

# AMRA-BBN

---

Systemdokumentation

**Studiengang Informatik**

**Projekt IP 5**

**HS 2008 / 09**

<b>Versions- Nr.:</b>	1.0
<b>Autor:</b>	Maxim Moschko, Michael Henninger
<b>Datum Letzte Revision:</b>	19.1.2009
<b>Dokument Status:</b>	Final
<b>File:</b>	081229_AMRA-BBN_Systemdokumentation.doc

# Inhaltsverzeichnis

1	Einleitung .....	3
1.1	Sinn und Zweck des Dokumentes .....	3
1.2	Ausgangslage .....	3
1.3	Problem- und Aufgabenstellung .....	3
2	Anforderungsanalyse .....	5
2.1	Funktionale Anforderungen .....	5
2.2	Forderungen .....	5
2.3	Wünsche .....	5
2.4	Nichtfunktionale Anforderungen .....	6
2.4.1	Lizenzfrage .....	7
2.5	Use Cases .....	7
2.5.1	Diagramm .....	7
2.5.2	Gesamtablauf .....	8
3	Lösungskonzept im Überblick .....	10
3.1	Existierende Software (FAI-BBN) .....	10
3.2	Überblick AMRA-BBN .....	11
3.3	AMRA-BBN Funktionalität .....	11
3.3.1	Kontrollflussdiagramm .....	12
3.3.2	Datenflussdiagramm .....	13
3.4	Konventionen und Einschränkungen .....	14
4	Datenbankdesign .....	15
4.1	Statische Tabellen .....	15
4.2	Dynamische Tabellen .....	16
4.2.1	Verknüpfung mit dem Bayesian-Belief-Netzwerk .....	17
4.3	Beziehungen .....	17
4.4	Views (Abfragen) .....	17
4.5	Datenbank bereitstellen .....	18
5	Umsetzung AMRA-BBN .....	20
5.1	Allgemein .....	20
5.2	Softwaredesign .....	20
5.3	Änderungen am bestehenden Code .....	21
5.3.1	Anpassungen an der GUI .....	21
5.3.2	Weitere Anpassungen .....	21
5.4	Neuer Code .....	22
5.4.1	DAO- Schicht .....	22
5.4.2	DTO- Schicht .....	23
5.4.3	Logik- Schicht .....	24
5.4.4	Wizard- Schicht .....	25
5.4.5	ConnectedDb Klasse .....	26
5.4.6	DBModule Klasse .....	26
5.4.7	Hilfsklassen .....	26
5.5	Probleme .....	27
6	Testen .....	28
7	Ergebnisse .....	32
7.1	Was wurde erreicht? .....	32
7.2	Was wurde nicht erreicht? .....	32
7.3	Schlussfolgerungen .....	32
8	Glossar .....	33
9	Referenzliste .....	33
10	Anhang .....	34
10.1	Neue Properties in den DSC-Parser einfügen .....	34
10.1.1	SimpelJdbcTemplate .....	35

# 1 Einleitung

## 1.1 Sinn und Zweck des Dokumentes

Diese Systemdokumentation bietet eine Übersicht über den Aufbau der Software. Es soll Programmierern einen Überblick über die Struktur der Software geben, damit sie sich schnell zurechtfinden. Aussenstehenden hilft diese Dokumentation die Funktionsweise und angewendeten Techniken zu verstehen.

Der Name der neuen Software ist AMRA-BBN. AMRA-BBN steht für „Air Medical Research Academy – Bayesian-Belief-Networks“ und ist der Name des aktuellen Projektes, welches vom Fliegerärztlichen Institut vergeben wurde.

Das Dokument geht überwiegend nur auf die durchgeführten Erweiterungen der Software FAI-BBN (Produkt unseres Vorgängers) ein. Für ein umfassendes Verständnis der gesamten Software (Berechnung, Parser, GUI usw.) muss auch die Dokumentation der Basissoftware berücksichtigt werden.

## 1.2 Ausgangslage<sup>1</sup>

„Die Risikobewertung und Prävention von Flugunfällen erfordert eine Modellvorstellung der Zusammenhänge zwischen den Fähigkeiten und Fertigkeiten des Piloten und seiner Tätigkeiten im Cockpit. Am Fliegerärztlichen Institut (FAI) der Schweizer Luftwaffe werden solche Zusammenhänge untersucht und mittels so genannter Bayesian-Belief-Networks (BBN) modelliert. Damit sind spezifische Anforderungen an eine Software zur Generierung eines BBN verbunden. Die auf dem Markt erhältlichen Tools können diese Anforderungen nicht erfüllen.

Einzelne Bayes-Netzwerke werden derzeit im FAI mit der Microsoft-Software MSBNx und der Software FAI-BBN (HSR-Studienarbeit) bearbeitet. Es fehlt allerdings dringend eine Software, um für eine rationale Datenerfassung und Verwaltung (die auch mobil erfolgen kann) auf Bayes-Netzwerke und deren Ergebniswerte automatisiert zugreifen zu können.“

## 1.3 Problem- und Aufgabenstellung

Die Aufgabenstellung besteht aus zwei Teilen. Zu Beginn ist ein Entwurf zu erarbeiten, welcher ermöglicht, Eingabewerte aus einer Datenbank in ein Bayesian-Belief-Netzwerk zu laden und Ausgabewerte wiederum in der Datenbank zu speichern. Anschliessend wird der Entwurf in eine lauffähige Software umgesetzt.

Das Programm soll im Stande sein, Bayesian-Belief-Netzwerke im DSC-Format einzulesen und Berechnungen durchzuführen. Input- und Output-Knoten müssen als solche gekennzeichnet werden. Auf einen Input-Knoten können aus einer Datenbank stammende Eingabewerte projiziert werden. Output-Knoten ermöglichen unter Berücksichtigung der aktuell gesetzten Wahrscheinlichkeiten eine Berechnung. Die Berechnungsergebnisse werden wiederum in der Datenbank gespeichert. Da die Struktur der Datenbank vom Bayesian-Belief-Netzwerk abhängig ist, muss die Software, ausgehend von einem Netzwerk, passende Datenbanktabellen erzeugen. Die Software soll mehrere Berechnungen automatisiert hintereinander (Stapelverarbeitung) durchführen können.

Die Softwarelösung muss keine Bayesian-Belief-Netzwerke erzeugen oder modifizieren können. Hierfür benutzt das FAI die von Microsoft stammende Software MSBNx.

Als Datenbank kommt Microsoft Access zum Einsatz. Es ist eine Programmiersprache zu wählen, welche mit der Datenbank problemlos zusammenarbeiten kann.

Die neue Software AMRA-BBN darf bereits vorhandene Softwarepakete benutzen. Vor allem für die Berechnung könnte es sinnvoll sein, Teile anderer Software-Pakete einzusetzen, da es bereits

---

<sup>1</sup> Christoph Stamm, Samuel Huber: Informatik Projektarbeit P5 (I1121) Bayesian-Belief-Netzwerke

viele brauchbare Lösungen gibt und das Schreiben eines neuen Algorithmus unter Umständen zu aufwändig wäre. Daher muss zu Beginn nachgeforscht werden, ob ein Softwarepaket existiert, welches einen passenden Algorithmus implementiert. Dieses soll einfach in das eigene Projekt integriert werden können und auch nicht durch Lizenzbestimmungen eingeschränkt sein.

## 2 Anforderungsanalyse

### 2.1 Funktionale Anforderungen

Nachfolgend sind alle funktionalen Anforderungen aus dem Dokument „Informatik Projektarbeit P5 (I1121) Bayesian-Belief-Netzwerke“ aufgelistet. Dabei wird zwischen Forderungen und Wünschen unterschieden. Forderungen müssen am Ende der Projektarbeit zwingend implementiert sein, wobei Wünsche implementiert werden können, jedoch nicht implementiert werden müssen.

### 2.2 Forderungen

- **Einlesen bestehender Modelle (DSC)**

Das Einlesen und Speichern von bereits bestehenden BBN-Modellen, welche mit der Microsoft-Software (MSBNX) oder einer anderen kompatiblen Software erstellt wurden und in dem DSC Dateiformat (Speicherformat von MSBNx) vorliegen, muss möglich sein.

- **Berechnungen ermöglichen**

Abfragen und Berechnungen im neuen Tool müssen möglich sein und exakt die gleichen Ergebnisse liefern, wie mit den bestehenden Programmen.

- **Datenbankstruktur basierend auf einem Modell generieren**

Anhand des Aufbaus eines eingelesenen Modells soll eine geeignete Datenbankstruktur erzeugt werden. Die Datenstruktur muss in der Lage sein, Eingabe- und Ausgabewerte für die Knoten des Modells zu speichern. Weiterhin muss die Datenstruktur über geeignete Beziehungen und Views verfügen, um eine einfache Verarbeitung zu ermöglichen.

Es soll möglich sein, Daten direkt durch Access-Anwendungen einzuspeisen und übernehmen zu können.

- **Einzelberechnung**

Eine Berechnung wird mit den Eingabedaten eines Piloten durchgeführt. Die Eingabedaten müssen hierzu aus der Datenbank gelesen werden. Die Resultate werden wiederum in den dafür vorgesehenen Tabellen gespeichert. Die Berechnung soll automatisiert durchgeführt werden. Der Benutzer muss nur den gewünschten Eingabedatenbereich auswählen. Das Resultat sollte mit einer manuellen Eingabe der Werte in das Modell und einer folgenden manuellen Berechnung identisch sein.

- **Stapelberechnung**

Eine Stapelberechnung ist eine Einzelberechnung für mehrere Piloten. Es gelten dieselben Anforderungen wie an die Einzelberechnung.

### 2.3 Wünsche

- **Import bestehender Modelle (XML)**

Modelle, die im XML Dateiformat vorliegen, müssen geladen werden können und die Erzeugung einer Datenbank sowie die Durchführung einer Berechnung ermöglichen. Die Funktionalität muss identisch zum DSC Dateiformat sein.

- **Grafische Darstellung eines Netzwerks**

Ein eingelesenes und berechnetes Bayesian-Belief-Netzwerk wird als Grafik mit Knoten und Kanten dargestellt. Die Grafik zeigt die Eingabewerte sowie die Berechnungsergebnisse an.

- **Grafische Auswertung der Berechnungsergebnisse**

Die Berechnungsergebnisse müssen in einer grafischen Form darstellbar sein. Die

Darstellung soll sich dazu eignen, um z.B. in einer Präsentation verwendet zu werden. Es soll möglich sein, eine komplette Auswertung für einen Piloten beziehungsweise für alle Piloten bei einem bestimmten Ausgabewert zu erstellen.

