Variablen verwenden und Daten zurückgeben.
- Sie können Daten mithilfe von Ausgabeparametern, Rückgabecodes, Resulttest von SELECT- Anweisungen oder globalen Cursorn zurückgeben.
- Der Befehl *Params* gibt z.B. die Parameter zurück, die an die *Stored Procedure* übergeben oder von ihr gelesen werden sollen.
- Mit dem Befehl *StoredProcName* wird der Name der auszuführenden *Stored Procedure* zurückgegeben.
- Die Methode *DescriptionsAvailable* zeigt an, ob die Parameterinformationen der *Stored Procedure* vom Server verfügbar sind. (Wenn FALSE, müssen die Parameter per Hand eingegeben werden)
- Die Methode *ExecProc* ruft die *Stored Procedure* auf.
- Mit der Methode Prepare geschieht die Vorbereitung einer SQL- Anweisung zur Ausführung (Der Server wird veranlasst, die *Stored Procedure* in seinen Arbeitsspeicher zu laden, und wenn nötig, zu kompilieren)
- Die Methode *ParamByName* erlaubt den Zugriff auf die Parameter der *Stored Procedure* über den Parameternamen.

Skalierbarkeit
Skalierbarkeit beschreibt die Möglichkeit der Anpassung einer Datenbank an sich ändernde Benutzer und Transaktionen.
Dabei sind folgende Arten von Systemänderungen zu verstehen:

- Vergrößerung von Datenbanken
- wachsende Anzahl von Benutzern und Netzwerklast
- zunehmende Transaktionslast
- steigende Anforderungen bei Anwendungen durch neue Programmiertechniken
- wachsende Anzahl von Servern

Skalierbare Systeme lösen Probleme, indem sie die Möglichkeit bieten, das Netzwerk,
die Serversysteme, die Datenbank und die Anwendungen durch das Hinzufügen von zusätzlicher Hardware zu erweitern.

Man unterscheidet bei der Skalierbarkeit eines Systems zwischen:

- Abwärtsskalierung
- Aufwärtsskalierung durch Wechsel auf leistungsfähigere Systeme
- Skalierung durch Hinzufügen von Knoten zu einem Clustersystem

Es werden im folgendem die beiden Begriffe *Clustering* und *Load Balancing*, die bei der Skalierbarkeit eine zentrale Rolle spielen, erklärt.

Clustering
Ein Cluster beschreibt eine Gruppe von Computern, die sich bei Fehlfunktionen gegenseitig sichern.
Der im *Abschnitt 2.1* definierte Datenbankadministrator, der für die Verwaltung und korrekte Arbeit der Datenbank verantwortlich ist, und somit das Leistungsniveau und die Wiederherstellungszeit beim Eintreten von Fehlern reduzieren muss, kann dies mithilfe der Clusterdienste erreichen.

Dabei werden grundsätzlich verschiedene Fehlerarten, die auftreten können unterschieden:

Ausfall eines Festplattenlaufwerks
Sie stellen die häufigste Fehlerquelle dar, da das Laufwerk ein mechanisches Gerät ist, das sich abnutzt.

Ausfall einer Hardwarekomponente
Hardwareausfälle sind möglich, da sich die Komponenten vorwiegend aufgrund der Wärmeentwicklung abnutzen.

Ausfall einer Softwarekomponente
Manche Softwarefehler werden nur unter bestimmten Bedingungen entdeckt. Das System kann dabei Monate oder Jahre laufen, bis ein bestimmtes Zusammentreffen von Umständen ein Problem aufdeckt.

Externer Fehler

Ein System kann aufgrund externer Ursachen ausfallen, z.B. aufgrund eines Stromausfalls.

Menschlicher Fehler
Ein Clustering schützt ein System nicht vor Fehlern, die von Menschen verursacht werden.
(z.B. Versehentliches löschen einer Tabelle)

Der *Zweck des Clusters* ist es, im Falle eines Fehlers oder einer geplanten Abschaltung den Clientzugriff auf Anwendungen und andere Ressourcen zu erhalten.
Ist einer der Server im Cluster aus irgendeinem Grund nicht verfügbar, werden die Ressourcen und Anwendungen auf einen anderen Knoten im Cluster verschoben.
Zur Bildung eines Clusters sind mehre Komponenten erforderlich:

1. Konten- Manager

Verwaltet die Mitgliedschaft im Cluster und sendet Echtzeitsignale an die Mitglieder (Knoten) des Clusters

2. Konfigurationsdatenbank- Manager

Verwaltet die Konfigurationsdatenbank des Clusters.

3. Failover- Manager

Startet und stoppt die Clusterdienste erhält Informationen wie z.B. den Verlust oder Neuaufnahme von Knoten vom Ressourcenmonitor und Knoten – Manager.

4. Ereignisprozessor

Initialisiert den Cluster und leitet Ergebnisinformationen unter den Komponenten des Clusters weiter.

5. Kommunikations- Manager

Verwaltet die Kommunikation zwischen den Knoten im Cluster.

Alle Systeme, die Clusterdienste ausführen haben folgende Vorteile:

1. Hohe Verfügbarkeit

Fällt ein Server aus, so werden alle Systemressourcen, wie Festplattenlaufwerke und IP- Adressen automatisch auf einen funktionsfähigen Server übertragen.
Fällt ein Anwendung im Cluster dagegen aus, wird diese entweder automatisch auf einem funktionsfähigen Server gestartet oder die Arbeit des ausgefallen Servers wird auf die anderen Clusterknoten verteilt.
Die beiden Vorgänge, die automatisch ausgelöst werden bezeichnet man als *Failover,* der erreich wird, in dem die Server erst miteinander verbunden und anschließend in ein gemeinsames Platensystem eingesetzt werden.
Die Verbindung zwischen den Servern kann entweder eine Hochgeschwindigkeitsverbindung sein (wie Ethernet- Netzwerk) oder eine andere Netzwerkhardware.
Die Serververbindung fungiert als Kommunikationskanal zwischen den Servern, über den Informationen zu Status und Konfiguration des Clusters übertragen werden können.
Das gemeinsam genutzte Plattensystem ermöglicht allen Servern gleichzeitigen Zugriff auf die Datenbank und anderen Datendateien.
Wenn das gemeinsam genutzte Plattensystem nicht fehlertolerant ist und ein Subsystem ausfällt, wechseln die Clusterdienste zwar zu einem anderen Server, der neue Server verwendet jedoch weiterhin das ausgefallene Subsystem.

- *Das Failover geschieht außerdem so schnell dass man als Benutzer nur ein kurzes Anhalten des Dienstes bemerkt.*

2. Failback

Wenn ein ausgefallener Server repariert und wieder online geschaltet wird, spricht man von einem *Failback.* Dabei wird die Arbeitsaufteilung im Cluster automatisch neu erzeugt.

3. Verwaltbarkeit

Die Clusterverwaltung ermöglicht das gesamte System wie ein einzelnes System zu verwalten. Dabei können innerhalb des Clusters die Anwendungen durch Ziehen der Clusterobjekte in der Clusterverwaltung problemlos auf andere Server verschoben werden. (Drag – and - Drop).
Auf diese Weise kann die Serverbelastung manuell geregelt oder ein Server entlastet und auf geplante Ausfallzeiten bzw. Wartungsarbeiten vorbereitet werden.
Die Clusterverwaltung ermöglicht außerdem von einem beliebigen Standort im Netzwerk aus die Überwachung des Clusterstaus, der einzelnen Knoten und aller verfügbaren Ressourcen.

4. Skalierbarkeit

Wenn die Anforderungen an das System wachsen, können Clusterdienste neu konfiguriert und entsprechend angepasst werden.
Abschließend gilt zu den Clusterdiensten noch anzumerken, dass sie zwar ein Datenbanksystem nicht fehlertolerant machen können aber zur Wiederherstellung des Systems nach auftreten von Fehlern entscheidend beitragen.

Load Balancing
Das Load Balancing ist ein Mechanismus, um Anforderungen auf mehrere Rechner zu verteilen, in dem es durch automatische Leistungsmessungen für eine "optimale Systemlast" sorgt.
Ein Server, welches das Load Balancing durchführt ist der SQL Server 2000, der jeweils vor dem Starten einer Abfrage die Ressourcen je nach Lastsituation auswählt und dadurch die Arbeitslast auf mehrere CPU's verteilt, was im Allgemeinem einer Steigerung der Leistungsfähigkeit sorgt.

Performance
Bei jeder Art von Performancetuning ist das Wichtigste zunächst eine Messung, wobei festzustellen gilt, was genau langsam ist, bevor man sich daran macht, bestimmte Dinge zu verändern.

- Wenn die Datenbank nicht auf derselben Maschine läuft wie der Webserver, dann findet die Kommunikation zwischen Datenbank-Client und Server nicht mehr über schnelle Kommunikationsmethoden (wie shared memory) statt, sondern über eine TCP / IP- Verbindung, die eine wesentlich geringere Kapazität und wesentlich höhere Latenzzeit hat.
- Dies hat besonders fatale Auswirkungen, wenn die Datenbank und der Webserver durch ein langsames Netzwerk getrennt sind oder wenn die Netzwerkbandbreite eingeschränkt ist. Hier kommt es ganz entscheidend darauf an, die Anzahl der Anfragen pro Seitenaufbau zu vermindern und die Menge der übertragenen Daten zu verringern.
- Die Anzahl der Abfragen lässt sich dadurch vermindern, in dem man SQL JOIN- Operationen, statt vieler Abfragen verwendet.

Natürlich können auch andere Ursachen die Performance der Datenbank beeinflussen wie zum Beispiel dass die Datenbank offsite steht oder das die Datenbank Queries enthält die nicht effizient sind,…. Auf diese Beispiele wird aber nicht näher eingegangen.

6.2 Welche Datenbank wird benutzt und weshalb ?

Für unser Projekt "Entwurf und Implementierung eines wissensbasierten Decision Support Systems zur personalisierten Unterstützung von Anfragen im Kunden Service" haben wir uns aus Performance und Sequirity Gründen für den Microsoft SQL-Server 2000 SP 3a entschieden.

Der Microsoft SQL Server 2000 ist ein *relationales Datenbankmanagementsystem* (*RDBMS)*, das die gesamten Daten in Tabellen speichert.
Dazu werden zunächst alle verwandten Daten in Tabellen gruppiert und können anschließend anhand dieser Tabellen wieder in Relation zueinander gesetzt werden.
Die Tatsache, dass die Datenbankstruktur verändert werden kann, indem man Tabellen hinzufügt oder entfernt, ohne Anwendungen ändern zu müssen, die auf der älteren Struktur gründeten, bezeichnet man als Flexibilität, was ebenfalls zu den Vorteilen der relationalen Datenbank gehört.

Die Verknüpfung der Tabellen erfolgt des weiteren über Primär- oder Fremdschlüssel, die im folgendem kurz erläutert werden.

- Ein *Primärschlüssel* ist ein eindeutiges Feld innerhalb der Tabelle, dessen Wert kein anderer Datensatz noch mal hat, wodurch jeder Datensatz in der Tabelle von allen anderen Daten unterschieden werden kann.

- Ein *Fremdschlüssel* repräsentiert dagegen den Wert des Primärschlüssels für eine verknüpfte Tabelle. Es dient als *struktureller Link*, der die Beziehung zwischen den verschiedenen Tabellen definiert und somit als *"Tragende Säule"* der relationalen Datenbanken gesehen wird.

Zum besseren Verständnis einer relationalen Datenbanken wird im folgendem ein einfaches Beispiel dargestellt.
Anhand der beiden Tabellen "Lohntabelle" und "Arbeitnehmer" können viele verschiede Aufgaben erledigt werden wie z.B.:

- Durchschnittslohn der Arbeitnehmer berechnen (1. Tabelle)
- Adressetiketten für Einladungen zur Personalversammlung drucken (2. Tabelle)
- Lohnbescheide, bestehend aus Lohn und Adresse, zusenden (1. + 2. Tabelle)

Lohntabelle	
Name	Lohn
Meier	4500
Müller	5100

Arbeitnehmer			
Name	Straße	PLZ	Ort
Meier	Am Weg 1	79618	Rheinfelden
Müller	Hinze-Str. 2	80689	München

Abbildung 2: Veranschaulichung von relationalen Datenbanken, die auf der Basis von Tabellen arbeiten.

Beim SQL Server handelt es sich außerdem um eine *skalierbare Datenbank*, so dass beachtliche Datenmengen gespeichert und gleichzeitig das Zugreifen vieler Benutzer auf die Daten unterstützt werden kann.
Obwohl der SQL Server 2000 als Datenspeichermodul für mehrere Tausend gleichzeitige Benutzer verwendet werden kann, die eine Verbindung über ein Netzwerk herstellen, kann es auch als eigenständige Datenbank direkt auf demselben Computer wie eine Anwendung verwendet werden ohne dass dabei zu viele Ressourcen beansprucht werden oder der eigenständige Benutzer Verwaltungsarbeiten durchführen muss.

Ein weiterer Vorteil von SQL- Server ist, dass es Funktionen zur Vermeidung der logischen Probleme besitzt, die auftreten, wenn ein Benutzer versucht, Daten zu lesen oder zu ändern, die aktuell von anderen Benutzern verwendet werden.

Im folgendem werden weitere Vorteile des SQL- Servers erläutert:

1. XML- Unterstützung
2. Datenbankverwaltungsoperationen
3. Volltextsuche
4. Indexverbesserungen
5. Failover Clustering
6. Neue Datentypen
7. SQL- Server Query Analyzer
8. Replikationsverbesserungen

XML- Unterstützung

Extensible Markup Language (XML) ist die Standardtechnologie für den Austausch von Daten im Web und wird zunehmend als bevorzugte Technologie für die Integration von E- Commerce- Systemen eingesetzt.
Unternehmen, die Business- to- Consumer-, Business-to-Business- und Extranetweblösungen entwickeln, benötigen für eine einfache Integration von Back-End-Systemen und eine problemlose Übertragung von Daten durch Firewalls Unterstützung für XML.

SQL- Server bieten unterschiedliche Möglichkeiten, mit denen die XML- Funktion unterstützt werden.

- Eine *FOR XML- Klausel*, die in SELCET – Anweisungen verwendet werden kann, so dass Daten als XML – Dokument und nicht als Standart- Ausgabe aufgerufen werden.
- Neue *gesepeicherte Systemprocedueren* zum Verwalten von XML –Daten.
- *XML- Aktualisierungsfunktionen* zum Einfügen, Aktualisieren und Löschen von Daten in der Datenbank.
- Verwenden von *Vorlagen und Dateien* zum Ausführen mehrere SQL – Anweisungen.

Datenbankverwaltungsoperation

Eine weitere Aufgabe des SQL- Servers 2000 ist es, die Leistung von Datenbankverwaltungsoperationen zu verbessern und zu vereinfachen, die von Administratoren durchgeführt werden.
Zu diesen Verbesserungen gehören schnellere differenzielle Sicherungen DBCC (parallel Database Consistency Checks) und paralleles Durchsuchen mit DBCC.
Differentielle Sicherungen können somit in Zeitabständen durchgeführt werden, die proportional zu der Datenmenge stehen., die seit der letzten vollständigen Sicherung bearbeitet wurden.

Volltextsuche

SQL Server 2000 bietet zwei neue Funktionen für eine verbesserte Volltextsuche diese sind:

1. Änderungsprotokollierung
Über die Änderungsprotokollierung wird ein Protokoll aller Änderungen an indizierten Volltextdaten geführt, so dass der Index mit diesen Änderungen aktualisiert werden kann.

2. Filtern von Bilddateien
Durch das Filtern von Bilddateien können Dokumente indiziert und abgefragt werden, die in Bildspalten gespeichert sind, indem aus den Bilddaten Textdaten extrahiert werden.

Indexverbesserungen
SQL- Server bietet eine Verbesserung bei der Indizierung. Die Arbeit mit Indizes ist vor allem dann einfach, wenn folgende Aufgaben durchgeführt werden:

- Erstellen von Indizes an verarbeitenden Spalten
- Festlegung der Reihenfolge, in der Indizes erstellt werden, also aufsteigend oder absteigend
- Festlegen, ob der Index mithilfe von parallelem Durchsuchen und Sortieren erstellt werden soll.

Failover Clustering
Ein Vorteil des SQL- Server 2000 ist die *Failover- Clusterunterstüzung*. Das Failover-Setup wird nicht mehr über den Failovercluster- Assistenten durchgeführt, sondern ist Teil des Setup- Prozesses für SQL – Server.
Einige der neuen Verwaltungsarbeiten sind:

- Verwalten von Failover- Clusterunterstützung über einen beliebigen Knoten im Cluster.
- Ermöglichen eines Failovers eines Knotens im Cluster auf einen beliebigen anderen Knoten im Cluster.
- Festlegen mehrere IP- Adressen für einen virtuellen Server.

Neue Datentypen
Durch die Definition von drei neuen Datentypen, soll die Programmierung flexibler werden:

- *bigint* ist ein 8- Byte langer Ganzzahltyp
- *sql_variant* ist ein Typ mit dem Werte verschiedener Datentypen gespeichert werden
- *table* ein Typ, mit dem über Anwendungen Ergebnisse für spätere Verwendung temporär gespeichert werden können.

Skalierbarkeit
In Verbindung mit Windows 2000 als Betriebssystem wurden für SQL Server 2000 bisherige Speicherbeschränkungen aufgehoben. Der Arbeitsspeicher kann mit Windows 2000 Advanced Server von 3 GB auf 8 GB und mit dem Windows 2000 Datacenter Server sogar auf 64 GB erweitert werden. In Vorbereitung ist auch eine 64-Bit-Version, die zeitnah mit der entsprechenden 64-Bit-Hardware verfügbar sein wird.

SQL- Server Query Analyzer
Der SQL- Server Query Analyzer enthält einen neuen Objektkatalog, der zum Navigeiren und Anzeigen von Datenbankobjekten verwendet werden kann.

Zuverlässigkeit
Da sich das zu verarbeitende Datenvolumen der geschäftskritischen Anwendungen stetig vom Gigabyte- in den Terabyte- Bereich verlagert, wird eine schnelle und zuverlässige Sicherung und Wiederherstellung der Daten immer wichtiger. SQL Server 2000 bietet hierfür zwei Neuheiten:

1. Die differenzielle Sicherung beschleunigt den Sicherungsvorgang, da nur die Daten kopiert werden, die seit der letzten vollständigen Datenbanksicherung geändert wurden.
2. Snapshot- Sicherungen nutzen die Funktionalität für “split mirror" Szenarien. Diese „split mirror" Funktion ist zugleich die schnellste Möglichkeit zur Initialisierung von Schattendatenbanken. Langwierige Prozesse bei der Wiederherstellung von Daten können so auf ein Minimum reduziert werden.

Sicherheit
SQL Server bietet den höchsten Grad an Sicherheit, den es in der Industrie gibt. SQL Server ist C2 zertifiziert – die höchste Zertifizierungsklasse für Behörden und Regierungen.

6.3 Datenbankzugriffslayer

Ein Datenbankzugriffslayer dient als *Schnittstelle* zwischen Applikation und Datenbank, so dass man der Applikation nur Rechte auf die Stored Procedures geben muss, und somit einen Direktzugriff auf die Datenbanktabellen verhindern kann.
Außerdem ist es möglich, erweiterte Überprüfungslogik in die Stored Procedures einzubauen - bevor ein *INSERT, UPDATE* oder *DELETE* ausgeführt wird.

Im Rahmen der Projektgruppe wurde ein objektorientierter Datenbankzugriffslayer benutzt, da im allgemeinem mit Objekten gearbeitet wird, wie z.B. mit Klassen, Attributen, Methoden, Vererbung,… die dann bei der Objekt orientierten Programmierung zu Algorithmen und zu Code verarbeitet werden.

6.3.1 Architektur des Datenbanklayers

Die Architektur basiert vor allem auf der Benutzung von Attributen.
Alle Klassen der Datenbank sind attributiert.

Attribute sind Metainformationen, und dienen dazu im Code zusätzliche Informationen zu Klassen, Methoden und Membern abzulegen, die dann zur Laufzeit mittels *Reflection* ausgelesen werden.

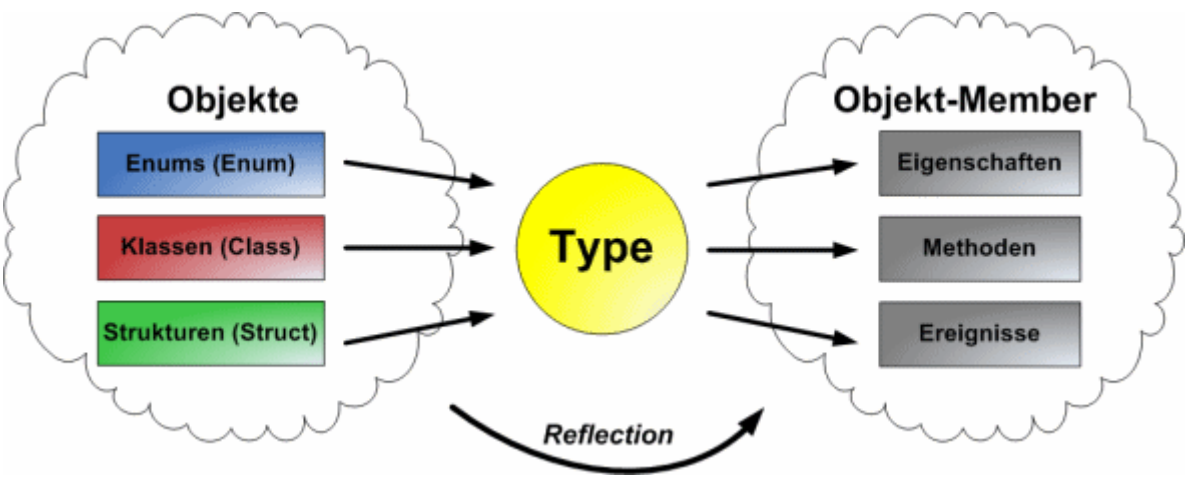


Abbildung 3: Arbeitsweise der Attribute

Des weiteren sind Attribute Objekte von Klassen, die von **System.Attribute** abgeleitet sind.
D.h., dass für jedes Attribut also eine entsprechende Attribut- Klasse definiert werden muss.
Attribute stehen in C# in eckigen Klammern und der Ausdruck, der innerhalb der eckigen Klammern steht, ist der Aufruf eines Konstruktors der entsprechenden Klasse.

6.4 Dokumentation der Datenbankklassen

Die Datenbank besteht aus den Klassen:

- Database_Model
- Database
- CommonElement und
- DataGen,

Als erstes wird die Funktion jeder Klasse erklärt und anschließend wird anhand der Klasse Database_Model, die Struktur und Arbeitsweise der Klassen verdeutlicht.

Grundsätzlich gilt, dass wir ein *Model* haben, an dem *Solutions, Questions, Answers* und *CPT's* hängen.
Zum besseren Verständnis werden an dieser Stelle die einzelnen Begriffe kurz erläutert.
Eine ausführlichere Erklärung kann in Kapitel … nachgelesen werden.

- Eine *Solution* stellt eine Lösung zu einem Problem des Users dar.
- Eine *Question* ist eine Frage, die das System dem User stellt, um die Problematik möglichst einzugrenzen.
- Die *Answers* sind Antworten auf die zuvor gestellten Fragen und enthalten *CPT's,* mit denen die Wahrscheinlichkeit definiert wird, die eine Answer zur Lösung beiträgt.

6.4.1 Zyklenvermeidung

Wie aus der Grafik zu erkennen ist, kann ein Ring bzw. eine Endlos- Schleife entstehen, die genau dann auftritt, wenn beim Lesen der Daten an einer Solution eine Question angehängt ist und diese Question wiederum dieselbe Solution hat.

Aus diesem Grund travesieren wir zuerst mit der Methode *AddToList* der Klasse Database_Model durch die verschiedenen Elemente und erzeugen vier verschiedene Listen (je eine Liste für die Question, Solution, Answer und CPT).
Die Methode arbeitet rekursiv und prüft dabei immer ob eine Solution, Question, oder Answer bereits in der Liste gespeichert ist.
Nur wenn sie noch nicht in der Liste steht wird das entsprechende Element hinzugefügt.
Gespeichert werden die Elemente mit folgender Syntax: Model<….> Table.
Wie die Umsetzung im Code genau geschieht kann im in der *Region CreateReference* nachgelesen werden.
Zu jedem Element wird außerdem die eindeutige ID sowie die Referenzen gespeichert. (Referenz auf das Model in dem sie enthalten sind.)

6.4.2 Speichern der Daten

Als nächstes müssen die in Abschnitt 4.1 erstellten Verknüpfungen im Model gespeichert werden.

- Das heißt, dass in einer neuen Tabelle *ModelSolution* alle Solution aufgelistet werden, die direkt am Model hängen. (Analog dazu sind in der Tabelle *ModelQuestion* alle Questions aufgelistet, die ebenfalls direkt am Model hängen.)

- Zu jeder Solution, die Questions besitzt, werden nun die Referenzen der Questions gespeichert (Speicherung der Referenzen zu ihren Kindreferenzen.).

Nachdem alle Verknüpfungen im Model gespeichert worden sind, können die Daten ausgelesen werden.

6.4.3 Lesen der Daten

Zum Lesen der Solutions wird die *ModelSolutionTable* mit der *SolutionTable* "gejoint".
Dabei erhalten wir alle Solutions zurück, die am Model hängen. (Analog gilt dies für die *Question – und AnswerTables.)*

Um alle Questions auszulesen die an der Solution hängen, wird die *Solution_QuestionTable* und die entsprechenden Referenzen gespeichert.
Aus der *Model_Question_Table* erhalten wir nun die ID's und in der entsprechenden *Element_Table* kann schließlich der Inhalt ausgelesen werden.

Dieselbe Prozedur wird auch für die *Questions* und *Answers* gemacht, so dass wir am Ende einen Graphen erhalten, der abschließend an das Model angehangen werden muss.

Als nächstes werden die wichtigsten Methoden, Attribute,… erklärt, die zur Umsetzung der oben genannten Punkte dienen.

6.4.4 Struktur der Klasse Database_Model

Zur Struktur der Klasse *Database-Model* gilt, dass sie aus zwei Vorgängen besteht.

1. Load
2. Save.

Zuerst erzeugt Load die Tabellen und Stored Procedures auf Basis der vorhandenen Klassen. Anschließend werden die Attribute angehängt.

In der *Region Member* werden 5 Arraylisten und 8 Hashtables erzeugt.

Die Listen sind:

- LoadQuestionsList,
- LoadSolutionsList,
- LoadAnswersList,
- LoadCptsList und
- LoadInfluenceList

Und die Hashtabels lauten:

- LoadReferenceSolutionQuestionList,
- LoadReferenceSolutionCptList,
- LoadReferenceQuestionAnswerList, LoadReferenceQuestionChildQuestionList,
- LoadReferenceQuestionCPTList,
- LoadReferenceQuestionParentQuestionList, LoadReferenceQuestionSolutionList,
- LoadReferenceAnswerInfluenceList.

In der *Region Method* gibt es zwei Klassen, wobei die *internal Klasse Load* zuerst den Inhalt der Listen löscht und danach die Answers, Cpts, Influences, Questions, Solutions geladen und gebunden werden.

In der *Region Bind Elements* gibt es 4 Teile die im folgendem alle detailliert erklärt werden.

1. <u>Teil</u>

BindToplevel binden jede Question und Solution der Liste an das entsprechende Model.

BindSolutions binden zu jeder Solution die entsprechenden CPT's und Questions.

BindQuestions binden zu jeder Question die entsprechenden Answers, ChildQuestions, CPT's, ParentQuestion und Solutions.

BindAnswers binden zu jeder Answer der die entsprechenden Influencen.

Um diese Objekte von der Liste auszulesen wird die Methode **GetObjectByID** benutzt.
Die Methode prüft die ID's aller Objekte, und gibt dann das Objekt, das mit der übergebenen ID übereinstimmt zurück.

2. Teil

Region LoadToList
Die im ersten Teil erhaltene ID wird an SQL-Database übergeben und mit dem Befehl Database.GetJoinData(LoadModel.ID) holen wir die zugehörige passende QuestionsList, SolutionsList, AnswersList, CptsList und InfluenceList zum Objekt.

Region LoadReference:
Die Methode sucht die Referenz des Objektes, dessen ID mit der gesuchten ID übereinstimmt und liest sie aus.

LoadModelSolution(), LoadModelQuestion(), LoadReferenceSolutionQuestion(Solution s), LoadReferenceSolutionCpt(Solution s), LoadReferenceQuestionA
nswer(Question q), LoadReferenceQuestionChildQuestion(Question q), LoadReferenceQuestionCPT(Question q), LoadReferenceQuestionParentQuestion(Question q), LoadReferenceQuestionSolution(Question q) und LoadReferenceAnswerInfluence(Answer a)

Als nächstes wird jede Question mit seinen Eltern- und KinderQuestions, sowie mit Solution, Answer, CPT und Influence verbunden.

Es gibt jetzt eine Referenz von Solutions nach Questions und von Questions nach Solutions, auf diese Weise wird ein Ring verhindert, dass entstehen kann, wenn zwei Solutions identisch sind.

In der *Region der Methode AddToSaveList* wird das Model *SaveModel* und weitere fünf Arraylisten erzeugt:

- SaveQuestionsList,
- SaveSolutionsList,
- SaveAnswersList,
- SaveCptsList und
- SaveInfluenceList

Als nächstes wird geprüft, ob es ein *Solution-Question-Ring* gibt.
Anschließend werden die Solutions, Questions, Answers, CPT's und Influence in Listen gespeichert.

- SaveSolutions(),
- SaveQuestions(),
- SaveAnswers(),
- SaveCpts(),
- SaveInfluences().

3. Teil

Region SaveList
Hier wird das Objekt und die entsprechende Referenz gespeichert
Die Methode *SaveList* speichert die Daten und die Datenstruktur.
Es gibt fünf Klassen:

- *SaveSolutions, SaveQuestions, SaveAnswers, SaveCpts und SaveInfluences.*

4. Teil

Region CreateReference:
In dieser Region werden neue Referenzen erzeugt.
Um eine neue Referenz zwischen zwei Objekten zu erstellen muss dazu die ID von beiden Objekten durch *Database.Save* gespeichert werden.
SaveList ruft dann die ID's auf und die neue refenz kann dann benutzt werden.

Um Zyklen in den Daten zu verhindern wird die Methode *AddToSaveList* benutzt.
Sie prüft alle Solutionslisten und wenn eine Soluiton zweimal auftaucht, wird die Tiefensuche abgebrochen.

6.4.5 Class CodeGen

Die in der Klasse CodeGen enthaltenen Methode *GetTypes* durchläuft zunächst alle Klassen und fragt mit den Attribut *CreateTableAttribut* die Attribute der Klassen (Typen) ab.
Aus diesen Typinformationen können dann Strings generiert werden, die auf den Datenbank Server Tabellen und Storeded Proceduren erzeugen.

6.4.6 Class CommonElement

Die Klasse Common_Element gilt als Basisklasse von allen anderen Datenbankklassen.

6.4.7 Class Database

Die Klasse Database dient dazu, Daten in die Datenbank zu schreiben und auszulesen.
Dazu benötigen wir zuerst die Information, von welcher Klasse aus der Befehl, die Daten zu schreiben oder zu lesen ausging.
Die Informationen hierzu werden von einem Call-Stack abgerufen.
Zurückgeliefert wird das entsprechende Objekt, (der Name) mit den entsprechenden Typinformationen, woraus eine Stored Procedure generiert wird.

Zudem wird noch die Methode *GetFieldUptime* aufgerufen.
Diese Methode bietet die Möglichkeit die aktuelle Zeit abzuspeichern.
Durch den Parameter <objekts> kann in der Klasse Database außerdem beliebig viele Objekte gespeichert werden.

Anhand dieser Informationen werden nun alle Felder durchlaufen und geprüft, ob sie *DataBaseTypeAttribute* haben.

Falls nicht, fügen wir die Attribute zu den Stored Procedures hinzu und führen sie aus.
Zusätzlich speichern die Elemente die Information, ob sie neu in die Datenbank aufgenommen worden sind oder bereits (zu einem anderen Zeitpunkt) in der DB gespeichert waren.

Nach dem die Daten gespeichert worden sind erfolgt die Umwandlung des Lesens durch die Methode *GetJoinData.*
Als erstes benötigen wir dazu die Stored Proceduren, aus dem Call- Stack.
Die zugehörigen Daten werden nun in einer Tabelle gespeichert.
Aus jeder Zeile der Tabelle wird danach ein Objekt generiert und aus den Objekten werden Listen erzeugt.
Mit der Methode *Activator.CreateObjekt.Type* durchlaufen wir alle Felder des neu erzeugten Objekts und prüfen ob es ein Objekt von GetAttribut ist.
Falls ja erfolgt abschließend ein Mapping der Daten mittels *System.Reflection*.
System.Reflection liest aus den Metadaten der Objekte (Attribute) die Informationen über die angesprochenen Objekte bzw. deren Typen aus, d.h. die Objekte reflektieren über sich selbst.

ANMERKUNG

Nach der Erklärung der Datenbankklassen gilt an dieser Stelle noch anzumerken, dass bei möglichen Fragen oder Unklarheiten im Code nachgelesen werden kann.

6.5 Literaturverzeichnis zu Adiuware.Data

Bücher

[1] M. Garcia, J. Reding: SQL Server 2000, Das Handbuch, ATLAS GmbH, München, 2000
[2] M. Young, B. Johnson, C. Skibo: Inside Microsoft Visual Studio. NET H.B. Fenn and Company Ltd. 2003
[3] D. Watkins, M. Hammond, B. Abrams: Programmierung in the .NET Addison-Wesley 2002

Internetseiten:

[4] www.onlineLexikon.de
[5] www.payer.de/dbaufbau/dbauf01.html
[6] www.microsoft.com/germany/ms/sql/
[7] http://v.hdm-stuttgart.de/~riekert/lehre/db-kelz/chap2.htm
[8] www.itb.uni-stuttgart.de/lectures/ Programmieren/Xiaolei/Bioinfor02_O.ppt
[9] www.bw.fh-degendorf.de-kurse-db-skripten-skript11.pdf
[10] http://de.gotdotnet.com/quickstart/aspplus/doc/webtemplates.aspx
[11] http://www.galileocomputing.de/openbook/vb_net/
[12] http://de.wikipedia.org/wiki/Relationale_Datenbank
[13]www.itb.uni-stuttgart.de/lectures/ Programmieren/Xiaolei/Bioinfor02_O.ppt

7 Dokumentation zu Adiuware.Logic

7.1 Generator

7.1.1 Zielsetzung

Das Ziel des Generators ist es, aus den vom Administrator eingegebenen Informationen korrekte CPT Einträge für die Fragen zu berechnen. Dabei stellt der Administrator folgende Informationen bereit:

- Welche Antwort welcher Frage beeinflusst welche Lösung wie stark
- Wie häufig ist die Lösung im Vergleich zu den anderen Lösungen

Bei der Beeinflussung werden positive und negative Beeinflussungen zugelassen, die jeweils in drei Stufen unterteilt sind (stark, mittel und schwach). Dabei meint eine positive Beeinflussung, dass wenn auf Frage A mit Antwort a geantwortet wird Lösung 1 wahrscheinlicher wird, während eine negative Beeinflussung Lösung 1 unwahrscheinlicher werden lässt. Aufgabe ist nun alle diese Beeinflussungen in den CPTs zusammenzufassen. Problem dabei ist, dass in der gewählten Struktur Beziehungen zwischen Fragen auftreten, über die der Administrator keinerlei Informationen einfügt. Zusätzlich ist ein Utility Wert zu bestimmen.

7.1.2 Idee

Die Idee ist es einen Durchlauf durch das Programm zu simulieren und die dabei gewonnenen Informationen in den CPTs zu speichern. Um den Ablauf zu simulieren wählt der Algorithmus zunächst eine Lösung aus. Er legt damit das Problem fest, das vorliegt. Anschließend beginnt er eine Frage – Antwort Kombination um das Problem zu bestimmen. Dabei werden die Fragen quasi zufällig ausgewählt. Die Antworten werden gemäß den Beeinflussungen gegeben. Es wird demnach immer die Antwort ausgewählt, die die positivste Beeinflussung auf die oben gewählte Lösung hat und wenn es keine Antwort mit eine positiven Beeinflussung gibt, diejenige, die die negativste Beeinflussung auf eine andere Lösung hat. Die entstandenen Frage – Antwort Folgen werden dann ausgewertet und in die CPTs übertragen.

7.1.3 Der Algorithmus

Anfangs wird die Lösung ausgewählt. Dies geschieht gemäß der a Priori Wahrscheinlichkeiten der Lösungen. Eine Lösung die wahrscheinlicher ist erhält somit einen größeren Einfluss auf die CPTs als eine weniger wahrscheinliche Lösung. Danach wird zufällig eine Frage ausgewählt. Die ausgewählte Frage wird aus der Liste der zu stellenden Fragen entfernt. Nun sucht der Algorithmus nach der Antwort, die unsere vorab gewählte Lösung maximal positiv beeinflusst. Gibt es keine Antwort, die die Lösung positiv beeinflusst, so wird die Antwort gewählt, die eine andere Lösung möglichst schlecht beeinflusst. Die letztlich gegebene Antwort beeinflusst nun die Wahrscheinlichkeiten der Lösungen. Es wird überprüft welche Lösungen die Antwort wie beeinflusst und die Wahrscheinlichkeit der Lösungen gemäß der Beeinflussungen angepasst. Wenn nach der Anpassung die ausgewählte Lösung einen Schwellenwert überschreitet, beendet der Algorithmus das Fragen mit einer Erfolgsnachricht, wenn alle Fragen gestellt wurden, hört er ebenfalls auf, jedoch ohne Erfolg.

Sonst stellt er die nächste Frage. Nachdem das Fragen beendet ist, wird das entstandene Sample ausgewertet, wenn es erfolgreich war. Für jede im Sample vorkommende Frage wird überprüft ob sie in der zugrunde liegenden Struktur Eltern hat. Falls ja, wird überprüft ob ihre Eltern ebenfalls im Sample vorkommen. Aus diesen Informationen ergibt sich nun ein CPT Eintrag (falls alle Eltern vorkommen, oder es keine Eltern gibt) oder mehrere CPT Einträge (falls nicht alle Eltern vorkommen). Jeder dieser, sich aus der Elternkombination und der gegebenen Antwort ergebender Eintrag, wird um 1 inkrementiert. Zusätzlich wird in jeder entsprechenden Zeile der Utility Wert um einen von der Sample Länge abhängigen Wert erhöht. (Je kürzer das Sample ist umso besser ist es).
Der Fragenzähler wird zurückgesetzt und ein neues Sample wird generiert. Wenn hinreichend viele Samples erzeugt wurden (ca. 100.000 Stück) endet der Algorithmus. Abschließend werden nun noch die CPTs normalisiert.

7.2 Inferenz auf Adiuware-Netzen

Der Inferenzalgorithmus von Adiuware ist ein rekursiver Algorithmus, der auf der exakten Inferenz auf Bayes'schen Netzen basiert. Es wird ein Knoten eines Adiuware-Netzes, eine Evidenzliste, sowie eine Belegung übergeben und eine Wahrscheinlichkeit zurückgegeben.

Ein Adiuware-Netz ist ein Bayes'sches Netz, dessen Blätter Lösungen repräsentieren und dessen restliche Knoten Fragen darstellen. Die CPTs der Fragen wurden außerdem um eine Spalte erweitert, die Utility-Werte enthält mit denen eine Fragereihenfolge berechnet werden kann.

Ist der übergebene Knoten eine Lösung, so wird die Wahrscheinlichkeit berechnet mit der diese spezifische Lösung die korrekte ist. Ist der Knoten eine Frage, so wird die Wahrscheinlichkeit berechnet mit der diese Frage die zurzeit sinnvollste ist.

Die Belegung legt fest welche Antwort auf eine Frage gegeben worden ist und ist beim ersten Aufruf des Algorithmus nicht relevant. Da für Fragen der Utility-Wert berechnet werden soll und bei Lösungen die Belegung immer eins ist, kann beim ersten Aufruf jeder beliebige Wert eingetragen werden. Erst bei rekursiven Aufrufen, bei denen die Wahrscheinlichkeiten für bestimmte Antworten relevant sind, werden vom Algorithmus automatisch Belegungen eingetragen.

Die Evidenzliste ist eine Liste von Frage-Knoten und zugehörigen Belegungen, spezifisch die Liste aller Fragen die schon beantwortet worden sind, jeweils mit der gegebenen Antwort.

Grundsätzlich berechnet der Algorithmus die Wahrscheinlichkeit eines Knotens durch aufaddieren der Wahrscheinlichkeiten seiner Eltern für alle Belegungen. Falls nötig wird Algorithmus rekursiv auf den Elterknoten aufgerufen. Dies läuft identisch zur exakten Inferenz bei Bayes'schen Netzen ab (siehe entsprechendes Referat im Zwischenbericht). Da Adiuware-Netze aber eine Erweiterung Bayes'scher Netze sind, gibt es einige Besonderheiten zu beachten.

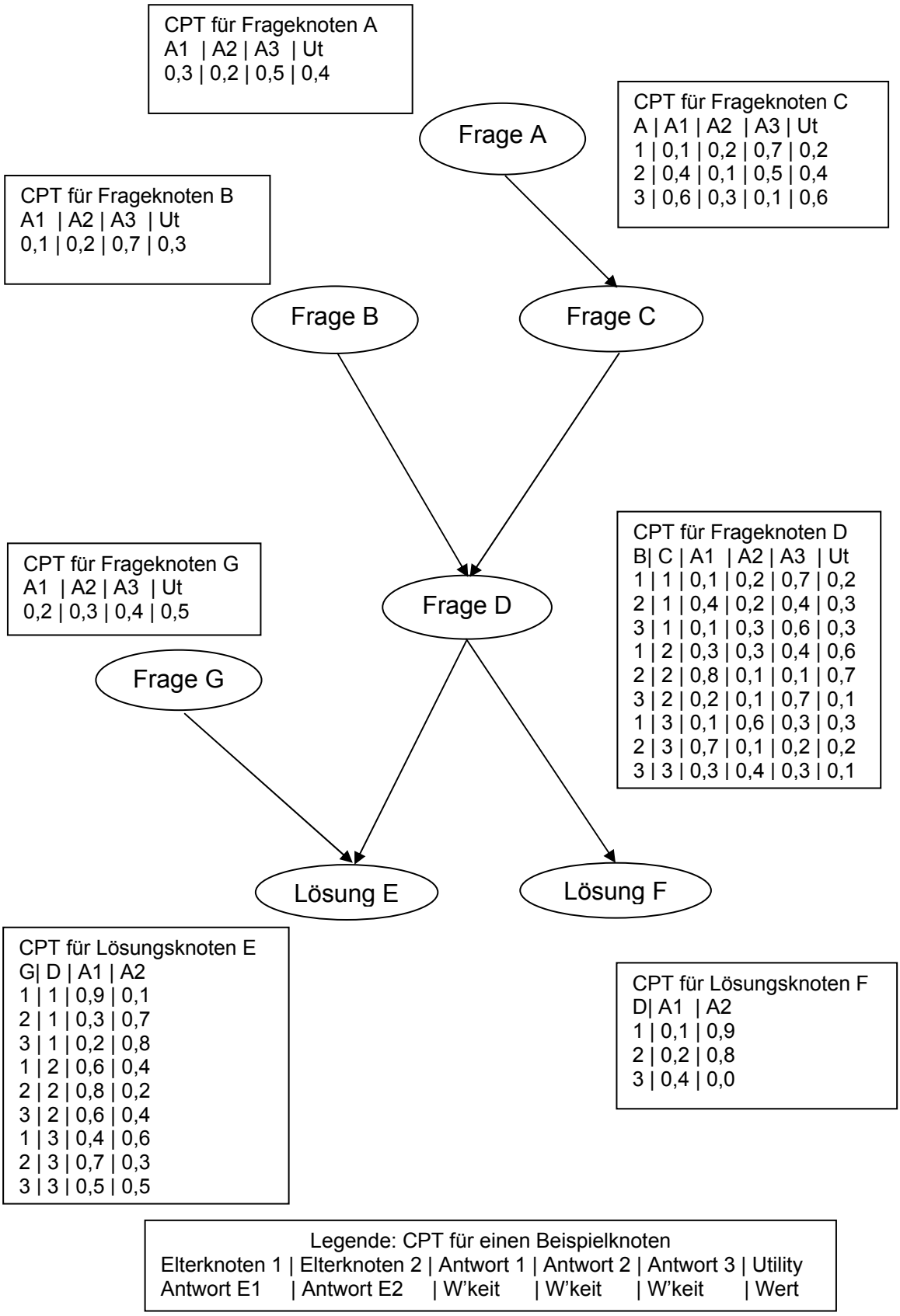
Zunächst muss unterschieden werden, ob der übergebene Knoten eine Frage oder eine Lösung ist. Im ersten Fall muss der Utility-Wert berechnet werden, ansonsten

wird wie beim normalen Algorithmus für Bayes'sche Netze verfahren und die Wahrscheinlichkeit berechnet, dass die Lösung wahr ist.

Dies gilt jedoch nur beim ersten Aufruf des Algorithmus. Bei rekursiven Aufrufen können keine Lösungen mehr auftreten, da diese nur in Blättern des Baumes auftreten und rekursive Aufrufe nur auf Elterknoten möglich sind. Dafür muss für die Fragen jetzt eine Belegung angegeben werden, da nicht mehr die Utility berechnet werden muss, sondern die Wahrscheinlichkeit für eine bestimmte Antwort. (Diese Wahrscheinlichkeiten werden dann verwendet, um die Utility im ursprünglich aufgerufenen Knoten zu berechnen.)

7.2.1 Beispiel eines Adiuware- Netzes

(s. nächste Seite)


CPT für Frageknoten A
A1 | A2 | A3 | Ut
0,3 | 0,2 | 0,5 | 0,4
Frage A
CPT für Frageknoten C
A | A1 | A2 | A3 | Ut
1 | 0,1 | 0,2 | 0,7 | 0,2
2 | 0,4 | 0,1 | 0,5 | 0,4
3 | 0,6 | 0,3 | 0,1 | 0,6
CPT für Frageknoten B
A1 | A2 | A3 | Ut
0,1 | 0,2 | 0,7 | 0,3
Frage B
Frage C
CPT für Frageknoten D
B| C | A1 | A2 | A3 | Ut
1 | 1 | 0,1 | 0,2 | 0,7 | 0,2
2 | 1 | 0,4 | 0,2 | 0,4 | 0,3
3 | 1 | 0,1 | 0,3 | 0,6 | 0,3
1 | 2 | 0,3 | 0,3 | 0,4 | 0,6
2 | 2 | 0,8 | 0,1 | 0,1 | 0,7
3 | 2 | 0,2 | 0,1 | 0,7 | 0,1
1 | 3 | 0,1 | 0,6 | 0,3 | 0,3
2 | 3 | 0,7 | 0,1 | 0,2 | 0,2
3 | 3 | 0,3 | 0,4 | 0,3 | 0,1
CPT für Frageknoten G
A1 | A2 | A3 | Ut
0,2 | 0,3 | 0,4 | 0,5
Frage D
Frage G
Lösung E
Lösung F
CPT für Lösungsknoten E
G| D | A1 | A2
1 | 1 | 0,9 | 0,1
2 | 1 | 0,3 | 0,7
3 | 1 | 0,2 | 0,8
1 | 2 | 0,6 | 0,4
2 | 2 | 0,8 | 0,2
3 | 2 | 0,6 | 0,4
1 | 3 | 0,4 | 0,6
2 | 3 | 0,7 | 0,3
3 | 3 | 0,5 | 0,5
CPT für Lösungsknoten F
D| A1 | A2
1 | 0,1 | 0,9
2 | 0,2 | 0,8
3 | 0,4 | 0,0
Legende: CPT für einen Beispielknoten
Elterknoten 1 | Elterknoten 2 | Antwort 1 | Antwort 2 | Antwort 3 | Utility
Antwort E1 | Antwort E2 | W'keit | W'keit | W'keit | Wert

7.2.2 Beispielanfragen

1. **Gegebene Evidenz:** Keine Evidenz im Netz vorhanden

   Gesuchter Wert: Die Nützlichkeit der Frage C, also der Utilitywert an C

   Rekursive Berechnung für Knoten C:
   Da C nur den Elter A hat und dieser ein Wurzelknoten ist, ist keine Rekursion nötig, so dass die Gewichtungen der einzelnen Utilitywerte direkt abgelesen werden können.

   Es folgt die Berechnung an Knoten C:
   Utility in C[0,4]: 0,2 * W'keit das A mit 1 beantwortet wird: 0,3

   Utility in C[1,4]: 0,4 * W'keit das A mit 2 beantwortet wird: 0,2

   Utility in C[2,4]: 0,6 * W'keit das A mit 3 beantwortet wird: 0,5

   W'keit für Knoten C: 0,44

2. **Gegebene Evidenz:** Es sind alle Eltern evident. Knoten D wurde mit Antwort 2 beantwortet und Knoten G mit Antwort 1.

   Gesuchter Wert: Die W'keit das Lösung E gilt, also Antwort 1 für Knoten E. Antwort 2 repräsentiert die W'keit, das Lösung E nicht gilt.

   Rekursive Berechnung für Knoten E:
   Da alle Eltern evident sind, ist hier ebenfalls keine Rekursion nötig. Es werden die die Gewichtungen der einzelnen W'keiten wiederum direkt abgelesen.

   Es folgt die Berechnung an Knoten E:
   W'keit in E[0,2]: 0,9 * W'keit das G mit 1 beantwortet wird: 1
   * W'keit das D mit 1 beantwortet wird: 0

   W'keit in E[1,2]: 0,3 * W'keit das G mit 2 beantwortet wird: 0
   * W'keit das D mit 1 beantwortet wird: 0

   W'keit in E[2,2]: 0,2 * W'keit das G mit 3 beantwortet wird: 0
   * W'keit das D mit 1 beantwortet wird: 0

   W'keit in E[3,2]: 0,6 * W'keit das G mit 1 beantwortet wird: 1
   * W'keit das D mit 2 beantwortet wird: 1

   W'keit in E[4,2]: 0,8 * W'keit das G mit 2 beantwortet wird: 0
   * W'keit das D mit 2 beantwortet wird: 1

   W'keit in E[5,2]: 0,6 * W'keit das G mit 3 beantwortet wird: 0
   * W'keit das D mit 2 beantwortet wird: 1

   W'keit in E[6,2]: 0,4 * W'keit das G mit 1 beantwortet wird: 1

* W'keit das D mit 3 beantwortet wird: 0

W'keit in E[7,2]: 0,7 * W'keit das G mit 2 beantwortet wird: 0
* W'keit das D mit 3 beantwortet wird: 0

W'keit in E[8,2]: 0,5 * W'keit das G mit 3 beantwortet wird: 0
* W'keit das D mit 3 beantwortet wird: 0

W'keit für für Knoten E: 0,6

3. **Gegebene Evidenz:** Frageknoten B wurde mit Antwort 3 beantwortet

   Gesuchter Wert: Die Nützlichkeit der Frage D, also der Utilitywert an D

   Rekursive Berechnung für Knoten D:

Die eingerückten Zeilen stellen einen rekursiven Aufruf dar für Knoten C, der zur Berechnung der W'keit eines Elternknotens gemacht wird, damit diese berechnete W'keit zur Berechnung des eigentlich gesuchten Wertes verwendet werden kann. Da B gegeben, bzw. A keine Eltern hat braucht für diese Knoten kein rekursiver Aufruf gemacht werden.

Es folgt die Berechnung an Knoten D:

Utility in D[0,5]: 0,2 * W'keit das B mit 1 beantwortet wird: 0
    Berechnungen an Knoten: C
    W'keit in C[0,1]: 0,1 * W'keit das A mit 1 beantwortet wird: 0,3
    W'keit in C[1,1]: 0,4 * W'keit das A mit 2 beantwortet wird: 0,2
    W'keit in C[2,1]: 0,6 * W'keit das A mit 3 beantwortet wird: 0,5
* W'keit das C mit 1 beantwortet wird: 0,41

Utility in D[1,5]: 0,3 * W'keit das B mit 2 beantwortet wird: 0
    Berechnungen an Knoten: C
    W'keit in C[0,1]: 0,1 * W'keit das A mit 1 beantwortet wird: 0,3
    W'keit in C[1,1]: 0,4 * W'keit das A mit 2 beantwortet wird: 0,2
    W'keit in C[2,1]: 0,6 * W'keit das A mit 3 beantwortet wird: 0,5
* W'keit das C mit 1 beantwortet wird: 0,41

Utility in D[2,5]: 0,3 * W'keit das B mit 3 beantwortet wird: 1
    Berechnungen an Knoten: C
    W'keit in C[0,1]: 0,1 * W'keit das A mit 1 beantwortet wird: 0,3
    W'keit in C[1,1]: 0,4 * W'keit das A mit 2 beantwortet wird: 0,2
    W'keit in C[2,1]: 0,6 * W'keit das A mit 3 beantwortet wird: 0,5
* W'keit das C mit 1 beantwortet wird: 0,41

Utility in D[3,5]: 0,6 * W'keit das B mit 1 beantwortet wird: 0
    Berechnungen an Knoten: C
    W'keit in C[0,2]: 0,2 * W'keit das A mit 1 beantwortet wird: 0,3
    W'keit in C[1,2]: 0,1 * W'keit das A mit 2 beantwortet wird: 0,2
    W'keit in C[2,2]: 0,3 * W'keit das A mit 3 beantwortet wird: 0,5
* W'keit das C mit 2 beantwortet wird: 0,23

Utility in D[4,5]: 0,7 * W'keit das B mit 2 beantwortet wird: 0

Berechnungen an Knoten: C
W'keit in C[0,2]: 0,2 * W'keit das A mit 1 beantwortet wird: 0,3
W'keit in C[1,2]: 0,1 * W'keit das A mit 2 beantwortet wird: 0,2
W'keit in C[2,2]: 0,3 * W'keit das A mit 3 beantwortet wird: 0,5
* W'keit das C mit 2 beantwortet wird: 0,23

Utility in D[5,5]: 0,1 * W'keit das B mit 3 beantwortet wird: 1
Berechnungen an Knoten: C
W'keit in C[0,2]: 0,2 * W'keit das A mit 1 beantwortet wird: 0,3
W'keit in C[1,2]: 0,1 * W'keit das A mit 2 beantwortet wird: 0,2
W'keit in C[2,2]: 0,3 * W'keit das A mit 3 beantwortet wird: 0,5
* W'keit das C mit 2 beantwortet wird: 0,23

Utility in D[6,5]: 0,3 * W'keit das B mit 1 beantwortet wird: 0
Berechnungen an Knoten: C
W'keit in C[0,3]: 0,7 * W'keit das A mit 1 beantwortet wird: 0,3
W'keit in C[1,3]: 0,5 * W'keit das A mit 2 beantwortet wird: 0,2
W'keit in C[2,3]: 0,1 * W'keit das A mit 3 beantwortet wird: 0,5
* W'keit das C mit 3 beantwortet wird: 0,36

Utility in D[7,5]: 0,2 * W'keit das B mit 2 beantwortet wird: 0
Berechnungen an Knoten: C
W'keit in C[0,3]: 0,7 * W'keit das A mit 1 beantwortet wird: 0,3
W'keit in C[1,3]: 0,5 * W'keit das A mit 2 beantwortet wird: 0,2
W'keit in C[2,3]: 0,1 * W'keit das A mit 3 beantwortet wird: 0,5
* W'keit das C mit 3 beantwortet wird: 0,36

Utility in D[8,5]: 0,1 * W'keit das B mit 3 beantwortet wird: 1
Berechnungen an Knoten: C
W'keit in C[0,3]: 0,7 * W'keit das A mit 1 beantwortet wird: 0,3
W'keit in C[1,3]: 0,5 * W'keit das A mit 2 beantwortet wird: 0,2
W'keit in C[2,3]: 0,1 * W'keit das A mit 3 beantwortet wird: 0,5
* W'keit das C mit 3 beantwortet wird: 0,36

Utility für Knoten D: 0,182

7.3 Implementierung der Suche

Dieser Abschnitt des Dokuments beschreibt die Implementierung der Suche und ihre Funktionalität.

7.3.1 Indexgenerierung

Um eine effiziente Suche zu ermöglichen, müssen die Schlagwörter eines Dokumentes bzw. eines Textes identifiziert werden. Dies geschieht durch eine Indexierung. Von jedem Dokument, in unserem Fall von den Solutions und CCCases, wird ein Index erzeugt, der alle sinntragenden Wörter des entsprechenden Dokuments enthält.

7.3.1.1 Eliminierung von Stoppwörtern

Um im Index eine Liste aller sinntragenden Wörter zu erhalten, muss man nicht-sinntragende Wörter eliminieren. Dies geschieht über eine Stoppwortliste, welche alle nicht-sinntragenden Wörter, die 'noisy'-words enthält. Nichtsinntragende Wörter im Englischen sind z.B. my, never oder of. Nach Eliminierung aller Stoppwörter, erhalten wir als Ergebnis eine Liste aller Schlagwörter.

7.3.1.2 Lexikographische Grundformenreduktion nach Kuhlen

Da sich die Schlagwörter in einer beliebigen Flexion befinden, muss man sie auf ihre Grundform zurückführen um eine hohe Übereinstimmung zwischen Such-Schlagwort und Schlagwort im Index zu erhalten. Dies geschieht über eine lexikographische Grundformenreduktion nach Kuhlen, die für die englische Sprache angewendet werden kann und aus 13 Regeln besteht.

Endung	Ersatz	Bedingung
IES	Y	
xyES	xy	xy = kO, CH, SH, SS, ZZ oder xX
xyS	xy	xy = xk, xE, vY, vO, OA oder EA
IES'	Y	
xES'	x	
xS'	x	
x'S	x	
x'	x	
xyING	xy	xy = kk, xv, xX
xyING	xyE	xy = vk
IED	Y	
xyED	xy	xy = kk, xv, xX
xyED	xyE	xy = vk

Angeführt sind Wortenden der vollen Wortformen und die Enden durch die diese überführt werden. Dabei bezeichnen x und y einen beliebigen aber festen Buchstaben, k bezeichnet einen beliebigen Konsonanten und v einen beliebigen Vokal.

7.3.1.3 Positionsspeicherung

Zu jedem Schlagwort, das im Index gespeichert wird, wird ein int[] array verwaltet, in dem die Positionsangaben des Wortes innerhalb des ursprünglichen Dokuments beinhaltet. Durch die Information in welcher Reihenfolge die Wörter im Text stehen, können Mehrwörter identifiziert werden.

7.3.1.4 Wortfrequenz

Die Wortfrequenz, die für die Berechnung des Rankings benötigt wird, kann über die Länge des int[] arrays der Positionsangaben ermittelt werden.

7.3.2 Datenstruktur der Indexierung

Für den Indexer implementieren wir für alle Objekte vom Typ CCCase und Solution eine AutomatedWordElementList und eine ManualWordElementList. Die AutomatedWordElementList enthält dabei die Liste der automatisch erzeugten Schlagwörter. Sie enthält WordElements mit den Attributen 'string word' und 'int[] positions'. Die ManualWordElementList dagegen soll eine Liste von Wörtern enthalten, die vom Admin zusätzlich eingegeben werden können, sofern ein Schlagwort im Text nicht vorkommt, dieses den Text aber gut repräsentiert. In dieser Liste stehen Objekte mit einem Attribut 'string word'.

7.3.3 Invertierte Indexlisten / Lookup-Listen

Die bisher beschrieben Vorgehensweise dient der Vorbereitung und Ermöglichung einer effizienten Suche. Die eigentliche Suche arbeitet auf invertierten Indexlisten, den sogenannten Lookup-Listen. Eine solche Liste enthält alle Schlagwörter in alphabetischer Sortierung mit dem Verweis auf die jeweilige DokumentenID (ccCaseID oder SolutionID). In unserem Fall benötigen wir zwei solcher Listen, eine für die Primärdokumente (Solutions) und eine für die Sekundärdokumente (ccCases).

7.3.4 Suchmethoden

Es werden zwei Suchmethoden angeboten. Eine, die alle Primärdokumente nach einem bestimmten SuchString durchsucht und eine zweite, die alle Sekundärdokumente durchsucht. Sucheingaben können in der Form von Schlagwörtern erfolgen, die entweder mit AND oder mit OR verkettet sind. Das AND ist dabei stärker bindend als das OR. Bei einer Sucheingabe von mehreren Wörtern ohne AND oder OR, wie z.B. bei einer ausformulierten Frage, werden die Wörter als OR-verknüpft aufgefasst, wobei die darin vorkommenden Stoppwörter generell vom Suchergebnis ausgeschlossen werden.
Die Suchmethode der Primärdokumente (searchInPrimaryList(String searchString) liefert eine ArrayList zurück, die sowohl die gefundenen Solutions, als auch ihre zugehörigen ModelTreeNodes enthält in der Form ModelTreeNode1, Solution1.1, Solutions 1.2, ModelTreeNode2, Solutions2.1, Solution 2.2, Solution 2.3, ModelTreeNode3, Solution3.1, usw..

Die Suchmethode der Sekundärdokumente (searchInSecondaryList(String searchString) liefert ebenfalls eine Arraylist, die analog alle gefundenen CCCases enthält.

7.3.5 Ranking der Suchergebnisse

Das Ranking aller gefunden Treffer bzw. Suchergebnissen, wird über die einfache Funktion Wortfrequenz / Dokumentenfrequenz berechnet. Die Berechnung des Ranking geschieht intern und beeinflusst lediglich die Sortierung innerhalb der zurückgegebenen ArrayLists. Die gefundenen Solutions bzw. CCCases werden in Reihenfolge ihres Matchinggrades von vorne nach hinten in der ArrayList einsortiert, wobei die Solutions bzw. CCCases mit dem potentiell höchsten Matchinggrad am Anfang der ArrayList stehen.

7.4 Literaturangaben

[CHAR91] E. Charniak, *Bayesian Networks without Tears*, American Association for Artificial Intelligence, Menlo Park, 1991

[PEARL88] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, 1988

[RUSS/NOR95] S. Russell, P. Norvig, *Artificial Intelligence – A Modern Approach*, Prentice Hall, Upper Saddle River, 1995

[RIED03] M. Riedmiller, *Vorlesungsskript „Einführung in die künstliche Intelligenz“*, Dortmund, 2003

[JENS01] F. V. Jensen, Bayesian *Networks and Decision Graphs*, Springer, 2001

8 Klassentest Adiuware.Logic + Adiuware.Data

Bemerkung: Die Tests zu den beiden Komponenten Adiuware.Logicund Adiuware.Data wurden zusammen durchgeführt.

8.1 Klasse: IndexGenerator

Autor: Alexander Ulbrich

Die Klasse IndexGenerator ist die Grundlage für die Suche. Sie generiert über die Dokumente, welche in Form von Solutions und CCCases vorliegen, einen Index über den im Anschluss eine schnelle Suche stattfindet. Dabei werden die Dokumente zunächst in einzelne Wörter zerlegt und im Anschluss alle nicht-sinntragenden Wörter über eine Stoppwortliste eliminiert. Außerdem werden die einzelnen Wörter auf ihre Grundform zurückgeführt und dann in dieser Form als Index in der Datenbank abgespeichert.

8.1.1 Teststrategie

8.1.1.1 Idee

Es werden alle Methoden getestet, die die Erstellung des Indexes unterstützen. Die internen Methoden werden auf Testumgebungen getestet, d.h. sie werden auf extra geschriebenen Testklassen und Dummydaten getestet. Die öffentlichen Methoden, durch die der Index generiert wird, wurden durch Vergleichen einer Solution mit ihrem erzeugten Index bzw. eines CCCases mit seinem erzeugten Index, auf Korrektheit überprüft.

8.1.1.2 Klassentestplan

Es folgt die Reihenfolge der Tests der internen Methoden:

1. checkIfNoisyWord(String word)
2. deflectToBasicForm(String word)

Die anschließend zu testenden öffentlichen Methoden:

3. createIndex(Data.CCCase ccCase)
4. createIndex(Data.Solution solution)

8.1.2 Testumgebung

8.1.2.1 Treiber für checkIfNoisyWord(String word), deflectToBasicForm(String word), createIndex(Data.CCCase ccCase) und createIndex(Data.Solution)

```
using System;
using Adiuware.Data.Release.Indexing;

namespace Adiuware.Data.TestClasses
{
	/// <summary>
	/// Testtreiber für die Klasse IndexGenerator.
	/// </summary>
	public class IndexGeneratorTest
	{
```

```
            public static void Main(string[] args)
            {
                  bool ergebnis;
                  /// Prüfen ob ein Wort auf der Stoppwortliste als
                  /// Stoppwort erkannt wird
                  ergebnis = IndexGenerator.checkIfNoisyWord("the");
                  Console.WriteLine("Das Ergebnis ist: {0}",ergebnis);

                  /// Prüfen ob ein Wort, welches nicht auf der
                  /// Stoppwortliste steht, korrekt behandelt wird
                  ergebnis = IndexGenerator.checkIfNoisyWord("printer");
                  Console.WriteLine("Das Ergebnis ist: {0}",ergebnis);

                  String word;
                  /// Prüfen ob das eingegebene Wort in seine korrekte
                  /// Grundform gewandelt wird
                  word = IndexGenerator.deflectToBasicForm("walking");
                  Console.WriteLine("Grundform des Wortes ist: {0}",word);

                  Console.WriteLine("IndexGeneratorTest beendet");
                  Console.ReadLine();
            }

      }
}
```

8.1.2.2 }Verwendete Klasse IndexGenerator

```
using System;
using System.Collections;
using System.IO;
using System.Text;


namespace Adiuware.Data.Release.Indexing
{
      /// <summary>
      /// Diese Klasse ist zuständig für die Indexgenerierung
      /// </summary>
      public class IndexGenerator
      {

            #region Solution-Indexierung
            /// <summary>
            /// Generiert einen Index aus der übergebenen Solution in der
Datenbank
            /// </summary>
            /// <param name="Solution">Data.Solution die indexiert werden
soll</param>
            public static void createIndex(Data.Solution solution)
            {
                  SortedList sl = new SortedList();
                  createIndex(solution.Description,sl);
                  createIndex(solution.Text,sl);
                  createIndex(solution.ModelTreeNode.Description,sl);


                  Data.WordElement wordElement;
                  char[] separator = new char[] {'&'};
                  string positionen;
                  int frequenz;
                  for(int i=0;i<sl.Count;i++)
```

```
                {
                        positionen = sl.GetByIndex(i).ToString();
                        frequenz = positionen.Split(separator).Length;
                        foreach (string sPosition in posi-
tionen.Split(separator))
                        {
                                wordElement = solution.Index.Automated.Add();
                                wordElement.Word = sl.GetKey(i).ToString();
                                wordElement.WordPosition =
int.Parse(sPosition);
                                wordElement.Count = frequenz;
                                wordElement.Save();
                        }
                }
        }


        private static void createIndexSolutionsComplete()
        {
                Console.WriteLine("Die Indexgenerierung der Solutions
wurde gestartet...");
                foreach(Data.ModelTreeNode node in
Data.DSSMain.ModelTreeNodeRoot.Nodes.GetEnumerator())
                {
                        //Console.WriteLine("");
                        //Console.WriteLine("Bezeichnung eines ModelTreeNo-
deRoots: {0}",node.Description);
                        if(node.HasModel)
                        {
                                foreach(Data.Solution solution in
node.Model.Solutions.GetEnumerator())
                                {
                                        createIndex(solution);
                                }
                        }
                        naechsteEbene(node,1);
                }
                Console.WriteLine("Die Indexgenerierung der Solutions
wurde erfolgreich abgeschlossen!");
        }


        private static void naechsteEbene(Data.ModelTreeNode node, int
i)
        {
                foreach(Data.ModelTreeNode nodeChild in
node.Nodes.GetEnumerator())
                {
                        //Console.WriteLine("Bezeichnung Ebene {0}: {1}",
i,nodeChild.Description);
                        if(nodeChild.HasModel)
                        {
                                foreach(Data.Solution solution in
nodeChild.Model.Solutions.GetEnumerator())
                                {
                                        createIndex(solution);
                                }
                        }
                        if(node.Nodes.GetEnumerator().Count!=0) naechsteE-
bene(nodeChild,i+1);
                }
        }
        #endregion Solution-Indexierung
```

```
            #region CCCase-Indexierung
            /// <summary>
            /// Generiert einen Index aus dem übergebenen ccCase in der Da-
tenbank
            /// </summary>
            /// <param name="CCCase">Data.CCCase der indexiert werden
soll</param>
            public static void createIndex(Data.CCCase ccCase)
            {
                  Data.WordElement wordElement;
                  char[] separator = new char[] {'&'};
                  string positionen;
                  int frequenz;
                  SortedList sl = new SortedList();
                  sl = createIndex(ccCase.Name,sl);
                  foreach(ProblemDescription pd in
ccCase.ProblemDescriptions.GetEnumerator())
                  {
                        sl = createIndex(pd.Text,sl);
                        sl = createIndex(pd.Title,sl);
                  }
                  for(int i=0;i<sl.Count;i++)
                  {
                        positionen = sl.GetByIndex(i).ToString();
                        frequenz = positionen.Split(separator).Length;
                        foreach (string sPosition in posi-
tionen.Split(separator))
                        {
                              wordElement = ccCase.Index.Automated.Add();
                              wordElement.Word = sl.GetKey(i).ToString();
                              wordElement.WordPosition =
int.Parse(sPosition);
                              wordElement.Count = frequenz;
                              wordElement.Save();
                        }
                  }
            }

            public static void createIndexCCCasesComplete()
            {
                  Console.WriteLine("Die Indexgenerierung der CCCases wurde
gestartet...");
                  foreach(Data.CCCase ccCase in
Data.DSSMain.GlobalBox.Cases)
                  {
                        createIndex(ccCase);
                  }
                  Console.WriteLine("Die Indexgenerierung der CCCases wurde
erfolgreich abgeschlossen!");
            }
            #endregion CCCase-Indexierung


            #region Hilfsmethoden
            /// <summary>
            /// Generiert den kompletten Index neu aus allen CCCases und
Solutions, die momentan
            /// in der Datenbank vorhanden sind.
            /// </summary>
            public static void createIndexComplete()
            {
```

```
                Console.WriteLine("Die Indexgenerierung wurde gestar-
tet...");
                createIndexCCCasesComplete();
                createIndexSolutionsComplete();
                Console.WriteLine("Die Indexgenerierung wurde erfolgreich
abgeschlossen!");
            }


            /// <summary>
            /// Generiert einen Index aus dem übergebenen Text
            /// </summary>
            /// <param name="text">Text, der indexiert werden soll</param>
            /// <returns>
            /// den Index als SortedList
            /// </returns>
            private static SortedList createIndex(string text, SortedList
sl)
            {
                String buchstaben="abcdefghijklmnopqrstuvwxyz";
                char[] separators = new char[] {'
',',',',','.','!',';',':','?','`','(',')','\\','<','>','/','{','}','[',']'};
                text = text.Replace("'s","");
                int iPosition=0;
                int index;
                string smallWord;
                foreach (string word in text.Split(separators))
                {
                    smallWord = word.ToLower();
                    if(smallWord.Length>0 && buchsta-
ben.IndexOf(smallWord.Substring(0,1))!=-1)
                    {
                        iPosition++;
                        if(!checkIfNoisyWord(smallWord))
                        {
                            smallWord=deflectToBasicForm(smallWord);
                            if(sl.ContainsKey(smallWord))
                            {
                                index = sl.IndexOfKey(smallWord);

      sl.SetByIndex(index,sl.GetByIndex(index).ToString()+"&"+iPosition);
                            }
                            else sl.Add(smallWord,iPosition);
                        }
                    }
                }
                return sl;
            }



            /// <summary>
            /// Wird von der Methode createIndex verwendet und überprüft,
ob das Wort eines Textes in der
            /// Stoppwortliste noise.eng bzw. noise.deu auftritt und lie-
fert entsprechenden Rückgabewert.
            /// </summary>
            /// <param name="word">das Wort das geprüft werden soll in
Kleinbuchstaben</param>
            /// <returns>
            /// true wenn das Wort in der Stoppwortliste auftaucht.
            /// false wenn das Wort nicht in der Stoppwortliste steht.
            ///   </returns>
```

```
            public static bool checkIfNoisyWord(String word)
            {
                  try
                  {
                        String path="";
                        //Pfad für Datei noise.eng für Startprojekt Win-
dows:

      if(Environment.CurrentDirectory.IndexOf("Windows\\bin")!=-1)

      path="../../../Data/Release/Indexing/noise.eng";
                        //Pfad für Datei noise.eng für Startprojekt Data-
Source:

      if(Environment.CurrentDirectory.IndexOf("Adiuware.Data.DataSource")!=
-1)
                              path="../Data/Release/Indexing/noise.eng";
                        //Pfad für Datei noise.eng für Startprojekt Data-
Source:

      if(Environment.CurrentDirectory.IndexOf("Adiuware.Data.TestClasses")!
=-1)

      path="../../../Data/Release/Indexing/noise.eng";

                        // Erzeugen einer Instanz von StreamReader, um aus
einer Datei zu lesen.
                        // Das using statement beendet den StreamReader.
                        using (StreamReader sr = new StreamReader(path))
                        {
                              String line;
                              // Lesen der Datei, Zeile für Zeile, bis das
Ende der Datei errreicht wird oder
                              // eine Übereinstimmung gefunden wird.
                              while ((line = sr.ReadLine()) != null)
                              {
                                    if(word.Equals(line.Trim().ToLower()))
return true;
                              }
                        }
                  }
                  catch (Exception e)
                  {
                        // Fehlerbenachrichtigung.
                        Console.WriteLine("Logic.DSSMain.checkIfNoisyWord:
The file could not be read:");
                        Console.WriteLine(e.Message);
                  }
                  return false;
            }



            /// <summary>
            /// Formt englische Wörter regelbasiert auf ihre Grundform zu-
rück. Insgesamt werden 13 Regeln berücksichtigt.
            /// </summary>
            /// <param name="word">das englische Wort, das in Grundform ge-
bracht werden soll</param>
            /// <returns>
            /// die Grundform des englischen Wortes
            ///   </returns>
            public static string deflectToBasicForm(string word)
```

```
            {
#if DEBUG
                  Console.Write("IN: {0}  ",word);
#endif
                  String vokale = "aeiou";
                  String konsonanten = "bcdfghjklmnpqrstvwxyz";
                  String xy;
                  String x;
                  String y;
                  //Regel 1
                  if(word.EndsWith("ies") && !word.Equals("series"))
                  {
#if DEBUG
                        Console.Write("(REGEL 1)");
#endif
                        word=word.Substring(0,word.Length-3)+"y";
#if DEBUG
                        Console.Write("  OUT: {0}\n",word);
#endif
                        return word;
                  }
                  //Regel 7 <-- vorgezogen da Regel 3+2 Regel 7 negativ be-
einflusst.
                  if(word.EndsWith("'s") || word.EndsWith("`s") ||
word.EndsWith("´s"))
                  {
#if DEBUG
                        Console.Write("(REGEL 7)");
#endif
                        word=word.Substring(0,word.Length-2);
#if DEBUG
                        Console.Write("  OUT: {0}\n",word);
#endif
                        return word;
                  }
                  //Regel 2+3
                  if(word.EndsWith("es") && word.Length>=4)
                  {
#if DEBUG
                        Console.Write("(AUF REGEL 2+3 CHECKEN)");
#endif
                        xy = word.Substring(word.Length-4,2);
                        x = xy.Substring(0,1);
                        y = xy.Substring(1,1);
                        if((konsonanten.IndexOf(x)!=-1 && y.Equals("o")) ||
xy.Equals("ch") || xy.Equals("sh") || xy.Equals("ss") || xy.Equals("zz") ||
y.Equals("x"))
                        {
#if DEBUG
                              Console.Write("(REGEL 2)");
#endif
                              word=word.Substring(0,word.Length-2);
                        }
                        else
                        {
                              xy = word.Substring(word.Length-3,2);
                              x = xy.Substring(0,1);
                              y = xy.Substring(1,1);
                              if(konsonanten.IndexOf(y)!=-1 ||
y.Equals("e") || (vokale.IndexOf(x)!=-1 && (y.Equals("y") ||
y.Equals("o"))) || xy.Equals("oa") || xy.Equals("ea"))
                              {
#if DEBUG
```

```
                                    Console.Write("(REGEL 3)");
#endif
                                    word=word.Substring(0,word.Length-1);
                              }
                        }
#if DEBUG
                        Console.Write("  OUT: {0}\n",word);
#endif
                        return word;
                  }
                  //Regel 3 nur noch der Wortlänge genau gleich 3 ansonsten
wird Regel 3 davor bearbeitet
                  if(word.EndsWith("s") && word.Length>=3 &&
!word.Equals("press"))
                  {
#if DEBUG
                        Console.Write("(REGEL 3)");
#endif
                        xy = word.Substring(word.Length-3,2);
                        x = xy.Substring(0,1);
                        y = xy.Substring(1,1);
                        if(konsonanten.IndexOf(y)!=-1 || y.Equals("e") ||
(vokale.IndexOf(x)!=-1 && (y.Equals("y") || y.Equals("o"))) ||
xy.Equals("oa") || xy.Equals("ea"))
                        {
                              word=word.Substring(0,word.Length-1);
                        }
#if DEBUG
                        Console.Write("  OUT: {0}\n",word);
#endif
                        return word;
                  }
                  //Regel 4
                  if(word.EndsWith("ies'"))
                  {
#if DEBUG
                        Console.Write("(REGEL 4)");
#endif
                        word=word.Substring(0,word.Length-4)+"y";
#if DEBUG
                        Console.Write("  OUT: {0}\n",word);
#endif
                        return word;
                  }
                  //Regel 5
                  if(word.EndsWith("es'"))
                  {
#if DEBUG
                        Console.Write("(REGEL 5)");
#endif
                        word=word.Substring(0,word.Length-3);
#if DEBUG
                        Console.Write("  OUT: {0}\n",word);
#endif
                        return word;
                  }
                  //Regel 6
                  if(word.EndsWith("s'"))
                  {
#if DEBUG
                        Console.Write("(REGEL 6)");
#endif
                        word=word.Substring(0,word.Length-2);
```

```
#if DEBUG
                        Console.Write("  OUT: {0}\n",word);
#endif
                        return word;
                    }
                    //Regel 8
                    if(word.EndsWith("'"))
                    {
#if DEBUG
                        Console.Write("(REGEL 8)");
#endif
                        word=word.Substring(0,word.Length-1);
#if DEBUG
                        Console.Write("  OUT: {0}\n",word);
#endif
                        return word;
                    }
                    //Regel 9+10
                    if(word.EndsWith("ing"))
                    {
                        xy = word.Substring(word.Length-5,2);
                        x = xy.Substring(0,1);
                        y = xy.Substring(1,1);
                        //Regel 9
                        if((konsonanten.IndexOf(x)!=-1 && konsonan-
ten.IndexOf(y)!=-1) || vokale.IndexOf(y)!=-1 || y.Equals("x") ||
word.Equals("following"))
                        {
#if DEBUG
                            Console.Write("(REGEL 9)");
#endif
                            word=word.Substring(0,word.Length-3);
                            if(word.Equals("runn")) word="run";
#if DEBUG
                            Console.Write("  OUT: {0}\n",word);
#endif
                            return word;
                        }
                            //Regel 10
                        else if(vokale.IndexOf(x)!=-1 && konsonan-
ten.IndexOf(y)!=-1)
                        {
#if DEBUG
                            Console.Write("(REGEL 10)");
#endif
                            word=word.Substring(0,word.Length-3)+"e";
#if DEBUG
                            Console.Write("  OUT: {0}\n",word);
#endif
                            return word;
                        }
                    }
                    //Regel 11
                    if(word.EndsWith("ied"))
                    {
#if DEBUG
                        Console.Write("(REGEL 11)");
#endif
                        word=word.Substring(0,word.Length-3)+"y";
#if DEBUG
                        Console.Write("  OUT: {0}\n",word);
#endif
                        return word;
```

```
                }
                //Regel 12+13
                if(word.EndsWith("ed") && !word.Equals("feed") &&
word.Length>=4)
                {
                      xy = word.Substring(word.Length-4,2);
                      x = xy.Substring(0,1);
                      y = xy.Substring(1,1);
                      //Regel 12
                      if(word.EndsWith("played") || !word.Equals("speed")
|| !word.Equals("garbled") || word.Equals("stapled") || (konsonan-
ten.IndexOf(x)!=-1 && konsonanten.IndexOf(y)!=-1) || vokale.IndexOf(y)!=-1
|| y.Equals("x") ||word.Equals("loaded"))
                      {
#if DEBUG
                            Console.Write("(REGEL 12)");
#endif
                            word=word.Substring(0,word.Length-2);
                      }
                            //Regel 13
                      else if(vokale.IndexOf(x)!=-1 && konsonan-
ten.IndexOf(y)!=-1)
                      {
#if DEBUG
                            Console.Write("(REGEL 13)");
#endif
                            word=word.Substring(0,word.Length-1);
                      }
#if DEBUG
                      Console.Write("  OUT: {0}\n",word);
#endif
                      return word;
                }
#if DEBUG
                Console.Write("  OUT: {0}\n",word);
#endif
                return word;
            }
            #endregion Private-Stuff

      }
}
```

8.1.3 Operation 1: checkIfNoisyWord(String word)

Diese Methode stellt fest, ob sich ein Wort auf der Stoppwortliste, die in Form einer eigenen Textdatei vorliegt, befindet oder nicht.

8.1.3.1 Äquivalenzklassen in der Testdatenmenge

1. **Das Wort befindet sich auf der Stoppwortliste:** In diesem Fall wird true zurückgeliefert.
2. **Das Wort befindet sich nicht auf der Stoppwortliste:** In diesem Fall wird false zurückgeliefert.