## 2.4 Nichtfunktionale Anforderungen

Nachfolgend sind die wichtigsten nichtfunktionalen Anforderungen an dieses Projekt aufgelistet. Sie wurden in die Abschnitte Funktionalität, Benutzbarkeit, Zuverlässigkeit, Sicherheit, Leistung, Änderbarkeit und Übertragbarkeit unterteilt und mit einer Gewichtung versehen, wobei diese die folgende Bedeutung haben:

- 1 sehr wichtig
- 2 wichtig
- 3 mittelmässig
- 4 weniger wichtig
- 5 vernachlässigbar

Anforderungen	Gewichtung				
	1	2	3	4	5
<b>Vollständigkeit und Korrektheit</b> Die Software muss vollständig und korrekt sein, d.h. es muss alle funktionalen und nicht funktionalen Anforderungen erfüllen.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>Funktionalität</b>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>Richtigkeit:</b> Die angezeigten berechneten Wahrscheinlichkeiten müssen unbedingt stimmen, da sonst falsche Informationen in Entscheidungsfindungen einfließen könnten.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>Ordnungsmässigkeit:</b> Die zur Verfügung gestellten Funktionen dürfen ausschliesslich die gewollten Aufgaben erfüllen.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>Benutzbarkeit</b>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>Verständlichkeit:</b> Die angezeigten Informationen sollen für alle Benutzer verständlich und übersichtlich dargestellt werden.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>Erlernbarkeit:</b> Um die neue Software AMRA-BBN innert nützlicher Frist verwenden zu können, soll die allfällige Einarbeitungszeit nicht mehr als eine dreiviertel Stunde betragen.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>Bedienbarkeit:</b> Um die neue Software im Alltag bedienen zu können, sollen keine erweiterten Bedien-Fähigkeiten von Nöten sein.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>Zuverlässigkeit</b>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>Fehlertoleranz:</b> Fehler könnten inkonsistenten Daten bedeuten, was wiederum dazu führt, dass die Eintrittswahrscheinlichkeiten im schlimmsten Fall nicht mehr der Wahrheit entsprechen. Deshalb soll die Auftretenswahrscheinlichkeit solcher Fehler maximal 1% betragen.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>Sicherheit:</b> Es sind keine speziellen Anforderungen an die Sicherheit gestellt, da die neue Software keine besondere Netzwerkfunktionalität besitzen muss und die gespeicherten Daten nicht verschlüsselt werden sollen.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<b>Leistung</b>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

<b>Zeitverhalten:</b> Die Oberfläche soll schnell auf Eingaben reagieren. Zeitintensive Operationen sollen gekennzeichnet werden.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>Verbrauchsverhalten:</b> Bei den Zielcomputern handelt es sich um gewöhnliche Workstations auf denen parallel noch andere Prozesse bzw. Tasks laufen. Diese dürfen nicht beeinträchtigt werden. Die durchschnittliche CPU Auslastung beim Einsatz der Software darf höchstens zusätzlich 10% betragen. Bei rechenintensiven Operationen darf, abhängig von der Eingabemenge, der Wert überschritten werden.	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>Änderbarkeit</b>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>Modifizierbarkeit:</b> Die neue Software soll in einem gewissen Masse modifizierbar sein. Die angezeigten Texte sollen konfigurierbar sein. Es soll möglich sein, Datenbanktabellen und -spalten umzubenennen. Jedoch wird ein gewisses Verständnis dafür vorausgesetzt.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>Stabilität:</b> Stabilität ist für die neue Software wichtig. Unvollständige oder ungültige Eingabemengen dürfen nicht zum Absturz des Programms führen.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>Prüfbarkeit:</b> Angezeigte Werte können mit den Werten der bereits bestehenden Software MSBNx verglichen werden. Zusätzlich werden alle Vorgänge in einer separaten Datei geloggt.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>Wartbarkeit:</b> Wiederkehrende Abläufe sollen verständlich dokumentiert werden.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>Übertragbarkeit</b>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<b>Installierbarkeit:</b> Die Installation soll schnell, verständlich und bequem mit einem Assistenten geschehen. Es soll bisher nur das Windows Betriebssystem unterstützt werden.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

### 2.4.1 Lizenzfrage

Es dürfen in der FAI-BBN Software keine Code-Bestandteile von Drittpersonen eingesetzt werden, die unter einer Lizenz für eingeschränkte Benutzung stehen. D.h. die neue Software muss in Zukunft auch kommerziell (z.B. in einer Dienstleistung) uneingeschränkt benutzt werden dürfen.

## 2.5 Use Cases

### 2.5.1 Diagramm

Während der Arbeit am Projekt konnten die oben beschriebenen funktionalen Anforderungen genauer definiert und gegliedert werden. Das Ergebnis ist nachfolgendes Use Case Diagramm. Die Use Cases in der Boundary (Begrenzung) „Microsoft Access“ stehen bereits durch die Software Access zur Verfügung. Use Cases im Boundary „AMRA-BBN“ werden in der neuen Software AMRA-BBN implementiert.

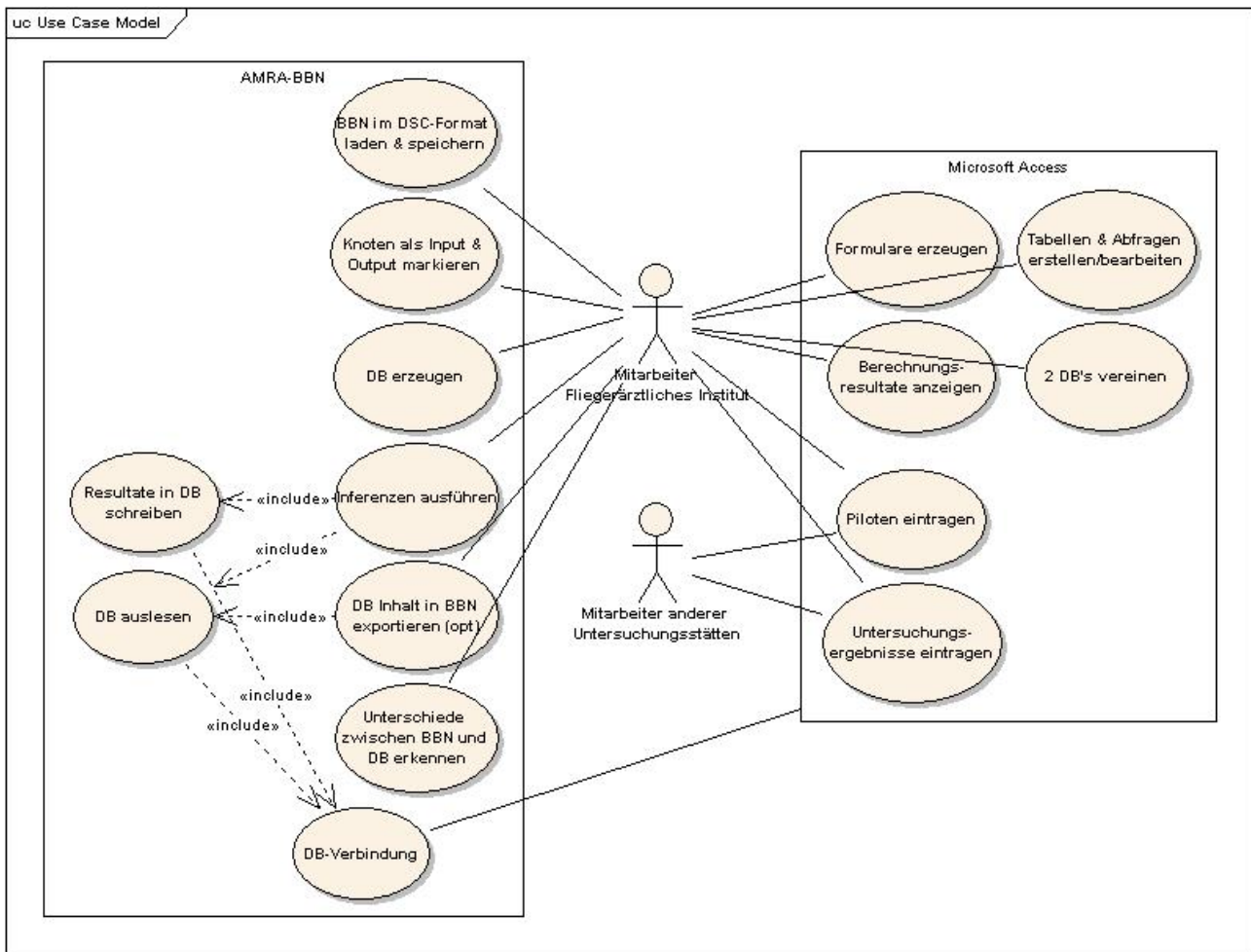


Abbildung 1: Use Case Diagramm

## 2.5.2 Gesamtablauf

Es wird auf eine detaillierte Beschreibung jedes einzelnen Use Cases verzichtet und nur der Gesamtablauf (Hauptverwendung) beschrieben. Dieser zeigt, wofür die neue Software AMRA-BBN überhaupt eingesetzt wird. Use Cases werden im Text fett hervorgehoben.

Ein Mitarbeiter des Fliegerärztlichen Instituts (FAI) hat mit der bereits bestehenden Software MSBNx oder dem Programm FAI-BBN ein BBN modelliert und als DSC-Datei gespeichert. Dieses öffnet er in der neuen Software AMRA-BBN (**BBN im DSC-Format laden**). Das Tool liest die Daten aus der DSC-Datei und stellt das Modell tabellarisch oder grafisch dar. Der Mitarbeiter markiert bestehende Knoten als Input- oder Output-Knoten (**Knoten als Input & Output markieren**) und speichert das geänderte Modell als DSC-Datei (**BBN im DSC-Format speichern**). Diese geänderte DSC-Datei kann nun wieder mit dem Tool MSBNx eingelesen und bearbeitet werden.

Hat der Mitarbeiter seine beabsichtigten Änderungen am BBN durchgeführt, kann er das BBN mit der Software AMRA-BBN öffnen und eine Microsoft Access Datenbank zum geladenen BBN erstellen lassen (**DB erzeugen**). Die Datenbank stellt Tabellen und Abfragen bereit, um Untersuchungsergebnisse einzutragen und abzufragen.

Um einfach Daten in die Datenbank einzutragen, werden von einem FAI Mitarbeiter Access Formulare anhand der Abfragen erstellt (**Formulare erzeugen**). Über diese Formulare können Mitarbeiter vom FAI und anderen Instituten Piloten (**Piloten eintragen**) und ihre Untersuchungsergebnisse (**Untersuchungsergebnisse eintragen**) in die Datenbank eintragen. Diese Datenerfassung kann über einen längeren Zeitraum andauern.

Sobald Daten in der Datenbank verfügbar sind, kann ein Mitarbeiter des FAI zu den eingetragenen Untersuchungsergebnissen Berechnungen durchführen (**Inferenzen ausführen**). Hierfür werden die Daten aus der Datenbank gelesen (**DB auslesen**), eine Inferenz durchgeführt und die

Resultate wieder in die Datenbank geschrieben (**Resultate in DB schreiben**). In Access lassen sich die Resultate über Abfragen ansehen (**Berechnungsergebnisse ansehen**) und weiterverarbeiten.

Optional soll eine Möglichkeit bereitstehen, die Daten aus der Datenbank in ein BBN zu exportieren (**DB Inhalt in BBN exportieren**). Dieses BBN wird als DSC-Datei abgespeichert und kann von der MSBNx oder FAI-BBN Software gelesen, dargestellt und bearbeitet werden. Dies ist unter anderem für Demonstrationszwecke sehr hilfreich.

Es ist möglich, dass durch neue oder geänderte Untersuchungsmethoden / -resultate auch neue Input- oder Output-Knoten hinzukommen. Allgemein kann man sagen, dass sich das BBN verändern kann. Die Datenbank muss an diese Veränderungen angepasst werden. Der FAI Mitarbeiter kann entweder eine neue Datenbank erstellen lassen und diese mit der bestehenden vereinen (**2 DB's vereinen**) oder er passt die Änderungen direkt in der Datenbank an (**Tabellen & Abfragen erstellen/bearbeiten**). AMRA-BBN soll den Betreuer dabei unterstützen und erkennen, wo das aktualisierte BBN und die Datenbank nicht mehr übereinstimmen (**Unterschiede zwischen BBN und DB erkennen**).

## 3 Lösungskonzept im Überblick

### 3.1 Existierende Software (FAI-BBN)

Im Auftrag des Fliegerärztlichen Instituts wurde bereits eine Software (FAI-BBN) zum Erstellen, Bearbeiten und Berechnen von Bayesian-Belief-Netzwerken programmiert. Das Programm hat Remo Siegwart an der Fachhochschule für Technik in Rapperswil in seiner Diplomarbeit entwickelt.

Grund für die Entwicklung einer neuen Software war die Lizenz des bisher eingesetzten Programms MSBNx. Dieses stammt von Microsoft und kann nicht uneingeschränkt benutzt werden. Deshalb wollte das FAI eine eigene Software entwickeln, welche später evtl. auch mal kommerziell, z.B. als Werkzeug in einer Dienstleistung, eingesetzt werden könnte.