8.1.3.2 Test von Das Wort befindet sich auf der Stoppwortliste:

Getestete Eingabedaten

- the

Sollergebnisse
Es soll true zurückgeliefert werden.

Istergebnisse
True.

8.1.3.3 Test von Das Wort befindet sich nicht auf der Stoppwortliste:

Getestete Eingabedaten

- printer

Sollergebnisse
Es soll false zurückgeliefert werden.

Istergebnisse
False.

8.1.4 Operation 2: deflectToBasicForm(String word)

Diese Methode führt ein englisches Wort mit Hilfe der 13 Deflexions-Regeln nach Kuhlen auf seine Grundform zurück.

8.1.4.1 Äquivalenzklassen in der Testdatenmenge

1. **Regel 1 - Das Wort endet mit IES:** In diesem Fall wird IES durch Y ersetzt.
2. **Regel 2 - Das Wort endet mit xyES, wobei xy = kO, CH, SH, SS, ZZ oder xX ist (mit k ein Konsonant, x ein beliebiger Buchstabe):** In diesem Fall wird die Endung ES des Wortes entfernt.
3. **Regel 3 - Das Wort endet mit xyS, wobei xy = xk, xE, vY, vO, OA oder EA ist (mit k ein Konsonant, v ein Vokal, x ein beliebiger Buchstabe):** In diesem Fall wird die Endung S des Wortes entfernt.
4. **Regel 4 - Das Wort endet mit IES`:** In diesem Fall wird IES` durch Y ersetzt.
5. **Regel 5 - Das Wort endet mit ES`:** In diesem Fall wird die Endung ES` entfernt.
6. **Regel 6 - Das Wort endet mit S`:** In diesem Fall wird die Endung S' entfernt.
7. **Regel 7 - Das Wort endet mit `S:** In diesem Fall wird die Endung `S entfernt.
8. **Regel 8 - Das Wort endet mit `:** In diesem Fall wird die Endung ` entfernt.
9. **Regel 9 - Das Wort endet mit xyING und xy = kk, xv, xX (mit k ein Konsonant, v ein Vokal, x ein beliebiger Buchstabe):** In diesem Fall wird die Endung ING entfernt.
10. **Regel 10 - Das Wort endet mit xyING und xy = vk (mit k ein Konsonant, v ein Vokal):** In diesem Fall wird die Endung ING durch E ersetzt.
11. **Regel 11 - Das Wort endet mit IED:** In diesem Fall wird die Endung IED durch Y ersetzt.
12. **Regel 12 Das Wort endet mit xyED und xy = kk, xv, xX (mit k ein Konsonant, v ein Vokal, x ein beliebiger Buchstabe):** In diesem Fall wird die Endung ED entfernt.
13. **Regel 13 Das Wort endet mit xyED und xy = vk (mit k ein Konsonant, v ein Vokal):** In diesem Fall wird die Endung ED durch die Endung E ersetzt.

8.1.4.2 Test von Regel 1:

Getestete Eingabedaten
- copies

Sollergebnisse
Es soll copy zurückgeliefert werden.

Istergebnisse
copy.

8.1.4.3 Test von Regel 2:

Getestete Eingabedaten
- continues

Sollergebnisse
Es soll continues zurückgeliefert werden.

Istergebnisse
continues.

8.1.4.4 Test von Regel 3:

Getestete Eingabedaten
- settings bzw. stops

Sollergebnisse
Es soll setting bzw. stopp zurückgeliefert werden.

Istergebnisse
setting bzw. stop.

8.1.4.5 Test von Regel 4:

Getestete Eingabedaten
- copies`

Sollergebnisse
Es soll copy zurückgeliefert werden.

Istergebnisse
copy.

8.1.4.6 Test von Regel 5:

Getestete Eingabedaten
- goes`

Sollergebnisse
Es soll go zurückgeliefert werden.

Istergebnisse

go.

8.1.4.7 Test von Regel 6:

Getestete Eingabedaten

- printers`

Sollergebnisse
Es soll printer zurückgeliefert werden.

Istergebnisse
printer.

8.1.4.8 Test von Regel 7:

Getestete Eingabedaten

- printer`s

Sollergebnisse
Es soll printer zurückgeliefert werden.

Istergebnisse
printer.

8.1.4.9 Test von Regel 8:

Getestete Eingabedaten

- printer`

Sollergebnisse
Es soll printer zurückgeliefert werden.

Istergebnisse
printer.

8.1.4.10 Test von Regel 9:

Getestete Eingabedaten

- working

Sollergebnisse
Es soll work zurückgeliefert werden.

Istergebnisse
work.

8.1.4.11 Test von Regel 10:

Getestete Eingabedaten

- operating

Sollergebnisse
Es soll operate zurückgeliefert werden.

Istergebnisse
operate.

8.1.4.12 Test von Regel 11:

Getestete Eingabedaten
- flied

Sollergebnisse
Es soll fly zurückgeliefert werden.

Istergebnisse
fly.

8.1.4.13 Test von Regel 12:

Getestete Eingabedaten
- connected

Sollergebnisse
Es soll connect zurückgeliefert werden.

Istergebnisse
connect.

8.1.4.14 Test von Regel 13:

Getestete Eingabedaten
- stored

Sollergebnisse
Es soll store zurückgeliefert werden.

Istergebnisse
store.

8.1.5 Operation 3: createIndex(Data.CCCase ccCase)

Um diese Operation zu testen, wurde ein Dummy-CCCase in der Datenbank angelegt und anschließend über diese Methode indexiert. Im Anschluss wurde der Index mit den Wörtern des CCCases verglichen und auf Korrektheit überprüft.

8.1.6 Operation 4: createIndex(Data.Solution solution)

Um diese Operation zu testen, wurde eine Dummy-Solution in der Datenbank angelegt und anschließend über diese Methode indexiert. Im Anschluss wurde der Index mit den in der Solution vorkommenden Wörtern verglichen und auf Korrektheit überprüft.

8.2 Klasse: ComplexSearchAndRank

Autor: Alexander Ulbrich

Die Klasse ComplexSearchAndRank implementiert die eigentlichen Suchmethoden. In ihr werden zwei getrennte aber ähnliche Suchvorgänge bearbeitet. Einmal um Solutions aufzufinden und einmal um CCCases aufzufinden.

8.2.1 Teststrategie

8.2.1.1 Idee

Es werden beide Suchmethoden auf Testdaten, die in die Datenbank importiert wurden, getestet. Dabei sucht man sich sämtliche Solutions bzw. CCCases aus der Datenbank heraus, die mit dem Such-AnfrageString korrelieren. Außerdem wurden die Rückgabewerte darauf geprüft, ob sie die entsprechenden Suchwörter auch tatsächlich enthalten.

8.2.1.2 Klassentestplan

Es folgt die Reihenfolge der Tests der Methoden:

1. searchInPrimaryList(String searchString)
2. searchInSecondaryList(String searchString)

8.2.2 Testumgebung

8.2.2.1 Treiber für searchInPrimaryList(String searchString), searchInSecondaryList(String searchString)

```
using System;
using System.Collections;
using Adiuware.Data.Release.Indexing;


namespace Adiuware.Data.TestClasses
{
	/// <summary>
	/// Testtreiber für die Klasse ComplexSearchAndRank
	/// </summary>
	public class SearchTest
	{
		public static void Main(string[] args)
		{
			DSSMain.ConnectionString = "server = egon; uid = admin;
pwd = adiuware; database = Adiuware";
			Object o = Adiuware.Data.DSSMain.CurrentAgent;

			//1. primäre Suche
			Console.WriteLine("Erste Suche:");
			ArrayList al = DSSMain.searchInPrimaryList("work");
			Console.WriteLine("Anzahl Suchergebniss: {0}",al.Count);

			Solution s = new Solution();
			ModelTreeNode n = new ModelTreeNode();
			foreach(Object ob in al)
			{
				if(ob.GetType() == s.GetType())
					Console.WriteLine("Eine mögliche Lösung:
{0}",((Solution)ob).Description);
				if(ob.GetType() == n.GetType())
					Console.WriteLine("Knoten zur gefundenen Lö-
sung: {0}",((ModelTreeNode)ob).Description);
			}
```

```
                  //2. primäre Suche
                  Console.WriteLine("Zweite Suche:");
                  al = DSSMain.searchInPrimaryList("work AND correct OR
complex");
                  Console.WriteLine("Anzahl Suchergebniss: {0}",al.Count);

                  foreach(Object ob in al)
                  {
                        if(ob.GetType() == s.GetType())
                              Console.WriteLine("Eine mögliche Lösung:
{0}",((Solution)ob).Description);
                        if(ob.GetType() == n.GetType())
                              Console.WriteLine("Knoten zur gefundenen Lö-
sung: {0}",((ModelTreeNode)ob).Description);
                  }

                  //1. sekundäre Suche
                  ArrayList al2 = DSSMain.searchInSecondaryList("display");
                  foreach(CCCase c in al2)
                  {
                        Console.WriteLine("Gefundener Case: {0}",c.Name);
                  }

                  //2. sekundäre Suche
                  al2 = DSSMain.searchInSecondaryList("display AND mac");
                  foreach(CCCase c in al2)
                  {
                        Console.WriteLine("Gefundener Case: {0}",c.Name);
                  }

                  Console.WriteLine("Das wars");
                  Console.ReadLine();
            }

      }
}
```

8.2.2.2 Verwendete Klasse ComplexSearchAndRank

```
using System;
using System.Collections;
using System.Text;

namespace Adiuware.Data.Release.Indexing
{
      /// <summary>
      /// Diese Klasse enthält alle für die Suchfunktionalität
      /// nötigen Methoden.
      /// </summary>
      public class ComplexSearchAndRank
      {
            private static SortedList sl = new SortedList();
            private static SortedList slGerankt = new SortedList();
            private static ArrayList al = new ArrayList();


            private static void reset()
            {
                  sl.Clear();
                  slGerankt.Clear();
```

```
                al.Clear();
            }

            public static ArrayList complexSearchSolution(String search-
String)
            {
                reset();
                analyzeSolution(searchString);

                String b=" abcdefghijklmnopqrstuvwxy";
                if(sl.Count>0)
                {
                    String key=sl.GetKey(0).ToString().Substring(0,36);
                    int
summe=((Data.InverseSolutionWordElement)sl.GetByIndex(0)).Count;
                    for(int i=1;i<sl.Count;i++)
                    {

      if(sl.GetKey(i).ToString().Substring(0,36).Equals(key))
                        {

      summe+=((Data.InverseSolutionWordElement)sl.GetByIndex(i)).Count;
                        }
                        else
                        {
                            slGer-
ankt.Add(summe+b.Substring((i%25),i/25+1),((Data.InverseSolutionWordElement
)sl.GetByIndex(i-1)).Solution);

      summe=((Data.InverseSolutionWordElement)sl.GetByIndex(i)).Count;

      key=sl.GetKey(i).ToString().Substring(0,36);
                        }
                    }
                    slGer-
ankt.Add(summe+"z",((Data.InverseSolutionWordElement)sl.GetByIndex(sl.Count
-1)).Solution);
                }

                al = new ArrayList(slGerankt.Values);

                ArrayList ergebnis = new ArrayList();
                Array ar = al.ToArray();
                Solution s;
                bool eingefuegt=false;
                for(int i=ar.Length-1; i>=0;i--)
                {
                    s = (Solution)ar.GetValue(i);
                    ModelTreeNode n = new ModelTreeNode();
                    foreach(Object ob in ergebnis)
                    {
                        if(ob.GetType() == n.GetType() && ((Model-
TreeNode)ob).Description==s.ModelTreeNode.Description)
                        {
                            ergeb-
nis.Insert(ergebnis.IndexOf((ModelTreeNode)ob)+1,s);
                            eingefuegt=true;
                            break;
                        }
                    }
                    if(!eingefuegt)
                    {
                        ergebnis.Add(s.ModelTreeNode);
```

```
                        ergebnis.Add(s);
                    }
                    eingefuegt=false;
                }
                return ergebnis;
            }


            private static void analyzeSolution(String searchString)
            {
                int splitPosition;
                String search1;
                while (searchString.IndexOf(" OR ")>0)
                {
                    splitPosition=searchString.IndexOf(" OR ");
                    search1=searchString.Substring(0,splitPosition);
                    searchString = search-
String.Substring(splitPosition+4,searchString.Length-(splitPosition+4));
#if DEBUG
                    Console.WriteLine("ODER-Term wurde zerlegt, gesucht
wird nach:{0}",search1);
#endif
                    searchSolution(search1);
                }
                while (searchString.Trim().IndexOf(" ")>0 && search-
String.IndexOf(" ")!=searchString.IndexOf(" AND "))
                {
                    splitPosition=searchString.IndexOf(" ");
                    search1=searchString.Substring(0,splitPosition);
                    searchString = search-
String.Substring(splitPosition+1,searchString.Length-(splitPosition+1));
#if DEBUG
                    Console.WriteLine("ODER-Term wurde zerlegt, gesucht
wird nach:{0}",search1);
#endif
                    searchSolution(search1);
                }
                searchSolution(searchString);
            }

            private static void searchSolution(String searchString){
                SortedList slAND = new SortedList();
                int splitPosition;
                ArrayList alreadyFound = new ArrayList();
                String searchWord="";
                if(searchString.IndexOf(" AND ")>0)
                {
                    splitPosition=searchString.IndexOf(" AND ");
                    search-
Word=searchString.Substring(0,splitPosition).Trim().ToLower();

      if(!Data.Release.Indexing.IndexGenerator.checkIfNoisyWord(searchWord)
)
                    {
                        foreach(Data.InverseSolutionWordElement we in
Data.DSSMain.SearchAutomatedSolutions(searchWord))
                        {

      if(!alreadyFound.Contains(we.SolutionID.ToString()))
                            {
                                //Ergebnisse des ersten Suchworts
eines AND-Terms
```

```
                                          already-
Found.Add(we.SolutionID.ToString());

      slAND.Add(we.SolutionID+we.Word,we);
#if DEBUG
                                          Console.WriteLine("Wort {0} gefun-
den in: {1}",searchWord,we.SolutionID);
#endif
                                    }
                              }
                        }
                        searchString = search-
String.Substring(splitPosition+5,searchString.Length-(splitPosition+5));
                  }
                  else
                  {

      if(!Data.Release.Indexing.IndexGenerator.checkIfNoisyWord(searchStrin
g))
                        {
                              foreach(Data.InverseSolutionWordElement we in
Data.DSSMain.SearchAutomatedSolutions(searchString.Trim().ToLower()))
                              {

      if(!sl.ContainsKey(we.SolutionID+we.Word))
                                    {
                                          //Sichere Ergebnisse, da OR und
kein AND im Term
                                          sl.Add(we.SolutionID+we.Word,we);
#if DEBUG
                                          Console.WriteLine("Wort {0} gefun-
den in: {1}",searchString.Trim().ToLower(),we.SolutionID);
#endif
                                    }
                              }
                        }
                        return;
                  }
                  ArrayList tempAlreadyFound = new ArrayList();
                  while(searchString.IndexOf(" AND ")>0)
                  {
                        splitPosition = searchString.IndexOf(" AND ");

                        search-
Word=searchString.Substring(0,splitPosition).Trim().ToLower();

      if(!Data.Release.Indexing.IndexGenerator.checkIfNoisyWord(searchWord)
)
                        {
                              foreach(Data.InverseSolutionWordElement we in
Data.DSSMain.SearchAutomatedSolutions(searchWord))
                              {

      if(alreadyFound.Contains(we.SolutionID.ToString()) &&
!slAND.ContainsKey(we.SolutionID+we.Word))
                                    {
                                          //Suchwoerter geklammert von AND´s

      slAND.Add(we.SolutionID+we.Word,we);
                                          tempAlready-
Found.Add(we.SolutionID.ToString());
#if DEBUG
```

```
                                          Console.WriteLine("Wort {0} gefun-
den in: {1}",searchWord,we.SolutionID);
#endif
                                    }
                              }
                        }
                        alreadyFound = tempAlreadyFound;
                        searchString = search-
String.Substring(splitPosition+5,searchString.Length-(splitPosition+5));
                  }


                  searchWord=searchString.Trim().ToLower();

      if(!Data.Release.Indexing.IndexGenerator.checkIfNoisyWord(searchWord)
)
                  {
                        foreach(Data.InverseSolutionWordElement we in
Data.DSSMain.SearchAutomatedSolutions(searchWord))
                        {

      if(alreadyFound.Contains(we.SolutionID.ToString()) &&
!slAND.ContainsKey(we.SolutionID+we.Word))
                              {
                                    //Letztes Suchwort eines AND-Terms
                                    slAND.Add(we.SolutionID+we.Word,we);
                                    tempAlready-
Found.Add(we.SolutionID.ToString());
#if DEBUG
                                    Console.WriteLine("Wort {0} gefunden in:
{1}",searchWord,we.SolutionID);
#endif
                              }
                        }
                  }
                  String tempKey;
                  for(int i=0;i<slAND.Count;i++)
                  {

                        tempKey=slAND.GetKey(i).ToString().Substring(0,36);
                        if(!sl.ContainsKey(slAND.GetKey(i)) && tempAlready-
Found.Contains(tempKey))
                        {
                              sl.Add(slAND.GetKey(i),slAND.GetByIndex(i));
                        }

                  }
            }


            public static ArrayList complexSearchCCCase(String search-
String)
            {
                  reset();
                  analyzeCCCase(searchString);

                  String b=" abcdefghijklmnopqrstuvwxy";
                  if(sl.Count>0)
                  {
                        String key=sl.GetKey(0).ToString().Substring(0,36);
                        int
summe=((Data.InverseCCCaseWordElement)sl.GetByIndex(0)).Count;
                        for(int i=1;i<sl.Count;i++)
```

```
                        {

      if(sl.GetKey(i).ToString().Substring(0,36).Equals(key))
                              {

      summe+=((Data.InverseCCCaseWordElement)sl.GetByIndex(i)).Count;
                              }
                              else
                              {
                                    slGer-
ankt.Add(summe+b.Substring((i%25),i/25+1),((Data.InverseCCCaseWordElement)s
l.GetByIndex(i-1)).CCCase);

      summe=((Data.InverseCCCaseWordElement)sl.GetByIndex(i)).Count;

      key=sl.GetKey(i).ToString().Substring(0,36);
                              }
                        }
                        slGer-
ankt.Add(summe+"z",((Data.InverseCCCaseWordElement)sl.GetByIndex(sl.Count-
1)).CCCase);
                  }

                  al = new ArrayList(slGerankt.Values);

                  ArrayList ergebnis = new ArrayList();
                  Array ar = al.ToArray();
                  CCCase c;
                  for(int i=ar.Length-1; i>=0;i--)
                  {
                        c = (CCCase)ar.GetValue(i);
                        ergebnis.Add(c);
                  }
                  return ergebnis;
            }


            private static void analyzeCCCase(String searchString)
            {
                  int splitPosition;
                  String search1;
                  while (searchString.IndexOf(" OR ")>0)
                  {
                        splitPosition=searchString.IndexOf(" OR ");
                        search1=searchString.Substring(0,splitPosition);
                        searchString = search-
String.Substring(splitPosition+4,searchString.Length-(splitPosition+4));
#if DEBUG
                        Console.WriteLine("ODER-Term wurde zerlegt, gesucht
wird nach:{0}",search1);
#endif
                        searchCCCase(search1);
                  }
                  while (searchString.Trim().IndexOf(" ")>0 && search-
String.IndexOf(" ")!=searchString.IndexOf(" AND "))
                  {
                        splitPosition=searchString.IndexOf(" ");
                        search1=searchString.Substring(0,splitPosition);
                        searchString = search-
String.Substring(splitPosition+1,searchString.Length-(splitPosition+1));
#if DEBUG
                        Console.WriteLine("ODER-Term wurde zerlegt, gesucht
wird nach:{0}",search1);
```

```
#endif
                        searchCCCase(search1);
                  }
                  searchCCCase(searchString);
            }

            private static void searchCCCase(String searchString)
            {
                  SortedList slAND = new SortedList();
                  int splitPosition;
                  ArrayList alreadyFound = new ArrayList();
                  String searchWord="";
                  if(searchString.IndexOf(" AND ")>0)
                  {
                        splitPosition=searchString.IndexOf(" AND ");
                        search-
Word=searchString.Substring(0,splitPosition).Trim().ToLower();

      if(!Data.Release.Indexing.IndexGenerator.checkIfNoisyWord(searchWord)
)
                        {
                              foreach(Data.InverseCCCaseWordElement we in
Data.DSSMain.SearchAutomatedCases(searchWord))
                              {

      if(!alreadyFound.Contains(we.CCCaseID.ToString()))
                                    {
                                          //Ergebnisse des ersten Suchworts
eines AND-Terms
                                          already-
Found.Add(we.CCCaseID.ToString());
                                          slAND.Add(we.CCCaseID+we.Word,we);
#if DEBUG
                                          Console.WriteLine("Wort {0} gefun-
den in: {1}",searchWord,we.CCCaseID);
#endif
                                    }
                              }
                        }
                        searchString = search-
String.Substring(splitPosition+5,searchString.Length-(splitPosition+5));
                  }
                  else
                  {

      if(!Data.Release.Indexing.IndexGenerator.checkIfNoisyWord(searchStrin
g))
                        {
                              foreach(Data.InverseCCCaseWordElement we in
Data.DSSMain.SearchAutomatedCases(searchString.Trim().ToLower()))
                              {
                                    if(!sl.ContainsKey(we.CCCaseID+we.Word))
                                    {
                                          //Sichere Ergebnisse, da OR und
kein AND im Term
                                          sl.Add(we.CCCaseID+we.Word,we);
#if DEBUG
                                          Console.WriteLine("Wort {0} gefun-
den in: {1}",searchString.Trim().ToLower(),we.CCCaseID);
#endif
                                    }
                              }
                        }
```

```
                        return;
                    }
                    ArrayList tempAlreadyFound = new ArrayList();
                    while(searchString.IndexOf(" AND ")>0)
                    {
                        splitPosition = searchString.IndexOf(" AND ");

                        search-
Word=searchString.Substring(0,splitPosition).Trim().ToLower();

      if(!Data.Release.Indexing.IndexGenerator.checkIfNoisyWord(searchWord)
)
                        {
                            foreach(Data.InverseCCCaseWordElement we in
Data.DSSMain.SearchAutomatedCases(searchWord))
                            {

      if(alreadyFound.Contains(we.CCCaseID.ToString()) &&
!slAND.ContainsKey(we.CCCaseID+we.Word))
                                {
                                    //Suchwoerter geklammert von AND´s
                                    slAND.Add(we.CCCaseID+we.Word,we);
                                    tempAlready-
Found.Add(we.CCCaseID.ToString());
#if DEBUG
                                    Console.WriteLine("Wort {0} gefun-
den in: {1}",searchWord,we.CCCaseID);
#endif
                                }
                            }
                        }
                        alreadyFound = tempAlreadyFound;
                        searchString = search-
String.Substring(splitPosition+5,searchString.Length-(splitPosition+5));
                    }


                    searchWord=searchString.Trim().ToLower();

      if(!Data.Release.Indexing.IndexGenerator.checkIfNoisyWord(searchWord)
)
                    {
                        foreach(Data.InverseCCCaseWordElement we in
Data.DSSMain.SearchAutomatedCases(searchWord))
                        {

      if(alreadyFound.Contains(we.CCCaseID.ToString()) &&
!slAND.ContainsKey(we.CCCaseID+we.Word))
                            {
                                //Letztes Suchwort eines AND-Terms
                                slAND.Add(we.CCCaseID+we.Word,we);
                                tempAlready-
Found.Add(we.CCCaseID.ToString());
#if DEBUG
                                Console.WriteLine("Wort {0} gefunden in:
{1}",searchWord,we.CCCaseID);
#endif
                            }
                        }
                    }
                    String tempKey;
                    for(int i=0;i<slAND.Count;i++)
                    {
```

```
                        tempKey=slAND.GetKey(i).ToString().Substring(0,36);
                        if(!sl.ContainsKey(slAND.GetKey(i)) && tempAlready-
Found.Contains(tempKey))
                        {
                              sl.Add(slAND.GetKey(i),slAND.GetByIndex(i));
                        }

                  }
            }

            public static void analyzeSearchString(String searchString)
            {
                  Char[] brackets = new char[]{'(',')'};

                  if(searchString.IndexOfAny(brackets)!=-1)
                  {
                        Console.WriteLine("Klammerung entdeckt
{0}:",searchString);
                        int start = searchString.IndexOf("(");
                        int end = searchString.IndexOf(")");
                        String term = searchString.Substring(start,end-
start);
                        analyzeSearchString(term);
                  }
                  else{
                        char[] separators = new char[] {' ',',',';'};
                        foreach (string word in search-
String.Split(separators))
                        {
                              SortedList sl = new SortedList();
                              foreach(Data.InverseSolutionWordElement we in
Data.DSSMain.SearchAutomatedSolutions(word))
                              {
                                    if(!sl.ContainsKey(we.SolutionID))
                                    {
                                          sl.Add(we.SolutionID,we);
                                    }
                              }
                        }
                  }
            }
      }
}
```

8.2.3 Operation 1: searchInPrimaryList(String searchString)

Diese Methode liefert alle Solutions, die mit dem searchString korrelieren, mit ihren zugehörigen Knoten in Form von einer gerankten ArrayList der Gestalt Node1, Solution1.1, Solution1.2, Node2, Solution2.1, usw..

8.2.3.1 Äquivalenzklassen in der Testdatenmenge

1. **Es wird nach einem einzigen Wort gesucht:** In diesem Fall werden alle Solutions mit ihren zugehörigen Knoten gerankt zurückgeliefert, die das Wort enthalten.

2. **Es wird nach zwei Wörtern gesucht, die mit AND verknüpft sind:** In diesem Fall werden alle Solutions mit ihren zugehörigen Knoten gerankt zurückgeliefert, die sowohl das erste Wort, als auch das zweite Wort enthalten.
3. **Es wird nach zwei Wörtern gesucht, die mit OR verknüpft sind:** In diesem Fall werden alle Solutions mit ihren zugehörigen Knoten gerankt zurückgeliefert, die entweder das erste Wort oder das zweite Wort enthalten.
4. **Es wird nach drei Wörtern gesucht, wobei zwei Wörter durch AND verknüpft sind und das dritte mit OR, dabei ist das AND stärker bindend als das OR:** In diesem Fall werden alle Solutions mit ihren zugehörigen Knoten gerankt zurückgeliefert, die sowohl das erste Wort, als auch das zweite Wort oder das dritte Wort enthalten.
5. **Es wird eine Frage eingegeben, wobei die Wörter der Frage behandelt werden, als wären sie mit OR verknüpft:** In diesem Fall werden alle Solutions mit ihren zugehörigen Knoten gerankt zurückgeliefert, die eines der Wörter unter nicht Berücksichtigung der Stoppwörter enthalten.

8.2.3.2 Test von es wird nach einem einzigen Wort gesucht:

Getestete Eingabedaten

- work

Sollergebnisse

Es sollten 17 Solutions mit ihren zugehörigen ModelTreeNodes zurückgeliefert werden und das Wort work enthalten.

Istergebnisse

Es werden 17 Solutions mit dem dazugehörigen ModelTreeNode zurückgeliefert, die das Wort work enthalten.

8.2.3.3 Test von es wird nach zwei Wörtern gesucht, die mit AND verknüpft sind:

Getestete Eingabedaten

- work AND correct

Sollergebnisse

Es sollten 3 Solutions mit ihren zugehörigen ModelTreeNodes zurückgeliefert werden, die sowohl das Wort work als auch das Wort correct enthalten.

Istergebnisse

Es werden 3 Solutions mit dem dazugehörigen ModelTreeNode zurückgeliefert, die sowohl das Wort work, als auch das Wort correct enthalten.

8.2.3.4 Test von es wird nach zwei Wörtern gesucht, die mit OR verknüpft sind:

Getestete Eingabedaten

- work OR correct

Sollergebnisse

Es sollten 25 Solutions mit ihren zugehörigen ModelTreeNodes zurückgeliefert werden, die entweder das Wort work oder das Wort correct oder auch beide enthalten.

Istergebnisse
Es werden 25 Solutions mit dem dazugehörigen Knoten zurückgeliefert, die entweder das Wort work oder das Wort correct enthalten.

8.2.3.5 Test von es wird nach drei Wörtern gesucht, wobei zwei Wörter durch AND verknüpft sind und das dritte mit OR, dabei ist das AND stärker bindend als das OR:

Getestete Eingabedaten
- work AND correct OR complex

Sollergebnisse
Es sollten 6 Solutions mit ihren zugehörigen ModelTreeNodes zurückgeliefert werden, die entweder das Wort work und das Wort correct enthalten oder das Wort complex.

Istergebnisse
Es werden 6 Solutions zurückgeliefert mit dem dazugehörigen Knoten, die entweder das Wort work und das Wort correct enthalten oder das Wort complex.

8.2.3.6 Test von es wird eine Frage eingegeben, wobei die Wörter der Frage behandelt werden, als wären sie mit OR verknüpft:

Getestete Eingabedaten
- it does not work correct

Sollergebnisse
Es sollten 25 Solutions mit ihren zugehörigen ModelTreeNodes zurückgeliefert werden, die mindestens eines der eingegebenen Wörter enthalten.

Istergebnisse
Es werden 25 Solutions zurückgeliefert mit dem dazugehörigen Knoten, die alle mindestens eines der eingegebenen Wörter enthalten.

8.2.4 Operation 2: searchInSecondaryList(String searchString)

Diese Methode liefert alle CCCases, die mit dem searchString korrelieren in Form von einer gerankten ArrayList.

8.2.4.1 Äquivalenzklassen in der Testdatenmenge

6. **Es wird nach einem einzigen Wort gesucht:** In diesem Fall werden alle CCCases gerankt zurückgeliefert, die das Wort enthalten.
7. **Es wird nach zwei Wörtern gesucht, die mit AND verknüpft sind:** In diesem Fall werden alle CCCases gerankt zurückgeliefert, die sowohl das erste Wort, als auch das zweite Wort enthalten.

8. **Es wird nach zwei Wörtern gesucht, die mit OR verknüpft sind:** In diesem Fall werden alle CCCases gerankt zurückgeliefert, die entweder das erste Wort oder das zweite Wort enthalten.
9. **Es wird nach drei Wörtern gesucht, wobei zwei Wörter durch AND verknüpft sind und das dritte mit OR, dabei ist das AND stärker bindend als das OR:** In diesem Fall werden alle CCCases gerankt zurückgeliefert, die sowohl das erste Wort, als auch das zweite Wort oder das dritte Wort enthalten.
10. **Es wird eine Frage eingegeben, wobei die Wörter der Frage behandelt werden, als wären sie mit OR verknüpft:** In diesem Fall werden alle CCCases gerankt zurückgeliefert, die eines der Wörter unter nicht Berücksichtigung der Stoppwörter enthalten.

8.2.4.2 Test von es wird nach einem einzigen Wort gesucht:

Getestete Eingabedaten
- display

Sollergebnisse
Es sollten drei CCCases zurückgeliefert werden, die das Wort display enthalten.

Istergebnisse
Es werden drei CCCases zurückgeliefert, die das Wort display enthalten.

8.2.4.3 Test von es wird nach zwei Wörtern gesucht, die mit AND verknüpft sind:

Getestete Eingabedaten
- display AND mac

Sollergebnisse
Es sollte ein CCCase zurückgeliefert werden, der sowohl display, als auch mac enthält.

Istergebnisse
Es wird ein CCCase zurückgeliefert, der sowohl display, als auch mac enthält.

8.2.4.4 Test von es wird nach zwei Wörtern gesucht, die mit OR verknüpft sind:

Getestete Eingabedaten
- display OR work

Sollergebnisse
Es sollten drei CCCases zurückgeliefert werden, die entweder das Wort display oder das Wort work oder auch beide enthalten.

Istergebnisse
Es werden drei CCCases zurückgeliefert, die entweder das Wort display oder das Wort work oder auch beide enthalten.

8.2.4.5 Test von es wird nach drei Wörtern gesucht, wobei zwei Wörter durch AND verknüpft sind und das dritte mit OR, dabei ist das AND stärker bindend als das OR:

Getestete Eingabedaten

- display AND work OR complex

Sollergebnisse
Es sollten zwei CCCases zurückgeliefert werden, die entweder die Wörter display und work oder das Wort complex enthalten.

Istergebnisse
Es werden zwei CCCases zurückgeliefert werden, die entweder die Wörter display und work oder das Wort complex enthalten.

8.2.4.6 Test von es wird eine Frage eingegeben, wobei die Wörter der Frage behandelt werden, als wären sie mit OR verknüpft:

Getestete Eingabedaten

- The display does not work

Sollergebnisse
Es sollten drei CCCases zurückgeliefert werden, die jeweils mindestens eines der Wörter aus den Eingabedaten enthalten unter Ausschluss der Stoppwörter.

Istergebnisse
Es werden drei CCCases zurückgeliefert, die jeweils mindestens eines der Wörter aus den Eingabedaten enthalten ohne Berücksichtigung der Stoppwörter.

8.3 Klasse: Model

Autor: Frank Gietmann, Daniel Müller, Nils Müller, Richard Süselbeck

Die Klasse Model ist eine der wichtigsten control Klassen des gesamten Projektes. Sie dient der Erstellung und der Manipulation eines Bayeschen Netzes, also dem Kernstück, des Projektes. Sie stellt außerdem ein Interface dar, auf das die GUI zugreifen kann und Informationen an und aus dem Bayeschen Netz ziehen kann. Es kann mehrere Instanzen dieser Klasse geben, die jeweils an einer Instanz der Klasse ModelTreeNode hängen.

8.3.1 Teststrategie

8.3.1.1 Idee

Es werden alle Methoden getestet, die die Manipulation und die Erstellung eines Bayeschen Netzes unterstützen. Dazu gehören auch die Methoden, auf die die GUI zugreift. Da dies Methoden sind, die die entscheidende Funktionalität des Systems gewährleisten, werden sie entsprechend intensiv getestet.
Die internen Methoden werden auf Testumgebungen getestet, d.h. sie werden auf extra geschriebenen Testklassen und Dummydaten getestet. Die gilt insbesondere für die Methoden zum Betreiben der Inferenz. Daraus folgt das die Reihenfolge innerhalb der Tests der internen Methoden nicht die logische Aufrufreihenfolge des

Systems sein muss. Aus organisatorischen Gründen werden daher die internen Methoden in einer anderen Reihenfolge getestet, als dies logisch notwendig wäre. Die Testergebnisse werden aufgrund der verwendeten Testumgebung davon nicht beeinflusst. Die Methoden, die die Schnittstelle zur GUI darstellen, werden auf vorhandenen Daten in der Datenbank des Systems getestet. Daraus folgt auch, das zuerst die internen Methoden getestet werden und anschließend die darauf zugreifenden Methoden, für das Interface.

8.3.1.2 Klassentestplan

Es folgt die Reihenfolge der Tests der internen Methoden:

5. makeInference (Object o, int answer)

Die anschließend zu testenden Methoden für das Interface werden in folgender Reihenfolge getestet:

6. calculateEvidenceList()
7. getBestQuestions()
8. getBestSolutions()
9. oneStepBack(HistoryElement h)

8.3.2 Testumgebung

8.3.2.1 Treiber für makeInference(Object o, int answer)

```
using System;
using System.Collections;

namespace Inference
{
        /// <summary>
        /// Zusammenfassung für Test.
        /// </summary>
        public class Test
        {
                public Test()
                {
                }
                [STAThread]
                static void Main(string[] args)
                {
                        InferenceMain inf = new InferenceMain();

                        //Ausgabe aller CPT aller Knoten
                        inf.printCPT();

                        //Berechnung der W'keiten für einen Knoten
                        inf.sw.WriteLine("W'keit für für Knoten E: "+
                                Adiuware.Data.Model.makeInference(inf.e,1));
                        inf.sw.Close();
                }

        }
}
```

8.3.2.2 Treiber für calculateEvidenceList(), getBestQuestions(),

getBestSolutions(), oneStepBack(HistoryElement h)

```
using System;
```

```
using System.Collections;
using System.IO;


namespace Adiuware.Data
{
	/// <summary>
	/// Zusammenfassung für InferenceTest.
	/// </summary>
	public class InferenceTest
	{
		private StreamWriter sw;
		public InferenceTest()
		{
		}

		public static void Main (string[] args)
		{
			DSSMain.ConnectionString = "server = egon; uid = admin; pwd = adiuware;
			database = Adiuware";
			InferenceTest test = new InferenceTest();
			test.testInference ();
		}

		public void testInference () {

			Model testModel = DSSMain.ModelTreeNodeRoot.Nodes[0].Nodes[0].Model;


			sw = File.CreateText("C:\\Dokumente und Einstellungen\\Daniel Mül-
			ler\\Adiuware\\Data\\testOut.txt");
			sw.WriteLine("Inferenztest in Model: " + testModel.ID);
			sw.WriteLine("Inferenz ohne eine beantwortete Frage");
			printProbOfAllNodes (testModel);
			Question testQuestion = printBestMethods(testModel);

			testModel.givenAnswer(testQuestion,1);
			HistoryElement hist = new HistoryElement(testQuestion,1);
			sw.WriteLine();
			sw.WriteLine("Inferenz für eine beantwortete Frage");
			printProbOfAllNodes (testModel);
			printBestMethods(testModel);

			sw.WriteLine();
			sw.WriteLine("Inferenz für zurückgenomme Antwort");
			testModel.oneStepBack(hist);
			printProbOfAllNodes (testModel);
			printBestMethods(testModel);

			sw.Close();
		}

		internal Question printBestMethods (Model m)
		{
			Question[] bestTestQuestions = m.getBestQuestions();
			Solution[] bestTestSolutions = m.getBestSolutions();
			sw.WriteLine("Die 1. der 3 besten Fragen ist: " + bestTestQuestions[0].Text);
			sw.WriteLine("Die Frage hat Wkeit: " + bestTestQuestions[0].getWkeit());
			sw.WriteLine("Die 2. der 3 besten Fragen ist: " + bestTestQuestions[1].Text);
			sw.WriteLine("Die Frage hat Wkeit: " + bestTestQuestions[1].getWkeit());
			sw.WriteLine("Die 3. der 3 besten Fragen ist: " + bestTestQuestions[2].Text);
			sw.WriteLine("Die Frage hat Wkeit: " + bestTestQuestions[2].getWkeit());
```

```
                    sw.WriteLine("Die 1. der 3 besten Lösungen ist: " + bestTestSoluti
                    ons[0].Text);
                    sw.WriteLine("Die Frage hat Wkeit: " + bestTestSolutions[0].getWkeit());
                    sw.WriteLine("Die 2. der 3 besten Lösungen ist: " + bestTestSoluti
                    ons[1].Text);
                    sw.WriteLine("Die Frage hat Wkeit: " + bestTestSolutions[1].getWkeit());
                    sw.WriteLine("Die 3. der 3 besten Lösungen ist: " + bestTestSoluti
                    ons[2].Text);
                    sw.WriteLine("Die Frage hat Wkeit: " + bestTestSolutions[2].getWkeit());
                    sw.WriteLine();
                    return bestTestQuestions[0];
                }

                internal void printProbOfAllNodes (Model m)
                {
                    sw.WriteLine("Evidenzliste:");
                    m.erstelleEvidenzListe();
                    for (int i=0;i < m.getEvidenz().Count; i+=2)
                        sw.WriteLine(((Question)m.getEvidenz()[i]).Text +" : " +
                            m.getEvidenz()[i+1]);
                    sw.WriteLine();

                    sw.WriteLine("Wkeit aller Fragen:");
                    foreach (Question q in m.Questions)
                        sw.WriteLine("Wkeit für Knoten: "+ q.Text + " : " +
                            m.betreibeInferenz(q,1,m.getEvidenz()));
                    sw.WriteLine();

                    sw.WriteLine("Wkeit aller Lösungen:");
                    foreach (Solution s in m.Solutions)
                        sw.WriteLine("Wkeit für Knoten: "+ s.Text + " : " +
                            m.betreibeInferenz(s,1,m.getEvidenz()));
                    sw.WriteLine();
                }
        }
}
```

8.3.2.3 Verwendete Klasse InferenceMain

```
using System;
using System.IO;
using System.Collections;

namespace Inference
{
        class InferenceMain
        {
                public Knoten a;
                public Knoten b;
                public Knoten c;
                public Knoten d;
                public Knoten e;
                public Knoten f;
                public Knoten g;
                public ArrayList questions;
                public ArrayList solutions;
                public StreamWriter sw;


                public InferenceMain ()
                {
                    questions = new ArrayList();
```

```
            solutions = new ArrayList();
            sw = File.CreateText("D:\\Applikationen\\C#\\Inference\\Inference\\out.txt");
            baueNetz();
}

public void baueNetz ()
{
            //Knoten A wird generiert
            a = new Knoten("A", true,1,4);
            //Knoten B wird generiert
            b = new Knoten("B", true,1,4);
            //Knoten C wird generiert
            c = new Knoten("C", true,3,5);
            //Knoten D wird generiert
            d = new Knoten("D", true,9,6);
            //Knoten E wird generiert
            e = new Knoten("E", false,9,4);
            //Knoten F wird generiert
            f = new Knoten("F", false,3,3);
            //Knoten G wird generiert
            g = new Knoten("G", true, 1,4);

            //Verbindungen sezten
            ArrayList aKinder = new ArrayList();
            aKinder.Add(c);
            a.kinder = aKinder;
            questions.Add(a);

            ArrayList bKinder = new ArrayList();
            bKinder.Add(d);
            b.kinder = bKinder;
            questions.Add(b);

            ArrayList cKinder = new ArrayList();
            cKinder.Add(d);
            c.kinder = cKinder;
            ArrayList cEltern = new ArrayList();
            cEltern.Add(a);
            c.eltern = cEltern;
            questions.Add(c);

            ArrayList dKinder = new ArrayList();
            dKinder.Add(e);
            dKinder.Add(f);
            d.kinder = dKinder;
            ArrayList dEltern = new ArrayList();
            dEltern.Add(b);
            dEltern.Add(c);
            d.eltern = dEltern;
            questions.Add(d);

            ArrayList eEltern = new ArrayList();
            eEltern.Add(g);
            eEltern.Add(d);
            e.eltern = eEltern;
            solutions.Add(e);

            ArrayList fEltern = new ArrayList();
            fEltern.Add(d);
            f.eltern = fEltern;
            solutions.Add(f);
```

```
ArrayList gKinder = new ArrayList();
gKinder.Add(e);
g.kinder = gKinder;
questions.Add(g);

//CPT Eintraege setzen
a.cpt[0,0] = 0.3;
a.cpt[0,1] = 0.2;
a.cpt[0,2] = 0.5;
a.cpt[0,3] = 0.4;

b.cpt[0,0] = 0.1;
b.cpt[0,1] = 0.2;
b.cpt[0,2] = 0.7;
b.cpt[0,3] = 0.3;

c.cpt[0,0] = 1;
c.cpt[0,1] = 0.1;
c.cpt[0,2] = 0.2;
c.cpt[0,3] = 0.7;
c.cpt[0,4] = 0.2;
c.cpt[1,0] = 2;
c.cpt[1,1] = 0.4;
c.cpt[1,2] = 0.1;
c.cpt[1,3] = 0.5;
c.cpt[1,4] = 0.4;
c.cpt[2,0] = 3;
c.cpt[2,1] = 0.6;
c.cpt[2,2] = 0.3;
c.cpt[2,3] = 0.1;
c.cpt[2,4] = 0.6;

d.cpt[0,0] = 1;
d.cpt[0,1] = 1;
d.cpt[0,2] = 0.1;
d.cpt[0,3] = 0.2;
d.cpt[0,4] = 0.7;
d.cpt[0,5] = 0.2;
d.cpt[1,0] = 2;
d.cpt[1,1] = 1;
d.cpt[1,2] = 0.4;
d.cpt[1,3] = 0.2;
d.cpt[1,4] = 0.4;
d.cpt[1,5] = 0.3;
d.cpt[2,0] = 3;
d.cpt[2,1] = 1;
d.cpt[2,2] = 0.1;
d.cpt[2,3] = 0.3;
d.cpt[2,4] = 0.6;
d.cpt[2,5] = 0.3;
d.cpt[3,0] = 1;
d.cpt[3,1] = 2;
d.cpt[3,2] = 0.3;
d.cpt[3,3] = 0.3;
d.cpt[3,4] = 0.4;
d.cpt[3,5] = 0.6;
d.cpt[4,0] = 2;
d.cpt[4,1] = 2;
d.cpt[4,2] = 0.8;
d.cpt[4,3] = 0.1;
d.cpt[4,4] = 0.1;
d.cpt[4,5] = 0.7;
```

```
d.cpt[5,0] = 3;
d.cpt[5,1] = 2;
d.cpt[5,2] = 0.2;
d.cpt[5,3] = 0.1;
d.cpt[5,4] = 0.7;
d.cpt[5,5] = 0.1;
d.cpt[6,0] = 1;
d.cpt[6,1] = 3;
d.cpt[6,2] = 0.1;
d.cpt[6,3] = 0.6;
d.cpt[6,4] = 0.3;
d.cpt[6,5] = 0.3;
d.cpt[7,0] = 2;
d.cpt[7,1] = 3;
d.cpt[7,2] = 0.7;
d.cpt[7,3] = 0.1;
d.cpt[7,4] = 0.2;
d.cpt[7,5] = 0.2;
d.cpt[8,0] = 3;
d.cpt[8,1] = 3;
d.cpt[8,2] = 0.3;
d.cpt[8,3] = 0.4;
d.cpt[8,4] = 0.3;
d.cpt[8,5] = 0.1;

e.cpt[0,0] = 1;
e.cpt[0,1] = 1;
e.cpt[0,2] = 0.9;
e.cpt[0,3] = 0.1;
e.cpt[1,0] = 2;
e.cpt[1,1] = 1;
e.cpt[1,2] = 0.3;
e.cpt[1,3] = 0.7;
e.cpt[2,0] = 3;
e.cpt[2,1] = 1;
e.cpt[2,2] = 0.2;
e.cpt[2,3] = 0.8;
e.cpt[3,0] = 1;
e.cpt[3,1] = 2;
e.cpt[3,2] = 0.6;
e.cpt[3,3] = 0.4;
e.cpt[4,0] = 2;
e.cpt[4,1] = 2;
e.cpt[4,2] = 0.8;
e.cpt[4,3] = 0.2;
e.cpt[5,0] = 3;
e.cpt[5,1] = 2;
e.cpt[5,2] = 0.6;
e.cpt[5,3] = 0.4;
e.cpt[6,0] = 1;
e.cpt[6,1] = 3;
e.cpt[6,2] = 0.4;
e.cpt[6,3] = 0.6;
e.cpt[7,0] = 2;
e.cpt[7,1] = 3;
e.cpt[7,2] = 0.7;
e.cpt[7,3] = 0.3;
e.cpt[8,0] = 3;
e.cpt[8,1] = 3;
e.cpt[8,2] = 0.5;
e.cpt[8,3] = 0.5;
```

```
                f.cpt[0,0] = 1;
                f.cpt[0,1] = 0.1;
                f.cpt[0,2] = 0.9;
                f.cpt[1,0] = 2;
                f.cpt[1,1] = 0.2;
                f.cpt[1,2] = 0.8;
                f.cpt[2,0] = 3;
                f.cpt[2,1] = 0.6;
                f.cpt[2,1] = 0.4;

                g.cpt[0,0] = 0.2;
                g.cpt[0,1] = 0.3;
                g.cpt[0,2] = 0.4;
                g.cpt[0,3] = 0.5;
        }

        public void printCPT ()
        {
                foreach (Knoten k in questions)
                {
                        sw.WriteLine("CPT für Frageknoten "+ k.name);
                        for(int i = 0; i < k.cpt.GetLength(0); i++)
                        {
                                for (int j = 0; j < k.cpt.GetLength(1); j++)
                                        sw.Write(k.cpt[i,j]+" | ");
                                sw.WriteLine();
                        }
                        sw.WriteLine();
                }
                foreach (Knoten k in solutions)
                {
                        sw.WriteLine("CPT für Lösungsknoten "+ k.name);
                        for(int i = 0; i < k.cpt.GetLength(0); i++)
                        {
                                for (int j = 0; j < k.cpt.GetLength(1); j++)
                                        sw.Write(k.cpt[i,j]+" | ");
                                sw.WriteLine();
                        }
                        sw.WriteLine();
                }
                sw.WriteLine();
        }
    }
}
```

8.3.2.4 Verwendete Klasse Knoten

```
using System;
using System.Collections;

namespace Inference
{
    public class Knoten
    {
        public String name;
        public ArrayList kinder;
        public ArrayList eltern;
        public double[,] cpt;
        public bool frage;

        public Knoten(String name, bool f, int cpt_x, int cpt_y)
```

```
        {
            this.name = name;
            cpt = new double[cpt_x,cpt_y];
            frage = f;
            kinder = new ArrayList();
            eltern = new ArrayList();
        }

        public String getName () {return name;}

        public Knoten getKind(int index)
        {
            return (Knoten)kinder[index];
        }

        public Knoten getElter(int index)
        {
            return (Knoten)eltern[index];
        }

        public ArrayList getEintrag(ArrayList elternBelegung)
        {
            ArrayList a = new ArrayList();
            bool addresse = false;
            for (int i = 0; i < cpt.GetLength(0); i++)
            {
                for (int j = 0; j < elternBelegung.Count; j++)
                {
                    if (cpt[i,j] != (double)elternBelegung[j])
                    {
                        addresse = false;
                        break;
                    }
                    else addresse = true;
                }
                if (addresse)
                {
                    for (int k = elternBelegung.Count+1; k < cpt.GetLength(1);
                    k++)
                        a.Add(cpt[i,k]);
                    break;
                }
            }
            return a;
        }
    }

}
```

8.3.3 Operation 1: makeInference (Object o, int answer)

Diese Methode berechnet für einen übergeben Knoten des Bayeschen Netzes die Wahrscheinlichkeit auf der Basis der gegebenen Eltern. Dabei kann sowohl für eine Solution als auch für eine Question die Wahrscheinlichkeit berechnet werden. Der zweite Übergabeparameter legt fest für welche Antwort der Question die W'keit berechnet werden soll. Dieser ist nur für die Rekursionen notwendig, somit auch nicht für Solutions.

8.3.3.1 Äquivalenzklassen in der Testdatenmenge

3. **Alle Eltern einer Question evident:** In diesem Fall wird die W'keit für eine Question berechnet, die Eltern hat, für diese aber bereits eine Antwort gegeben ist, und somit die Wahrscheinlichkeits-verteilung auf den Antworten der Eltern feststeht.
4. **Alle Eltern einer Solution evident:** In diesem Fall wird die W'keit für eine Solution berechnet, die Eltern hat, für diese aber bereits eine Antwort gegeben ist, und somit die Wahrscheinlichkeits-verteilung auf den Antworten der Eltern feststeht.
5. **Keine Eltern:** In diesem Fall wird die W'keit für einen Knoten berechnet, der keine Eltern hat, also eine Wurzel des Bayeschen Netzes ist. Die Unterscheidung zwischen Question und Solution kann hier und in allen weiteren Äquivalenzklassen wegfallen, da der Unterscheid nur in der Auswertung der richtigen Spalte der CPT liegt. Ob die richtige Spalte für eine Question oder Solution ausgewertet wird, wurde schon in der beiden vorherigen Äquivalenzklassen getestet.
6. **Mindestens ein Elter nicht evident:** In diesem Fall wird eine Rekursion in Gang gesetzt um zuerst die W'keit für den, oder die nicht evidenten Eltern zu berechnen, um anschließend die W'keit für den gesuchten Knoten berechnen zu können. Die Anzahl der nicht evidenten Eltern hat keinen Einfluss auf die Korrektheit. Wenn der Algorithmus auf einem nicht evidenten Elter korrekt arbeitet, macht er dies auch für mehrere nicht evidente Eltern.

8.3.3.2 Test von alle Eltern einer Question evident:

Getestete Eingabedaten

- Question A, der Elter von C mit gegebener Antwort 2
- Question C

Sollergebnisse

An Question C soll im CPT nur der Utilitywert der Letzten Spalte (Index 4) und der 2. Zeile (Index 1) mit 1 gewichtet werden. Dabei soll 0.4 heraus kommen.

Istergebnisse

Berechnungen an Knoten: C

Utility in C[0,4] : 0,2 * W'keit das A mit 1 beantwortet wird: 0

Utility in C[1,4] : 0,4 * W'keit das A mit 2 beantwortet wird: 1

Utility in C[2,4] : 0,6 * W'keit das A mit 3 beantwortet wird: 0

W'keit für für Knoten D: 0,4

8.3.3.3 Test von alle Eltern einer Solution evident:

Getestete Eingabedaten

- Question G, der Elter von E mit gegebener Antwort 3
- Question D, der Elter von E mit gegebener Antwort 2
- Solution E

Sollergebnisse

An Solution E sollen im CPT die Werte der 3. Spalte (Index 2) mit den jeweiligen W'keiten der Antworten der Questions gewichtet werden, die den durch die Zeile des CPTs vorgegebenen Antworten entsprechen. Dabei soll 0.6 heraus kommen.

Istergebnisse

Berechnungen an Knoten: E

W'keit in E[0,2] : 0,9 * W'keit das G mit 1 beantwortet wird: 0
* W'keit das D mit 1 beantwortet wird: 0

W'keit in E[1,2] : 0,3 * W'keit das G mit 2 beantwortet wird: 0
* W'keit das D mit 1 beantwortet wird: 0

W'keit in E[2,2] : 0,2 * W'keit das G mit 3 beantwortet wird: 1
* W'keit das D mit 1 beantwortet wird: 0

W'keit in E[3,2] : 0,6 * W'keit das G mit 1 beantwortet wird: 0
* W'keit das D mit 2 beantwortet wird: 1

W'keit in E[4,2] : 0,8 * W'keit das G mit 2 beantwortet wird: 0
* W'keit das D mit 2 beantwortet wird: 1

W'keit in E[5,2] : 0,6 * W'keit das G mit 3 beantwortet wird: 1
* W'keit das D mit 2 beantwortet wird: 1

W'keit in E[6,2] : 0,4 * W'keit das G mit 1 beantwortet wird: 0
* W'keit das D mit 3 beantwortet wird: 0

W'keit in E[7,2] : 0,7 * W'keit das G mit 2 beantwortet wird: 0
* W'keit das D mit 3 beantwortet wird: 0

W'keit in E[8,2] : 0,5 * W'keit das G mit 3 beantwortet wird: 1
* W'keit das D mit 3 beantwortet wird: 0

W'keit für für Knoten E: 0,6

8.3.3.4 Test von Keine Eltern:

Getestete Eingabedaten

- Question B

Sollergebnisse

An Question B soll im CPT nur der Utilitywert der Letzten Spalte (Index 4) ausgegeben werden. Dabei soll 0.3 herauskommen.

Istergebnisse

W'keit für für Knoten B: 0,3

8.3.3.5 Test von Mindestens ein Elter nicht evident:

Getestete Eingabedaten

- Question A, der Elter von C mit gegebener Antwort 1
- Question B, der Elter von D mit gegebener Antwort 3
- Question C, der Elter von D mit keiner gegebenen Antwort
- Question D

Sollergebnisse

An Question D soll für den Elter C ein rekursiver Aufruf gemacht werden, der die W'keit für C berechnet und diese dann mit der Evidenz von B zur Berechnung von D verwendet werden. Dabei soll 0.12 herauskommen.

Istergebnisse

Berechnungen an Knoten: D
Utility in D[0,5] : 0,2 * W'keit das B mit 1 beantwortet wird: 0
Berechnungen an Knoten: C
W'keit in C[0,1] : 0,1 * W'keit das A mit 1 beantwortet wird: 1
W'keit in C[1,1] : 0,4 * W'keit das A mit 2 beantwortet wird: 0
W'keit in C[2,1] : 0,6 * W'keit das A mit 3 beantwortet wird: 0
* W'keit das C mit 1 beantwortet wird: 0,1

Utility in D[1,5] : 0,3 * W'keit das B mit 2 beantwortet wird: 0
Berechnungen an Knoten: C
W'keit in C[0,1] : 0,1 * W'keit das A mit 1 beantwortet wird: 1
W'keit in C[1,1] : 0,4 * W'keit das A mit 2 beantwortet wird: 0
W'keit in C[2,1] : 0,6 * W'keit das A mit 3 beantwortet wird: 0
* W'keit das C mit 1 beantwortet wird: 0,1

Utility in D[2,5] : 0,3 * W'keit das B mit 3 beantwortet wird: 1
Berechnungen an Knoten: C
W'keit in C[0,1] : 0,1 * W'keit das A mit 1 beantwortet wird: 1
W'keit in C[1,1] : 0,4 * W'keit das A mit 2 beantwortet wird: 0
W'keit in C[2,1] : 0,6 * W'keit das A mit 3 beantwortet wird: 0
* W'keit das C mit 1 beantwortet wird: 0,1

Utility in D[3,5] : 0,6 * W'keit das B mit 1 beantwortet wird: 0
Berechnungen an Knoten: C
W'keit in C[0,2] : 0,2 * W'keit das A mit 1 beantwortet wird: 1
W'keit in C[1,2] : 0,1 * W'keit das A mit 2 beantwortet wird: 0
W'keit in C[2,2] : 0,3 * W'keit das A mit 3 beantwortet wird: 0
* W'keit das C mit 2 beantwortet wird: 0,2

Utility in D[4,5] : 0,7 * W'keit das B mit 2 beantwortet wird: 0
Berechnungen an Knoten: C
W'keit in C[0,2] : 0,2 * W'keit das A mit 1 beantwortet wird: 1
W'keit in C[1,2] : 0,1 * W'keit das A mit 2 beantwortet wird: 0
W'keit in C[2,2] : 0,3 * W'keit das A mit 3 beantwortet wird: 0
* W'keit das C mit 2 beantwortet wird: 0,2

Utility in D[5,5] : 0,1 * W'keit das B mit 3 beantwortet wird: 1
Berechnungen an Knoten: C
W'keit in C[0,2] : 0,2 * W'keit das A mit 1 beantwortet wird: 1
W'keit in C[1,2] : 0,1 * W'keit das A mit 2 beantwortet wird: 0
W'keit in C[2,2] : 0,3 * W'keit das A mit 3 beantwortet wird: 0
* W'keit das C mit 2 beantwortet wird: 0,2

Utility in D[6,5] : 0,3 * W'keit das B mit 1 beantwortet wird: 0

Berechnungen an Knoten: C
W'keit in C[0,3] : 0,7 * W'keit das A mit 1 beantwortet wird: 1
W'keit in C[1,3] : 0,5 * W'keit das A mit 2 beantwortet wird: 0
W'keit in C[2,3] : 0,1 * W'keit das A mit 3 beantwortet wird: 0
 * W'keit das C mit 3 beantwortet wird: 0,7

Utility in D[7,5] : 0,2 * W'keit das B mit 2 beantwortet wird: 0
Berechnungen an Knoten: C
W'keit in C[0,3] : 0,7 * W'keit das A mit 1 beantwortet wird: 1
W'keit in C[1,3] : 0,5 * W'keit das A mit 2 beantwortet wird: 0
W'keit in C[2,3] : 0,1 * W'keit das A mit 3 beantwortet wird: 0
 * W'keit das C mit 3 beantwortet wird: 0,7

Utility in D[8,5] : 0,1 * W'keit das B mit 3 beantwortet wird: 1
Berechnungen an Knoten: C
W'keit in C[0,3] : 0,7 * W'keit das A mit 1 beantwortet wird: 1
W'keit in C[1,3] : 0,5 * W'keit das A mit 2 beantwortet wird: 0
W'keit in C[2,3] : 0,1 * W'keit das A mit 3 beantwortet wird: 0
 * W'keit das C mit 3 beantwortet wird: 0,7

W'keit für für Knoten D: 0,12

8.3.4 Operation calculateEvidenceList()

Diese Methode berechnet die Evidenz für das aktuelle Model. Dabei werden alle Questions in dem Model überprüft, ob sie beantwortet wurden, und wenn ja mit welcher Antwort. Diese Informationen werden in einer Liste gespeichert.

8.3.4.1 Äquivalenzklassen in der Testdatenmenge

1. **Nichts Evident:** Keine Questions wurden beantwortet
2. **Min. 1 Question gesetzt:** Eine oder mehrere Questions wurden gesetzt

8.3.4.2 Test von Nichts Evident:

Sollergebnisse
Da keine Frage beantwortet wurde, darf nichts in der Liste stehen.

Istergebnisse
Die Liste ist leer.

8.3.4.3 Test von Min. 1 Question gesetzt:

Getestete Eingabedaten

- Question B wurde mit 1 beantwortet
- Question G wurde mit 3 beantwortet

Sollergebnisse
Die beantworteten fragen müssen gefolgt von den jeweiligen Antworten in der Liste stehen.

Istergebnisse
Evidenz: B, 1, G, 3,

8.3.5 Operation getBestQuestions()

Diese wird von der GUI aufgerufen und liefert die drei besten Questions zurück, nachdem für alle die W'keiten mit makeInference(*) berechnet wurden. Die drei Besten sind die mit den höchsten W'keiten.

8.3.5.1 Äquivalenzklassen in der Testdatenmenge

1. **Alle W'keiten liegen vor:** Alle Questions habe eine W'keit.

8.3.5.2 Test von Alle W'keiten liegen vor:

Getestete Eingabedaten

- Es werden die Daten in der Datenbank benutzt

Sollergebnisse
Eine Liste aller berechneten W'keiten wird ausgegeben, sowie die drei ausgewählten Questions. Diese drei müssen die drei höchsten von allen sein.

Istergebnisse
Wkeit aller Fragen:
Wkeit für Knoten: What is the printing problem : 0,435
Wkeit für Knoten: Which lights are blinking? : 0,0234
Wkeit für Knoten: Does the printer cannot print a configuration page? : 0,246
Wkeit für Knoten: Does the printer turn on? : 0,5
Wkeit für Knoten: Does the printer draws paper from the wrong tray? : 0,1003

Die 1. der 3 besten Fragen ist: Does the printer turn on?
Die Frage hat Wkeit: 0,5
Die 2. der 3 besten Fragen ist: What is the printing problem
Die Frage hat Wkeit: 0,435
Die 3. der 3 besten Fragen ist: Does the printer cannot print a configuration page?
Die Frage hat Wkeit: 0,246

8.3.6 Operation getBestSolutions()

Diese wird von der GUI aufgerufen und liefert die drei besten Solutions zurück, nachdem für alle die W'keiten mit makeInference(*) berechnet wurden. Die drei Besten sind die mit den höchsten W'keiten.

8.3.6.1 Äquivalenzklassen in der Testdatenmenge

1. **Alle W'keiten liegen vor:** Alle Solutions habe eine W'keit.

8.3.6.2 Test von Alle W'keiten liegen vor:

Getestete Eingabedaten

- Es werden die Daten in der Datenbank benutzt

Sollergebnisse
Eine Liste aller berechneten W'keiten wird ausgegeben, sowie die drei ausgewählten Solutions. Diese drei müssen die drei höchsten von allen sein.

Istergebnisse
W'keit aller Lösungen:
Wkeit für Knoten: Make sure that a printer cover is not open : 0,017318715645454
Wkeit für Knoten: Ensure that you are using the correct printer driver. : 0,017318715645454
Wkeit für Knoten: Wait a few more minutes. : 0,0115458104303027
Wkeit für Knoten: Make sure control panel reads Processing Job. : 0,017318715645454
Wkeit für Knoten: Open the Top Cover. Reseat the toner cartridge. : 0,0230916208606054
Wkeit für Knoten: Open Trays 2 and 3. : 0,0230916208606054
Wkeit für Knoten: Make sure to select Manual Feed or Tray 1 through your software application and the size and type for which the tray is configured. : 0,0303747075162916
Wkeit für Knoten: Load the correct paper into the tray. : 0,0242997660130333
Wkeit für Knoten: Make sure that the appropriate paper size and type are selected from the software application for the paper size loaded. : 0,0303747075162916
Wkeit für Knoten: Adjust the tray correctly for the paper size loaded. : 0,018224824509775
Wkeit für Knoten: The power cord is not firmly plugged into both the printer and the power receptacle. : 0,018224824509775
Wkeit für Knoten: Check cables and connectors between the tray and the printer. : 0,0303747075162916

Die 1. der 3 besten Lösungen ist: Make sure to select Manual Feed or Tray 1 through your software application and the size and type for which the tray is configured.
Die Frage hat Wkeit: 0,0303747075162916
Die 2. der 3 besten Lösungen ist: Make sure that the appropriate paper size and type are selected from the software application for the paper size loaded.
Die Frage hat Wkeit: 0,0303747075162916
Die 3. der 3 besten Lösungen ist: Check cables and connectors between the tray and the printer.
Die Frage hat Wkeit: 0,0303747075162916

8.3.7 Operation oneStepBack(HistoryElement h)

Diese Methode wird von der GUI aufgerufen um eine gegebene Antwort zurückzunehmen. Dabei wird die Information, welches die zuletzt gemachte Aktion war, übergeben.

8.3.7.1 Äquivalenzklassen in der Testdatenmenge

1. **Aktion ausgeführt:** Eine Aktion die zurückgenommen werden kann wurde durchgeführt und kann übergeben werden.
2. **Aktion falsch:** Entweder ist eine Aktion die zu einem anderen Model gehört übergeben worden, oder die Aktion ist leer.

8.3.7.2 Test von Aktion ausgeführt:

Getestete Eingabedaten

- Es werden die Daten in der Datenbank benutzt
- Evidenzliste = „Does the Printer Turn on?“, „no“
- HistoryElement mit Question „Does the Printer Turn on?“ und Antwort „no“

Sollergebnisse
Die Frage die übergeben wurde und vorher in der Evidenzliste stand darf nun nicht mehr in der Liste stehen. Die Evidenz muss leer sein.

Istergebnisse
Evidenz: leer

8.3.7.3 Test von Aktion falsch:

Getestete Eingabedaten

- Es werden die Daten in der Datenbank benutzt
- Evidenzliste = „Does the Printer Turn on?“, „no“
- HistoryElement mit Question „What color do you like?“ und Antwort „green“

Sollergebnisse
Die Evidenzliste darf sich nicht verändert haben, da die im Parameter übergeben frage nicht im Model enthalten ist.

Istergebnisse
Evidenz: „Does the Printer Turn on?“, „no“

9 Klassentests Adiuware.Windows

9.1 Teststrategie

Das Windows-Frontend zur Benutzung und Verwaltung des Adiuware Decision Support Systems vereint die Funktionalität mehrerer Einzelmodule, von denen die Suchfunktionalität, die Datenbankanbindung sowie die generelle Berechnungslogik zu den wichtigsten gehören.

Da all diese Module unabhängig voneinander Entwickelt wurden, werden sie natürlich auch gesondert getestet, weshalb ihre korrekte Funktionsweise bei der Überprüfung der Windows-Oberfläche von Adiuware vorausgesetzt werden muss.

Ebenso liegt es in der Natur der Dinge, dass eine grafische Oberfläche auf einer gegebenen Bibliothek von Schaltflächen und Steuerelementen basiert. Auch die fehlerfreie Arbeit dieser Elemente eines „Drittanbieters“ wird als gegeben angenommen – wir werden also im Folgenden die entsprechenden, von Microsoft Visual Studio .NET automatisch erzeugten Programmteile nicht gesondert testen.

Die für das Frontend entwickelte Programmlogik verteilt sich größtenteils auf zwei Codedateien:

- In der Hauptdatei „AdiWinGui.cs“ sind sämtliche Methoden untergebracht, die für die korrekte Anzeige aller Daten verantwortlich sind, indem sie z.B. den Modellbaum auf der Suche nach einem bestimmten Knoten durchlaufen. Hierbei werden ausschließlich Klassen referenziert, welche speziell für die Windows-Oberfläche geschrieben wurden.

- Die Datei „Preprocessing.cs“ hingegen stellt eine Reihe statischer Methoden zur Verfügung, welche direkt mit den anderen Modulen von Adiuware kommunizieren und stellt somit die Schnittstelle zur Programmlogik sowie zur Datenbank bereit.

Weitere wichtige Klassen sind diverse Baumknoten wie z.B. „WinformsTreeNode.cs“, welche verschiedene Konstrukte der Programmlogik kapseln und für die Anzeige in der Oberfläche aufbereiten. Insbesondere die enthaltenen Methoden zur dynamischen Erzeugung des gesamten zugrunde liegenden Baumes müssen mehreren Tests unterzogen werden.

9.2 Vorbereitung

Das zentrale Element der Programmkategorien „Case Explorer“ und „Case Creator“ ist der Modellbaum. Als ersten Schritt erzeugen wir daher zunächst eine Reihe verschiedenerer Bäume, die folgende Eigenschaften besitzen:

- Mindestanzahl der Kinder der Wurzel: 3
- Mindesttiefe des gesamten Baumes: 8
- Nicht alle Blattknoten enthalten notwendigerweise ein Modell
- Jedes Modell enthält mindestens:
    - 4 Fragen mit je mindestens 2 Antworten

- 4 Lösungen

9.3 Case Creator

Durch die Erstellung dieser Bäume unterziehen wir den Case Creator bereits eines eingehenden Tests, der uns unter anderem folgende Fragen beantwortet:

- Lassen sich alle benötigten Elemente erzeugen?
- Können Relationen zwischen Fragen und Lösungen hergestellt werden?
- Ist eine Berechnung der neu erstellten sowie modifizierten Modelle möglich?
- Wird die Berechnung eines neu erstellten oder veränderten Modells vor dem Wechsel zu einem anderen Programmteil bzw. vor Beendung des gesamten Programms erzwungen?
- Werden Bedingungen, die eine erfolgreiche Berechnung eines Modells behindern würden, abgefangen? Dazu gehören z.B. Fragen, denen keine Antwort zugewiesen wurde.

✓ *Ergebnis: alle Fragen lassen sich positiv beantworten.*

Unter anderem werden auf diese Weise folgende wichtige Methoden getestet:

AdiWinGui.cs
- private void refreshCaseCreatorModelSelectionTreeView(bool expand)
- private void refreshCaseCreatorSubmodelTreeView(WinformsTreeNode myWinNode, bool expand)
- private void refreshCaseCreatorSubmodelSectionDetailsTreeView(TreeNode selected, bool expand)
- private void manageModelButtons(Adiuware.Data.ModelTreeNode selectedNode)
- private WinformsTreeNode getNodeByID(WinformsTreeNode currentNode, System.Guid searchID)

Preprocessing.cs
- public static void setUnproperQuestion(Adiuware.Data.Question q)
- public static void deleteUnproperQuestion(Adiuware.Data.Question q)
- public static void addChangedModel(Adiuware.Data.Model model)
- public static void recalculateModels(WinformsTreeNode root)
- private static void traverseTreeToRecalculateModels(WinformsTreeNode currentNode, ArrayList changedModels)

WinformsTreeNode.cs
- public WinformsTreeNode(Adiuware.Data.ModelTreeNode myModelNode, bool showModel)
- public void buildTree(bool excludeModels, WinformsTreeNode excludeNode)
- public void createNewModel()

9.4 Case Explorer

Die so erstellten Bäume dienen nun als Grundlage für den Test des Case Explorers, in welchem wir zunächst die Suchfunktionalität überprüfen. Zu diesem Zwecke geben wir in das Suchfeld unterschiedliche Begriffe ein, von denen wir wissen, dass sie

im aktuellen Baum nicht bzw. genau einmal bzw. mehr als einmal vorkommen. Dieser Test dient der Beantwortung folgender Fragen:

- Wird das Ergebnis einer erfolglosen Suche korrekt dargestellt?
- Wird eine gefundene Lösung korrekt angezeigt und ihrem jeweiligen Knoten zugeordnet?
- Funktioniert diese Zuordnung ebenfalls bei mehreren einem Knoten zugeordneten Lösungen?
- Arbeitet die Anzeige ebenfalls korrekt bei mehreren Lösungen, die sich z.T. auf unterschiedliche Knoten verteilen?
- Werden die Details einer gefundenen Lösung richtig wiedergegeben?
- Wird bei der Auswahl eines der gefundenen Knoten die Einordnung in den Modellbaum an der Position eben dieses Knotens fehlerfrei vorgenommen?

✓ *Ergebnis: alle Fragen lassen sich positiv beantworten.*

Unter anderem werden auf diese Weise folgende wichtige Methoden getestet:

AdiWinGui.cs
- private void caseExplorerSearchButton_Click(object sender, System.EventArgs e)
- private WinformsTreeNode getNoteByIDWithoutRootForCaseExplorer(System.Guid searchID)

Preprocessing.cs
- public static System.Collections.ArrayList searchInPrimaryList(String suchString)

SearchResultDialog.cs
- private void fillTreeView(ArrayList resultList)
- private void searchButton_Click(object sender, System.EventArgs e)
- private void showButton_Click(object sender, System.EventArgs e)

Im Anschluss unterziehen wir den eigentlichen Modellbaum einer genaueren Betrachtung, indem wir unterschiedliche Knoten expandieren und auswählen. Enthält der gewählte Knoten ein Modell, so beantworten wir exemplarisch einige der gestellten Fragen. Wir erhalten so Aufschluss über folgende Fragestellungen:

- Werden die Knoten des zuvor erzeugten Modellbaumes richtig dargestellt?
- Wird die Auswahl der derzeit „besten Fragen“ innerhalb eines Modells korrekt wiedergegeben?
- Gilt dies auch für die Auswahl der derzeit „besten Lösungen“?
- Lässt sich eine gewählte Frage beantworten, und wird durch diese Antwort eine Neuberechnung der besten Fragen und Lösungen initiiert?
- Können Detailinformationen zu Fragen und Lösungen abgerufen werden?

✓ *Ergebnis: alle Fragen lassen sich positiv beantworten.*

Unter anderem werden auf diese Weise folgende wichtige Methoden getestet:

AdiWinGui.cs

- private void caseExplorerTreeView_AfterSelect(object sender, System.Windows.Forms.TreeViewEventArgs e)
- private void questionSelected(object sender, System.EventArgs e)

Preprocessing.cs
- public static Adiuware.Data.Question[] getBestQuestion(Adiuware.Data.Question question, int answer)
- public static Adiuware.Data.Solution[] getBestSolution(Adiuware.Data.Question question, int answer)

9.5 Call Editor

Die dritte und letzte Kategorie, den Call Editor, prüfen wir durch die Erzeugung, Anzeige und Verwaltung verschiedener Boxen und Calls. Außerdem greifen wir erneut auf die Suchfunktion des Case Explorers zurück, in der wir aber nun die Sekundärsuche aktivieren, mit der wir nach zuvor erzeugten Calls suchen. Folgende Fragen lassen sich durch dieses Vorgehen beantworten:

- Können Boxen in den drei Hauptkategorien „Global Box“, „Global Group Box“ und „Personal Box“ anlegen?
- Lassen sich beliebige weitere Sub-Boxen erstellen?
- Können in jeder angelegten Box neue Calls angelegt werden?
- Werden die Detailinformationen eines gewählten Calls wie erwartet wiedergegeben?
- Lassen sich den verschiedenen Teilbereichen eines Calls („Problem“, „Action“, „Notes“, „Links“) neue Informationen hinzufügen, und werden diese Änderungen gespeichert?
- Wird das Ergebnis einer Suche, welche keinen/einen/mehrere Suchtreffer zutage fördert, richtig wiedergegeben?
- Hat eine Anwahl eines durch eine Suche gefundenen Calls einen Wechsel zum Call Editor sowie eine entsprechende Einordnung in den Call-Baum zur Folge?

✓ *Ergebnis: alle Fragen lassen sich positiv beantworten.*

AdiWinGui.cs
- private void caseExplorerSearchButton_Click(object sender, System.EventArgs e)
- private Adiuware.Windows.WinformsCaseTreeNode getCaseByIDWithoutRoot(System.Guid searchID)
- private void callEditorTreeView_AfterSelect(object sender, System.Windows.Forms.TreeViewEventArgs e)
- private void callEditorCreateBoxButton_Click(object sender, System.EventArgs e)
- private void callEditorNewCallButton_Click(object sender, System.EventArgs e)
- private void callEdtitorPublishCallButton_Click(object sender, System.EventArgs e)
- private void callEditorSaveCallButton_Click(object sender, System.EventArgs e)

- private void refreshCallEditorTabPage(Adiuware.Data.CCCase c)
- private void refreshCallEditorTreeView()

Preprocessing.cs

- public static System.Collections.ArrayList searchInSecondaryList(String such-String)

SearchResultDialog.cs

- private void fillTreeView(ArrayList resultList)
- private void searchButton_Click(object sender, System.EventArgs e)
- private void showButton_Click(object sender, System.EventArgs e)

10 Klassentests Adiuware.Web

10.1 Klasse Register User

Autor: Laith Raed

Mittels der RegisterUser Klasse soll ein neuer Account angelegt werden. Dieser Account besteht aus:

Vornamen
Namen
Telefonnr. Privat
Faxnr.
E-Mail Adresse
Passwort

Zusätzlich (optional) kann eine Geschäftsaccount angelegt werden, der ein entsprechendes Objekt mit folgenden Einträgen besteht

Telefonnr. geschäftl.
Land
Name des Geschäftes
ZIP Nr.
Strasse
Stadt

10.1.1 Teststrategie

10.1.1.1 Idee:

10.1.1.2 Es werden Daten in die entsprechenden Felder die Daten eingetragen. Dazu werden die Methoden aufgerufen, die von der Daten- bzw. Logikgruppe angeboten werden. Wurden die diese Methoden ausgeführt, wird entsprechend in der Datenbank überprüft ob die Daten aus der Eingabemaske in die Datenbank übernommen wurden. Die Methode die von der GUI dazu aufgerufen wird lautet:

10.1.1.3

```
private void btregister_Click(object sender, System.EventArgs e)
```

Diese Methode ist die Schnittstelle zu den Methoden, die die Methoden der Daten- und Logikgruppe zur Verfügung stellt.

10.1.2 Klassentestplan

Eingabe der Daten in die Registrierungsmaske.

10.1.3 Testumgebung

10.1.3.1 Verwendete Klasse UserData

Das UserControl UserData beinhaltet lediglich Properties, die die entsprechenden Daten auslesen und setzen. Als Beispiel:

```
public string FirstName
    {
        get
        {
            return txtFirstName.Text;
        }

        set
        {
            txtFirstName.Text = value;
        }
    }
```

Zudem enthält die Klasse UserData Validierungsmethoden die sicher stellen, dass alle Felder ausgefüllt sind.

10.1.3.2 Verwendete Klasse Customer

Die Customer Klasse enthält alle Attribute und Methoden, die benötigt werden um einen neuen Customer anzulegen.

Der Quellcode entstammt aus der Daten- und Logikgruppe!

10.1.3.3 Verwendete Klasse Company

Die Company Klasse enthält alle Attribute und Methoden, die benötigt werden um eine neue Company anzulegen.

Der Quellcode entstammt aus der Daten- und Logikgruppe!

10.1.4 Operation private void btregister_Click(Object sender, System.EventArgs e)

Die Daten, die in die Regsitrerungsmaske eingegben wurden sollen in die Datenbak geschrieben werden.

10.1.4.1 Äquivalenzklassen in der Testdatenmenge

- Alle Daten werden eingeben *einfachundrichtig*
- Es wird lediglich ein Customer Objekt angelegt *mittelschwierigaberrichtig*
- Es fehlen Daten (innerhalb des Customer Objektes bzw. innerhalb des Company Objektes) *schwieriundfalsch*

10.1.4.2 Test der Äquivalenzklasse einfachundrichtig

Getestete Eingabedaten

- Vorname: Max
- Namen: Mustermann
- Telefonnr. Privat: 00000
- Faxnr.:00001
- E-Mail Adresse: max@mustermann.de
- Passwort: MaxMuster33
- Telefonnr. Geschäftl.: 00002
- Land: Musterland
- Name des Geschäftes: Musterwaren
- ZIP Nr.: 1234
- Strasse: Musterstrasse
- Stadt: Musterstadt

Wenn alle Daten eingegeben wurden und der Eintrag in die Datenbank erfolgt ist, erscheint eine entsprechende Meldung, dass der Account erfolgreich angelegt wurde.

Sollergebnisse

Alle Daten werden in die Datenbank gespeichert und es wird eine entsprechende Meldung ausgegeben.

Istergebnisse

Die Daten werden in die entsprechenden Zellen der Datenbank geschrieben und in der GUI erfolgt eine entsprechende Erfolgsmeldung.

Hinweis: Alle Daten bis auf die Eigenschaft *Country* werden eingetragen!

10.1.4.3 Test der Äquivalenzklasse mittelschwierigaberrichtig

Getestete Eingabedaten

- Vorname: Max1
- Namen: Mustermann1
- Telefonnr. Privat: 00000_a

- Faxnr.:00001_a
- E-Mail Adresse: max@mustermann.de1
- Passwort: MaxMuster34

Wenn alle Daten eingegeben wurden und der Eintrag in die Datenbank erfolgt ist, erscheint eine entsprechende Meldung, dass der Account erfolgreich angelegt wurde.

Sollergebnisse

Alle Daten werden in die Datenbank gespeichert und es wird eine entsprechende Meldung ausgegeben.

Istergebnisse

Die Daten werden in die entsprechenden Zellen der Datenbank geschrieben und in der GUI erfolgt eine entsprechende Erfolgsmeldung.

10.1.4.4 Test der Äquivalenzklasse falsch

Getestete Eingabedaten

- Vorname: Max1
- Namen:
- Telefonnr. Privat: 00000_a
- Faxnr.:00001_a
- E-Mail Adresse: max@mustermann.de1
- Passwort: MaxMuster34

Mindestens ein Datum wird nicht eingegeben.