Hauptfunktionen dieser Software sind:

- Generieren und Verändern des BBN mittels eines ergonomischen GUI
- Visualisierung des Graphen
- Berechnung des BBN-Netzwerkes
- Setzen von Evidenzen für einzelne Knoten
- Einlesen und Speichern eines BBN-Netzwerkes im DSC Format
- Kompatibel mit der Software MSBNx

Die Oberfläche des Programms wurde der Microsoft Software MSBNx nachempfunden. Untenstehende Abbildung zeigt ein einfaches geöffnetes Bayesian-Belief-Netzwerk mit einer durchgeführten Inferenz auf dem Knoten MaryCalls.

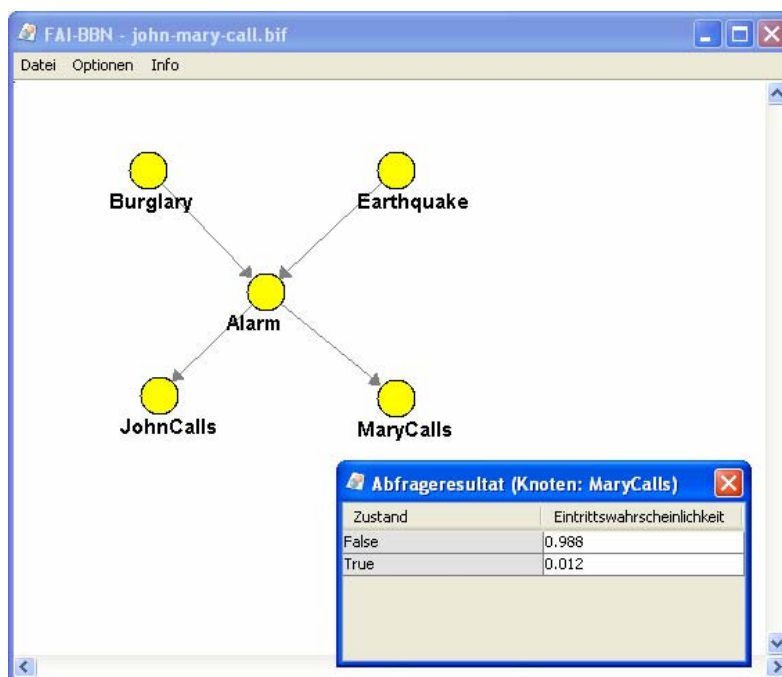


Abbildung 2: FAI-BBN

Die Software FAI-BBN ist keine komplette Neuentwicklung. Sie baut auf der Software JavaBayes auf. Remo Siegwart begründet den Einsatz von JavaBayes folgendermassen: „Aufgrund der weiten Verbreitung und der verwendeten Inferenz-Algorithmen wurde entschieden, dass für die Berechnung der Inferenz, Teile der bereits bestehenden Software JavaBayes verwendet werden sollen. Zudem sind die implementierten Berechnungsalgorithmen ausführlich dokumentiert und die Benutzung, da JavaBayes unter der GNU GPL (General Public License) vertrieben wird, durch

keine Lizenz eingeschränkt.“<sup>2</sup>

Ebenfalls ist eine kurze Beschreibung von JavaBayes in der Projektdokumentation der FAI-BBN Software aufgeführt: „JavaBayes wurde durch Fabio Cozman zwischen 1998 und 2001 an der University of Sao Paulo entwickelt. Es wurde in der Programmiersprache Java erstellt, wobei die Parser für die verschiedenen Dateiformate mit dem Parser-Generator javacc (Java Compiler Compiler) generiert wurden. Standardmässig werden die Formate XML und BIF (Bayes Interchange Format) unterstützt. JavaBayes wird unter der GNU GPL (General Public License) kostenlos vertrieben und wird zu Forschungszwecken von mehreren Hochschulen eingesetzt.“<sup>3</sup>

Weitere Informationen zu JavaBayes sind auf folgender Homepage verfügbar: <http://www.cs.cmu.edu/~javabayes/Home/>

Das FAI ist im Besitz des Quellcodes und der Dokumentation der Software FAI-BBN und stellt beides dem neuen Projekt AMRA-BBN zur Verfügung.

### 3.2 Überblick AMRA-BBN

Das existierende Programm FAI-BBN dient als Grundlage für die neue Software AMRA-BBN. Es werden einige Änderungen am bestehenden Programmcode vorgenommen, um neue Funktionalität bereitzustellen und das neue Modul einzubinden. Dieses neue Modul kümmert sich unter anderem um die Datenbankkommunikation. Weiterhin koordiniert es die Projektion der Eingabewerte auf das Bayesian-Belief-Netzwerk und die Abfrage der Resultate.

Der Benutzer nimmt die Erweiterungen der Software FAI-BBN nur als eine Anzahl von Wizards sowie einem erweiterten Kontextmenü wahr. Über die Wizards wird die neue Funktionalität gesteuert. Ein Wizard ist ein Assistent, mit dem eine Aufgabe Schritt für Schritt erledigt wird.

Das neue Modul ist nur schwach an den bestehenden Teil gekoppelt. Es finden lediglich in der GUI Aufrufe der neuen Wizards statt. Dabei wird an die Wizards ein Objekt der Klasse InferenceGraph übergeben, welches diesen nutzt, um die gewünschten Operationen auszuführen. Ansonsten bestehen keine Abhängigkeiten oder Verbindungen zwischen der ursprünglichen Software FAI-BBN und dem neuen Modul.

Ein InferenceGraph repräsentiert das gesamte Bayesian-Belief-Netzwerk und bietet Zugriff auf alle enthaltenen Knoten. Die Knoten werden durch Objekte der Klasse InferenceGraphNode dargestellt. Ein Knoten bietet unter anderem Zugriff auf den Namen, die Zustände sowie die Wahrscheinlichkeiten. Diese Werte lassen sich auch neu setzen.

Die Logik von AMRA-BBN nutzt die Informationen, die sie aus den InferenceGraphNode erhält, um beispielsweise die Datenbank zu erzeugen. Für eine Berechnung muss auf allen Knoten, zu denen ein Untersuchungsergebnis vorliegt, die Wahrscheinlichkeit im InferenceGraphNode eingetragen werden. Dieser Vorgang erfolgt automatisiert. Dazu liest das neue Modul die Werte aus der Datenbank und projiziert diese auf das Netzwerk. Dies geschieht für jeden Knoten, der als Input-Knoten markiert ist. Anschliessend kann das Programm auf beliebigen Knoten eine Inferenz ausführen. Dieser Schritt wird auf allen als Output-Knoten markierten Knoten durchgeführt und die Resultate in die Datenbank gespeichert.

### 3.3 AMRA-BBN Funktionalität

Dieser Abschnitt erläutert die erweiterte Funktionalität der AMRA-BBN Software. Funktionen die bereits in FAI-BBN realisiert sind werden an dieser Stelle nicht beschrieben.

- **Input- und Output-Knoten setzen:** Bei einem geöffneten BBN kann über einen Rechtsklick auf den Knoten im Kontextmenü ausgewählt werden ob der Knoten ein Input- oder Output-Knoten sein soll.
- **Eingabe- und Ausgabeknoten farbig hervorheben:** Im Netzwerkgraphen lassen sich markierte Knoten einfach von unmarkierten Knoten unterscheiden. Input-Knoten werden

<sup>2</sup> Projektdokumentation: Software zur Modellierung von Bayesian-Belief Netzwerken; Remo Siegwart; S. 83

<sup>3</sup> ebd.

grün dargestellt, Output-Knoten in der Farbe Magenta.

- **Datenbank generieren:** Ein Wizard fragt den gewünschten Datenbanknamen und das Zielverzeichnis ab. Daraufhin wird zu dem gerade geladenen BBN eine neue Datenbank erstellt, in der jeder Input- und Output-Knoten eine eigene Tabelle hat. Dazu kommen noch einige allgemeine Tabellen, die unabhängig vom Netzwerk sind. Generierte Views verknüpfen den Inhalt mehrerer Tabellen. Damit können Formulare einfach erzeugt sowie Resultate bequem dargestellt werden.
- **Datenbankverbindung herstellen:** Der Benutzer öffnet zuerst ein BBN und wählt anschliessend eine Datenbank aus. Es wird geprüft, ob die Datenbank und das BBN zueinander passen. Falls nicht, kann der Anwender eine andere Datenbank wählen oder die Software ermitteln lassen, wo das Problem liegt.
- **Unterschiede ermitteln:** Diese Operation wird gestartet, wenn beim Verbindungsvorgang erkannt wird, dass die Datenbank und das BBN nicht zusammenpassen. Zuerst wird überprüft, ob zu jedem markierten Knoten eine entsprechende Tabelle in der Datenbank existiert. Ist dies der Fall wird überprüft, ob die Zustände eines Knotens in den jeweiligen Tabellen als Spalten vorhanden sind. Der Benutzer erhält nach der Prüfung eine Liste mit festgestellten Problemen.
- **Datenbankinhalt prüfen:** Der Inhalt einer Datenbank kann auf unvollständige oder fehlerhafte Eingaben überprüft werden. Diese Option ist freiwillig, da dieser Vorgang je nach Datenbankgrösse lange dauern kann. Falls der Benutzer diese Option auswählt, erhält er am Ende eine Auflistung aller gefundenen Probleme.
- **Berechnung durchführen:** Ist eine Datenbank erfolgreich verbunden und enthält Daten, kann der Benutzer eine Berechnung starten. Dazu wird ihm im Wizard eine Liste mit verfügbaren Eingabedaten angezeigt. Aus dieser Liste kann er Eingabedaten auswählen, welche mit dem BBN kombiniert werden. Abschliessend führt AMRA-BBN auf allen Output-Knoten eine Berechnung durch. Das Resultat der Berechnung wird wiederum in die Datenbank gespeichert.
- **Daten exportieren:** Diese Funktion läuft sehr ähnlich wie die Berechnung ab. Der Benutzer wählt verfügbare Eingabedaten aus. Diese werden mit dem BBN kombiniert. Anschliessend wird das BBN im DSC-Format gespeichert. Der Benutzer kann es öffnen und eigene Berechnungen (z.B. an nicht Ausgabeknoten) durchführen.
- **Ungültige oder leere Eingabewerte ignorieren:** Beinhaltet die Datenbank einen nicht kompletten Satz Eingabewerte oder sind die Eingabewerte ungültig so werden die Standardwerte aus dem BBN benutzt. Leere oder ungültige Eingabewerte sind kein Grund für den Abbruch einer Berechnung.

### 3.3.1 Kontrollflussdiagramm

Untenstehende Abbildung ordnet die im vorigen Abschnitt beschriebenen Funktionen in einen Kontrollfluss. Dabei werden zwei Partitionen verwendet um abzugrenzen, welche Funktionen neu sind und was bereits mit FAI-BBN möglich war. Von FAI-BBN sind nur die wesentlichsten Aspekte in diesem Zusammenhang dargestellt.

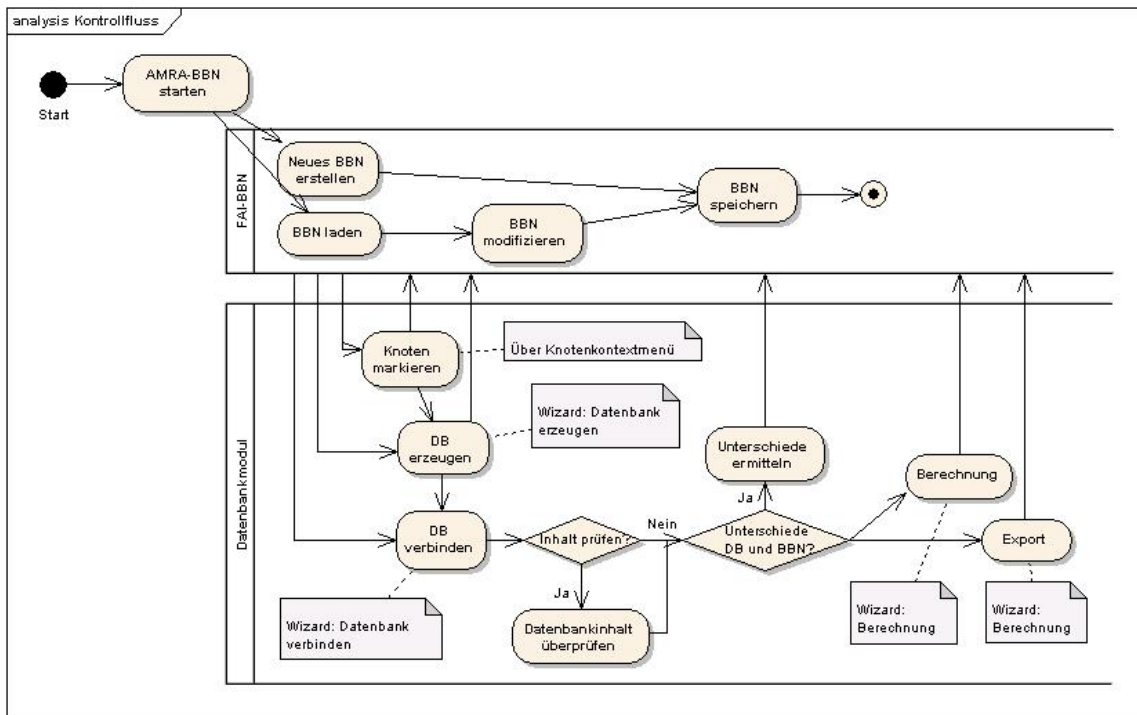


Abbildung 3: Kontrollflussdiagramm

### 3.3.2 Datenflussdiagramm

Untenstehende Abbildung stellt dar, welche Funktion welche Daten benötigt und von wo diese Daten stammen. Eckige Kästchen stellen Datenquellen dar. Kästchen mit abgerundeten Ecken sind Aktivitäten. Die Verbindungen zwischen den Datenquellen und den Aktivitäten stellt den Datenfluss dar.

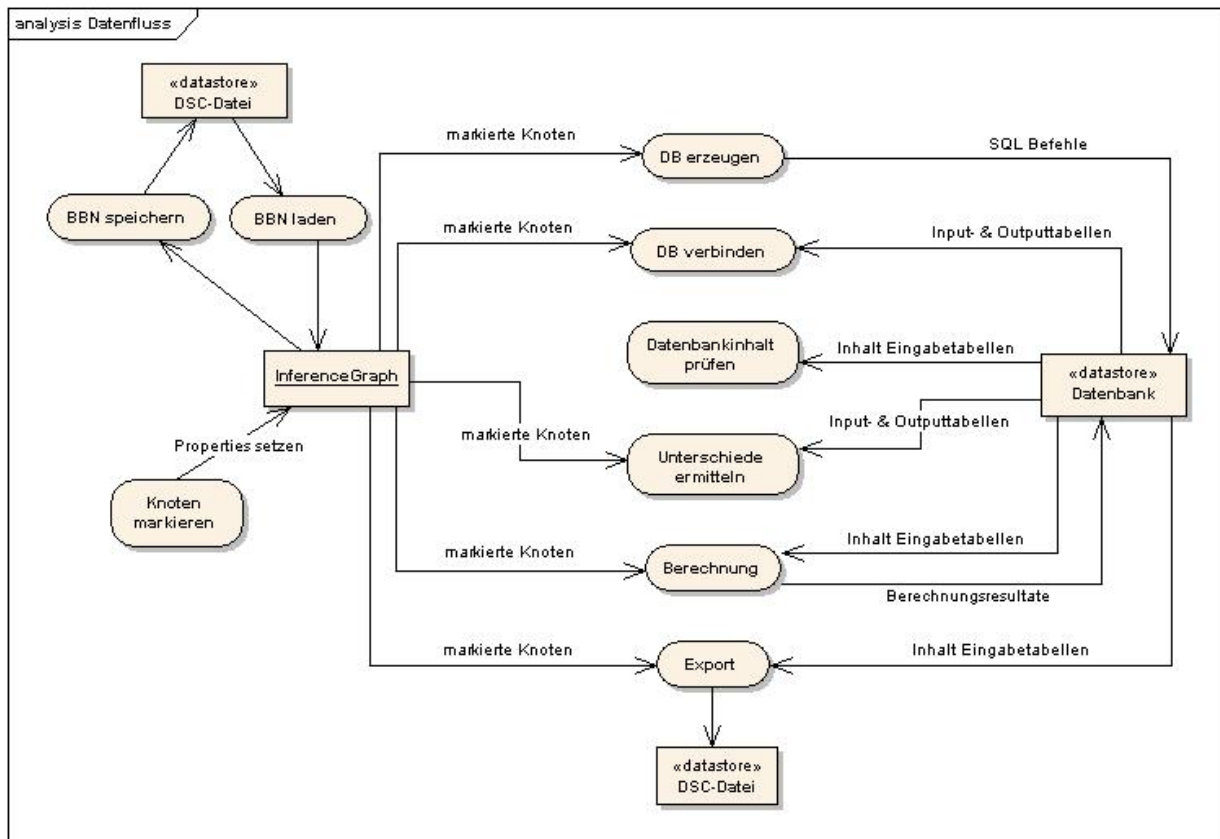


Abbildung 4: Datenflussdiagramm

### 3.4 Konventionen und Einschränkungen

Es bestehen einige Konventionen, damit das Datenbankmodul die Tabellen in der Datenbank findet und richtig ansprechen kann. Diese Vereinbarungen müssen für ein korrektes funktionieren der Software erfüllt sein. Über die Konfigurationsdatei lassen sich einige Punkte konfigurieren.

- Der Name einer Input-Tabelle beginnt mit dem Präfix „IN\_“ gefolgt von dem Knotennamen. Bei Output-Tabellen wird der Präfix „OUT\_“ verwendet. Die Präfixe lassen sich in der Konfigurationsdatei anpassen. Die Zuordnung von Knoten und Tabellen geschieht über den Namen. Daher muss dieser übereinstimmen. Falls nicht, wird die Verbindung mit der Datenbank fehlschlagen.
- Die Namen der Zustände eines Knotens entsprechen den Spaltenbezeichnungen einer Input- bzw. Output-Tabelle. Ebenfalls muss die Reihenfolge der Zustände mit der Anordnung in der Tabelle übereinstimmen.
- Die Eigenschaft, ob ein Knoten als Input- oder Output-Knoten verwendet werden soll, wird als Property abgespeichert. Der Name des Properties ist „DB\_Settings“. Diese Property darf die Werte „INPUT“, „OUTPUT“ oder „INPUT, OUTPUT“ enthalten. Das Property lässt sich auch über die MSBNx Software setzen, erfordert jedoch die Einhaltung oben genannter Schreibweisen.

Zudem existieren einige Einschränkungen, welche vorwiegend durch Designentscheidungen bedingt sind. Daher wird jeweils angegeben, wieso eine Einschränkung besteht.

- Input-Tabellen können nur zu äusseren Knoten (Knoten ohne Abhängigkeiten) erzeugt werden. Dies liegt an dem einfachen Aufbau der Input-Tabellen. Um auch innere Knoten als Input-Knoten definieren zu können, müssten (abhängig von den Elternknoten) sehr umfangreiche Wahrscheinlichkeitstabellen persistiert werden. Ein Datensatz wäre bei diesem Modell über mehrere Zeilen verteilt gespeichert. Dies erschwert die Verarbeitung und Eingabe von Daten.

Folgende Gründe sprechen für die vereinfachte Tabellenstruktur:

- Der Kunde benötigt diese Funktion nicht unbedingt
- Tabellenstruktur ist intuitiv verständlich
- Verarbeitung der Daten ist einfacher
- Formulare können einfacher erzeugt werden
- Die Verwendung eines Unterstrichs im Knotennamen führt zur Bildung einer Gruppe (siehe Bemerkung). Daher wird der Gebrauch von Unterstrichen in Knotennamen eingeschränkt. Möchte man also keine Gruppen, so muss man auf Unterstriche verzichten.

Diese Funktion wurde auf Wunsch des Auftraggebers realisiert. Die Gruppierung vereinfacht das Eintragen von Werten durch gruppenspezifische Abfragen und Formulare.

**Bemerkung:** Das BBN des Auftraggebers enthält Input-Knoten, die in Gruppen eingeteilt sind. Dabei werden zusammengehörende Knoten mit demselben Präfix gekennzeichnet. Ein Präfix wird über einen Unterstrich vom restlichen Namen separiert.

Beispiel: Beim Knoten „SIM\_SelEin\_5“ kennzeichnet „SIM“ die Gruppenzugehörigkeit.

## 4 Datenbankdesign

### 4.1 Statische Tabellen

Statische Tabellen sind unabhängig vom Bayesian-Belief-Netzwerk, welches für die Datenbankgenerierung benutzt wird. Diese Tabellen sind bereits in der leeren Datenbankdatei vorhanden und werden nicht generiert.

Eine Änderung ist sehr einfach möglich. Dazu muss die leere Datenbankdatei geöffnet und bearbeitet werden. Es ist nur darauf zu achten, dass bestehenden Spalten, die auch in der Konfigurationsdatei stehen, nicht gelöscht werden (In den untenstehenden Abbildungen mit einem Pfeil gekennzeichnet). Dies ist notwendig, weil AMRA-BBN auf diese Spalten zugreifen muss. Das Hinzufügen von neuen Spalten ist problemlos möglich. Falls eine Spalte umbenannt werden soll, muss das auch in der Konfigurationsdatei geschehen.

- **Piloten:** In dieser Tabelle werden die einzelnen Piloten abgelegt. Piloten werden über die „SPHAIR Nummer“ identifiziert. Diese Spalte ist daher der primäre Schlüssel. Alle weiteren Felder sind optional.

Piloten : Tabelle		
	Feldname	Felddatentyp
🔑	SPHAIR_Nr	Zahl
	FAI_Nr	Zahl
	Pilotenkurs	Zahl
	Name	Text
	Vorname	Text
	Geburtstag	Datum/Uhrzeit
	AHV	Text

Abbildung 5: Piloten Tabelle

- **Untersuchungen:** Ein Pilot kann mehrfach untersucht werden. Daher wird in dieser Tabelle einem Piloten eine Untersuchung zugeordnet. Der Primäre Schlüssel ist die Kombination der SPAIR Nummer und einer Untersuchungsnummer. Das Feld Untersuchung hat den Standardwert „1“. Der Standardwert erleichtert die Eingabe von Untersuchungen. Erst bei einer zweiten Untersuchung zu einem Piloten ist die Anpassung des Feldes notwendig. Die übrigen Felder sind optional.

Input- und Output-Tabellen besitzen ebenfalls eine Kombination aus SPHAIR- und Untersuchungsnummer, um Datensätze in den Tabellen einem Piloten zuordnen zu können.

Untersuchungen : Tabelle		
	Feldname	Felddatentyp
🔑	SPHAIR_Nr	Zahl
🔑	Untersuchung	Zahl
	Untersuchungsdatum	Datum/Uhrzeit
	Bemerkung	Text

Abbildung 6: Untersuchungen Tabelle

- **Berechnungen:** Berechnungen können mit den gleichen Daten mehrfach durchgeführt werden. Diese Tabelle hilft, Berechnungen zu unterscheiden und zu kennzeichnen. Der primäre Schlüssel ist die ID der Berechnung. Weiterhin werden das Datum der Berechnung sowie ein Benutzerkommentar gespeichert.