Sollergebnisse

Keine Daten werden in die Datenbank gespeichert und es wird eine entsprechende Meldung ausgegeben, dass alle Felder ausgefüllt werden müssen.

Istergebnisse

Es werden keine Daten in die Datenbank geschrieben und eine entsprechende Meldung wird ausgegeben, dass alle Felder für einen Customer Account ausgefüllt werden müssen.

10.2 Klasse WebForm1

Autor: David Mittag

Mittels der WebForm1 Klasse hat der Benutzer die Möglichkeit, sich anzumelden, nach dem er schon eine Account angelegt hat. Der Benutzer gibt dabei seine E-Mail und Passwort ein und wird am System angemeldet, falls seine Angaben richtig sind.

10.2.1 Teststrategie

10.2.1.1 Idee:

Es werden E-Mail und Passwort in den entsprechenden Feldern in der Eingabemaske eingegeben und geprüft, ob die Anmeldung erfolgreich ist, falls die Daten in der Datenbank schon vorhanden und richtig sind.

Es werden die beiden Methoden:

- private Customer CheckUser(string email, string password)
- private void btlogin_Click(object sender, System.EventArgs e)

getestet.

10.2.1.2 Klassentestplan

Eingabe der Daten in die LogIn maske.

10.2.2 Testumgebung

10.2.2.1 Operation private Customer CheckUser(string email, string password)

Die Methode CheckUser überprüft, ob der User angemeldet werden darf!Dabei greift sie direkt auf die „CustomerCollection“ in der Datenbank zu um die Daten zu überprüfen.

10.2.2.2 Äquivalenzklassen in der Testdatenmenge

- Alle Daten werden richtig eingeben „email & Password“ *einfachundrichtig*
- Es wird mindestens ein falsches Datum eingegeben „email oder Password“ *einfachundfalsch*
- Es wird lediglich ein Datum eingegeben „email oder Password *mittelschwierigaberfalsch*

10.2.2.3 Test der Äquivalenzklasse einfachundrichtig

Getestete Eingabedaten

- E- mail: max@mustermann.de
- Passwort: MaxMuster33
-
- Wenn die E- Mail und das Passwort richtig sind, dann gelingt der User in die internen Bereich und eine Begrüßungsmeldung mit Vornamen und Namen des Users wird dann erschienen.
-
- Sollergebnisse

Alle Daten werden in die Datenbank gespeichert und es wird eine entsprechende Meldung ausgegeben.

Istergebnisse

Die Daten werden in die entsprechenden Zellen der Datenbank geschrieben und in der GUI erfolgt eine entsprechende Erfolgsmeldung.

Hinweis: Alle Daten bis auf die Eigenschaft *Country* werden eingetragen!

10.2.2.4 Test der Äquivalenzklasse mittelschwierigaberrichtig

Getestete Eingabedaten

- Vorname: Max1
- Namen: Mustermann1
- Telefonnr. Privat: 00000_a
- Faxnr.:00001_a
- E-Mail Adresse: max@mustermann.de1
- Passwort: MaxMuster34

Wenn alle Daten eingegeben wurden und der Eintrag in die Datenbank erfolgt ist, erscheint eine entsprechende Meldung, dass der Account erfolgreich angelegt wurde.

Sollergebnisse

Alle Daten werden in die Datenbank gespeichert und es wird eine entsprechende Meldung ausgegeben.

Istergebnisse

Die Daten werden in die entsprechenden Zellen der Datenbank geschrieben und in der GUI erfolgt eine entsprechende Erfolgsmeldung.

10.2.2.5 Test der Äquivalenzklasse falsch

Getestete Eingabedaten

- Vorname: Max1
- Namen:
- Telefonnr. Privat: 00000_a
- Faxnr.:00001_a
- E-Mail Adresse: max@mustermann.de1
- Passwort: MaxMuster34

Mindestens ein Datum wird nicht eingegeben.

Sollergebnisse

Keine Daten werden in die Datenbank gespeichert und es wird eine entsprechende Meldung ausgegeben, dass alle Felder ausgefüllt werden müssen.

Istergebnisse

Es werden keine Daten in die Datenbank geschrieben und eine entsprechende Meldung wird ausgegeben, dass alle Felder für einen Customer Account ausgefüllt werden müssen.

10.3 Klasse EditUser

Autor: Laith Raed

Mittels der EditUser Klasse soll ein bestehender Account geändert werden. Dieser Account besteht aus:

- Vornamen
- Namen
- Telefonnr. Privat
- Faxnr.
- E-Mail Adresse
- Passwort

Zusätzlich (optional) kann auch der Geschäftsaccount geändert werden, der aus einem entsprechendes Objekt mit folgenden Einträgen besteht

- Telefonnr. Geschäftl.
- Land
- Name des Geschäftes
- ZIP Nr.
- Strasse
- Stadt

10.3.1 Teststrategie

10.3.1.1 Idee:

Es werden die bereits vorhandenen Nutzerdaten in die entsprechenden Felder geladen. Dazu werden die Methoden aufgerufen, die von der Daten- bzw. Logikgruppe angeboten werden. Wurden die diese Methoden ausgeführt, wird entsprechend in der Datenbank überprüft ob die geänderten Daten aus der Eingabemaske in die Datenbank übernommen wurden. Die Daten werden dabei in die entsprechenden Eingabfelder gefüllt. Diese können geändert werden, wobei diese Änderungen abgespeichert werden. Die Methode die von der GUI dazu aufgerufen wird lautet:

- private void btUpdate_Click(object sender, System.EventArgs e)

Diese Methode ist die Schnittstelle zu den Methoden, die die Methoden der Daten- und Logikgruppe zur Verfügung stellt.

10.3.1.2 Klassentestplan

Neueingabe der Daten in die Registrierungsmaske.

10.3.2 Testumgebung

10.3.2.1 Verwendete Klasse UserData

Das UserControl UserData beinhaltet lediglich Properties, die die entsprechenden Daten auslesen und setzen. Als Beispiel:

```
public string FirstName
        {
                get
                {
                        return txtFirstName.Text;
                }

                set
                {
                        txtFirstName.Text = value;
                }
        }
```

- Zudem enthält die Klasse UserData Validierungsmethoden die sicher stellen, dass alle Felder ausgefüllt sind.

10.3.2.2 Verwendete Klasse Customer

- Die Customer Klasse enthält alle Attribute und Methoden, die benötigt werden um einen neuen Customer anzulegen.
- Der Quellcode entstammt aus der Daten- und Logikgruppe!

10.3.2.3 Verwendete Klasse Company

- Die Company Klasse enthält alle Attribute und Methoden, die benötigt werden um eine neue Company anzulegen.
- Der Quellcode entstammt aus der Daten- und Logikgruppe!

10.3.3 Operation private void btUpdate_Click(object sender, System.EventArgs e)

Die Daten, die in die Registrierungsmaske eingegeben wurden sollen in die Datenbank geschrieben werden.

10.3.3.1 Äquivalenzklassen in der Testdatenmenge

- Alle Daten werden eingeben *einfachundrichtig*
- Es wird lediglich ein Customer Objekt angelegt *mittelschwierigaberrichtig*

- Es fehlen Daten (innerhalb des Customer Objektes bzw. innerhalb des Company Objektes) *schwieriundfalsch*

10.3.3.2 **Test der Äquivalenzklasse** einfachundrichtig

Getestete Eingabedaten

- Vorname: Maxi
- Namen: Mustermann1
- Telefonnr. Privat: 000001
- Faxnr.:000012
- E-Mail Adresse: max@mustermann.com
- Passwort: MaxMuster333
- Telefonnr. Geschäftl.: 00003
- Land: Musterland
- Name des Geschäftes: Musterwarenfachgeschäft
- ZIP Nr.: 1234567
- Strasse: Mustersträsschen
- Stadt: Musterstädtchen

Wenn alle Daten eingegeben wurden und der Eintrag in die Datenbank erfolgt ist, erscheint eine entsprechende Meldung, dass der Account erfolgreich geändert wurde.

Sollergebnisse

Alle Daten werden in die Datenbank gespeichert und es wird eine entsprechende Meldung ausgegeben das die Daten erfolgreich geändert wurden.

Istergebnisse

Die Daten werden in die entsprechenden Zellen der Datenbank geschrieben und in der GUI erfolgt eine entsprechende Erfolgsmeldung, das die Daten erfolgreich geändert wurden.

Hinweis: Alle Daten bis auf die Eigenschaft *Country* werden eingetragen!

10.3.3.3 Test der Äquivalenzklasse mittelschwierigaberrichtig

Getestete Eingabedaten

- Vorname: Maximilian
- Namen: Mustermann1000
- Telefonnr. Privat: 00000_aaaa
- Faxnr.:00001_aa
- E-Mail Adresse: max@mustermann.com
- Passwort: MaxMuster34567

Wenn alle Daten eingegeben wurden und der Eintrag in die Datenbank erfolgt ist, erscheint eine entsprechende Meldung, dass der Account erfolgreich angelegt wurde.

Sollergebnisse

Alle Daten werden in die Datenbank gespeichert und es wird eine entsprechende Meldung ausgegeben.

Istergebnisse

Die Daten werden in die entsprechenden Zellen der Datenbank geschrieben und in der GUI erfolgt eine entsprechende Erfolgsmeldung.

10.3.3.4 **Test der Äquivalenzklasse** falsch

Getestete Eingabedaten

- Vorname: Max1
- Namen:
- Telefonnr. Privat: 00000_a
- Faxnr.:00001_a
- E-Mail Adresse: max@mustermann.de1
- Passwort: MaxMuster34

Mindestens ein Datum wird nicht eingegeben.

Sollergebnisse

Keine Daten werden in die Datenbank gespeichert und es wird eine entsprechende Meldung ausgegeben, dass alle Felder ausgefüllt werden müssen.

Istergebnisse

Es werden keine Daten in die Datenbank geschrieben und eine entsprechende Meldung wird ausgegeben, dass alle Felder für einen Customer Account ausgefüllt werden müssen.

10.4 Klasse Search

Autor: Laith Raed

In der Searchklasse wird der Produkt Baum aufgebaut. Dies geschieht mittels Methodenaufruf und Zugriff auf die Produktwurzel die sich in der Datenbank befindet.

10.4.1 Teststrategie

10.4.1.1 Idee:

Es wird überprüft ob die Daten aus der Datenbank im Produktbaum in der GUI angezeigt werden.

- private void BuildTree()

Diese Methode ist die Schnittstelle zu der Datenbank und der Produktwurzel.

10.4.1.2 Klassentestplan

Mittels rekursivem Aufbau wird überprüft, ob die Daten aus der Datenbank vom Baum angezeigt werden

10.4.2 Testumgebung

10.4.2.1 Verwendete Klasse Microsoft.Web.UI.WebControls.TreeView

Die Klasse enthält die entsprechenden Methoden, die dafür sorgen dass der Baum aufgebaut wird. Da die Klasse direkt von Microsoft zur Verfügung gestellt wurde, kann auf entsprechende Testdokumentationen der Entwickler verwiesen werden.

10.4.2.2 Verwendete Klasse ModelTreeNode

Die Klasse ModelTreeNode wird von der Logikgruppe zur Verfügung gestellt und von Seiten der GUI verwendet, um die entsprechenden Objekte verwalten zu können.

10.4.2.3 Verwendete Klasse Model

Die Klasse Model wird von der Logikgruppe zur Verfügung gestellt und von Seiten der GUI verwendet, um die entsprechenden Objekte verwalten zu können.

10.4.3 Operation private void BuildTree()

Die Methode BuildTree() baut rekursiv den Produktbaum auf.

10.4.3.1 Äquivalenzklassen in der Testdatenmenge

- Es werden die Daten aus der Datenbank gelesen: *richtig*
- falls keine Daten vorhanden sind (z.B. durch fehlende Anbindung zur Datenbank): *falsch*

10.4.3.2 Test der Äquivalenzklasse richtig

Rekursiver Durchlauf des Objektes modelroot der Klasse ModelTreeNode.

Sollergebnisse

Alle Produktknoten aus der Datenbank werden im Baum abgezeigt.

Istergebnisse

Die Knoten werden richtig angezeigt

10.4.3.3 **Test der Äquivalenzklasse** falsch

Wenn kein Objekt vom Typ ModelTreeNode vorhanden ist (z.B. fehlende Anbindung zur Datenbank) wird ein leerer Baum angezeigt. Es darf keine

Exception geworfen werden. Diese Aktion wird innerhalb der von Microsoft verfassten Klasse Web.Controls verwaltet. Es wird auf entsprechende Testdokumentationen der Entwickler verwiesen.

Sollergebnisse

Leerer Baum (es wird nichts angezeigt)

Istergebnisse

Leerer Baum (es wird nichts angezeigt)

10.4.4 Operation private void btsubmit_Click(object sender, System.EventArgs e)

Mittels btsubmit_Click und dem entsprechend selektierten Knoten im Baum wird ein Model aus der Datenbank geladen.

10.4.4.1 Äquivalenzklassen in der Testdatenmenge

- Es wird ein Knoten mit Model geladen: *richtig*
- Es wird versucht einen Knoten zu laden, der kein Model hat: *falsch*

10.4.4.2 Test der Äquivalenzklasse **richtig**

- Es wird ein Knoten mit einem Objekt der Klasse *model* geladen.
-

Sollergebnisse

Die entsprechenden Fragen des Objektes vom Typ *model* werden in das QuestionControl geladen. Die entsprechenden Lösungen des Objektes vom Typ *model* werden in das SolutionControl geladen

Istergebnisse

Im QuestionControl erscheinen die zu dem Model zugehörigen Fragen und im SolutionControl die möglichen Lösungen.

10.4.4.3 Test der Äquivalenzklasse **falsch**

Wenn ein Knoten gewählt wird, der kein Objekt vom Typ *model* enthält, erscheint eine entsprechende Fehlermeldung.

Sollergebnisse

Es wird ein Label mit dem Hinweis „Selected node has no model.Please choose another one to start the dialog“ angezeigt.

Istergebnisse

Die entsprechende Fehlermeldung wird angezeigt

10.5 Klasse QuestionContainerControl & SelectionContainer-Control

Autor: David Mittag

In diesen beiden (zusammengehörenden) Klassen werden die zugehörigen Fragen und ihre Antwortmöglichkeiten angezeigt. Im SelectionControl befindet sich die Auswahlmöglichkeiten und das Event welches die gegebene Antwort an die Logik übergibt.

10.5.1 Teststrategie

10.5.1.1 Idee:

Es wird überprüft, ob die Fragen und Antwortmöglichkeiten in den beiden Controls angezeigt wird. Ist die Anzeige erfolgreich, so soll geprüft werden ob Antworten gegeben werden können und darauf folgende Fragen erneut angezeigt werden.

- public void CreateQuestionControls()
- public void CreateSolutionControls()
- private void submit_Click(object sender, EventArgs e)

10.5.1.2 Klassentestplan

Es wird geprüft, ob Fragen und Antwortmöglichkeiten angezeigt werden. Die Fragen werden entsprechend beantwortet.

10.5.2 Testumgebung

10.5.2.1 **Verwendete Klasse** QuestionControl

Die Fragen werden als LinkButtons angezeigt und mit einem entsprechendem Event ausgestattet. Die Fragen werden nacheinander im QuestionControl aufgebaut welches an das QuestionContainerControl übergeben wird.

10.5.2.2 Verwendete Klasse SelectionControl

Im SelectionControl werden die entsprechenden Antwortmöglichkeit einer Frage aufgelistet. Das SelectionControl wird sichtbar, wenn auf eine Frage geklickt wurde. Die Frage kann mittels eines Submit Buttons beantwortet werden.

10.5.2.3 Verwendete Klasse Model

Die Klasse Model wird von der Logikgruppe zur Verfügung gestellt und von Seiten der GUI verwendet, um die entsprechenden Objekte verwalten zu können.

10.5.3 Operation public void CreateQuestionControls()

Die Methode `CreateQuestionControls()`lädt die Fragen in das QuestionContainerControl.

10.5.3.1 Äquivalenzklassen in der Testdatenmenge

- Es werden Fragen in das QuestionContainerControl geladen: *richtig*
- Es werden keine Fragen in das QuestionContainerControl geladen: *falsch*
-

10.5.3.2 Test der Äquivalenzklasse richtig

Die Fragen werden angezeigt.

Sollergebnisse

Das Control zeigt alle Fragen an.

Istergebnisse

Alle Fragen werden angezeigt.

10.5.3.3 Test der Äquivalenzklasse falsch

Falls keine Fragen vorhanden sind wird das QuestionContainerControl ausgeblendet.

Sollergebnisse

Ausblendung des Controls

Istergebnisse

Control wird ausgeblendet.

10.5.4 Operation public void CreateSolutionControls()

Analog zu Operation public void CreateQuestionControls()

Test wurde erfolgreich durchgeführt.

10.5.5 Operation private void submit_Click(object sender, EventArgs e)

Die Methode `submit_Click ()`übergibt der Logik die gegebene Antwort.

10.5.5.1 Äquivalenzklassen in der Testdatenmenge

Eine Antwort wurde gegeben und neue Fragen werden geladen: *richtig*

Es wurde keine Antwort gegeben und Submit geklickt: *falsch*

10.5.5.2 Test der Äquivalenzklasse richtig

Die neuen Fragen werden im QuestionContainerControl angezeigt.

Sollergebnisse

Das Control zeigt alle neuen Fragen an.

Istergebnisse

Alle neuen Fragen werden angezeigt.

10.5.6 Test der Äquivalenzklasse falsch

Falls keine Antwort gegeben wurde wird eine entsprechende Meldung gegeben. Der Zustand des QuestionContainerControls und des SelectionControls bleibt unverändert

Sollergebnisse
Beibehlatung des Zustandes. Die ursprünglichen Fragen und Antwortmöglichkeiten bleiben erhalten. Es erscheint eine Fehlermeldung „Please select an answer“

Istergebnisse
Der Zustand bleibt erhalten. Eine entsprechende Fehlermeldung wird angezeigt.

10.6 Klasse SolutionContainerControl

Autor: David Mittag

Das SolutionContainerControl verhält sich analog zum QuestionContainerControl. Wenn ein Objekt vom Typ *model* angewählt wurde, werden gleichzeitig zu den Fragen Lösungen in das SolutionContainerControl geladen.
Die Tests verlaufen analog zum QuestionContainerControl und wurden erfolgreich durchgeführt.

Beide Controls sind vom Aufbau und der Verwendung identisch.

10.7 Klasse SolutionContainerControl (Suchtyp: Eingabe eines Strings)

Autor: David Mittag & Laith Raed

Es wird ein Eingabefeld für den Benutzer bereitgestellt, in dem dieser einen String eingeben kann. Dieser String wird an die Logik übermittelt. Es wird das gleiche SolutionContainerControl benutzt. Durch die entsprechenden Methoden Aufrufe werden die Lösungen in das Control geladen.

Die Tests wurden erfolgreich durchgeführt!

Teil III
Ist- Soll Vergleich; Der Ablauf der PG; Erfahrungsberichte; Fazit

11 Ist- Soll Vergleich

In diesem Kapitel soll kritisch betrachtet werden, welche der im Pflichtenheft festgelegten Ziele erreicht wurden und welche nicht.
Die grundsätzliche Teilung in primäre Dokumente, die durch die Beantwortung von Fragen gefunden werden können, und sekundäre Dokumente, die noch nicht aufbereitete Protokolle laufender oder abgeschlossener Fälle darstellen, wurde wie beschrieben umgesetzt. Dies trifft auch auf die grundsätzliche Organisation der Modelle in Fragen, Antworten und Lösungen zu. Ebenso stehen die Funktionen zur Bearbeitung der Modelle (Hinzufügen, Bearbeiten und Löschen von Fragen, Antworten und Lösungen) und Möglichkeiten zur Protokollierung von einzelnen Fällen wie geplant zur Verfügung.
Die Oberflächen erfüllen die festgelegten Grundanforderungen, einige Zusatzfunktionalitäten wurden jedoch nicht mehr implementiert. Zu nennen wären hier erweiterte Suchfunktionen (am häufigsten gesuchte Begriffe anzeigen, „meinten Sie…“ zum Auffinden ähnlicher Suchbegriffe, Suche nur über bestimmte Informationen), die Möglichkeit der Bewertung einer ausprobierten Lösung durch den Kunden, der „Contact Me“-Button (löst eine Bitte um Rückruf bei den zuständigen Agenten aus) sowie Druck- und Speicherfunktionen für vom Endkunden durchgeführte Anfragen.
Insgesamt wurden demnach alle wesentlichen Funktionen des Systems implementiert. Bei den fehlenden Komponenten handelt es sich vielmehr um zusätzliche Komfortfunktionen, deren Implementierung von Anfang an als Sekundärziel betrachtet wurde. Die Qualität der Ergebnisse der eingesetzten Logik-Komponenten wird in der Dokumentation der Anwendungslogik diskutiert.

12 Ablauf der Projektgruppe

12.1 Erstes Semester

Die Projektgruppe 446 "Entwurf und Implementierung eines wissensbasierten Decision Support Systems zur personalisierten Unterstützung von Anfragen im Kunden-Service" begann für uns Teilnehmer damit, dass am 09. Juli 2003 die Teilnehmerliste im Foyer des GB V ausgehangen wurde.
Der offizielle Starttermin war dann der 30. Juli 2003, an dem wir uns zum ersten Mal mit den Betreuern trafen.
Bei diesem Treffen wurden nach einer kurzen Vorstellung untereinander organisatorische Fragen wie das Stattfinden der Seminarfahrt nach Haus Nordhelle und die Themenvergabe für dieses Seminar geklärt.
Bis zu der Seminarfahrt, die vom 16.-17. Oktober 2003 statt fand, bestand unsere Hauptaufgabe in der Vorbereitung des Vortrags.
Die erste PG-Sitzung wurde für den 13. Oktober 2003 anberaumt.
Mit ihr begann dann die Arbeit der PG, die erst einmal darin bestand, gewisse Ämter wie folgt zu vergeben:

Schrankbeauftragter:	David Mittag
Hardwarebeauftragter:	Alexander Ulbrich
Kassenwart:	Ute Kersting

In den ersten Sitzungen einigten wir uns auch auf das Logo und den Slogan:

Des Weiteren wurde nach und nach die Hardware im PG-Raum aufgestockt und entsprechende Software installiert, bis wir eine lauffähige Entwicklungsumgebung hatten.
Wir erstellten eine Skillmatrix, in der jeder Teilnehmer seine bisherigen Fähigkeiten und gewünschten Vertiefungsthemen angeben konnte, um damit eine spätere Kleingruppenaufteilung zu vereinfachen.
Und natürlich begannen wir nach dem Seminar mit dem Entwurf unseres Programms.
Dazu betrachteten wir einige bestehende Desicion Support Systeme und identifizierten deren positive und negative Aspekte. Anschließend begannen wir mit dem Brainstorming für die Anforderungen an unser Programm. Nach der Konkretisierung der Anforderungen mit Hilfe von Userstories stand dann Anfang Januar 2004 die Erstellung des Pflichtenhefts an. Im Januar 2004 fand ein weiteres Seminar zur theoretischen Vertiefung des Themas statt.

Anfang Februar 2004 konnten wir dann die Arbeit in vier Kleingruppen aufnehmen und mit der Modellierung beginnen.
Die Kleingruppen setzen sich folgendermaßen zusammen:

Datenbank-Gruppe
- Suna Kaya
- Martin Saternus
- Yuan Zhang

Logik-Gruppe
- Frank Gietmann
- Daniel Müller
- Nils Müller
- Richard Süselbeck
- Alexander Ulbrich

Web-Gruppe
- David Mittag
- Laith Read

Windows-Gruppe
- Ute Kersting
- Christian Topnik

Bis zum Ende des Semesters waren das Datenmodell der Logikgruppe und die grobe Struktur der Oberflächen fertig.

12.2 Zweites Semester

Im zweiten Semester folgten nun die Erstellung des Zwischenberichts, der die Ausarbeitungen der Vorträge im Rahmen des Vertiefungsseminars im Januar 2004 beinhaltet, und die Umsetzung unseres Programms.
Nachdem die Datenbank-Gruppe ihren Teil implementiert hatte, erstellte sie Testdaten, um den anderen Gruppen die Möglichkeit zu geben, ihren Quellcode zu testen.
Dabei stellte sich heraus, dass die Logik-Gruppe einige Punkte in ihrem Datenmodel ändern musste, was zu weit reichenden Änderungen an der Datenbank führte.
Während die Logik und die Datenbankgruppe sich um diese Probleme kümmerten, erstellte die Web-Gruppe den Rumpf der Internet-Seiten und stattete sie mit Session-Verwaltungsfunktionalität aus. Die Windows-Gruppe erweiterte die bis dato bestehende grobe Struktur ihrer Anwendung um die nötigen Eingabemasken.
Anschließen begannen die Web- und die Windows-Gruppe mit dem Erstellen des Handbuchs, sowie der Dokumentation.
Nachdem die Probleme mit der Datenbank gelöst waren, implementierten die Web- und die Windows-Gruppe die Funktionalität.
Die Datenbank- und die Logik-Gruppe vervollständigten ihre Dokumentation und begannen mit dem Testen.
Im Juni 2004 erstellten wir die Gliederung des Endberichts und teilten die Abschnitte, die nicht zwangsläufig einer Gruppe zugeordnet sind, unter den PG-Teilnehmern auf.
Ab Mitte Juli 2004 nahm unser Programm Form an, die Web- und die Windows-Gruppe vervollständigten jeweils ihre Funktionalitäten und starteten in die Testphase.
David als Zwischen- und Endbericht-Beauftragter stellte den Endbericht fertig und wir präsentierten unser fertiges Programm am 12. August 2004 im Rahmen des Diplomanten und Doktoranten Seminars des Lehrstuhls I.

13 Erfahrungsberichte der Einzelgruppen

13.1 Adiuware.Logic

Dieser Bericht stellt aus persönlicher und subjektiver Sichtweise die Erlebnisse und Eindrücke dar, die sich während der einjährigen Zusammenarbeit der zwölf Auserwählten Mitglieder der PG 446 angesammelt haben. Wenn man also mal ein Jahr zurückgeht und sich überlegt wie die ganze Sache angefangen hat, so fällt einem als erstes ein, wie ambitioniert die ganze Gruppe an dieses Projekt rangegangen ist. Nicht zuletzt wegen des inoffiziellen Projektleiters, der es sich zur Aufgabe gemacht hatte alle Softwarepakete einer großen Softwarefirma einzusetzen, deren Gründer ebenso heißt wie das Pferd von Sam aus dem Herrn der Ringe in der ursprünglichen Sprache. Natürlich braucht an dieser Stelle nicht erwähnt zu werden, dass der inoffizielle Projektleiter bei selbiger Firma angestellt ist, weshalb das an dieser Stelle auch nicht geschieht. Jedoch die Idee an sich war, wie im Übrigen alle meinten, keine schlechte, aber ließ die Realisierung mangels Einsatzwillen seitens des Lehrstuhlbeauftragten für Hardwareangelegenheiten zu wünschen übrig. Somit vergingen Monate bis ein System existierte auf dem man theoretisch arbeiten konnte. Es sei hier angemerkt das das Attribut theoretisch nicht zufällig gewählt ist. Praktisch gesehen vergingen nämlich noch weitere Monate bis man selbiges benutzen konnte. Die Gründe dafür liegen im Verborgenen, jedoch gibt es Dunkle Mächte die Behaupten es hätte einzig und allein an jenem Softwarehersteller gelegen. Dazu erinnern wir uns noch mal an den Namensvetter des Pferdes und verweigern uns jedes weiteren Kommentars. Nachdem nun die anfänglichen Probleme beseitigt waren und man sich um die eigentlich essentiellen Dinge, wie Logo, Weihnachtsfeier und Kaffeekochen kümmern konnte, ging die Arbeit zügig voran. Das heißt nach zwei Wochen intensivem Arbeiten hatte die Datenbankgruppe wie versprochen die DB fertig. Wenn man mal von den wenigen, dreistelligen, Änderungen die im Weiteren noch gemacht werden mussten absieht, ging dieser Teil recht zügig. Schließlich wurden die letzten Änderungen an dem grundlegendsten Projektteil schon zwei Tage vor der Präsentation gemacht. Im Vergleich zur Logikgruppe, die erst 24 Std. vor Präsentationsbeginn aufgehört hat zu coden, ist das schon ziemlich bemerkenswert, zumal die Logiker auch nur deshalb aufgehört haben weil Ihnen wieder erwarten die Ideen ausgingen. Interessanter Weise war diese Ideenlosigkeit nicht Gegenstand von Beschwerden der anderen Kleingruppen, was sich vielleicht mit der Art und Weise, sowie den Auswirkungen jener Ideen erklären könnte. Etwaige Auswirkungen auf andere Teilbereiche wurden mittels modernster Kommunikationsformen, wie Sprache, übermittelt. Aufgrund des Einsatzes dieser modernen Methode kam es hin und wieder zu kleineren Missverständnissen, was sich jedoch nicht ausschließlich mit der Mannigfaltigkeit der Sprachtalente innerhalb der PG erklären lässt. An diesem Punkt sollte noch mal ein lob an alle Teilnehmer ausgesprochen werden, die es nebenbei geschafft haben ihr Diplom in chinesischer Übersetzung zu machen.

Was bleibt also übrig, nachdem man nun einen Eindruck von all den nicht ganz ernst gemeinten Erlebnissen bekommen hat? Zu aller erst die Erkenntnis, dass es tatsächlich möglich ein Jahr lang mit den unterschiedlichsten Charakteren ohne einen größeren Streit zusammenzuarbeiten. Man mag sich wundern oder nicht, es bleibt eine Tatsache, dass am Ende dieser PG alle Teilnehmer ein einigermaßen gutes Verhältnis zueinander haben. Kombiniert mit dem Erfolg in der Arbeit ist dies etwas von dem ich persönlich denke, dass wir darauf stolz sein können.

13.2 Adiuware.Web

Die Entwicklung einer Internetseite empfand ich persönlich als eine spannende Herausforderung, deren ich mich eigentlich schon länger annehmen wollte. Durch die PG sollte sich mir diese Gelegenheit auch einmal bieten. Ich hatte eine gewisse Vorfreude auf die PG, war aber auch froh diese endlich anpacken zu können um sie hinter mich zu bringen.
Dadurch dass sich die PG im 1. Semester hauptsächlich auf die Theorie stürzte und zuerst versuchte Konzepte zu entwickeln, bekam ich das Gefühl, dass der Ablauf klar strukturiert war und der Erfolg der PG gar kein Problem darstellen sollte. Jeder in der Gruppe machte einen kompetenten Eindruck auf mich. Mir persönlich graute es etwas vor der Programmierarbeit, da ich in diesem Bereich kaum Übung besitze. Aber ich wartete erst mal ab. Dann irgendwann wurden die Gruppen eingeteilt und ich bekam die Webseite mit Allaithy zugesprochen. Im ersten Moment dachte ich: gut, dann habe ich die wenigste Programmierarbeit. Also stürzten wir uns auf die Arbeit (ohne Bücher zu lesen) frei nach dem Prinzip Learning-by-doing und dachten lange auch wir wären auf den richtigen Weg. Ich verliess mich darauf, dass Allaithy C# programmieren kann und er verliess sich darauf, dass ich den Aufbau von mit .Net programmierten Seiten verstehe. Ich war dabei derjenige, der das Prinzip Learning-by-doing bevorzugte. Dies und die Annahme von uns beiden, uns blind auf den anderen zu verlassen war der grosse Fehler. Nachdem wir also dachten, nur noch die Schnittstellen zusammenführen zu müssen, erfuhren wir von einem Mitglied der PG, dass unsere Architektur „ungeeignet“ sei. Das war der Punkt an dem die PG für uns „heiss“ wurde, denn diesmal sollte es richtig laufen und wir verspürten so etwas wie Druck seitens der PG. Erkenntnis dieses kleinen Dramas: zwischen unserer einstigen Vorstellung einer Webseite und denen der .Net Entwickler liegen Welten.
Wir haben also damit begonnen, sehr viel Zeit für die PG zu verwenden und konnten uns dann auch irgendwann, nach viel Fleiss, Kampf und Opfern (verschobene Prüfung) freischwimmen. Da diese Zeit doch sehr anstrengend war, strebten wir also auch ziemlich zügig dem Ende entgegen, welches also das Zusammenführen der Schnittstellen bedeutete. Auch hier sollten sich erstmals Probleme eines Projektes im grösseren Rahmen zeigen, da auch andere Teilgruppen ihre Probleme hatten.

Meine Rolle in der PG habe ich mir anders vorgestellt: zum einen wollte ich natürlich solche Situationen vermeiden, wie die Neugestaltung unserer Seite und zum Anderen habe ich mir im Vorfeld der PG mehr Einfluss auf das Projekt erhofft. Ich habe versucht, mich an allen wesentlichen Dingen, die innerhalb der PG stattgefunden, haben zu beteiligen. Auch hier wurde ich enttäuscht. Da sich irgendwann der harte Kern der PG gebildet hatte, musste ich einsehen doch irgendwie aussen vor zustehen. Mein Fehler war, dies nicht einmal angesprochen zu haben und so fühlte ich mich immer mehr in eine Statistenrolle hineingedrängt worden zu sein. Dies sollte sich u.a. so äussern, dass wenn Nachfragen meinerseits gestellt worden sind, ich teilweise mit unfreundlichen Reaktionen als Antwortet bekam. Dies hatte für mich Auswirkungen auf das Klima der PG. Und das war meiner Meinung nach nicht besonders gut, da sich sämtliche Aktivitäten lediglich auf PG interne Dinge beschränkten.
So musste es dann auch letztendlich zu den (logischen) Spannungen zwischen einigen Mitgliedern kommen, in denen sich die Aufnahme von Kritik und deren Akzeptanz als schwierig herausstellen sollte. Da ich mit Kritik an meiner Person ganz gut leben kann, ist es für mich tlw. völlig unverständlich warum andere dies nicht können!

Auch der Ton, welcher in den Sitzungen einige Male verwendet wurde, war nicht immer der Feinste. Dies wurde sogar angesprochen, stiess aber kaum auf Resonanz.
Als Ergebnis möchte ich aber dennoch betonen, dass viele fähige Leute in der PG waren, die zu Recht einige Male gereizt gewesen sind und das Vorankommen des Projektes sichergestellt haben.
Ich für mich musste einsehen, dass das laxe Herangehen an diese Arbeit ein grosser Fehler war, welcher mich in einigen Sachen doch enorm zurückgeworfen hat. Die Arbeit an sich habe ich klassisch unterschätzt. Auch musste ich mir oft viele Lücken eingestehen, die es zu schliessen gilt.
Mit der Zusammenarbeit innerhalb meiner Gruppe (Allaithy) bin ich dennoch zufrieden, da wir uns auch ausserhalb der PG sehr gut verstehen und uns durch diese Sache gemeinsam geschlagen haben!!
Wir glauben beide unterm Strich, dass wir in dem letzten Jahr eine PG erlebt haben, wie sie in ihrer Form wahrscheinlich doch sehr häufig, zumindest hier an der Uni Dortmund vorkommt. Also bleibt uns als kleiner Trost: wir haben hier ein völlig „normale“ PG hinter uns gebracht!

14 PG 446 – Fazit

Nach einem Jahr, zahlreichen Gruppensitzungen und mehreren Tausend Zeilen Quellcode ist unsere Projektgruppe und die Entwicklung des Programms „Adiuware“ abgeschlossen.
In diesem Kapitel soll nun noch ein abschließender Blick auf den Projektverlauf geworfen werden und eine kurze Betrachtung der Erweiterungs- und Verbesserungsmöglichkeiten der Software erfolgen.

14.1 Die Gruppe

Unsere Gruppe bestand aus zwölf Personen mit sehr unterschiedlichen Fähigkeiten und Interessen. Dennoch gelang es uns, Kleingruppen zu bilden, in denen jeder die Möglichkeit bekam, seine Stärken einzubringen und neue Fähigkeiten zu entwickeln.
Die Integration aller Teilnehmer in den Arbeitsprozess war nicht immer problemlos möglich. Dennoch verlief die Zeit der Projektgruppe ohne größere Auseinandersetzungen und nennenswerte Konflikte.
Die positive Bilanz: alle Teilnehmer haben das Projekt erfolgreich zu Ende geführt.
Für Einblicke in persönliche Erfahrungen einzelner Teammitglieder sei hier auf das Kapitel → **13. *Erfahrungsberichte der Einzelgruppen*** verwiesen.

14.2 Das Projekt

Die Projektplanung kann in Grundzügen als gelungen bezeichnet werden. Der eigentlichen Softwareentwicklung ging vernünftiger Weise eine intensive Beschäftigung mit unterschiedlichen, für die Durchführung des Projektes wichtigen Themengebieten wie z.B. den mathematischen Grundlagen der Programmlogik voraus. Und auch die Architektur der zur Entwicklung nötigen Hard- und Softwareumgebung wurde eingehend diskutiert.
Erste Fehler schlichen sich jedoch bereits bei der Anforderungsdefinition der Software ein. Das größte Problem, welches jedoch für Projektgruppen keineswegs untypisch ist, war die Tatsache, dass wir am Anfang bei weitem zu ambitioniert waren und den Anspruch hatten, eine Software zu entwickeln, die im Bereich von Call Centern tatsächlich zum Einsatz kommen könnte. Und so wurde recht viel Zeit auf die Besprechung von Programmfunktionen verwandt, die man bestenfalls mit einer weiteren Projektgruppe hätte umsetzen können.
Offensichtlich war es uns aber möglich, diese (und weitere) Probleme zu lösen oder zu kompensieren, so dass wir pünktlich zur Endpräsentation unser Programm „Adiuware“ in einer funktionsfähigen Version vorstellen konnten.
Die Fähigkeiten der Software wären mit entsprechend besserer Projektplanung und –durchführung sicherlich ausbaufähig gewesen, doch muss man natürlich zu bedenken geben, dass sich unsere Gruppe zu einem großen Teil aus eher unerfahrenen „Softwareentwicklern“ zusammensetzte.
Von diesem Standpunkt kann das Ergebnis, welches wir nach einem Jahr Arbeit vorstellen konnten, sicherlich als gelungen bezeichnet werden.
Eine detailliertere Gegenüberstellung der von uns erbrachten Ergebnisse mit den ursprünglichen Anforderungen findet sich im Kapitel → **11. *Ist- Soll Vergleich*.**

14.3 Ausblick

Wie bereits aus der vorangegangenen Betrachtung des Ablaufes der Projektgruppe deutlich wird, ist „Adiuware“ nicht fehlerfrei.
Zum einen ergeben sich bei der eigentlichen Kernfunktionalität, nämlich der Auswahl möglichst passender Fragen und Lösungen, in einigen Fällen gewisse Qualitätsprobleme, die wir nicht abschließend lösen konnten.
Zum anderen müsste das Programm um zahlreiche Funktionen erweitert werden, um es für den realen Einsatz innerhalb eines Call Centers zu befähigen. Dazu gehört z.B. die Zuteilung aller Systemnutzer zu gewissen „Fachgruppen“, denen Anfragen bestimmter Art automatisch zugeführt werden, um eine spezialisierte Abarbeitung möglichst vieler Fälle zu ermöglichen (denkbar sind noch weitere Verbesserungen zur Integration in tatsächliche Workflowprozesse innerhalb eines Support Centers). Durch die Berücksichtigung von Nutzerpräferenzen (z.B. mittels Nutzenwerten) wäre zudem eine weitere Verbesserung der gelieferten Ergebnisse zu erwarten.

Offensichtlich wäre eine Weiterentwicklung der Software also möglich und sinnvoll. Insbesondere die Prüfung und Verbesserung der Programmlogik zur optimalen Auswahl von Fragen und Lösungen könnte hierbei auch ein interessantes Thema für Diplomarbeiten darstellen, ebenso wie Erweiterungen hinsichtlich des Strukturlernens, die die Praxistauglichkeit von „Adiuware“ stark verbessern würden.

Teil IV
Handbücher; Installationsanleitung

15 Handbuch Windows-Anwendung

15.1 Einleitung

15.1.1 Allgemeines

Herzlich Willkommen und Gratulation zum Erwerb des Decision Support Systems Adiuware. Unsere Software arbeitet auf der Basis von Bayes'schen Netzen, um Ihren Kunden den größtmöglichen Komfort bei der Beantwortung von Supportanfragen zu bieten. Im Gegensatz zu vielen bestehenden Systemen werden die Fragen kontextsensitiv und individuell für den Benutzer berechnet, um die Usability und damit die Akzeptanz bei den Anwendern zu steigern. Ihnen ist doch sicher das Abarbeiten von Expertensystemen, die immer wieder ähnliche Fragen stellen und sehr lange benötigen, um zu einer Lösung zu kommen, auch schon langweilig geworden, und Sie haben dann doch die Hotline angerufen. Unser Ziel ist es, für Ihre Kunden und Sie die Anzahl der Telefonsupportanfragen zu reduzieren, um damit für beide Seiten die Kosten zu minimieren.