Die Berechnungsergebnisse in den Output-Tabellen verfügen ebenfalls über eine Berechnungsnummer, damit eine Zuordnung zwischen durchgeführter Berechnung und einem Resultat möglich ist.

Berechnungen : Tabelle		
	Feldname	Felddatentyp
PK	ID	AutoWert
	Datum	Datum/Uhrzeit
	Bemerkung	Text

Abbildung 7: Berechnungen Tabelle

## 4.2 Dynamische Tabellen

Dynamische Tabellen sind abhängig vom Bayesian-Belief-Netzwerk, welches für die Datenbankgenerierung benutzt wird. Sie werden bei der Erstellung der Datenbank neu erstellt.

Die Präfixe beider Tabellentypen lassen sich in der Konfigurationsdatei ändern.

- **Input-Tabellen:** Pro Input-Knoten wird eine Input-Tabelle erzeugt. Der Name von Input-Tabellen setzt sich aus dem Präfix „IN\_“ und dem Knotennamen zusammen. Eine Input-Tabelle hat stets die Spalten „SPHAIR\_Nr“ und „Untersuchung“. Diese beiden Spalten bilden den Primären Schlüssel und verknüpfen den Datensatz mit einem Piloten. Dazu kommt pro Zustand des Knotens eine weitere Spalte hinzu, die denselben Namen wie der Zustand hat. In diese Zustandsspalten können die Wahrscheinlichkeiten eingetragen werden.

Input-Tabellen werden mit folgendem SQL Befehl erzeugt (Beispiel für einen Knoten mit 3 Zuständen):

```
CREATE TABLE IN_NodeXY (
  SPHAIR_Nr Integer NOT NULL,
  Untersuchung Integer NOT NULL,
  [ValueCol1] DOUBLE,
  [ValueCol2] DOUBLE,
  [ValueCol3] DOUBLE,
  PRIMARY KEY (SPHAIR_Nr, Untersuchung));
```

- **Output-Tabellen:** Pro Output-Knoten wird eine Output-Tabelle erzeugt. Der Name von Output-Tabellen setzt sich aus dem Präfix „OUT\_“ und dem Knotennamen zusammen. Eine Output-Tabelle hat stets die Spalten „Berechnung“, „SPHAIR\_Nr“, und „Untersuchung“. Die drei Spalten bilden den Primärschlüssel der Tabellen. Dazu kommt pro Zustand des Knotens eine weitere Spalte hinzu, die denselben Namen wie der Zustand hat. In diesen Zustandsspalten werden die Ergebnisse einer Berechnung durch AMRA-BBN eingetragen.

Output-Tabellen werden mit folgendem SQL Befehl erzeugt (Beispiel für einen Knoten mit 3 Zuständen):

```
CREATE TABLE OUT_NodeXY (
  Berechnung Integer NOT NULL,
  SPHAIR_Nr Integer NOT NULL,
  Untersuchung Integer NOT NULL,
  [ValueCol1] DOUBLE,
  [ValueCol2] DOUBLE,
  [ValueCol3] DOUBLE,
  PRIMARY KEY (Berechnung, SPHAIR_Nr, Untersuchung));
```

- **Gruppen-Tabellen:** Zu jeder Gruppe wird eine Tabelle mit dem Präfix „GR\_“ gefolgt vom Gruppennamen erstellt. Diese Tabelle hat nur die Spalten „SPHAIR\_Nr“, und „Untersuchung“, welche den Primärschlüssel bilden. Die Tabelle wird benötigt, um die Abfrage einer ganzen Gruppe zu realisieren.

Gruppen-Tabellen werden mit folgendem SQL Befehl erzeugt:

```
CREATE TABLE GR_XYZ (
  SPHAIR_Nr Integer NOT NULL,
  Untersuchung Integer NOT NULL);
```

**Bemerkung:** Hat ein Input-Knoten keinen Präfix, wird er auch keiner Gruppe zugeordnet. Wenn keine Gruppen gebildet werden können, so gibt es auch keine Gruppen-Tabellen.

**Bemerkung:** Die Namen der Zustände können Sonderzeichen wie das Leer- oder das Prozentzeichen enthalten. Die eckigen Klammern in den obigen SQL-Befehl ermöglichen, dass die Abfrage korrekt ausgeführt werden kann.

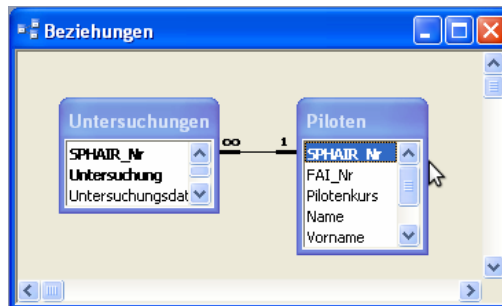
#### 4.2.1 Verknüpfung mit dem Bayesian-Belief-Netzwerk

Die Verknüpfung der dynamischen Tabellen mit den Knoten im BBN geschieht über die Namen. Der Name einer Tabelle beginnt mit dem Input- oder Output-Präfix, gefolgt von dem Knotennamen. Wird ein Knoten umbenannt, muss auch der Tabellename angepasst werden.

Dasselbe gilt für die Zustände eines Knoten. Wird ein neuer Zustand hinzugefügt, ein bestehender Zustand entfernt oder ein Zustand umbenannt, muss dies in der Datenbank ebenfalls angepasst werden. Ansonsten kann sich AMRA-BBN nicht mit der Datenbank verbinden.

### 4.3 Beziehungen

Folgende Grafik stellt die Beziehungen zwischen den Tabellen „Piloten“ und „Untersuchungen“ dar. Es handelt sich hierbei um eine 1:n Beziehung, da ein Pilot mehrere Untersuchungen haben kann.



**Abbildung 8: Beziehung zwischen Pilot und Untersuchungen**

Die übrigen Beziehungen werden in Access nicht definiert und somit auch nicht dargestellt. Grund hierfür ist, dass Access höchstens 32 Indizes für eine Tabelle unterstützt. Würde man also die Input und Output-Tabellen mit der Tabelle Untersuchungen oder den Gruppen-Tabellen referenzieren, wäre dieser Grenzwert schnell erreicht. Die Erstellung weiterer Referenzen wird dann verweigert.

Die Input-, Output-, Gruppen-Tabellen und die Untersuchungstabelle sind über die Kombination aus SPHAIR\_Nr und Untersuchungsnummer miteinander verknüpft.

### 4.4 Views (Abfragen)

Views (in Access als Abfragen bezeichnet) helfen auf die Daten in den Tabellen zuzugreifen. Zudem lassen sich aus Views einfach Formulare erzeugen. Daher werden bereits einige Views bereitgestellt. Der Anwender kann beliebig eigene zusätzliche Views erstellen.

Views werden zur besseren Unterscheidung mit dem Präfix „V\_“ gekennzeichnet. Das Präfix lässt sich in der Konfigurationsdatei anpassen.

AMRA-BBN bedient sich ebenfalls der View „V\_Untersuchungen“, um einfach Daten abzufragen. Daher gilt hier das gleiche wie bei den Tabellen. Bereits existierende Views dürfen nicht gelöscht werden. Erweiterungen und Änderungen sind in Ordnung, solange die Konfigurationsdatei angepasst wird.

- **V\_Untersuchungen:** Diese View verknüpft die Tabellen Piloten und Untersuchungen über die SPHAIR\_Nr. Das Resultat ist eine Auflistung aller Piloten und ihrer Untersuchungen. Diese View ist bereits in der leeren Datenbank vorhanden.

Der dazugehörige SQL-Befehl sieht folgendermassen aus:

```
SELECT Untersuchungen.SPHAIR_Nr, Piloten.Name, Piloten.Vorname,
```

```
Piloten.Pilotenkurs, Untersuchungen.Untersuchung,
Untersuchungen.Untersuchungsdatum, Untersuchungen.Bemerkung
FROM Piloten INNER JOIN Untersuchungen
ON Piloten.SPHAIR_Nr = Untersuchungen.SPHAIR_Nr;
```

- **V\_Berechnungsergebnisse:** Um alle Berechnungsergebnisse auszugeben wird diese View bereitgestellt. Sie verknüpft alle Output-Tabellen und die Berechnungstabelle über die drei Felder „Berechnung“, „SPHAIR\_Nr“ und „Untersuchung“. Diese View wird bei der Erzeugung der Datenbank generiert.

Die Tabellen sind in Reihe miteinander verknüpft. Dies ist aus Performance-Gründen so gelöst worden. Wenn alle Tabellen mit einer zentralen Tabelle verknüpft werden, dauert das Öffnen einer Abfrage ab etwa 15 Tabellen sehr lange. Je mehr Tabellen es werden, desto mehr Zeit benötigt Access, um die Tabellen zu verknüpfen.

Die View wird mit folgendem SQL-Befehl erzeugt (Beispiel für drei Output-Tabellen mit jeweils zwei Zuständen):

```
SELECT OUT_K1.Berechnung, OUT_K1.SPHAIR_Nr, OUT_K1.Untersuchung,
OUT_K1.Col1 AS [BeschrK1_Col1], OUT_K1.Col2 AS [BeschrK1_Col2],
OUT_K2.Col1 AS [BeschrK2_Col1], OUT_K2.Col2 AS [BeschrK2_Col2],
OUT_K3.Col1 AS [BeschrK3_Col1], OUT_K3.Col2 AS [BeschrK3_Col2],
FROM (OUT_K1 INNER JOIN OUT_K2 ON (OUT_K1.Berechnung = OUT_K2.Berechnung)
AND (OUT_K1.SPHAIR_Nr = OUT_K2.SPHAIR_Nr) AND (OUT_K1.Untersuchung =
OUT_K2.Untersuchung)) INNER JOIN OUT_K3 ON (OUT_K2.Berechnung =
OUT_K3.Berechnung) AND (OUT_K2.SPHAIR_Nr = OUT_K3.SPHAIR_Nr) AND
(OUT_K2.Untersuchung = OUT_K3.Untersuchung);
```

- **Gruppen-Views:** Zu jeder Gruppe wird eine View generiert, welche die gesamte Gruppe darstellt. Dazu werden alle Input-Tabellen, die zu einer Gruppe gehören, mit der Gruppentabelle verknüpft. Neben allen Wertspalten werden die Spalten „SPHAIR\_Nr“ und „Untersuchung“ angezeigt. Die Gruppen Views erleichtern die Erstellung von Formularen.

Diese Views haben eine zentrale Tabelle (die Gruppentabelle), mit der die übrigen Input-Tabellen verknüpft sind. Hier konnte nicht die verkettete Struktur wie bei der Berechnungsergebnisse-View verwendet werden (Begründung weiter unten). Daher dauert bei grossen Gruppen (>15 Tabellen) das Öffnen der View einige Zeit lang.

Die Views werden mit folgendem SQL-Befehl erzeugt:

```
SELECT GR_XY.SPHAIR_Nr, GR_XY.Untersuchung,
IN_XY_K1.Col1 AS BeschrK1_Col1, IN_XY_K1.Col2 AS BeschrK1_Col2,
IN_XY_K2.Col1 AS BeschrK2_Col1, IN_XY_K2.Col2 AS BeschrK2_Col2,
FROM (GR_XY LEFT JOIN IN_XY_K1 ON (GR_XY.SPHAIR_Nr = IN_XY_K1.SPHAIR_Nr)
AND (GR_XY.Untersuchung = IN_XY_K1.Untersuchung)) LEFT JOIN IN_XY_K2 ON
(GR_XY.SPHAIR_Nr = IN_XY_K2.SPHAIR_Nr) AND (GR_XY.Untersuchung =
IN_XY_K2.Untersuchung);
```

**Bemerkung:** Hat ein Input-Knoten keinen Präfix, wird er auch keiner Gruppe zugeordnet. Wenn keine Gruppen gebildet werden können, so gibt es auch keine Gruppen-Views.