15.1.2 History

Adiuware wurde von der PG 446 des Lehrstuhl Automaten- und Schaltwerktheorie, Computational Intelligence
des Fachbereichs Informatik der Universität Dortmund in der Zeit vom 1.Oktober 2003 bis zum 31.Juli 2004 entwickelt. Die Basissoftware bietet die Möglichkeit des Einsatzes als Callcenter Software, aber auch als Webfrontend mit reduzierter Funktionalität für Kunden.

15.1.3 Technologien

Wie bereits erwähnt, nutzt die Adiuware Software Bayes'sche Netze zur Auswahl der Fragen, die dem Nutzer als nächstes gestellt werden. Wir möchten hier einen kleinen Einblick in diese Technologie geben, da dies beim Verständnis einiger Bedienelemente nützlich sein kann.

Um zu entscheiden, welche Fragen die Lösungen am meisten zu Gunsten einer schnellen Lösung beeinflussen, ist eine Möglichkeit mit Wahrscheinlichkeiten zu arbeiten. Diese Wahrscheinlichkeiten sind jedoch meist sehr unhandlich, denn sie benötigen viel Speicherplatz und sind schlecht zu visualisieren.

Bayes'sche Netze sind Modelle die Wahrscheinlichkeitstheorie und Graphentheorie miteinander kombinieren und es dadurch erlauben, Wahrscheinlichkeiten effizient abzuspeichern und zu verwalten, sowie die Zusammenhänge zwischen Antworten und Fragen einfach graphisch darzustellen.

Sind also die Wahrscheinlichkeiten, mit denen Fragen Antworten beeinflussen bekannt, so kann daraus eine sinnvolle und möglichst kurze Abfolge von Fragen berechnet werden.

Bei tiefer gehendem Interesse an Bayes'schen Netzen empfehlen wir die nachfolgende Literatur:

- S. Russell, P. Norvig, Artificial Intelligence - A Modern Approach, Prentice Hall, Upper Saddle River, 1995

- F. V. Jensen, Bayesian Networks and Decision Graphs, Springer, 2001

Wir bitten Sie, das Handbuch vor dem ersten Gebrauch sorgfältig zu lesen.

15.2 Case Explorer

15.2.1 Allgemeines

Der Case Explorer bietet dem Call Center Agent dieselbe Ansicht, die den Kunden Im Internet zur Verfügung gestellt wird. Hier kann er, wenn er telefonischen Support gibt, die Fragen durchgehen und sie dem Kunden stellen.

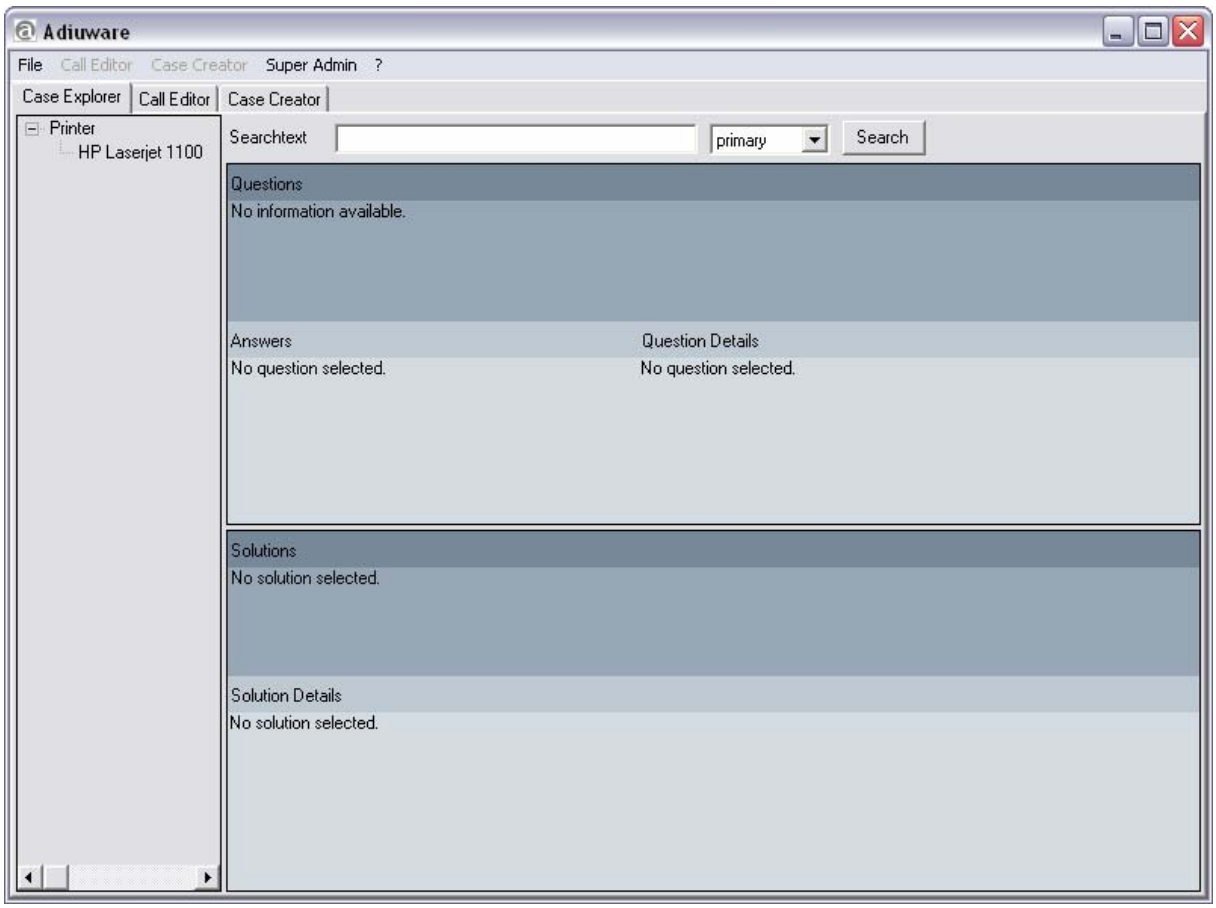


Abbildung 15.2-1 Case Creator

15.2.2 Bedienung

Dem Benutzer stehen zu Beginn grundsätzlich die Aktionen "Eingrenzung der Lösungen mittels Suche" vgl. 15.2.2.1 oder "Eingrenzung der Lösungen mittels Navigationsbaum" vgl. 15.2.2.2 zur Verfügung.

15.2.2.1 Eingrenzung der Lösungen mittels Suche

Um das Auffinden einer passenden Lösung zu vereinfachen, kann der Nutzer Begriffe, die mit seinem Problem assoziiert sind, in das Suchfeld eingeben. Mehrere Begriffe werden dabei standardmäßig mit einem logischen "Oder" verknüpft, es ist aber auch eine explizite Angabe eines "Und"-Operators möglich ("printer AND error").

Abbildung 15.2-2 Suchmaske

In der neben dem Suchfeld zu findenden Combobox kann nun noch zwischen einem primären und einem sekundären Suchvorgang gewählt werden. Ein Betätigen der "Search"-Schaltfläche startet schließlich die Suche.

Wurden keine Suchtreffer erzielt, wird dies in einem Informationsfenster mitgeteilt; andernfalls öffnet sich ein Suchdialog, welcher alle Suchergebnisse in geordneter Form darstellt: alle gefundenen Solutions sind ihrem jeweils entsprechenden Knoten zugeordnet. Um seine Suche weiter einzugrenzen, kann der Nutzer weitere Suchanfragen in das Textfeld am unteren Rand des Dialoges eingeben, welches sich wie das entsprechende Textfeld des Case Explorers verhält.

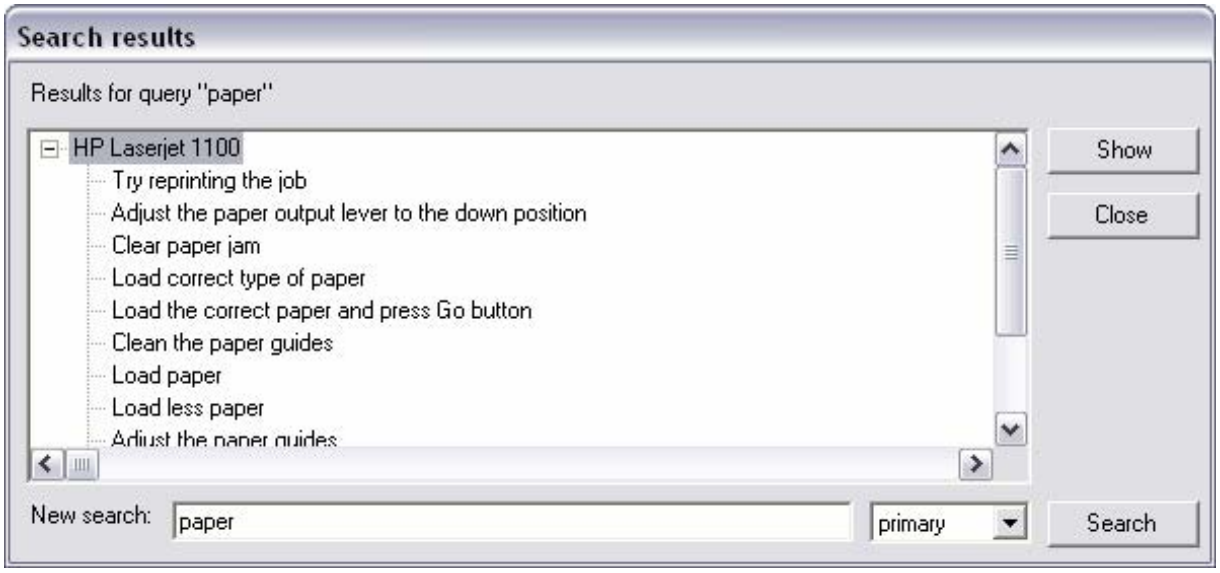


Abbildung 15.2-3 Anzeige der Ergebnisse der primären Suche

Wurde eine viel versprechend aussehende Solution gefunden, kann diese ausgewählt und mittels der "Show"-Schaltfläche in einer Detailansicht angezeigt werden.

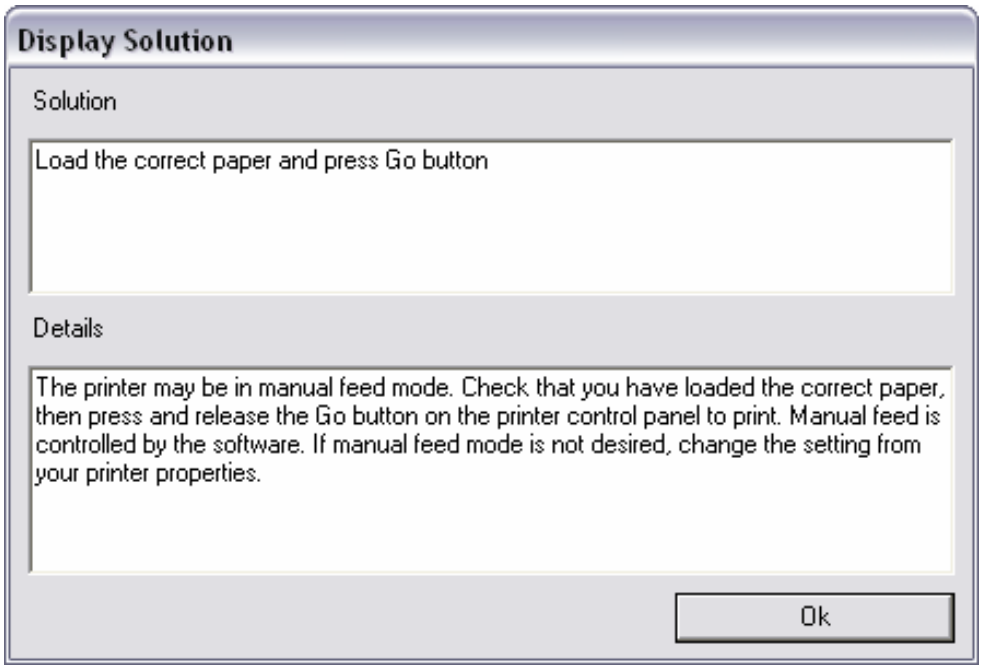


Abbildung 15.2-4 Anzeige einer Solution aus der Ansicht der primären Suche

Ist keine der gefundenen Lösungen zutreffend, obwohl einer der angezeigten Knoten eine dem Problem möglicherweise zuzuordnende Beschreibung enthält (z.B. Produkttyp), kann dieser Knoten ebenfalls per Mausklick angewählt werden. Ein Betätigen der Schaltfläche "Show" bewirkt in diesem Fall ein Schließen des Dialoges und eine automatische Einordnung in den Navigationsbaum an der Stelle des soeben gewählten Knotens.

15.2.2.2 Eingrenzung der Lösungen mittels Navigationsbaum

In dem Suchbaum kann der Benutzer sein Problem einordnen, indem er sich durch den Baum klickt, bis der Knoten keine Unterknoten mehr hat.

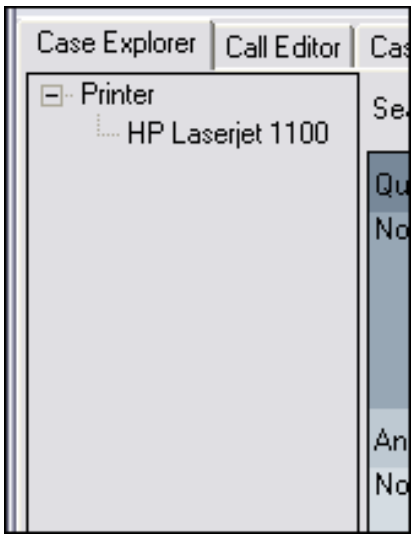


Abbildung 15.2-5 Baum zur Einordnug

Erreicht er einen solchen Knoten werden in dem rechten Teil des Fensters Fragen angezeigt, und er kann sein Problem nun mittels Beantwortung von Fragen vgl. 15.2.2.3 weiter eingrenzen.

15.2.2.3 Eingrenzung der Lösungen mittels Beantworten der Fragen

Der Benutzer hat sich im Suchbaum eingeordnet. Sobald er einen Knoten anklickt, der keine Unterknoten mehr hat, erscheinen auf der rechten Seite Fragen.

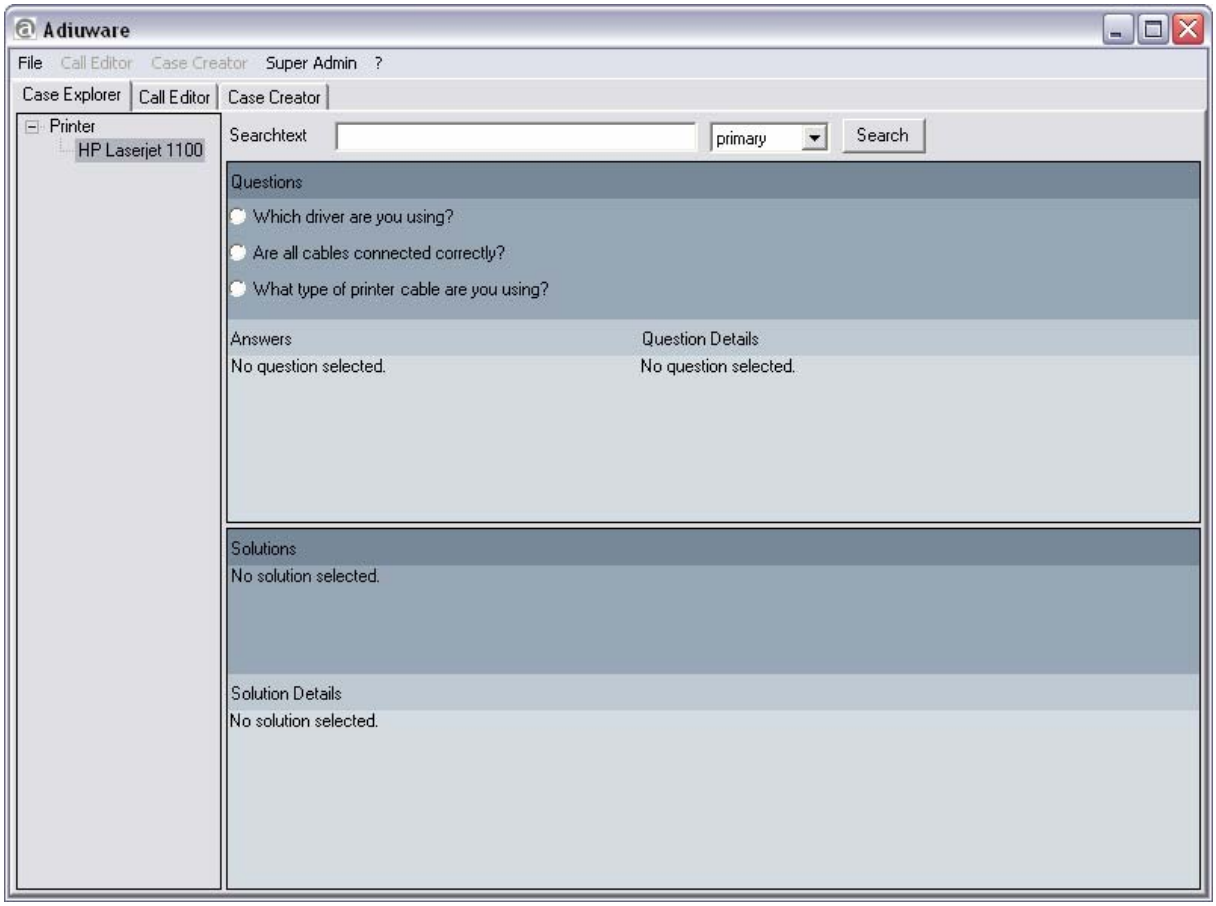


Abbildung 15.2-6 Anzeige der Fragen

Aus diesen Fragen kann der Benutzer nun eine auswählen, von der er meint sie beantworten zu können. Dazu klickt er mit dem Mauszeiger in den weißen Kreis links neben der Frage.

Sogleich erscheinen eine Detailbeschreibung der Frage und die Antwortmöglichkeiten.

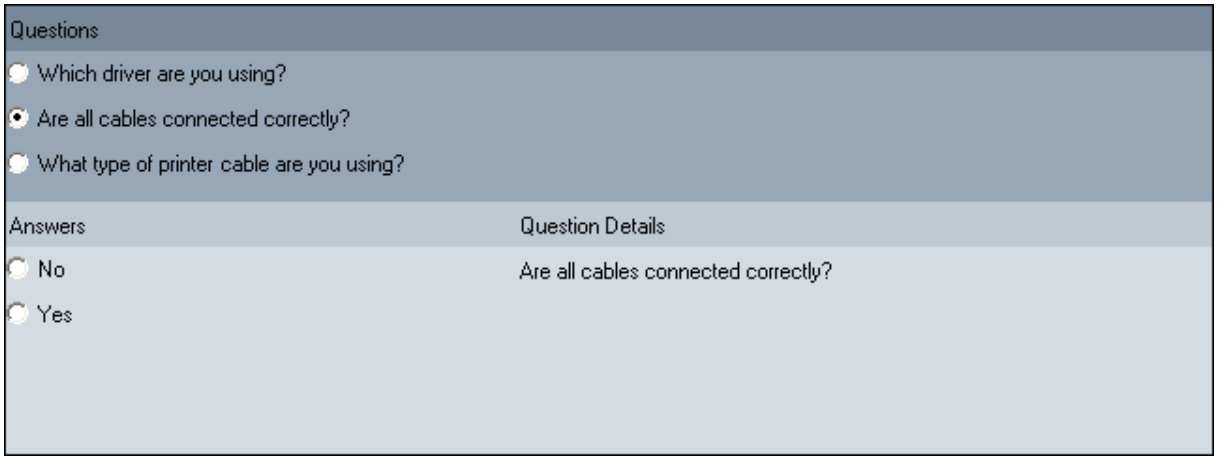


Abbildung 15.2-7 Anzeige Fragen und Antworten

Ist der Benutzer nun unsicher, ob er nicht doch eine andere Frage hätte wählen sollen, kann er dies tun und die angezeigten Detailbeschreibung der Frage und die Antwortmöglichkeiten werden angepasst.

Hat sich der Benutzer für eine Frage entschieden, so beantwortet er sie, indem er mit dem Mauszeiger in den weißen Kreis links neben der Antwort klickt.

Sobald der Benutzer geantwortet hat, wird in dem Antwortteil die soeben beantwortete Frage mit den Antworten angezeigt, die jedoch nicht mehr anklickbar sind. Auch die Detailbeschreibung bleibt erhalten.

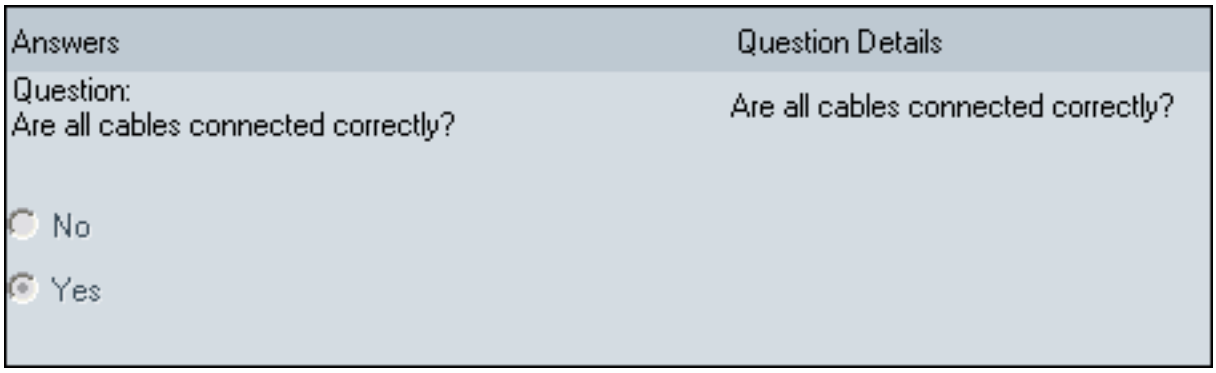


Abbildung 15.2-8 Antwort ausgewählt

In den Bereichen für Fragen und Lösungen erscheinen nun neue Einträge.

Zunächst kann der Benutzer Lösungen auswählen und es erscheint eine Detailbeschreibung, mit deren Hilfe der Benutzer nun versuchen kann, sein Problem zu lösen.

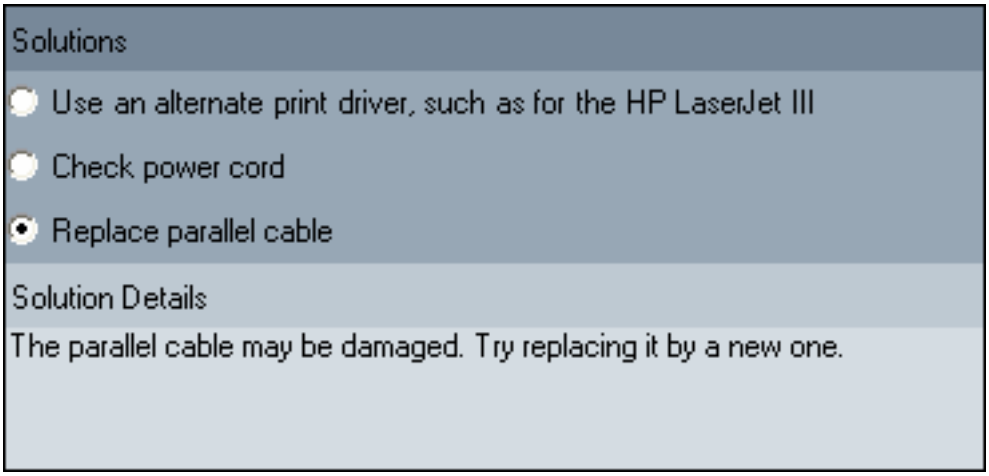


Abbildung 15.2-9 Ansicht der Lösung und Anzeige der Details der ausgewählten Lösung

Führt keine der vorgeschlagenen Lösungen zu einem Erfolg, muss der Benutzer in dem Bereich für Fragen, eine weitere auswählen und diese beantworten.

Dieser Vorgang wird wiederholt, bis eine zufrieden stellende Lösung gefunden wurde, oder die Vermutung nahe liegt, dass für das Problem keine Lösung zu finden ist.

15.3 Case Creator

Im Case Creator werden neue Fälle in das Decision Support System eingepflegt. Dabei ist zu beachten, dass während der Eingabe zunächst nur die Daten gespeichert werden, die Verarbeitung dieser Daten jedoch erst nach Beendigung aller Eingaben geschieht. Dies bedeutet auch, dass in der Zeit von der ersten Eingabe bis zum Abschluss der Berechnung das System für den Support nicht Einsatzbereit ist.

15.3.1 Allgemeines

Um neue Fälle einzupflegen, muss man einen groben Überblick über die Struktur des Adiuware Systems haben.

Grundsätzlich gibt es den Suchbaum, in den alle Knoten eingebettet sind.
Die wichtigsten Knoten sind die Blätter, also die, die keine Unterknoten mehr haben, denn alle anderen Knoten dienen nur der Strukturierung.
Unter den Blättern verbergen sich die eigentlichen wichtigen Daten, die in einem so genannten Model gekapselt sind.
In diesem Model befinden sich die Fragen mit ihren jeweiligen Antworten, die Lösungen, sowie Details der jeweiligen Beziehungen zwischen den Fragen, Antworten und Lösungen.
Wichtig ist weiterhin, dass nur Blattknoten ein solches Model besitzen dürfen.
Besitzt ein Blattknoten ein Model, so wird dies durch das Zeichen "(M)" hinter dem Namen gekennzeichnet.

15.3.2 Bedienung

Bei der Bedienung ist zu beachten, dass nur die Menüpunkte und Schaltflächen aktiv sind, die zu dem jeweiligen Zeitpunkt eingesetzt werden können.

Der Case Creator gliedert sich in zwei Teilbereiche. Im oberen Teil, der Model Section, kann der Benutzer die Baumstruktur editieren und neue Verbindungen zwischen Lösungen, Fragen und Antworten erstellen (vgl. 15.3.2.1), während der untere Teil, die Submodel Section, dem Hinzufügen der Fragen, Antworten und Lösungen dient (vgl. 15.3.2.2) .

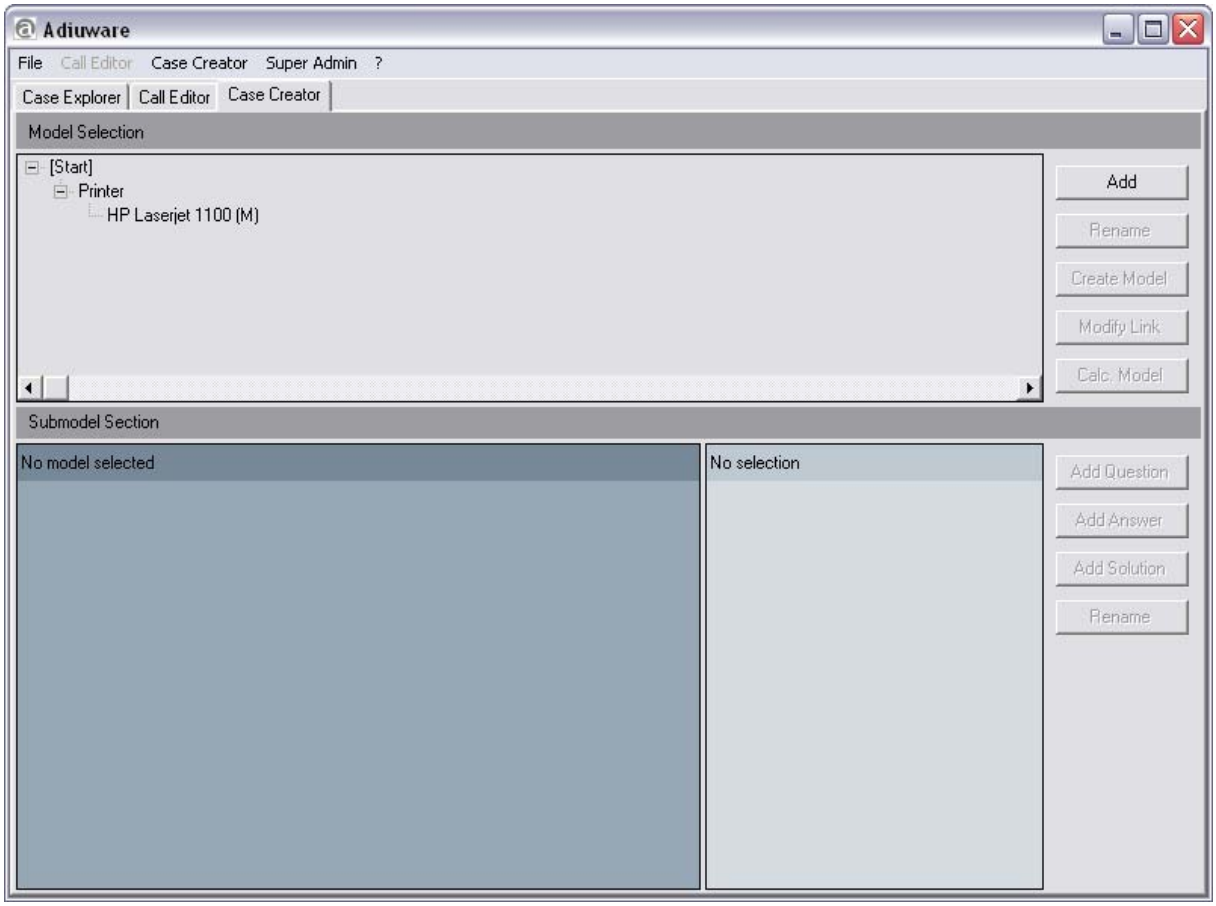


Abbildung 15.3-1 Case Creator Ansicht

15.3.2.1 Model Section

Einen Knoten hinzufügen

Einen Knoten kann man auf verschiedene Arten hinzufügen: zunächst einmal offensichtlich mit Hilfe des "Add"-Knopfs auf der rechten Seite. Eine weitere Möglichkeit findet sich im Hauptmenü unter dem Punkt "Case Creator".

Hier muss der Menüpunkt "Add Model" ausgewählt werden.

Der eigentliche Ablauf des Hinzufügens ist jedoch in beiden Fällen gleich.

Es öffnet dich ein Fenster, in dem der Name des neuen Knoten eingegeben werden kann. Drückt der Benutzer den "OK"- Knopf, so wird der neue Kno-

ten gespeichert, drückt er den "Cancel"-Knopf, so wird der Vorgang abgebrochen.

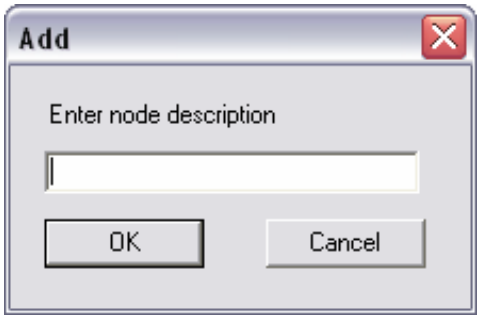


Abbildung 15.3-2 "Add node"-Dialog

Einen Knoten umbenennen

Auch für das Umbenennen gibt es die Möglichkeiten, entweder direkt den Knopf "Rename" zu drücken, oder im Hauptmenü unter dem Punkt "Case Creator" den Punkt "Rename Model" auswählen.

Sobald der Benutzer eine dieser Aktionen durchgeführt hat, öffnet sich ein Fenster. Zunächst befindet sich in der Textbox der aktuelle Name des Knotens, den der Benutzer beliebig ändern darf. Drückt der Benutzer den "OK"-Knopf, so wird die Änderung übernommen, drückt er den "Cancel"-Knopf, so wird der Vorgang abgebrochen.

Abbildung 15.3-3 "Rename node"-Dialog

Ein neues Model anlegen

Hat der Benutzer einen Knoten ausgewählt, der keine Unterknoten mehr besitzt, kann er ihm ein Model hinzufügen. Wichtig ist zu beachten, dass ein Model nur einem solchen Knoten hinzugefügt werden kann.

Um dem Knoten das Model hinzuzufügen, drückt er entweder auf den "Add Model" Knopf oder er wählt "Add Model" unter dem Hauptmenüpunkt "Case Creator".

Sofort erscheint hinter dem Namen des Knotens das Zeichen dafür, dass der Knoten ein Model hat "(M)".

Beziehungen zwischen Fragen, Antworten und Lösungen editieren

Wie bei den anderen Bedienungselementen, bieten sich dem Benutzer auch hier wieder zwei Möglichkeiten, diese Aktion auszuführen.

Es stehen ihm der direkte Knopf "Modify Link" und der Menüpunkt "Modify Link" unter dem Hauptmenüpunkt "Case Creator" zur Verfügung.

Ruft der Benutzer die Aktion "Modify Link" auf, so öffnet sich folgendes Fenster:

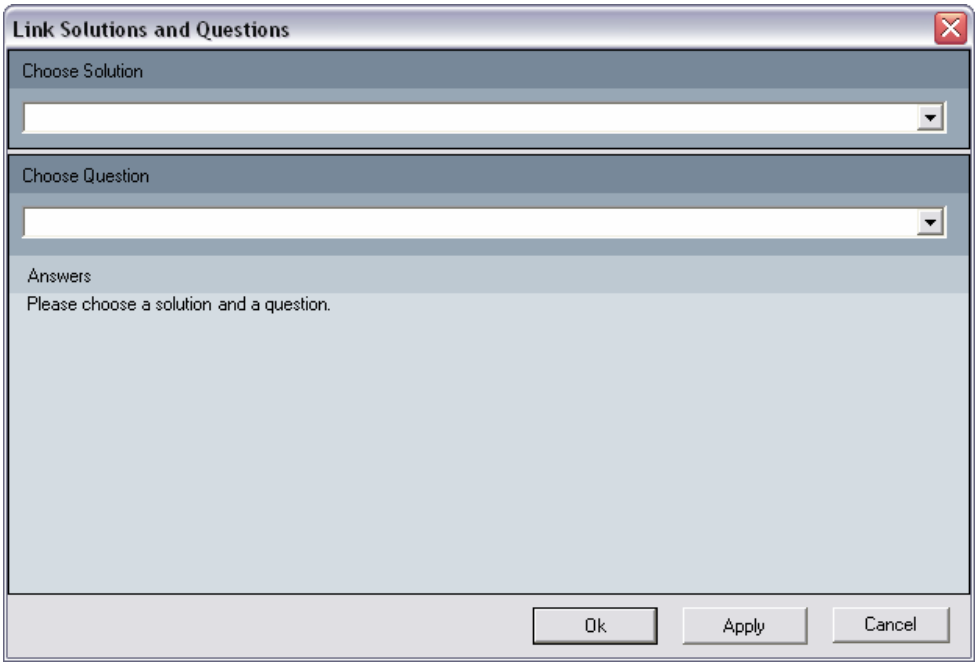


Abbildung 15.3-4 "Modify Link"-Dialog

In den oberen beiden Auswahlboxen, muss der Benutzer nun die Lösung bzw. die Frage auswählen, zwischen denen die Beziehung geändert werden soll.

Ist dies geschehen, werden im unteren Bereich die Antworten angezeigt.

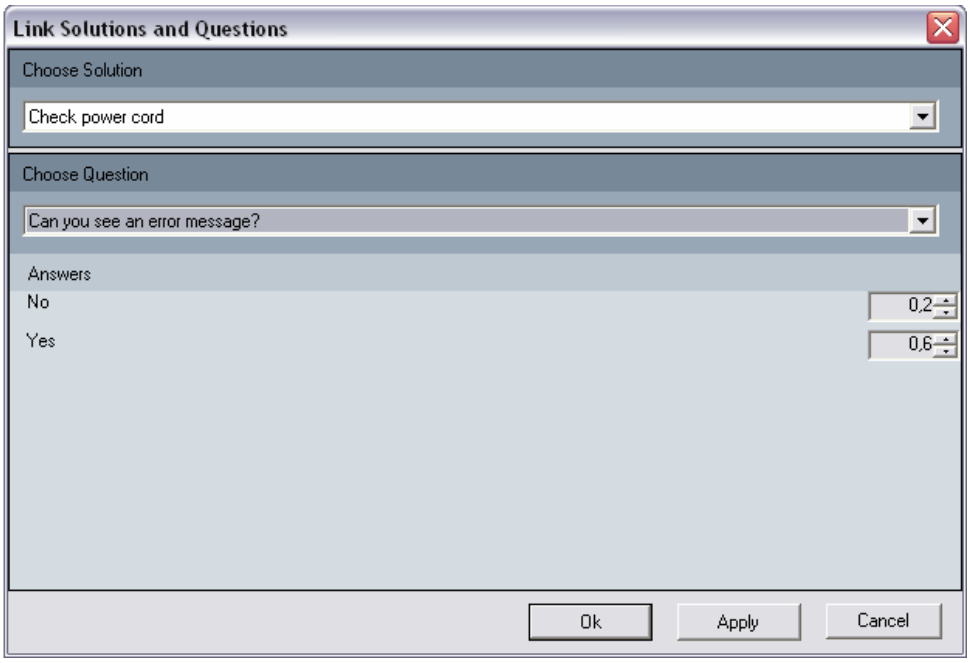


Abbildung 15.3-5 Einstellen der Beeinflussung von Lösungen und Antworten einer Frage

In den Eingabefeldern neben den entsprechenden Antworten kann nun ein neuer Wert eingetragen werden.

Damit die Werte gespeichert werden, muss der Benutzer entweder "Apply"-Knopf oder den "OK"-Knopf drücken.

Drückt der Benutzer nun den "Apply"-Knopf, so wird die Eingabe übernommen, und er kann eine neue Frage oder neue Lösung wählen und auch dort die Beeinflussung ändern.

Drückt er den "OK"-Knopf, so werden die Änderungen gespeichert und das Fenster geschlossen.

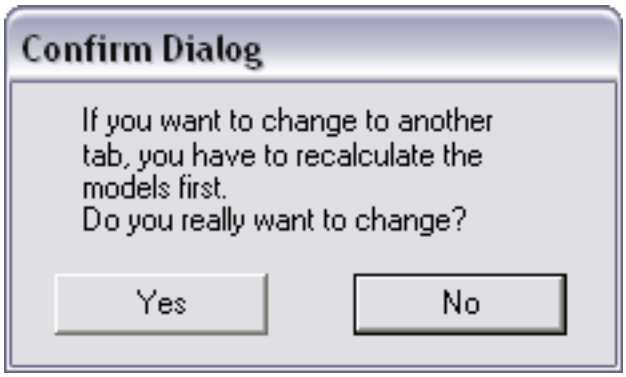


Abbildung 15.3-6 Abfrage Dialog zur Bestätigung der Neuberechnung der Modelle

Drückt er den "Cancel"-Knopf, so wird das Fenster mit vorheriger Sicherheitsabfrage ohne Speichern geschlossen.

Die Modelle neu berechnen

Um die im Case Creator gespeicherten Daten im Model zu berücksichtigen, muss dieses neu berechnet werden. Dies geschieht automatisch, sobald man den Reiter wechselt und die entsprechende Sicherheitsabfrage bestätigt.

Der Benutzer kann diesen Vorgang aber auch von Hand anstoßen, indem er entweder den Knopf "Calc. Model" drückt oder unter dem Punkt "Case Creator" im Hauptmenü den Punkt "Calculate Model" auswählt.

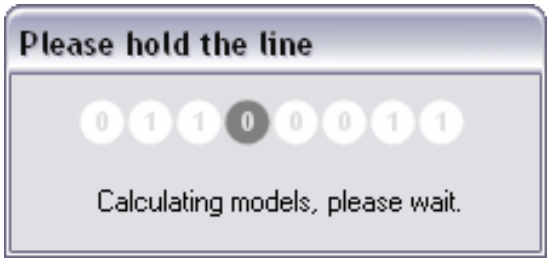


Abbildung 15.3-7 Anzeige, dass die Modelle neu berechnet werden

15.3.2.2 Submodel Section

Eine Frage hinzufügen

Um eine Frage hinzufügen zu können, muss in der "Model Section" ein Knoten mit Model ausgewählt sein.

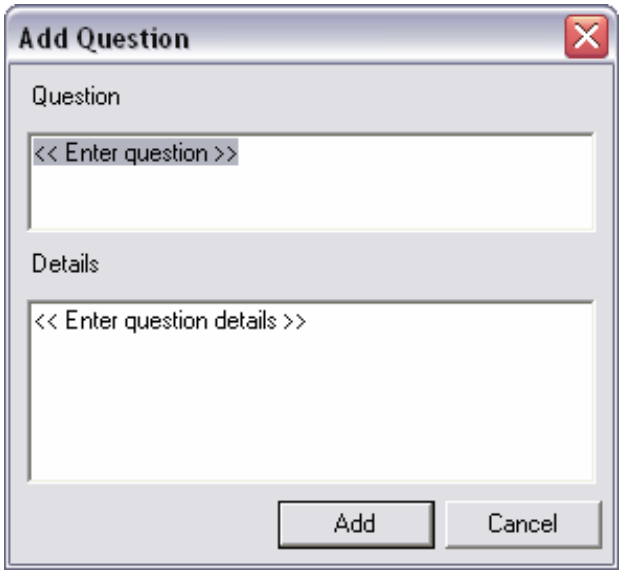


Abbildung 15.3-8 "Add Question"-Dialog

Der Benutzer wählt im Hauptmenü unter dem Punkt Case Creator den Punkt "Add Question" oder drückt den Knopf "Add Question " auf der rechten Seite.

Es öffnet sich ein Fenster, in dem er den Namen und die Details der Frage eintragen kann. Zur Bestätigung drückt er den "OK"-Knopf, zum Abbrechen den "Cancel"-Knopf.

Eine Antwort hinzufügen

Um eine Antwort hinzuzufügen, muss in der "Submodel Section" eine Frage ausgewählt sein.

Der Benutzer kann nun entweder den Knopf "AddAnswer" oder den Menüpunkt "Add Answer" unter dem Hauptmenüpunkt "Case Creator" benutzen.

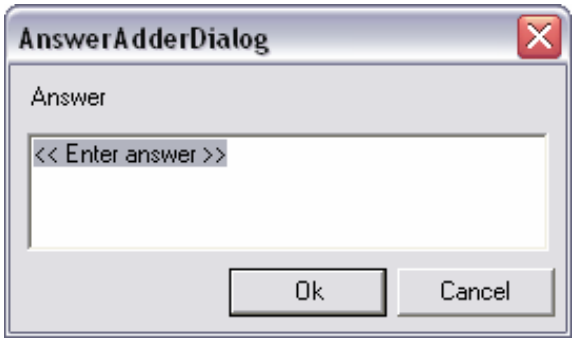


Abbildung 15.3-9 " Add Answer"-Dialog

Es öffnet sich ein Dialog, in den der Benutzer nun die Antwort schreiben kann.

Er bestätigt mit dem "OK"-Knopf, um abzubrechen, drückt er den "Cancel" Knopf.

Eine Lösung hinzufügen

Um eine Lösung hinzufügen zu können, muss in der "Model Section" ein Knoten mit Model ausgewählt sein.

Auch für diese Aktion hat der Benutzer wieder die Möglichkeiten unter dem Hauptmenüpunkt "Case Creator" den Punkt "Add Solution" oder den Knopf "Add Solution" zu wählen.

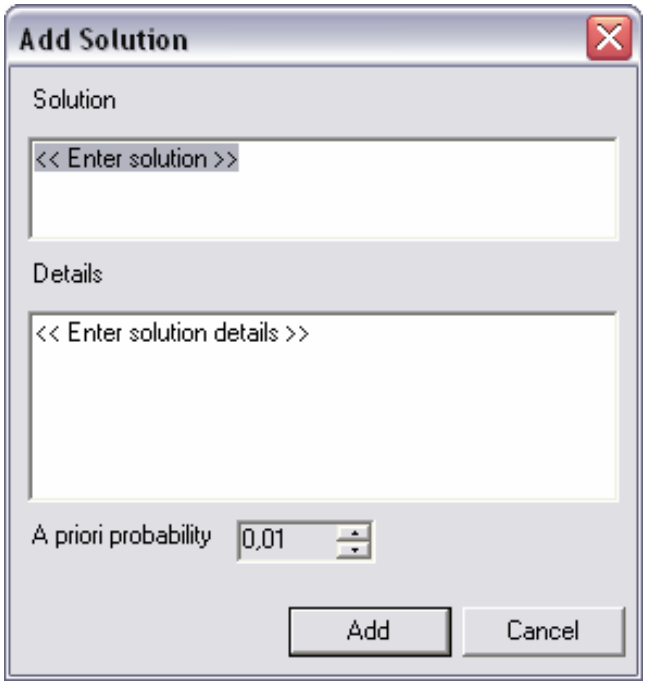


Abbildung 15.3-10 " Add Solution"-Dialog

Hat er eine dieser Möglichkeiten ausgeführt, so öffnet sich ein Fenster, in dem er die Lösung und deren Detailbeschreibung eintragen sowie ihre "A Priori"-Wahrscheinlichkeit angeben kann.

Die Bestätigung erfolgt mit dem "OK"-Knopf, der Abbruch mit dem "Cancel"-Knoppf.

Eine Frage oder Lösung umbenennen

Zunächst muss der Benutzer den Knoten, den er umbenennen möchte, im der "Submodel Section" auswählen. Anschließend drückt er den Knopf "Rename" oder wählt den Punkt "Rename Submodel" unter dem Hauptmenüpunkt "Case Creator ". Es erscheint ein Dialog, in dessen Textbox sich der aktuelle Name des Knotens befindet. Diesen darf der Benutzer beliebig ändern.

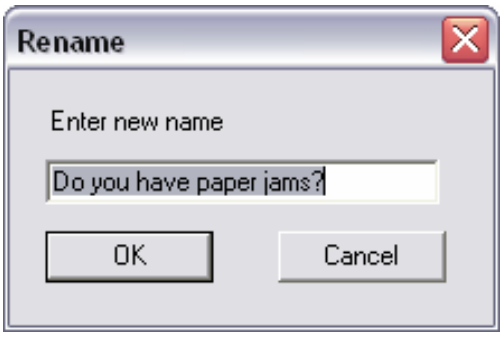


Abbildung 15.3-11 "Rename-Dialog für Fragen oder Lösungen in der Submodel Section

Die Eingabe wird mit dem "OK"-Knopf bestätigt oder mit dem "Cancel"-Knopf abgebrochen.

15.4 Call Editor

15.4.1 Allgemeines

Der Call Editor bietet dem Mitarbeiter eines Callcenters alle nötigen Funktionen, um die Anfragen unterschiedlicher Kunden zu verwalten und für spätere Abfragen zu archivieren.
Zur kategorisierten Ablage behandelter Fälle stehen verschiedene Ablageflächen zur Verfügung.
Diese unterteilen sich zunächst in drei bereiche: die "Global Box" enthält Fälle, die keinem Mitarbeiter und keiner Mitarbeitergruppe zugeordnet sind, in der "Global Group Box" befinden sich Fälle, die der Gruppe, welcher der aktuell eingewählte Mitarbeiter angehört, aber keinem konkreten Mitglied dieser Gruppe zugewiesen sind, und in der "Personal Box" sind schließlich alle Fälle abgelegt, die von dem derzeit eingewählten Mitarbeiter verwaltet werden.
Unterhalb dieser Kategorien kann eine beliebige hierarchische Struktur weiterer Subkategorien erzeugt werden um das schnelle Auffinden bestimmter Anfragen zu erleichtern.

15.4.2 Bedienung

In der Beschreibung der Bedienung wird im Folgenden nur noch von den Knöpfen gesprochen. Für den Benutzer ist es wichtig zu wissen, dass zu jedem dieser Knöpfe auch ein gleichnamiger Menüpunkt unter dem Hauptmenüpunkt "Call Editor" mit derselben Funktionalität zu finden ist.

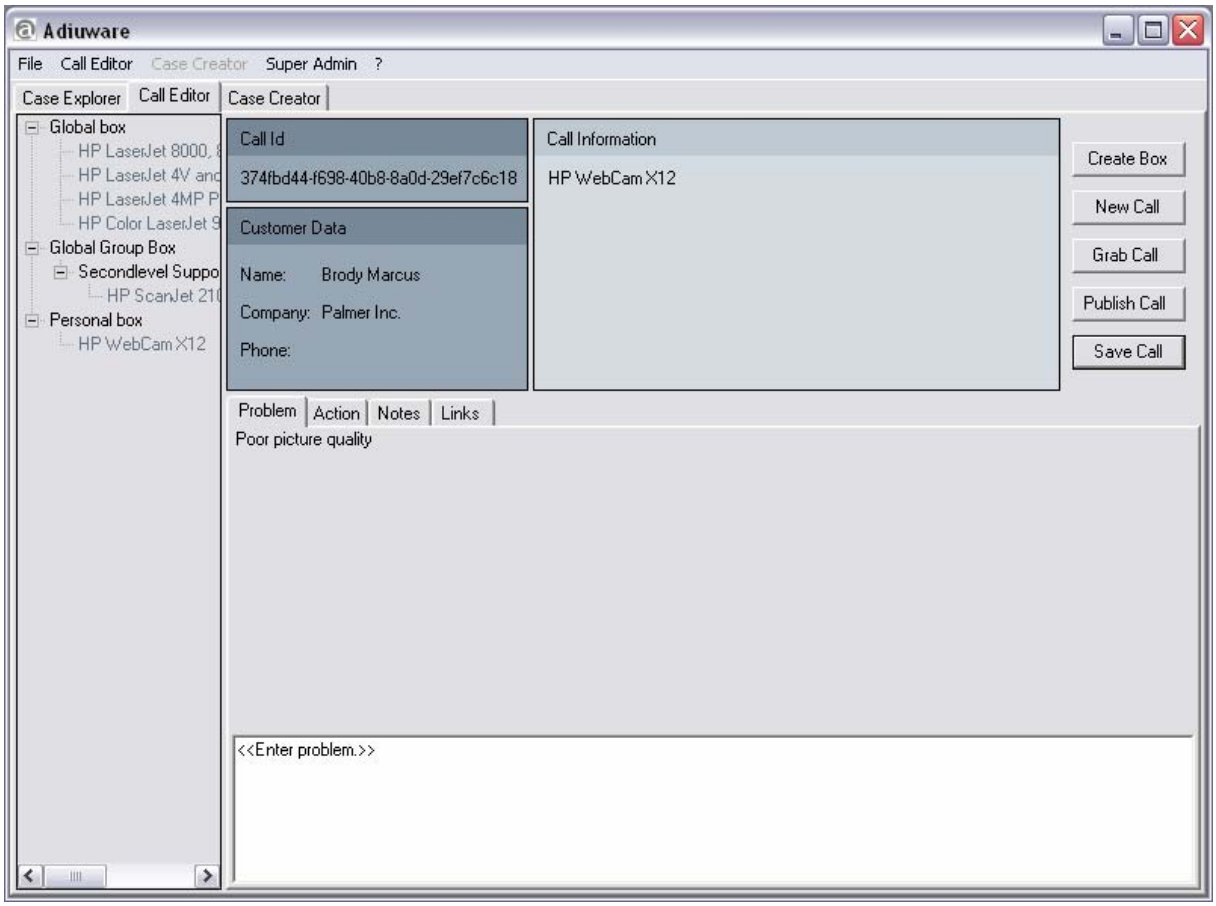


Abbildung 15.4-1 Call Editor Ansicht

Die so genannten "Calls", welche einzelne, von einem Callcenter-Mitarbeiter bearbeitete Fälle darstellen, können in der links befindlichen Baumansicht unterschiedlichen Kategorien, dargestellt durch so genannte "Boxen", zugeordnet werden.

Die Details eines gewählten Calls werden im mittleren Bereich des Fensters angezeigt (um Informationen zu "Actions", "Notes" und "Links" zu erhalten, müssen die entsprechenden Karteireiter angewählt werden). Mittels der am rechten Rand angeordneten Schaltflächen, welche im Folgenden näher beschrieben werden, ist eine Verwaltung der bestehenden bzw. die Erzeugung neuer Calls und Boxen möglich.

15.4.2.1 Eine neue Box hinzufügen

Durch Betätigung der "Create Box"-Schaltfläche lässt sich eine neue Box unterhalb der aktuell gewählten erzeugen. In dem erscheinenden Dialogfeld muss lediglich eine Bezeichnung für die zu erzeugende Box eingetragen und die Eingabe durch Drücken der "Enter"-Taste abgeschlossen werden.

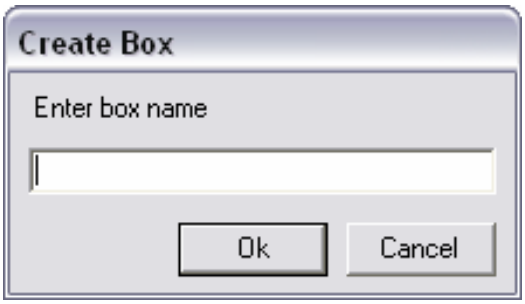


Abbildung 15.4-2 Dialog zum erzeugen einer neuen Box im Call Creator

15.4.2.2 Einen neuen Call eröffnen

Das Dialogfeld, welches nach Betätigung der Schaltfläche "New Call" erscheint, fragt den Nutzer einige Daten bezüglich der zu erzeugenden Anfrage ab, wozu unter anderem auch die Kontaktdaten des betroffenen Kunden gehören.

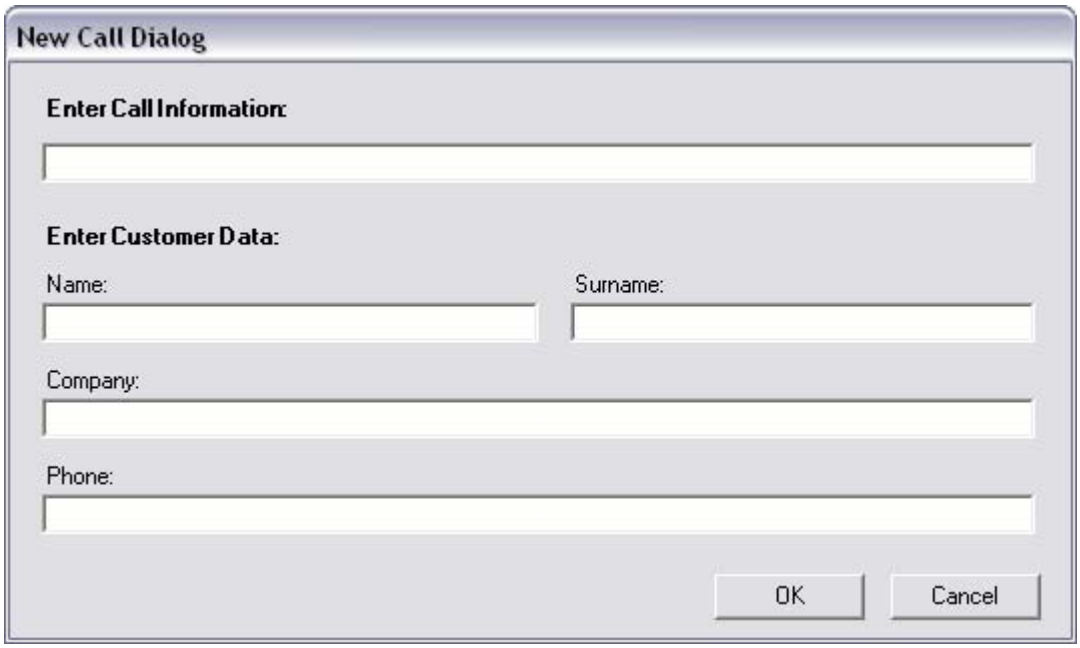


Abbildung 15.4-3 Dialog zum Anlegen eines neuen Calls

Die Betätigung der "Ok"-Schaltfläche schließt den Eingabevorgang ab und erzeugt einen neuen Call in der aktuell gewählten Box.

15.4.2.3 Einen Call in seine "Personal Box" holen

Um einen Call, der derzeit von keinem Mitarbeiter bearbeitet wird, in seine Personal Box zu verschieben, muss der aktuell eingewählte Nutzer lediglich die "Grab Call"-Schaltfläche betätigen und in dem daraufhin erscheinenden Dialog den entsprechenden Call auswählen. Nachdem die Auswahl mittels der "Ok"-Schaltfläche bestätigt wurde, wird der Call entsprechend verschoben.

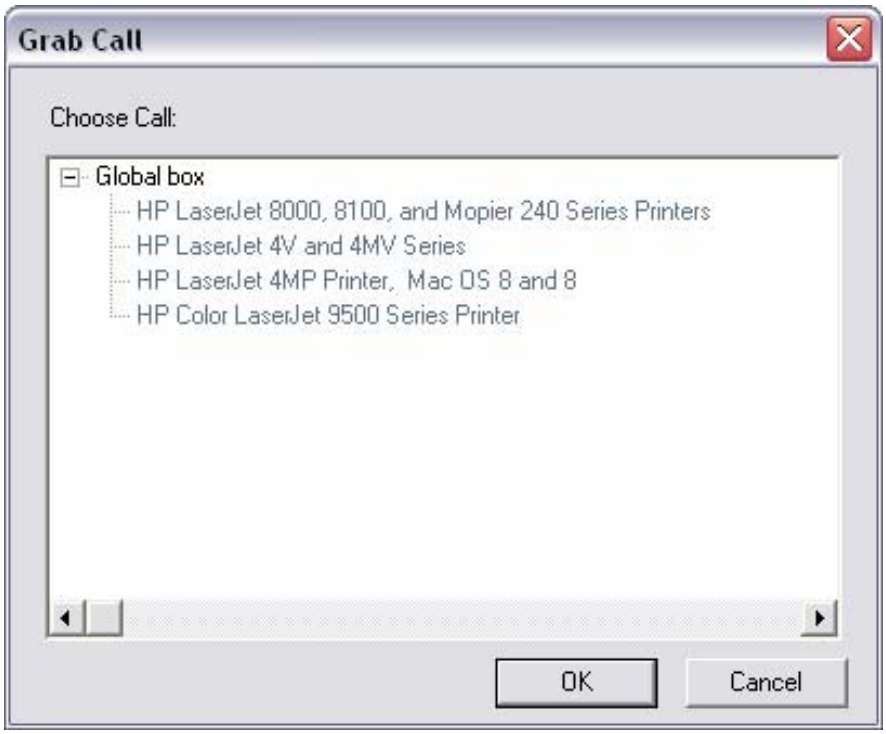


Abbildung 15.4-4 Auswahl Dialog zum Holen eines Calls aus der "Global box"

15.4.2.4 Einen Call an den Secondlevel Support weiterleiten

War ein Mitarbeiter bei der Bearbeitung eines Calls nicht erfolgreich (z.B. weil das behandelte Themengebiet außerhalb seines Kompetenzbereiches liegt), so kann er mittels der Schaltfläche "Publish Call" eine Weiterleitung des Calls an den Secondlevel Support veranlassen. Damit einhergehend wird der Call in die entsprechende Box "Secondlevel Support Box" der jeweiligen Nutzergruppe verschoben.

15.4.2.5 Einen Call speichern

Bei der Beabreitung eines Calls kann der Nutzer Einträge in den Kategorien "Problem", "Actions", "Notes" und "Links" (welche jeweils durch den entsprechenden Karteireiter gleichen Namens erreichbar sind) vornehmen, indem er einen Text in das am unteren Fensterrand befindliche Textfeld eingibt. Zur expliziten Speicherung derartiger Änderungen dient die Schaltfläche "Save Call" (vorgenommene Einträge werden aber auch automatisch beim Wechsel zu einem anderen Call gespeichert).

15.4.2.6 Sekundärsuche

Die in Kapitel 15.2.2.1 besprochene Suchfunktion des Case Explorers lässt sich auch für Calls verwenden, indem die Suchmethode von "primär" auf "sekundär" umgestellt wird. Das Suchfenster zeigt dementsprechend keine Lösungen, sondern ausschließlich gefundene Calls an. Durch Betätigung der "Show"-Schaltfläche wird automatisch ein Wechsel zum Call Editor und eine Auswahl des entsprechenden Calls vorgenommen. Die sonstige Benutzung der Sekundär-Suche verläuft analog zur Primär-Suche.

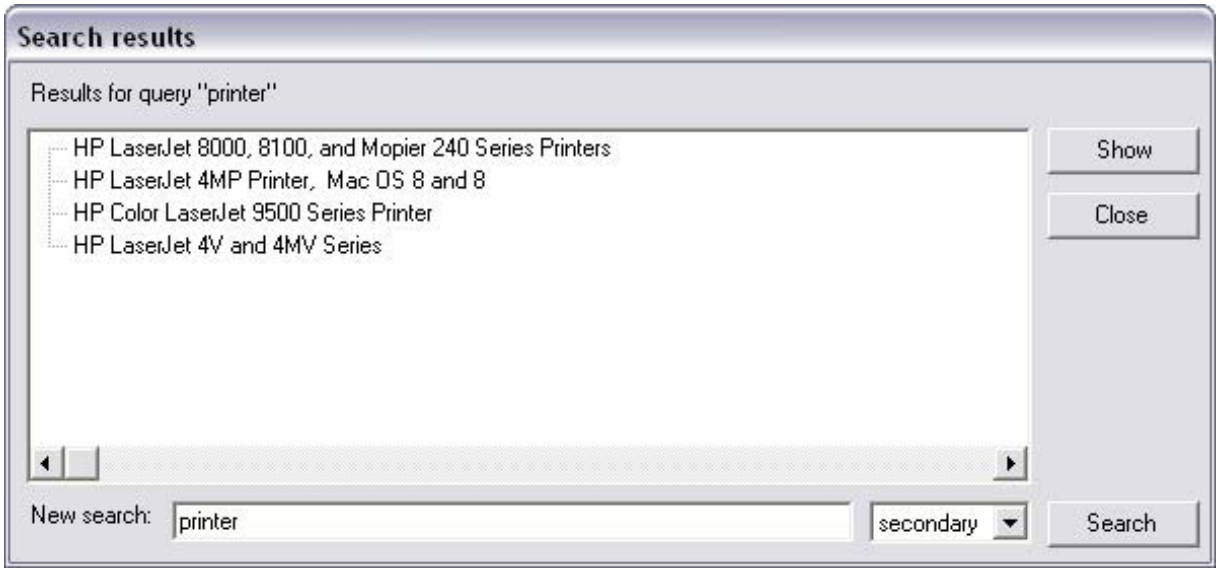


Abbildung 15.4-5 Ergebnisse der sekundär Suche

15.5 Einen neuen Call Center Agenten hinzufügen

Um einen neuen Call Center Agenten anzulegen, muss der Benutzer unter dem Hauptmenüpunkt "Super Admin" den Punkt "Create Agent" wählen.
Es öffnet sich ein Fenster, in die die Daten eingetragen werden müssen.
Da Adiuware das Prinzip der Windows Authentifizierung nutzt, muss der Name mit dem Windows Login übereinstimmen.

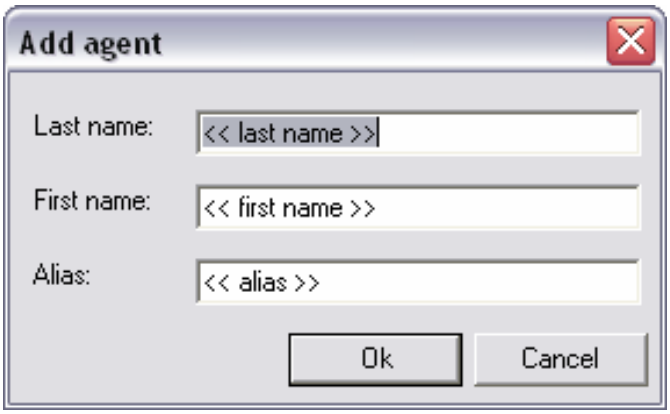


Abbildung 15.5-1 Dialog zum Anlegen eines neuen Agenten.

16 Handbuch zur Web- Anwendung

16.1 Startbildschirm

Der Startbildschirm ist in drei 3 Teile gegliedert: Kopfleiste, Menüleiste und (Informationsfeld). **(s. Abbildung 16.1: Startbildschirm)** Dabei ändert sich bei jeder Aktion nur das Informationsfeld

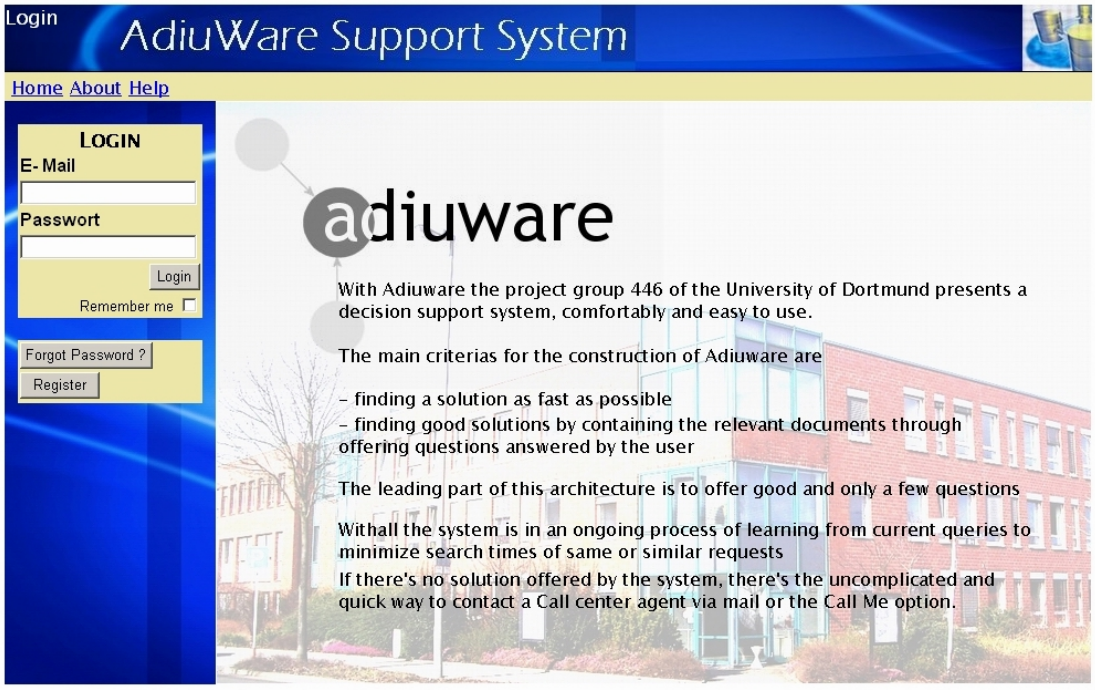


Abbildung 16.1: Startbildschirm

16.1.1 Kopfleiste des Startbildschirmes

Die Kopfleiste beinhaltet lediglich Buttons, die kurze Informationen rund um das Projekt liefert. (**s. Abbildung 16.2: Kopfleiste des Startbildschirmes**)

Abbildung 16.2: Kopfleiste des Startbildschirmes

Die Kopfleiste beinhaltet folgende Funktionen bzw. Links:

- Home
- About
- Help

Desweitern beinhaltet die Kopfleiste ein Textfeld Navigation in der der Benutzer sieht auf welcher Seite er sich befindet. Im Falle der Login Seite ist dies: *Start.*

16.1.1.1 Home

Über diesen Link gelangt man aus jedem Untermenü wieder auf die Startseite, also die Seite die bei einem Aufruf der Internetseite von Adiuware gela-

den wird. Sie wird im Informationsfeld angezeigt und beinhaltet eine kleine Einführung zu Adiuware und seinen innovativen Funktionen

16.1.1.2 About

Hier werden die Mitwirkenden und ihre Teilbereiche Im Informationsbereich aufgeführt.

16.1.1.3 Hilfe

Help

Bei Betätigung des Help Buttons öffnet sich ein Pop up. Gerade für Einsteiger soll die Hilfe eine einfache Einführung geben, wie Adiuware für Erstanmelder zu bedienen ist. Ist man im System noch nicht angemeldet werden so zum Beispiel die notwendigen Schritte erklärt, um eine erfolgreiche Anmeldung durchzuführen.

16.1.2 Menüleiste des Startbildschirms

Die linke Leiste bleibt während des gesamten Arbeitsvorganges (also auch im eingeloggten Zustand) die zentrale Einheit der Webanwendung (s. Abbildung 16.3: Menüleiste des Startbildschirms).

Abbildung 16.3: Menüleiste des Startbildschirms

Im Adiuware- Startbildschirm sind folgende Funktionen in dieser Leiste vorhanden:

Login
- E-Mail
- Password
- Login- Button
- Remember Me- Kontrollkästchen

- Sonst.
- Forgot Password?
- Not registered?

16.1.3 Login

Der Login ist dafür verantwortlich, die Authentifizierung eines Benutzers am System durchzuführen (s. Abbildung 16.4: Login).

Abbildung 16.4: Login

16.1.3.1 E- Mail

In diesem Feld wird die E-Mailadresse zur Anmeldung angegeben. In Verbindung mit dem richtigen → 16.1.3.2 Passwort meldet sich der Benutzer im System an.

16.1.3.2 Passwort

Password

In dieses Feld wird das Passwort eingetragen, mit welchem man sich in Verbindung mit der → 16.1.3.1 E- Mailadresse am System anmeldet. Dieses Feld zeigt das Passwort aus Sicherheitsgründen nur als Sternchenstring an.

Mit dem Login Button meldet man sich am System an, vorausgesetzt der Benutzer hat seine E-Mail Adresse und das korrekte Passwort in die dafür vorgesehenen Felder eingetragen.

Hinweis zum Login: Sollte der Benutzer falsche Eingaben oder generell einen inkorrekten Login vornehmen, so erscheint in der untersten Zeile (unter dem Register- Button)e in Hinweis, der den Benutzer darauf hinweist.

16.1.3.3 Remember Me- Kontrollkästchen

Wenn man das Remember Me- Kontrollkästchen aktiviert, kann bei dem nächsten Besuch auf der Adiuware- Webseite auf die Anmeldung verzichten und wird automatisch im System angemeldet und landet direkt im Arbeitsbereich von Adiuware. Die erfolgt mittels sogenannter Cookies, die dafür im Browser aktiviert sein müssen.

Hinweis zum Verwenden von Cookies:

Wird das Kontrollkästchen Remember Me aktiviert wird für die laufende Sitzung ein so genanntes persistentes Cookie erstellt. Das bedeutet, dass das Cookie nur solange existiert bis eine Sitzung beendet wird. Wenn der Benut-

zer also eine Sitzung ohne Logout beendet bleibt das Cookie bestehen. Das hat bei einem Aufruf der Login Seite eine automatische Weiterleitung auf den internen Arbeitsbereich von Adiuware zur Folge. Hat der Benutzer die vorherige Sitzung durch einen Logout beendet, wurde das Cookie gelöscht und die automatische Weiterleitung findet nicht statt!

16.1.4 Sonstiges in der Menüleiste des Startbildschirms

(s. Abbildung 16.5: Sonstiges in der Menüleiste des Startbildschirms)

Abbildung 16.5: Sonstiges in der Menüleiste des Startbildschirms

16.1.4.1 Passwort vergessen

Forgot Password:

Falls der Benutzer sein Passwort vergessen haben sollte, ist das für das System kein Problem. Durch die Funktion Forgot Password wird dem Benutzer das Passwort direkt an seine E-Mail Adresse gesendet. Um diese Funktion nutzen zu können, muss das Feld → E-Mail mit der korrekten Adresse ausgefüllt sein.

Hinweis: Diese Funktion wurde nicht implementiert!!

16.1.4.2 Neuen Account anlegen

Register?

(s. Abbildung 16.6: Neuen Account anlegen)

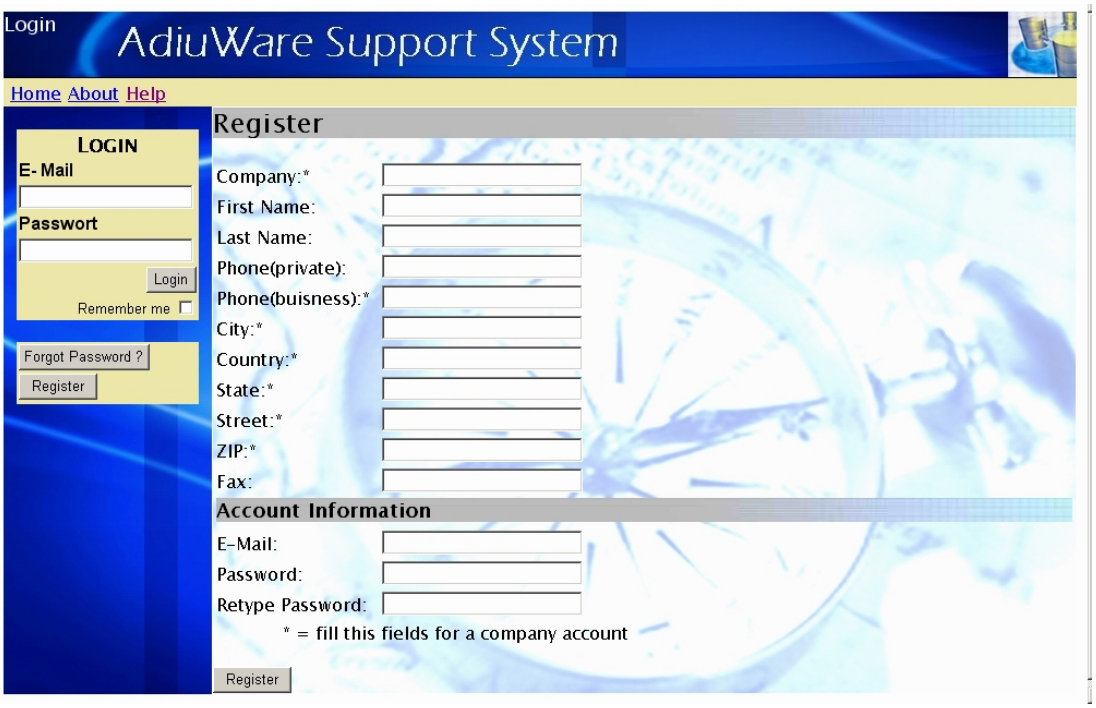


Abbildung 16.6: Neuen Account anlegen

Für Erstanmeldungen im System ist es notwendig, sich ein neues Benutzerkonto anzulegen. In die dafür vorgesehenen Felder werden die Daten eingetragen. Ein Benutzerkonto kann unterschieden werden. Es kann ein Konto für Privatnutzer oder ein Konto für Firmenangestellte sein. Die Felder, die mit einem (*) markiert sind, müssen für ein Firmenkonto ausgefüllt werden. Dabei ist es wichtig, dass ALLE Felder, die mit einem (*) markiert sind, ausgefüllt werden. Durch Bestätigung des Buttons Register werden die Daten an das System übermittelt und das Benutzerkonto ist angelegt.

16.2 Hauptmenü von Adiuware im angemeldeten Zustand

Hat sich der Benutzer erfolgreich eingeloggt, befindet er sich im Hauptmenü von Adiuware. Das Menü ist aufgebaut wie das des Startbildschirmes, es gibt also die Einteilung in Kopfleiste, Menüleiste und Informationsfeld (**s. Abbildung 16.7: Hauptmenü von Adiuware im angemeldeten Zustand)**.

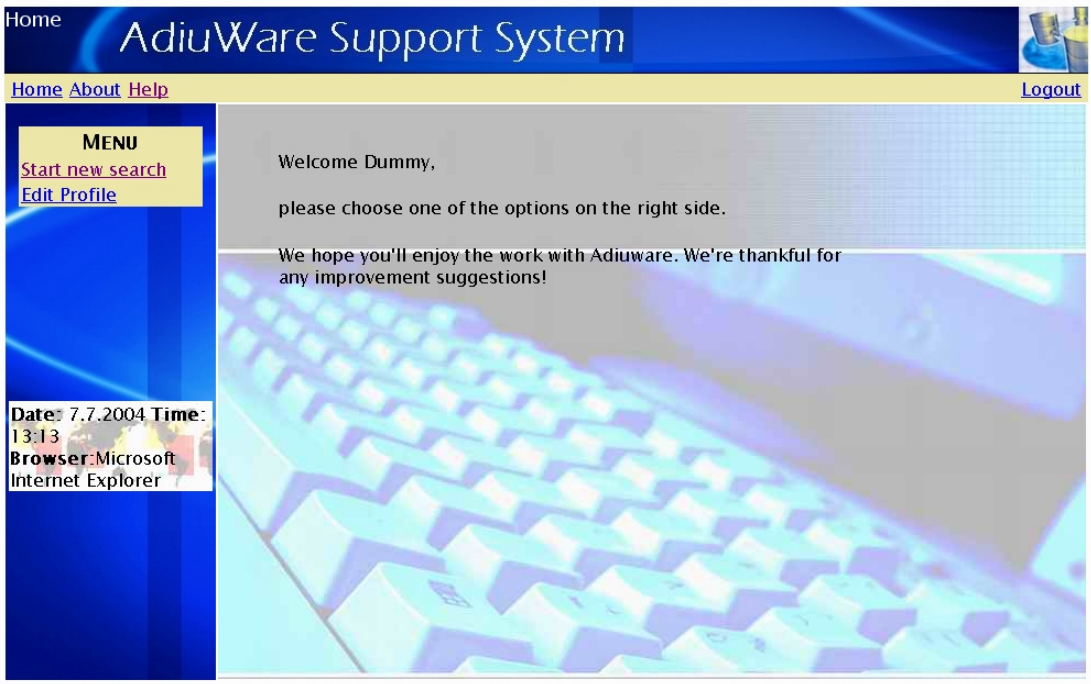


Abbildung 16.7: Hauptmenü von Adiuware im angemeldeten Zustand

16.2.1 Kopfleiste im angemeldeten Zustand

Die Kopfleiste beinhaltet lediglich Buttons, die kurze Informationen rund um das Projekt liefert. Zudem wird hier die sichere Abmeldung vom System behandelt. (**s. Abbildung 16.2: Kopfleiste des Startbildschirmes**)

Abbildung 16.8: Kopfleiste im angemeldeten Zustand

Die Kopfleiste beinhaltet folgende Funktionen:

- Home
- About
- Help
- Log Out

16.2.1.1 Home

Über diesen Link gelangt man aus jedem Untermenü wieder auf die Startseite, die die ersten Schritte für die Arbeit mit Adiuware angibt. Die Anzeige erfolgt wieder im Informationsfeld

16.2.1.2 About

Hier werden die Mitwirkenden und ihre Teilbereiche im Informationsfeld aufgeführt.

16.2.1.3 Hilfe

Help

s. → **Fehler! Verweisquelle konnte nicht gefunden werden.**

16.2.1.4 Log Out

Mittels des *Logout*- Buttons ist der Benutzer jeder Zeit in der Lage, sich von dem System abzumelden. Das Cookie wird dabei gelöscht und bei einem späteren Aufruf der Seite muss sich der Benutzer neu anmelden!

16.2.2 Menüleiste im angemeldeten Zustand

Die linke Leiste bleibt genauso wie bei dem Startbildschirm die zentrale Einheit der Webanwendung (**s. Abbildung 16.9: Menüleiste im angemeldeten Zustand**).

Abbildung 16.9: Menüleiste im angemeldeten Zustand

Im Adiuware- Hauptmenü sind folgende Funktionen in dieser Leiste vorhanden. Im Hauptmenü gibt es folgende Funktionen

- New Search
- Edit Profile

16.2.2.1 Neue Suche beginnen

New Search:

Eine neue Suchanfrage wird gestartet. (→ s. 16.3. Suche)

16.2.2.2 Profil ändern

Edit Profile

(s. Abbildung 16.10: Profil ändern)

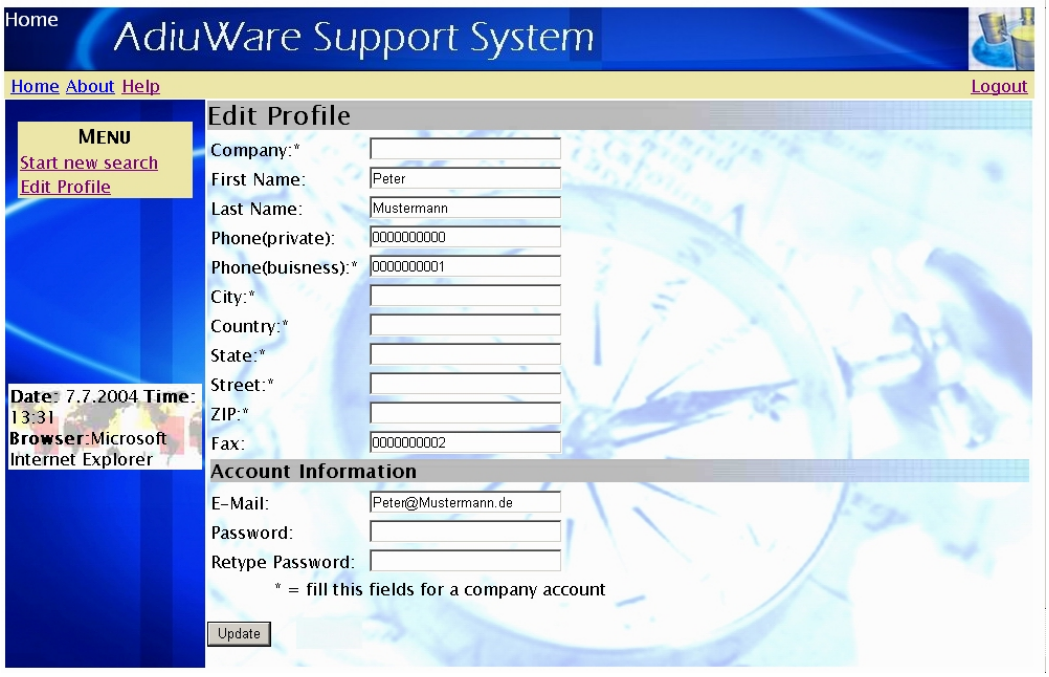


Abbildung 16.10: Profil ändern

Sollten sich persönliche Daten geändert haben, können diese aktualisiert werden. Dabei wird dem Benutzer die vertraute Maske angezeigt, die er bei seiner Erstanmeldung ausgefüllt hat. Die Felder sind mit den bisherigen Daten ausgefüllt. Soll ein Feld aktualisiert werden, kann der alte Eintrag einfach überschrieben werden und das Benutzerkonto wird durch den Button *Update* aktualisiert.