**Begründung zentrale Verknüpfung:** Bei einer verketteten Struktur zeigt die View nur ein Resultat an, wenn in allen Tabellen der gleiche Eintrag vorhanden ist. Dies würde für den Benutzer bedeuten, dass er in alle Input-Tabellen, die zu der Gruppe gehören, etwas eintragen muss. Vergisst er eine Tabelle oder hat er keine Werte, die er eintragen kann (die Kette wird unterbrochen), so wird der komplette Datensatz nicht in der Abfrage angezeigt. Dies erweckt den Eindruck, dass die Daten gar nicht gespeichert wurden. Dabei steckt das Problem lediglich in der Abfrage.

## 4.5 Datenbank bereitstellen

Man kann eine leere Access Datenbank nicht direkt aus Java heraus erzeugen. Andere Datenbanken unterstützen die Möglichkeit, beim Holen der Verbindung das Property "create=true"

anzugeben. Falls die Datenbank nicht existiert, wird eine neue erzeugt. Access versteht dieses Property nicht.

Daher bleiben zwei Möglichkeiten, dem Benutzer eine leere Datenbank bereitzustellen:

- Java setzt einen Kommandozeilenbefehl an das Betriebssystem ab (Process p = Runtime.getRuntime().exec(cmd);)
- Eine leere Datenbank manuell im Ressource-Verzeichnis erstellen und diese bei Gebrauch an die gewünschte Stelle kopieren.

AMRA-BBN verwendet die zweite Möglichkeit, da diese nicht so sehr vom Betriebssystem abhängig ist. Zudem können an der leeren Datenbank einfacher Einstellungen verändert werden und statische Tabellen lassen sich bereits in der leeren Datenbank erstellen.

## 5 Umsetzung AMRA-BBN

### 5.1 Allgemein

AMRA-BBN arbeitet intern mit Untersuchungen. Mit Untersuchung ist ein Menge von Eingabewerten gemeint (maximal eine Zeile pro Input-Tabelle), welche in der Datenbank dieselbe SPHAIR\_Nr und Untersuchungsnummer besitzt.

Die Kombination aus SPHAIR\_Nr und Untersuchungsnummer, die in der Datenbank ein zusammengesetzter Primärschlüssel ist, wird in AMRA-BBN als durch einen Bindestrich getrennter String dargestellt.

Beispiel: Hat ein Pilot die SPHAIR\_Nr „1122“ und wird das erste Mal untersucht (UntersuchungsNr: 1) ergibt das die Untersuchung „1122-1“.

### 5.2 Softwaredesign

Die Erweiterung der FAI-BBN Software wurde in verschiedene Schichten aufgeteilt. Die folgende Grafik zeigt die Schichten und ihre Anordnung. Weiter unten wird dann die Rolle der einzelnen Schichten erläutert. Implementierungsdetails zu den einzelnen Schichten werden in den folgenden Abschnitten näher beschrieben. An dieser Stelle soll erst einmal ein grober Überblick vermittelt werden.

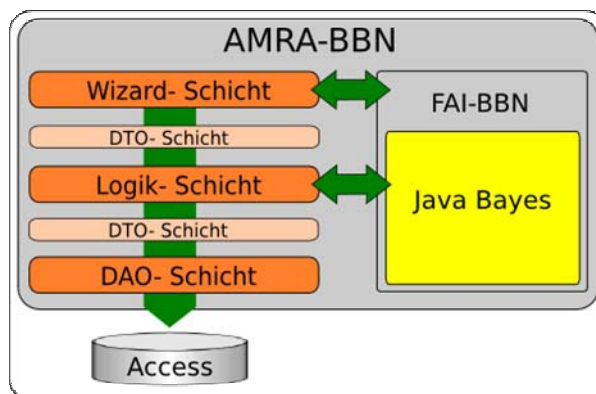


Abbildung 9: AMRA-BBN Schichtenmodell

Das Schichtenmodell wurde wegen der schwachen Kopplung gewählt. Änderungen innerhalb einer Schicht lassen sich problemlos durchführen. Lediglich wenn die Schnittstellen angepasst werden, wirkt sich das auf die benachbarte Schicht aus.

- **DAO- Schicht:** Die „Data Access Object“-Schicht befindet sich unmittelbar über der Datenbank. Alle Anfragen an die Datenbank laufen über diese Schicht. Klassen höherer Schichten dürfen nicht direkt auf die Datenbank zugreifen.
- **DTO- Schicht:** Viele Informationen lassen sich in Form von Objekten besser bzw. einfacher an andere Methoden übergeben und verarbeiten. In den beiden „Data Transfer Object“-Schichten befinden sich Klassen, deren Objekte zum Transfer zwischen der Logik und der Datenbank bzw. zwischen der Logik und der GUI dienen. Diese Objekte beinhalten in der Regel kaum bis keine Logik. Diese Schicht ist an einigen Stellen löchrig, da nicht für alle Informationen ein explizites Objekt notwendig ist.
- **Logik- Schicht:** Hier wird die eigentliche Arbeit der Software erledigt bzw. koordiniert. Die Datenbankinformationen in Form von DTO-Objekten und das geladene Bayesian-Belief-Netzwerk werden miteinander kombiniert und beispielsweise Berechnungen auf dem Netzwerk gestartet. Die Verbindung mit FAI-BBN (grüner Pfeil) symbolisiert die Benutzung der Objekte, welche das Netzwerk darstellen.
- **Wizard- Schicht:** Diese Schicht kommuniziert mit dem Benutzer. Sie ruft Methoden in der

Logik- Schicht auf und übergibt Eingaben des Benutzers an diese. Resultate der aufgerufenen Operationen werden wieder dem Benutzer präsentiert. Der grüne Pfeil stellt die Integration der Wizard-Schicht in die bestehende grafische Oberfläche dar.

## 5.3 Änderungen am bestehenden Code

Der Programmcode des FAI-BBN Projektes musste an einigen Stellen erweitert werden. Diese Stellen sollen in diesem Abschnitt beschrieben werden. Da der restliche Programmcode unverändert ist, kann für weitergehende Informationen die Dokumentation des ursprünglichen Projektes konsultiert werden.

### 5.3.1 Anpassungen an der GUI

- **Mainframe (ch.hsr.faibbn.gui.MainFrame.java)**

Das Mainframe wurde um einen weiteren Menüeintrag „Datenbank“ erweitert. Der Menüeintrag und seine Untereinträge sind nur verfügbar, wenn sie ausgewählt werden können. Ist kein Bayesian-Belief-Netzwerk geladen, steht das gesamte Menü nicht zur Auswahl. Ist keine Datenbank verbunden, kann die Berechnungs- und Exportfunktion auch nicht ausgewählt werden. Die Untermenüs starten jeweils einen Wizard.



Abbildung 10: Datenbankmenü

Weiterhin wird unten in der grafischen Oberfläche die momentan verbundene Datenbank angezeigt. Die Statuszeile ist ebenfalls neu hinzugekommen.

Die Erweiterungen sind jeweils mit einem kleinen Kommentar im Quellcode gekennzeichnet.

- **NetworkComponent (ch.hsr.faibbn.gui.NetworkComponent.java)**

Um Input- und Output-Knoten farblich hervorzuheben, wurde die paint()-Methode angepasst. Die Farben können in der Konfigurationsdatei eingestellt werden.

Um die Properties überhaupt setzen zu können, musste die showPopupMenu()-Methode dieser Klasse angepasst werden. Es wurde ein zusätzliches Untermenü definiert, welches zwei JCheckBoxMenuItem's besitzt. Wird eine Property in diesem Menü gesetzt, so wird es automatisch auch als Eigenschaft des Knotens (variable-Property) übernommen. Dies hat zur Folge, dass die Eigenschaft auch beim Abspeichern des DSC-Files persistiert wird.

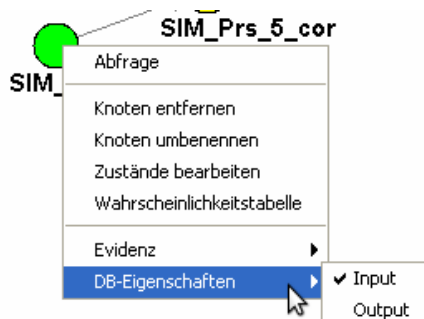


Abbildung 11: Knoten markieren

### 5.3.2 Weitere Anpassungen

- **DB\_Settings Property in den DSC-Parser einfügen (ch.hsr.faibbn.pd.parsers.DSC)**

Über das DB\_Settings Property wird einem Knoten mitgeteilt, ob er ein Input- oder Output-

Knoten ist. Diese Information wird als Property im Bayesian-Belief-Netzwerk hinterlegt. Damit der Parser beim Einlesen eines dsc-Files nicht über dieses neue Property stolpert und den Lesevorgang abbricht, muss dem Parser das neue Property bekannt gemacht werden.

Wie das funktioniert wird als HowTo im Anhang beschrieben.

- **Speicherfunktion angepasst**

Das Abspeichern der Knoteneigenschaft (DB\_Settings) geschieht automatisch. Damit das Programm MSBNx die neue Knoteneigenschaft akzeptiert, muss diese noch bei den Eigenschaften des Netzwerkes definiert werden.

```
network "Netzwerkname" {}
properties{
    ...
    type DB_Settings = string;
    ...
}
```

Dazu wurde in der Klasse „BayesNet“ die Methode „save\_dsc“ angepasst. Die genauen Anpassungen sind im Quellcode mit Kommentaren versehen.

- **Buildscript (build.xml)**

Im Ant Buildscript wurde das neue Benutzerhandbuch hinzugefügt sowie ein kleiner Fehler behoben (das Compile-Target hatte eine falsche Abhängigkeit). Weiterhin wurde die neue Version 1.1 sowie der neue Projektname eingetragen. Ansonsten waren keine Änderungen notwendig, da das Script bereits alle wichtigen Targets enthielt. Besonders bequem ist die automatische Generierung von Windows-Installern.

Damit die Installer mit den Änderungen am Programmcode und im Buildscript zurecht kommen, waren Anpassungen in der Datei „run.bat“ (neue Ressourcen und erweiterte Paketstruktur einbinden) notwendig.

- **Logger (ch.hsr.faibbn.util.Recorder.java)**

Der bereits vorhandene Log-Mechanismus wurde ebenfalls weiterverwendet. Ein Log-Eintrag wird über den folgenden Aufruf erstellt: Recorder.rec.debug|info|...(...). Die Log-Datei wird über ein Properties-File (log4j.properties) konfiguriert. Der Log-Level lässt sich in der Konfigurationsdatei einfach einstellen.

- **Konfigurationsdatei (res/properties/faibbn.properties)**

Die bestehende Konfigurationsdatei wurde erweitert. Hinzugekommen ist die Konfiguration von Tabellennamen und –spalten, welche in der Datenbank benutzt werden. Ebenfalls lassen sich hier die Präfixe und beispielsweise die Farben der Knoten einstellen.

- **I18N (res/properties/text.properties)**

Ebenfalls wird die Internationalisierung weiter verwendet. Alle Texte, die der Benutzer sieht, lassen sich in dieser Datei anpassen.

## 5.4 Neuer Code

### 5.4.1 DAO- Schicht

Package: ch.fhnw.amrabbn.dao

Für jede Datenbanktabelle existiert ein DAO-Interface, welches die benötigten Methoden spezifiziert. Dieses Interface wird in einer weiteren Klasse implementiert.

Die DAO Klassen benutzen das „SimpleJdbcTemplate“, um mit der Datenbank zu kommunizieren. Hierbei handelt es sich um einen JDBC-Wrapper. Der Vorteil für den Programmierer liegt darin, dass sich der JDBC-Wrapper um den gesamten Verbindungsaufbau kümmert. Die Verbindung wird

mit Hilfe eines „DataSource“-Objektes automatisch geöffnet und am Ende geschlossen. Die „DataSource“ abstrahiert die Datenbank bzw. die Datenbankverbindung. SQL-Befehle können einfacher ausgeführt und die Resultate bequemer durchsucht werden. Zudem werden bei Fehlern sinnvolle Exceptions geworfen. Im Anhang befinden sich zwei kleine Beispiele, welche die Vorzüge des „SimpleJdbcTemplate“ aufzeigen. Das „SimpleJdbcTemplate“ wird im Spring-Framework bereitgestellt. Daher besteht eine Abhängigkeit zu dieser Bibliothek.

In den Klassen werden viele String-Konkatenationen durchgeführt, um die SQL-Befehle zusammzusetzen. Aus Darstellungs- und vor allem aus Performancegründen wird in den meisten Fällen ein „StringBuilder“ benutzt.

Die DAO-Schicht wirft keine Exceptions an höhere Schichten weiter. Falls ein Fehler z.B. beim Ausführen eines Queries auftritt, wird dieser geloggt und das Fehlverhalten über den Rückgabeparameter signalisiert.

Klasse	Beschreibung
<b>CalculationTbIDAOImpl</b>	<ul style="list-style-type: none"> <li>• Implementiert CalculationTbIDAO</li> <li>• - Schreibt Datensätze in die Berechnungen Tabelle und liefert den verwendeten Primärschlüssel zurück. Dieser wird in den Outputtabellen eingetragen</li> </ul>
<b>CheckupDAOImpl</b>	<ul style="list-style-type: none"> <li>• Implementiert CheckupDAO</li> <li>• Liest den Inhalt der V_Untersuchungen View. Die Resultate werden in den Wizards Berechnung und Export verwendet.</li> </ul>
<b>DatabaseDAOImpl</b>	<ul style="list-style-type: none"> <li>• Implementiert DatabaseDAO</li> <li>• Erzeugt Input- und Output-Tabellen</li> <li>• Erzeugt Views</li> <li>• Liest verfügbare Tabellen aus</li> </ul>
<b>InputTbIDAOImpl</b>	<ul style="list-style-type: none"> <li>• Implementiert InputTbIDAO</li> <li>• Liest Inhalte der Inputtabellen und gibt diese an die Logik- Schicht</li> </ul>
<b>OutputTbIDAOImpl</b>	<ul style="list-style-type: none"> <li>• Implementiert OutputTbIDAO</li> <li>• Speichert Resultate in den zugehörigen Outputtabellen</li> </ul>

### 5.4.2 DTO- Schicht

Package: ch.fhnw.amrabbn.model

Bis auf NodeConfiguration sind alle Klassen dieser Schicht reine Datencontainer, die Informationen von einer Schicht an eine andere geben.

NodeConfiguration beinhaltet zusätzlich einige Methoden um herauszufinden, ob die aus der Datenbank gelesenen Werte problemlos auf das Bayesian-Belief-Netzwerk übertragen werden können. Bei negativen Wahrscheinlichkeitswerten, Wahrscheinlichkeitswerten die grösser als 1 sind und falls die Summe der Wahrscheinlichkeiten nicht 1 oder 0 (nicht eingetragene Werte) ist, wird die Knotenkonfiguration als ungültig gekennzeichnet.

Klasse	Beschreibung
<b>PilotCheckups</b>	Enthält eine Untersuchung, welche aus SPHAIR_Nr, Untersuchungsnummer, sowie Namen und Kurs des Piloten besteht. Diese Informationen werden im Berechnungs- und Export- Wizard zur Auswahl des Piloten angezeigt.
<b>OperationResult</b>	Beinhaltet eine Liste mit Fehler- und Erfolgsmeldungen. Die Liste wird während dem Durchführen einer Operation gefüllt und am Ende dem Benutzer angezeigt.

<b>NodeConfiguration</b>	Repräsentiert eine Zeile einer Input-Tabelle mit der SPHAIR- und Untersuchungsnummer sowie den Wahrscheinlichkeitswerten.
<b>Checkup</b>	Steht für eine Untersuchung und fasst alle NodeConfigurations mit derselben SPHAIR- und Untersuchungsnummer zusammen.
<b>DbNodeResult</b>	Berechnungsergebnis eines einzelnen Output-Knotens.
<b>DbInferenceResult</b>	Zusammenfassung aller DbNodeResults einer Berechnung.

### 5.4.3 Logik- Schicht

Package: ch.fhnw.amrabbn.logic

Die einzelnen Funktionen werden in separaten Klassen nach ihren Aufgaben gegliedert. Speziell ist die Klasse BBNRepresentation, welche Funktionen rund um das BBN zusammenfasst, die von den anderen Logik-Klassen aufgerufen werden.

- **BBNRepresentation – Bayesian-Belief-Netzwerk Wrapper**

BBNRepresentation enthält einen InferenceGraph und bietet zusätzliche Methoden an, welche die Tabellenerstellung oder die Berechnung unterstützen. Folgende wichtige Methoden können über diese Klasse aufgerufen werden:

- Input- und Output-Knoten des Bayesian-Belief-Netzwerkes holen
- Backup der Wahrscheinlichkeitswerte von Input-Knoten erstellen und zurückspielen
- Wahrscheinlichkeitswerte einer Untersuchung auf das Bayesian-Belief-Netzwerk mappen
- Durchführen einer Berechnung

- **Fingerprint – Datenbank mit Bayesian-Belief-Netzwerk vergleichen**

Eine Instanz dieser Klasse wird erzeugt, um ein geladenes Bayesian-Belief-Netzwerk mit einer verbundenen Datenbank zu vergleichen. Er werden zwei Maps erzeugt. Eine für das Modell und die andere für die Datenbank. Als Schlüssel wird der Name der Input- und Output-Knoten verwendet. Der Wert ist ein kommaseparierter String aller Zustände des entsprechenden Knotens. Respektive wird bei der Map für die Datenbank der Name der Tabelle (ohne Präfix) als Schlüssel und die kommaseparierten Spaltenbezeichnungen als Wert benutzt.

Der Hashwert beider Maps wird verglichen. Sind die Hashwerte nicht identisch, wurde das Bayesian-Belief-Netzwerk seit der Datenbankgenerierung verändert und die Verbindung mit der Datenbank darf nicht zustande kommen.

- **TableCreator - Datenbankstruktur generieren**

Zu allen Input- und Output-Knoten werden Datenbanktabellen erstellt. Bei den Input-Knoten findet erst einmal eine Überprüfung statt, ob dieser Knoten keine Abhängigkeiten hat. Falls doch, wird eine InvalidInputNodeException geworfen.

Anschliessend werden, aufbauend auf den Knotentabellen, die Views generiert. Die View „V\_Berechnungsergebnisse“ verknüpft alle Output-Tabellen miteinander. Input-Tabellen werden erst in Gruppen aufgeteilt und anschliessend alle Tabellen einer Gruppe verknüpft.

- **Validation – Datenbankeinträge überprüfen**

Die Einträge aller Input-Tabellen werden auf korrekte Werte überprüft. Bei ungültigen Einträgen wird der Benutzer informiert, wo (Welche Tabelle und Untersuchung) sich der fehlerhafte Eintrag befindet.

Die Validierung ist freiwillig und kann beim Verbinden mit der Datenbank auch übersprungen werden. Ungültige Einträge werden beim Export oder bei der Berechnung allerdings nicht berücksichtigt.

- **Inference – Berechnung ausführen**

Die Klasse führt Berechnungen auf einer Menge von Untersuchungen durch. Dazu werden zu jeder Untersuchung die eingetragenen Werte aus der Datenbank gelesen, auf das Bayesian-Belief-Netzwerk gemappt und anschliessend auf allen Outputknoten die Berechnung gestartet. Die Resultate werden gesammelt und am Ende der Berechnung in die Output-Tabellen gespeichert.

- **Export – Datenbankinhalt in Form von DSC-Dateien exportieren**

Der Export funktioniert ähnlich wie die Berechnung. Einziger Unterschied ist, dass nach dem Mapping keine Berechnung durchgeführt wird, sondern das aktuelle BBN im spezifizierten Verzeichnis als DSC-Datei gespeichert wird.

- **Compatibility – Unterschiede zwischen BBN und Datenbank ermitteln**

Wenn festgestellt wurde, dass das BBN und die Datenbank nicht zueinander passen, werden die Klassen des Compatibility-Unterpackages (`ch.fhnw.amrabbn.logic.compatibility`) aktiv.

Die Klasse `CompatibilityCoordinator` steuert die beiden verfügbaren Tests. Erst wird geprüft, ob die Input- und Output-Knoten im BBN mit den Datenbanktabellen übereinstimmen (`NodeCompatibility`). Falls nicht, wird dem Benutzer eine Liste mit Problemen dargestellt. Ist der erste Test ohne festgestellte Probleme durchgelaufen, werden die Zustände der Input- und Output-Knoten mit den Spaltennamen in der Datenbank verglichen. Auch hier wird eine Liste mit Problemen zurückgegeben, wenn Unstimmigkeiten gefunden wurden.

Neue Tests können einfach hinzugefügt werden. Sie müssen lediglich das Interface `Compatibility` implementieren und im Koordinator eingebunden werden.

Es war beabsichtig, dass der Benutzer die Möglichkeit erhält festgestellte Probleme als „zu lösen“ zu kennzeichnen. AMRA-BBN hätte anschliessend versucht, die Datenbank mit dem BBN zu synchronisieren. Allerdings konnte diese Funktion wegen Zeitmangel nicht implementiert werden. Die Methode `resolveProblems()` ist deswegen nicht implementiert und wurde als TODO gekennzeichnet.

#### 5.4.4 Wizard- Schicht

Die Wizard-Schicht ist die oberste Schicht in der Schichten-Hierarchie. Sie bildet die Schnittstelle zwischen dem Benutzer und dem Datenbank-Modul.

Zur Erstellung der Wizards wurde das Framework von `wizard.spi.netbeans.org` verwendet. Diese Wizards sind so aufgebaut, dass es einen Hauptwizard gibt, welcher ein Array mit allen Wizard-Pages besitzt. Diese Pages werden in der Reihenfolge, wie sie im Array stehen, angezeigt. Alle Eingabewerte werden automatisch im Hintergrund in einer Map abgelegt. Der Key ist dabei der Name der Komponente, in welcher der Wert eingegeben wird. Wenn man beispielsweise im Textfeld „txtName“ den Wert „Mein Vorname“ einträgt, ist dieser später über den Key „txtName“ verfügbar. Die Wizard-Pages bieten weiterhin eine Möglichkeit, den Input zu validieren und den „Next“-Button erst bei einem gültigen Input zu aktivieren. Die Klasse `ch.fhnw.amrabbn.gui.wizard.InputValidator` ist für diese Input-Validierungen zuständig.

Des Weiteren muss eine (Sub-)Klasse von „DeferredResult“ angegeben werden. Diese ist am Ende, nach Betätigen des „Finish“-Buttons, für das Ausführen der entsprechenden Operationen zuständig. Danach wird eine Summary-Page präsentiert, die den Benutzer über Erfolg oder Fehlschlag informiert. Für diese Summary-Page ist die Klasse `ch.fhnw.amrabbn.gui.wizard.SummaryPage` verantwortlich.

Für alle vier Wizards der Software wurde ein eigenes Unterpackage im Package `ch.fhnw.amrabbn.gui.wizard` erstellt. All diese enthalten ein Wizard, welcher eine Unterklasse der abstrakten Klasse `ch.fhnw.amrabbn.gui.wizard.BaseWizard` ist. In Base-Wizard wurden die grundlegenden Einstellungen vorgenommen. In den Unterklassen muss dann nur noch der Konstruktor überschrieben werden. Auch befinden sich in diesen Unterpackages alle WizardPages und ein `DeferredResult`.