Möchte der Benutzer sein Konto aus dem System löschen, so betätigt er den *Delete* Button. Es erscheint ein Dialog, der noch mal eine Bestätigung erfordert und der Account kann gelöscht werden.

16.3 Suche

Die Suchmaske ist die zentrale Arbeitsmaske von Adiuware. Hier finden die Suchanfragen und der Dialog mit dem System statt (**s. Abbildung 16.11: Suche**).

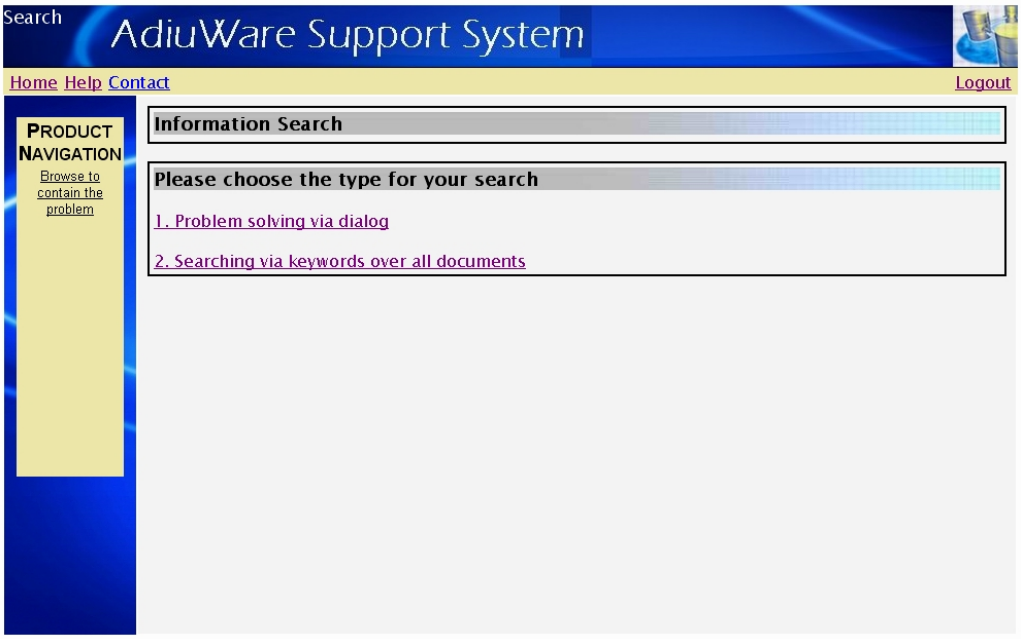


Abbildung 16.11: Suche

Die Suchmaske ist in folgende Teilbereiche gegliedert: Kopfleiste, Suchbaum, Suchfeld, später auch die Anzeige der Fragen und der Lösungen.

16.3.1 Kopfleiste im Suchbereich

(s. Abbildung 16.12: Kopfleiste im Suchbereich)

Abbildung 16.12: Kopfleiste im Suchbereich

Die Kopfleiste beinhaltet folgende Funktionen:

- Back to menu
- Help
- Contact
- Logout

16.3.1.1 Home

Durch Betätigung des *Home* Funktion gelangt man in das → 16.2 **Hauptmenü** von Adiuware zurück.

16.3.1.2 Hilfe

Help

s. → 16.1.1.3 **Hilfe**

16.3.1.3 Kontakt

Contact

Die Contact- Funktion bietet dem Benutzer die Möglichkeit an, bei erfolgloser Problemlösung Kontakt mit einem Call Center Mitarbeiter aufzunehmen. Dabei öffnet sich folgendes Pop Up Fenster (**s. Abbildung 16.13: Kontakt**):

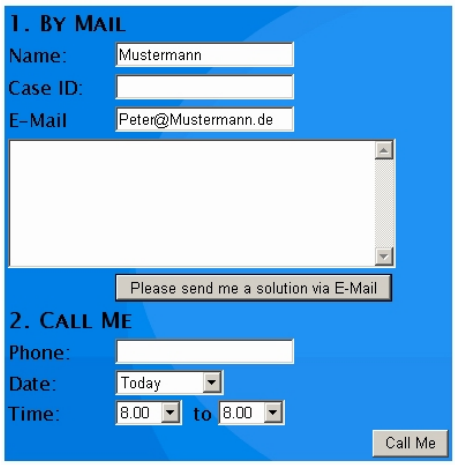


Abbildung 16.13: Kontakt

Der Benutzer hat zwei Möglichkeiten, Kontakt herzustellen:

- E-Mail
- Call Me (Telefonisch)

1.2.1.1.1 E-Mail

Es gibt vier Felder, CASE ID, Name, E-Mail und Problem Description. Die ersten drei Felder werden automatisch vom System ausgefüllt. In das vierte Feld, Problem Description kann der Benutzer sein Problem beschreiben. Bitte beachten: Je genauer das Problem umschrieben wird, desto besser kann geholfen werden! Mit dem Button Please send me a solution via E-Mail wird die Nachricht an das Call Center weitergeleitet und dieses meldet sich widerrum per Mail mit entsprechenden Lösungsschritten

1.2.1.1.2 Call Me

Die Call Me Funktion ist die zweite Möglichkeit einfach und problemlos Kontakt aufzunehmen. Diese Funktion bietet sich an, wenn eine Beschreibung des Problems für den Benutzer nicht mehr möglich ist. Es sind drei Felder vorgesehen: Telephon, Day, Time. Im Feld „Telephon“ kann der Benutzer seine Nummer hinterlassen unter der er erreichbar ist. Im Feld Day kann der der Benutzer angeben, an welchem Tag er erreichbar ist.Im Feld „Time“ kann der Benutzer einen Zeitraum angeben, in dem er erreichbar ist. Sind alle Felder ausgefüllt, werden die Daten mittels des Call Me Buttons übermittelt.
Hinweis: Diese Funktion ist nicht implementiert!

16.3.1.4 Log Out

s. 16.2.1.4 **Log Out**

16.3.2 Problemlösung mit Adiuware

Die Suche in Adiuware.Web gliedert sich in 2 Arten (s. Abbildung 16.14: Problemlösung mit Adiuware)

- Suche via Dialog mit dem System (*Problem solving via Dialog*)
- Suche via Stichwörtern über alle vorhanden Lösungsdokumente (*Searching via keywords over all documents*)

Abbildung 16.14: Problemlösung mit Adiuware

Vor Beginn der Suche kann der Benutzer zwischen diesen beiden Suchtypen wählen.

16.3.2.1 Suche via Dialog

Um eine Suche per Dialog zu beginnen muss im → 1.2.1.1.3 Such-/ Navigationsbaum1.2.1.1.3 ein Knoten angewählt werden (s. **Abbildung 16.15: Suche via Dialog**). Ist dies geschehen, wird mittels des *Search* Buttons der Dialog gestartet

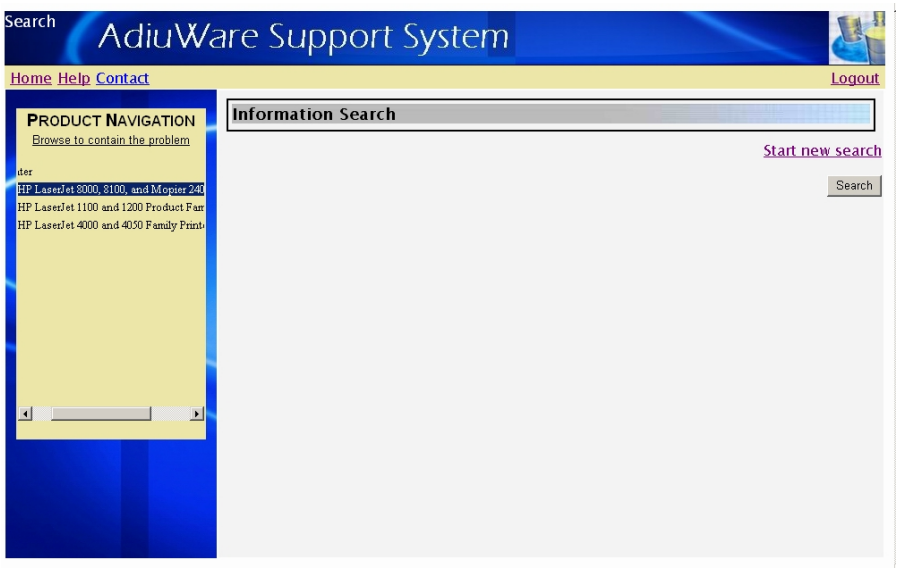


Abbildung 16.15: Suche via Dialog

Nur Blätter enthalten Fragen, Antworten und Lösungen.Wurde kein Knoten angewählt, der ein Modell enthält, mittels diesem Fragen und Lösungen angeboten werden, erscheint eine Fehlermeldung (*Please select a node*).

1.2.1.1.3 Such-/ Navigationsbaum

Der Suchbaum hat die Funktion, den Produkttypen einzugrenzen. Die Struktur ist zu der eines Verzeichnisbaumes völlig identisch, dass heisst er ist hierarchisch aufgebaut. Markiert man entsprechend einen Knoten im Suchbaum und stellt entsprechend eine Suchanfrage wird nur in den vom markierten Knoten relevanten Dokumenten gesucht. Einen markierten Knoten erkennt man an der blauen Umrandung. Zudem wird der markierte Knoten im Suchfeld angezeigt:

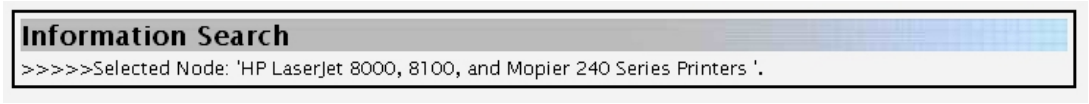


Abbildung 16.16: Anzeige des ausgwählten Knotens

1.2.1.1.4 Anzeige der Fragen/ Lösungen vom System

Please select one of the questions to answer/
Please answer one of the following questions

(**s. Abbildung 16.17: Anzeige der Fragen/ Lösungen vom System**)

Hat man den *Search* Button aktiviert, erscheinen zwei Felder. In dem oberen befinden sich die Fragen, die vom System gestellt werden, um ein Problem einzugrenzen.
Darunter befinden sich bereits gefundene Lösungen, die vom System angeboten werden (**s. 1.2.1.1.6** Lösungsanzeige).

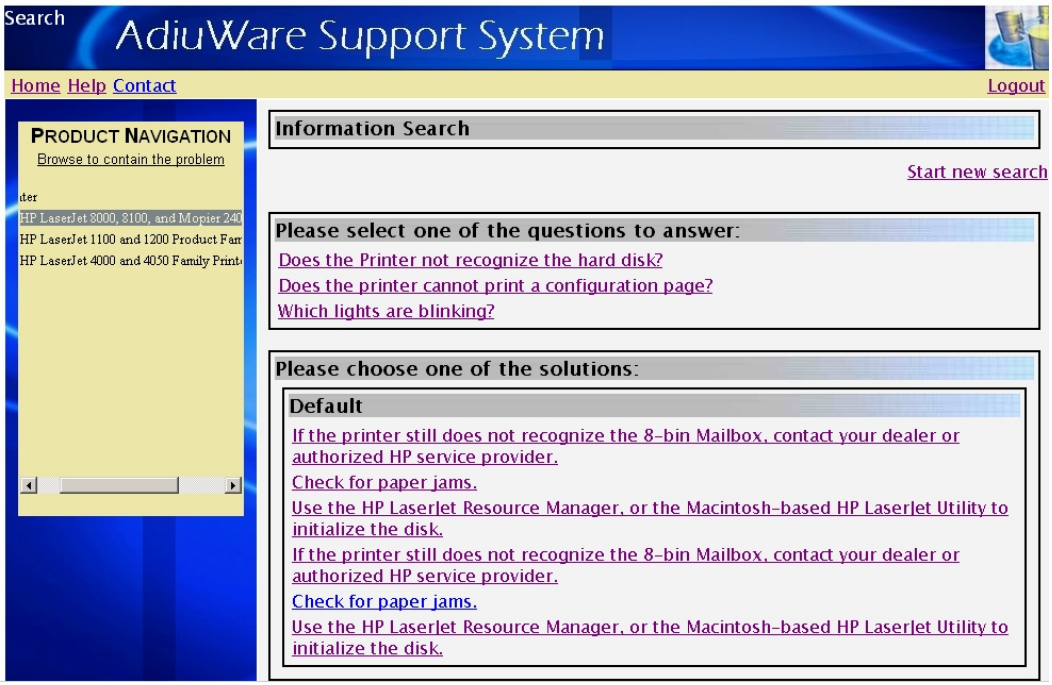


Abbildung 16.17: Anzeige der Fragen/ Lösungen vom System

Durch Klick auf die jeweilige Frage, wird ein Feld aufgeklappt (→ **1.2.1.1.5 Fragen Details und deren Beantwortung**

Question *Details/*

Please choose one of the answers

In diesem Feld werden die Fragen aufgelistet, die aufgrund der Suchanfrage des Benutzers vom System gestellt werden. Die Beantwortung der Fragen im unteren Bereich hat zur Folge, dass die Anzahl der zu durchsuchenden Dokumente wesentlich verringert wird (s. 6 **Abbildung 16.18: Fragen Details und deren Beantwortung).**

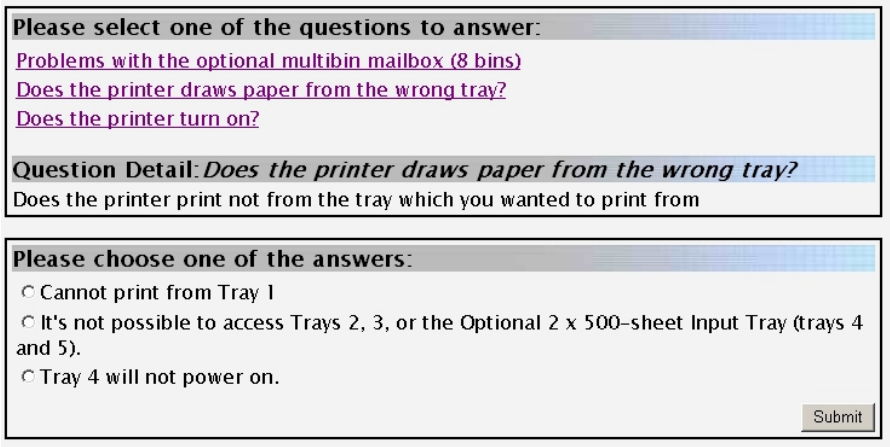


Abbildung 16.18: Fragen Details und deren Beantwortung

) in dem zum einen die Details der jeweiligen Frage aufgelistet werden und zum anderen darunter die entsprechenden Antwortmöglichkeiten.

1.2.1.1.5 Fragen Details und deren Beantwortung

Question Details/
Please choose one of the answers

In diesem Feld werden die Fragen aufgelistet, die aufgrund der Suchanfrage des Benutzers vom System gestellt werden. Die Beantwortung der Fragen im unteren Bereich hat zur Folge, dass die Anzahl der zu durchsuchenden Dokumente wesentlich verringert wird (s. 6 **Abbildung 16.18: Fragen Details und deren Beantwortung**).

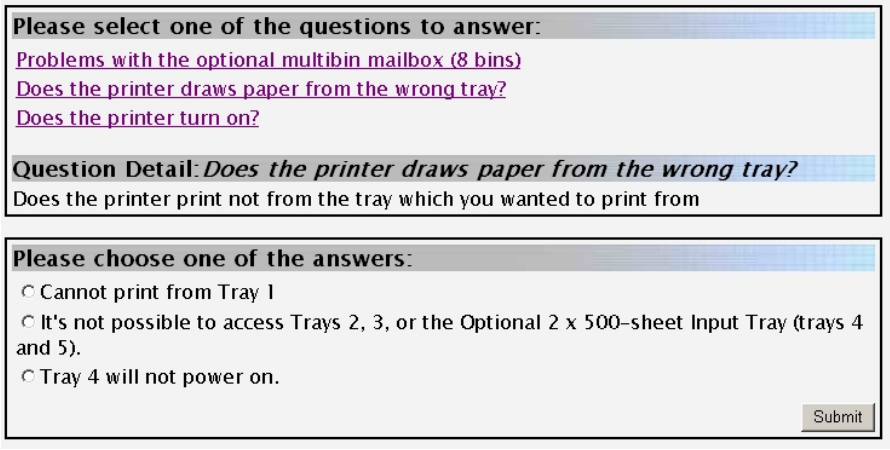


Abbildung 16.18: Fragen Details und deren Beantwortung

Sollte eine der von dem System gestellten Fragen unklar sein, kann die entsprechende Detailbeschreibung durch durch einen Mausklick auf die entsprechende Frage aufgerufen werden. Daraufhin wird in dem Feld *Question Detail* eine entsprechende Detailanzeige der Frage angezeigt, die evtl. Unklarheiten beheben sollte. Um die Übersicht in der Suchmaske zu behalten, kann dieses Feld durch Klick auf die entsprechende Frage wieder minimiert werden.
Hinweis: Sollte die Detailbeschreibung zu einer Frage bereits geöffnet sein und der Benutzer möchte sich die Beschreibung zu einer zweiten Frage anzeigen lassen, so ist es nicht nötig, dass Feld erst zu minimieren. Durch einen Mausklick auf die entsprechende Frage, wird eine evtl vorher angezeigte Detailansicht einer Frage einfach durch die neue Detailansicht ersetzt.

16.3.2.2 Beantworten einer Frage

Der Benutzer hat die Möglichkeit, die von ihm favorisierte Frage zu beantworten. Dazu aktiviert er den entsprechenden Radio Button und übermittelt mittels des *Submit*- Buttons die Antwort. Entsprechend werden im Feld → **1.2.1.1.4 Anzeige der Fragen/ Lösungen vom System** neue Fragen aufgelistet, die bei Bedarf beantwortet werden können.

Da permanent Lösungen vom System angeboten werden, entscheidet der Benutzer selbst, wann die Suche beendet ist.

1.2.1.1.6 Lösungsanzeige und deren Details

Please choose one of the solutions/
Solution Detail

Hat der Benutzer eine Frage eingegeben und die des System beantwortet, werden bei entsprechender Eingrenzung des Problems auf Grundlage der Schlagwörter bzw. der beantworteten Fragen Lösungsdokumente angeboten, die mit einer Kurzbeschreibung gekennzeichnet sind (**s. Abbildung 16.19: Lösungsanzeige und deren Details**).

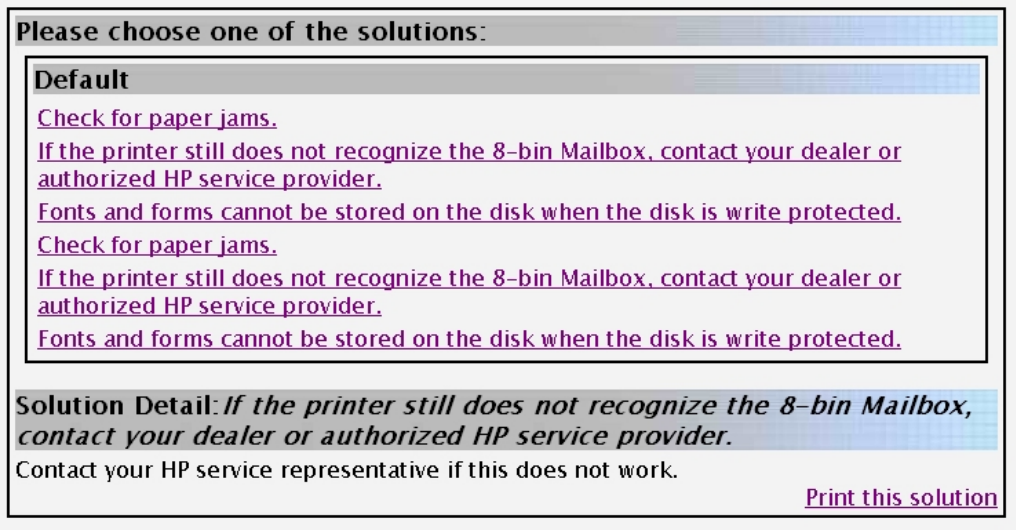


Abbildung 16.19: Lösungsanzeige und deren Details

Sollte der Benutzer ein Verständnisproblem mit der Lösung haben, hat er die Möglichkeit, Details zu dieser Frage abzurufen. Dabei klickt man auf die entsprechende Frage und unterhalb des Fragenfeldes erscheint ein zusätzliches Feld direkt unter allen gefundenen Lösungen (*Solution Details*)
Der Benutzer hat die Möglichkeit, die Detailbeschreibung einer Lösung auszudrucken. Unter der Detailbeschreibung befindet sich ein LinkButton *Print this solution.* Durch Mausklick drauf öffnet sich ein neues Pop- Up, in dem die Lösung samt Detailbeschreibung aufgeführt wird. Darunter befindet sich der *Print* –Button, der bei Betätigung ein Drucker Dialog Feld öffnet (**s. Abbildung 16.20: Drucken der Lösung**).

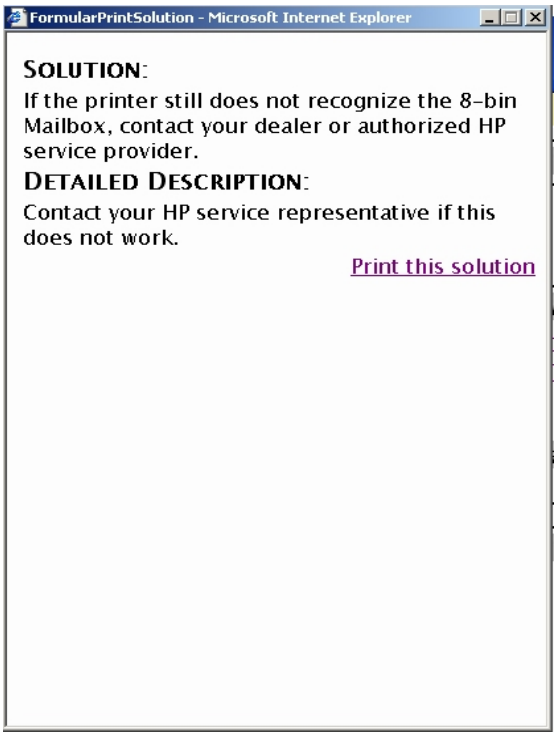


Abbildung 16.20: Drucken der Lösung

Hinweis: Sollte die Detailbeschreibung zu einer Lösung bereits geöffnet sein und der Benutzer möchte sich die Beschreibung zu einer zweiten Lösung anzeigen lassen, so ist es nicht nötig, dass Feld erst zu minimieren. Durch einen Mausklick auf die entsprechende Lösung, wird eine evtl vorher angezeigte Detailansicht einer Lösung einfach durch die neue Detailansicht ersetzt.

1.2.1.1.7 Zu einem früheren Zeitpunkt der Suche zurückspringen

<< One step back

Die *One step back* - Funktion bietet dem Benutzer die Möglichkeit an, zu einem früheren Zeitpunkt der Suche zurückzuspringen und von diesem aus die Suche in eine andere Richtung fortzuführen.
Diese Funktion soll dann aus Sackgassen führen, wenn der Benutzer merkt, dass er sich zu weit von seinem Ursprungsproblem entfernt hat. Auf diese Weise muss die Suche nicht nochmals von vorne begonnen werden.
Hinweis zur *One Step back*- Funktion: Sollte der Benutzer die Suchseite von Adiuware verlassen, wird auch die History vergessen. Das heisst, beim erneuten Aufruf der *Search*- Seite ist die vorherige Suche aus dem Verlauf gelöscht worden und es kann nur eine neue Suche angefangen werden.

16.3.2.3 Begin einer neuen Suchanfrage

Start new search

Sollte der Benutzer sich dazu entscheiden den Suchtyp zu ändern oder eine neue Anfrage zu beginnen, gelangt er mittels des *Start new search* Links zum Bildschirm 6 **16.3.2 Problemlösung mit Adiuware** zurück und kann den Suchtypus wählen!

16.3.3 Suche via Stichwörtern

Grundsätzlich hat man die Möglichkeit über folgende Art von Dokumenttypen zu suchen:

- Primärdokumente
- Sekundärdokumente

(s. Abbildung 16.21: Suche via Stichwörtern)

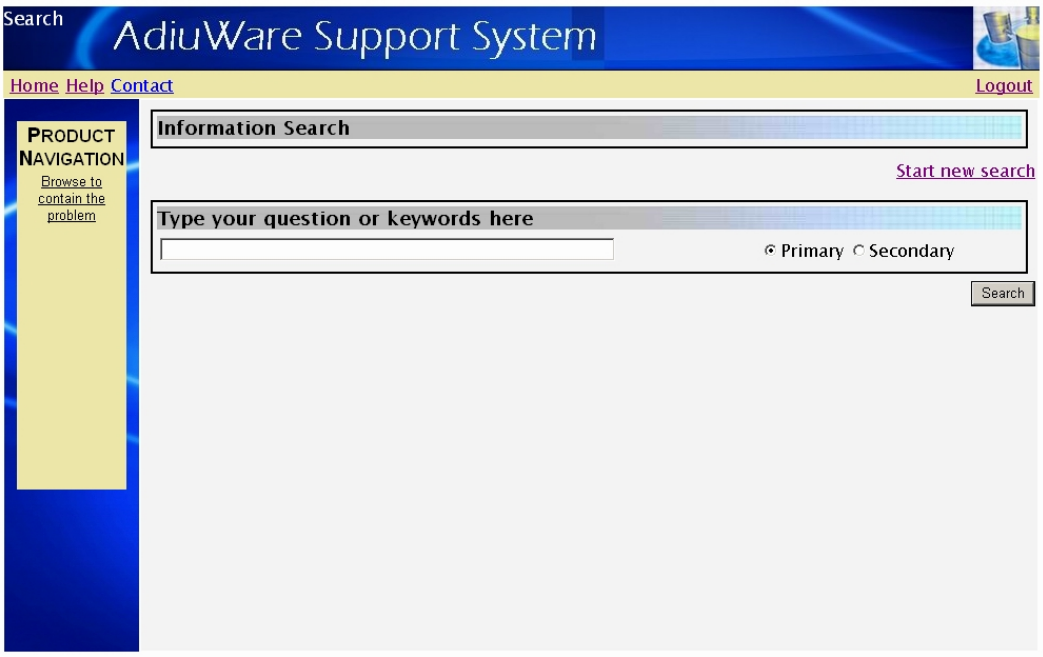


Abbildung 16.21: Suche via Stichwörtern

Diese Zweiteilung hat zum Vorteil, dass der Suchraum über die Dokumente schon zu Anfang stark eingeschränkt ist. Dennoch wird die Suche über Sekundärdokumente angeboten, da immer der Fall eintreten kann, dass die gelieferten Primärdokumente keine Lösung enthalten.

Hinweis: Der Produktbaum wird bei der Suche über Stichwörter ausgeblendet.

16.3.3.1 Frageeingabefeld

Type your keywords here/
Primary; Secondary

(s. Abbildung 16.22: Frageeingabefeld)

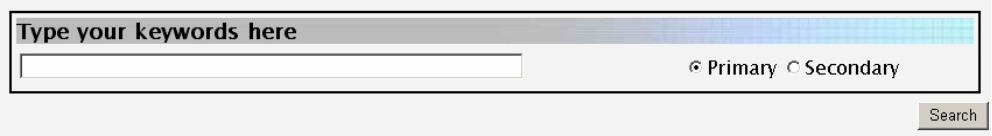


Abbildung 16.22: Frageeingabefeld

Der Benutzer kann sein Suche nun mittels Eingabe von Schlagwörtern durchführen, die durch AND oder OR miteinander verknüpft sind Mit dem *Search*- Buttons wird die Anfrage an das System gesendet.

Über den RadioButton, der *Primary* und *Secondary* enthält, kann der Benutzer die Suche über die entsprechenden Dokumente festlegen.

Anzeige der Suchergebnisse für Primärdokumente:

Die Anzeige der gefunden Lösungen in der Primärsuche entsprechen der Anzeige der 6 1.2.1.1.6 Lösungsanzeige und deren Details unter dem Suchtyp „Dialog“.

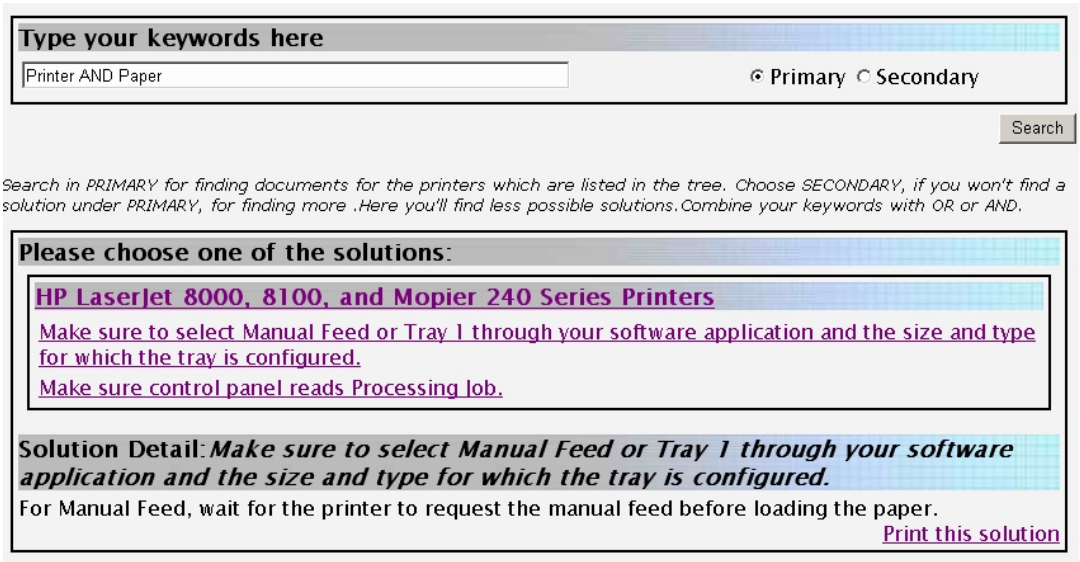


Abbildung 16.23: Anzeige der Suchergebnisse für Primärdokumente

Dabei werden die Lösungen unterschiedlichen Typen zugeordnet. In **Abbildung 16.24: Anzeige der Suchergebnisse für Sekundärdokumente:** entspricht dies der Überschrift „*HP LaserJet 8000, 8100 and Mopier 240 Series Printer*“. Es können mehrere dieser Überschriften vorhanden sein.

Ein Klick auf diese Überschriften hat die 1.2.1.1.8 Einordnung in den Produktbaum und Start eines Dialoges zur Folge.

Anzeige der Suchergebnisse für Sekundärdokumente:

Die Anzeige der Suchresultate für Sekundärdokumente zeigt im Unterschied zu den Primärdokumenten, direkt die Lösungsdetails an. Die fett gedruckten Sätze sind dabei die Kurzbeschreibung und darunter befinden sich die Details zu diesen. Alle Lösungen sind durch Striche voneinander abgegrenzt.

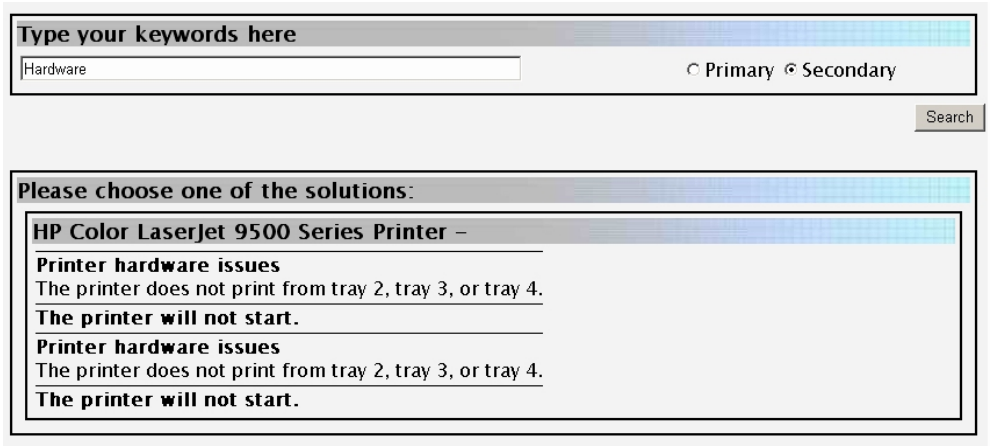


Abbildung 16.24: Anzeige der Suchergebnisse für Sekundärdokumente:

Dabei werden die Lösungen unterschiedlichen Typen zugeordnet. In **Abbildung 16.24: Anzeige der Suchergebnisse für Sekundärdokumente:** entspricht dies der Überschrift „*HP Color LaserJet 9500 Series Printer*“. Es können mehrere dieser Überschriften vorhanden sein.

1.2.1.1.8 Einordnung in den Produktbaum und Start eines Dialoges

Sollten die gefunden Dokumente dennoch keinen Aufschluss über das Problem geliefert haben, hat der Benutzer noch einmal die Möglichkeit, mittels eines Dialoges, das Problem zu lösen. Dabei klickt man auf den Knoten (s. **Abbildung 16.25: LinkButton zum Starten des Dialoges)**, unter dem die Lösungen aufgelistet sind und sofort wird eine neue **16.3.2.1 Suche via Dialog** gestartet.

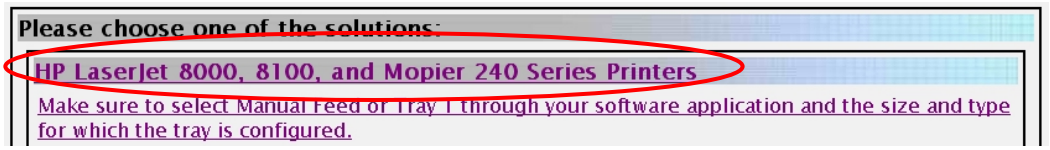


Abbildung 16.25: LinkButton zum Starten des Dialoges

Im weiteren Ablauf ordnet man sich entsprechend unter dem Produkttypen im Produktbaum ein.

17 Installationsanleitung

Im Folgenden werden die Voraussetzungen und die notwendigen Schritte für eine erfolgreiche Installation von Adiuware beschrieben.

17.1 Voraussetzungen

Um Adiuware auf ihrem Rechner nutzen zu können, müssen Sie einen Computer als ihren Server-Rechner festlegen. Dieser Server sollte unter einem Windows Betriebssystem betrieben werden. Zusätzlich ist es erforderlich VOR der Installation von Adiuware den Microsoft SQL Server 2000 oder höher zu installieren. Lesen Sie für diesen Schritt die Installationsanleitung von MS SQL Server 2000.

17.2 Installation

Die Installation von Adiuware läuft in zwei Schritten ab. Im ersten Schritt müssen Sie auf ihrem Server Computer, auf dem ihr MS SQL Server 2000 installiert ist, die Datenbank von Adiuware installieren. Im zweiten Schritt müssen Sie dann den entsprechenden Client auf jedem Computer installieren, von dem aus Sie Adiuware nutzen möchten.

17.2.1 Installation der Datenbank

Bevor Sie diesen Schritt ausführen, stellen Sie unbedingt sicher, dass Sie Ihren MS SQL Server 2000 erfolgreich installiert haben.
Um die Datenbank auf ihrem SQL Server 2000 zu installieren rufen Sie die Datei Adiuware_DB_Installer_Setup.exe auf.

Abbildung 26

Im erscheinenden Fenster (Abbildung 26) klicken Sie auf die Schaltfläche ‚Weiter'.

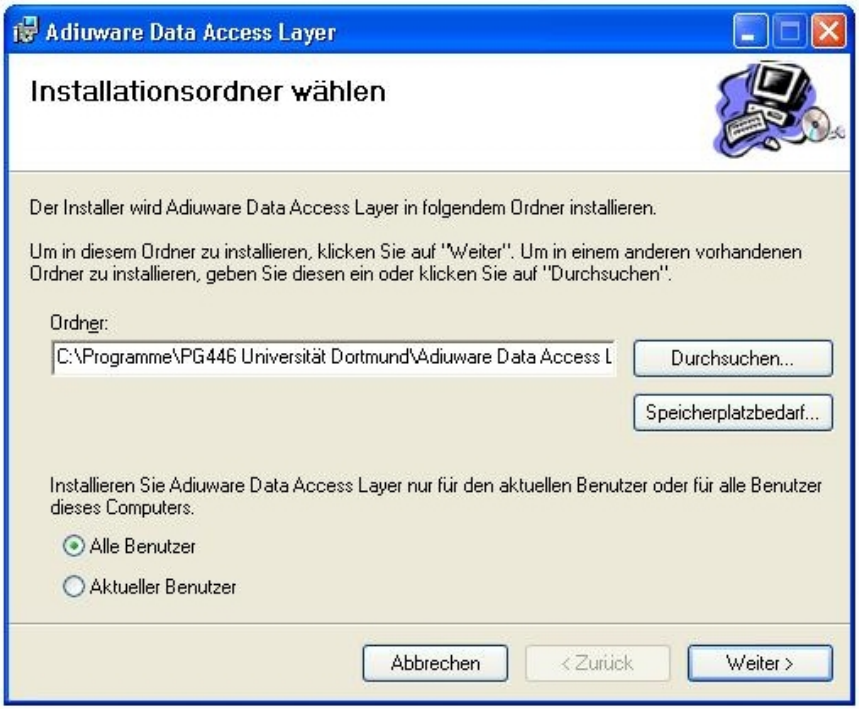


Abbildung 27

Wählen Sie dann den entsprechenden Zielordner aus und entscheiden Sie ob die Datenbank für alle Benutzer (empfohlen) oder nur für den aktuellen Benutzer installiert werden soll. Klicken Sie anschließend auf die Schaltfläche ‚Weiter'.

Abbildung 28

Im folgenden Fenster (Abbildung 28) geben Sie den Namen Ihres Server Computers ein, auf dem sich der MS SQL Server 2000 befindet. Außerdem geben Sie einen

Namen für Ihre Datenbank (Adiuware wird empfohlen), sowie einen Benutzernamen und ein Passwort ein. Dem eingegebenen Benutzer werden die vollständigen Zugriffrechte zu Ihrer Datenbank erteilt. Bestätigen Sie Ihre Eingaben, indem Sie auf die Schaltfläche ‚Weiter' klicken.

Abbildung 29

Im letzen Fenster (Abbildung 29) müssen Sie nur noch die Installation mit ‚Weiter' bestätigen.

17.2.2 Installation eines Clients

Um den Client von Adiuware auf einem Windows-basierten Computer zu installieren, starten Sie die Datei Adiuware_Setup.exe.

Abbildung 30

Bestätigen Sie das erscheinende Fenster (Abbildung 30) mit einem Klick auf ‚Weiter'.

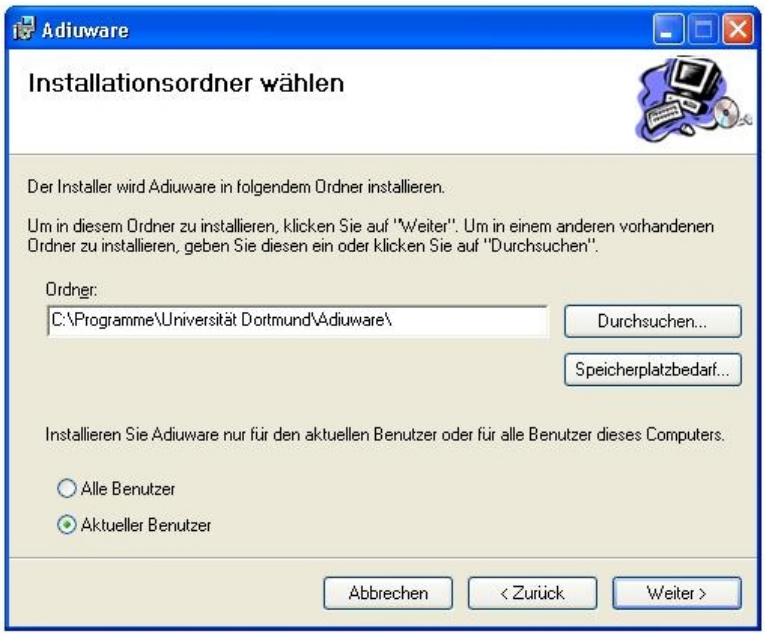


Abbildung 31

Geben Sie den Zielordner für die Installation an und Klicken Sie auf die Schaltfläche ‚Weiter'.

Abbildung 32

Bestätigen Sie abschließend (Abbildung 32) noch die eingegebenen Daten mit einem Klick auf ‚Weiter'.