Das GUI dieser Wizards wurde mit dem Netbeans GUI-Builder erstellt und anschliessend in die

entsprechende Klasse kopiert. Es handelt sich hierbei um automatisch erzeugten Code. Dieser ist zwar nicht besonders gut lesbar, lässt sich jedoch schnell erstellen und funktioniert sehr zuverlässig.

#### 5.4.5 ConnectedDb Klasse

Eine besondere Bedeutung hat die Klasse ConnectedDB. Diese Klasse repräsentiert die Verbindung zur Datenbank. Über sie können Klassen der Logik- Schicht auf DAO- Instanzen zugreifen. Daher wurde ConnectedDB als Singleton implementiert.

Sie wird erzeugt, sobald der Benutzer eine gültige Datenbank angibt und existiert solange, wie die Verbindung mit dieser besteht. Wichtigstes Attribut ist das DataSource Objekt. Es wird bei der Instanziierung erzeugt und bei der Erzeugung von DAO-Instanzen übergeben. Das SimpleJdbcTemplate der DAO- Klassen benötigt eine DataSource, um zu wissen, auf welche Datenbank die Abfragen ausgeführt werden sollen.

#### 5.4.6 DBModule Klasse

Das DBModule ist eine zentrale Stelle in der Applikation. Sie stellt einen Controller dar, welcher die Klassen der Logik- Schicht benutzt, um eine gewünschte Operation auszuführen und das Resultat der Operation wieder an den Aufrufer zurückreicht. Sie ist als Singleton implementiert, damit die Wizards die einzelnen Funktionen einfach aufrufen können. Die grafische Schicht ist somit sehr gut von der Funktionalität entkoppelt und könnte sehr einfach ausgetauscht werden.

Bei der Instanziierung benötigt die Klasse ein InferenceGraph Objekt, welches das geladene Bayesian-Belief-Netzwerk repräsentiert. Falls beim Instanzieren kein InferenceGraph übergeben und beispielsweise die „CreateDb“-Methode aufgerufen wird, kommt es zu einer IllegalStateException. Der InferenceGraph wird in ein eigenes Objekt (BBNRepresentation) gewrappt, welches erweiterte Methoden bereitstellt.

Einige Methoden dürfen erst ausgeführt werden, wenn eine Datenbankverbindung besteht. Die Verbindung wird als Objekt der Klasse ConnectedDB dargestellt. Ist dieses Objekt nicht gesetzt und es wird die Export- oder Berechnungsfunktion aufgerufen, kommt es ebenfalls zu einer IllegalArgumentException.

Die grafische Oberfläche muss sicherstellen, dass die bereitgestellten Funktionen nur bei Erfüllung der Abhängigkeiten ausführbar sind. Hierzu existieren weitere Methoden, welche beispielsweise darüber informieren, ob eine Datenbankverbindung besteht.

#### 5.4.7 Hilfsklassen

- **StringDefs (ch.fhnw.amrabbn.StringDefs.java)**

Hierbei handelt es sich um eine Hilfsklasse, welche alle Strings enthält, die in der Logik benutzt werden. Beispielsweise sind hier die Bezeichnungen der Tabellen und Spalten enthalten oder es wird definiert, welche Datentypen und Beschränkungen eine Spalte haben soll.

Die Tabellen- und Spaltennamen werden aus der Konfigurationsdatei (amrabbn.properties) geladen. Datentypen und Beschränkungen sind fest in der Software codiert und können nicht um konfiguriert werden.

Der Vorteil an der StringDefs-Klasse besteht darin, dass beim ersten Zugriff alle Informationen aus der Konfigurationsdatei geladen werden. Fehlt in der Konfigurationsdatei ein Property, wird eine Exception geworfen. Dadurch wird bei der ersten Benutzung des Datenbankmoduls sichergestellt, dass alle notwendigen Informationen in der Konfigurationsdatei spezifiziert sind. Weiterhin müssen oft benutzte Definitionen nicht mehrmals aus der Konfigurationsdatei gelesen werden.

Nachteil bei der ganzen Sache ist, dass alle Strings geladen werden, auch wenn sie beispielsweise gar nicht benötigt werden.

- **I18N (ch.fhnw.amrabbn.I18N.java)**

Diese Klasse hat denselben Existenzgrund wie die Klasse StringDefs. Der Fokus liegt hier aber auf der Internationalisierung.

- **DBUtil (ch.fhnw.amrabbn.util.DBUtil.java)**

Kopiert die leere Datenbank in ein beliebiges Verzeichnis.

## 5.5 Probleme

**JDBC - Access:** Nachfolgend sind einige Hindernisse aufgelistet, die bei der Arbeit mit JDBC und Access aufgetreten sind.

- In Microsoft Access gibt es die Möglichkeit einer Spalte eine Beschreibung zu geben. Die Beschreibung wird anstelle des Spaltennamens angezeigt. Mit JDBC kann die Beschreibung zwar aus dem ResultSet herausgelesen, jedoch nicht gesetzt werden.
- In SQL gibt es die Anweisung DEFAULT, welche einer Spalte einen Standardwert zuweist. Access(siehe Bemerkung) sowie JDBC unterstützen beiden diese Anweisung. Trotzdem war es nicht möglich, ein Query, welches den DEFAULT Parameter enthielt, über JDBC auszuführen. Die Fehlersuche führte auf zahlreiche Internetseiten, die dasselbe Problem meldeten, jedoch ohne eine Lösung zu kennen.

**Bemerkung:** In Access muss die Option "Benutze SQL Server-kompatible Syntax (ANSI 92)" aktiviert sein, damit der Parameter DEFAULT richtig erkannt wird.

## 6 Testen

Folgende Tests wurden abschliessend durchgeführt, um zu prüfen, ob AMRA-BBN korrekt funktioniert. Ein Test darf erst begonnen werden, wenn der vorherige erfolgreich war.

Während der Entwicklung wurden die einzelnen Funktionen intensiver durch den jeweiligen Programmierer geprüft. Dieser abschliessende Test prüft mehr das gesamte System.

Installer	
<b>Beschreibung</b>	Funktionsfähigkeit des Installers und des Buildscripts überprüfen
<b>Vorbedingung</b>	<ul style="list-style-type: none"> <li>Der Quellcode enthält keine Fehler</li> </ul>
<b>Testablauf</b>	<ol style="list-style-type: none"> <li>1. Target <i>installers</i> in Eclipse ausführen</li> <li>2. Normale Installation durchführen</li> <li>3. Programm kurz ausprobieren</li> <li>4. Online Installation durchführen</li> <li>5. Programm kurz ausprobieren</li> <li>6. Direkt ausführbare Version starten</li> <li>7. Programm kurz ausprobieren</li> </ol>
<b>Erwartung</b>	Buildscript und Installationen laufen ohne Fehler durch. Die Software kann problemlos gestartet und benutzt werden.
<b>Resultat</b>	Erwartungen sind eingetroffen

Input- und Output-Knoten markieren	
<b>Beschreibung</b>	Überprüfen der Knotenmarkierung
<b>Vorbedingung</b>	<ul style="list-style-type: none"> <li>BBN NEA_Struktur_92.dsc ist ein funktionierendes Modell und kann mit FAI-BBN oder MSBNx geöffnet werden</li> </ul>
<b>Testablauf</b>	<ol style="list-style-type: none"> <li>1. AMRA-BBN starten</li> <li>2. BBN öffnen: NEA_Struktur_92.dsc</li> <li>3. Äussere Knoten als Input-Knoten markieren</li> <li>4. Beliebige Knoten als Output-Knoten markieren</li> <li>5. BBN speichern</li> <li>6. BBN im Texteditor analysieren =&gt; Properties müssen richtig gesetzt sein</li> <li>7. BBN mit MSBNx öffnen =&gt; Properties müssen richtig gesetzt sein</li> </ol>
<b>Erwartung</b>	Das BBN lässt sich problemlos öffnen. Bei äusseren Knoten lässt sich Input und Output auswählen. Bei inneren Knoten lässt sich nur Output auswählen. Markierte Knoten werden farbig hervorgehoben. Das BBN lässt sich mit den Änderungen speichern. Im Texteditor ist bei den Knotenelementen das Property „DB_Settings“ mit den Werten „INPUT“, „OUTPUT“ oder „INPUT, OUTPUT“ hinzugekommen. MSNBx kann das veränderte BBN öffnen und erkennt das neue Property korrekt.
<b>Resultat</b>	Erwartungen sind eingetroffen

Datenbank generieren	
<b>Beschreibung</b>	Zu einem BBN eine Datenbank erzeugen
<b>Vorbedingung</b>	
<b>Testablauf</b>	<ol style="list-style-type: none"> <li>1. AMRA-BBN starten</li> <li>2. BBN öffnen: NEA_Struktur_92.dsc</li> <li>3. Alle äusseren Knoten als Input-Knoten markieren</li> <li>4. Etwa 15 beliebige Knoten als Output-Knoten markieren</li> <li>5. Wizard öffnen: Datenbank &gt; Generieren</li> <li>6. Datenbankname angeben</li> </ol>

	<ol style="list-style-type: none"> <li>7. Verzeichnis auswählen</li> <li>8. Wizard abschliessen =&gt; Datenbank wird erstellt</li> <li>9. Datenbank analysieren =&gt; wurden die richtigen Tabellen erstellt?</li> </ol>
<b>Erwartung</b>	Wizard zeigt einige Informationen zur bevorstehenden Operation an. Datenbankname und Verzeichnis lassen sich eingeben. Falls bereits eine Datei mit demselben Namen existiert wird darauf hingewiesen. Die Operation wird durchgeführt und eine Übersichtsseite zeigt erfolgreichen bzw. fehlerhaften Operationen an. Die Datenbank enthält zu jedem Input- und Output-Knoten eine Tabelle mit dem jeweiligen Präfix. Es wurde eine View für die Resultate erzeugt sowie eine weitere View pro Input-Präfix.
<b>Resultat</b>	Erwartungen sind eingetroffen

<b>Datenbank validieren</b>	
<b>Beschreibung</b>	Prüfen, ob ungültige Datenbankeinträge erkannt werden
<b>Vorbedingung</b>	<ul style="list-style-type: none"> <li>• Datenbank existiert</li> </ul>
<b>Testablauf</b>	<ol style="list-style-type: none"> <li>1. Datenbank öffnen</li> <li>2. Ungültige Werte in die Input-Tabellen eintragen (negative Werte, Werte grösser 1, Summe einer Zeile ungleich 1 oder 0)</li> <li>3. AMRA-BBN öffnen</li> <li>4. Zur Datenbank gehörendes Netzwerk laden</li> <li>5. Mit Datenbank verbinden: Datenbank &gt; Verbinden</li> <li>6. „Datenkonsistenz prüfen“ muss ausgewählt sein</li> </ol>
<b>Erwartung</b>	Es wird eine Liste mit Tabellen und darin betroffenen Untersuchungen angezeigt, bei denen die Eingabewerte ungültig sind. Der Benutzer kann anhand der Ausgabe schnell die besagten Stellen finden und korrigieren.
<b>Resultat</b>	Erwartungen sind eingetroffen

<b>Änderungen am BBN erkennen (Knoten)</b>	
<b>Beschreibung</b>	Die Software erkennt, falls Knoten (Input, Output) seit der Datenbankeinstellung im BBN hinzugekommen oder weggefallen sind
<b>Vorbedingung</b>	<ul style="list-style-type: none"> <li>• Datenbank existiert</li> </ul>
<b>Testablauf</b>	<ol style="list-style-type: none"> <li>1. BBN mit dem die Datenbank erstellt wurde in MSBNx öffnen</li> <li>2. Neue Input-Knoten hinzufügen</li> <li>3. Neue Output-Knoten hinzufügen</li> <li>4. Neue normale Knoten hinzufügen</li> <li>5. Einen bestehenden Knoten umbenennen</li> <li>6. BBN speichern</li> <li>7. AMRA-BBN starten</li> <li>8. Verändertes BBN laden</li> <li>9. Mit Datenbank verbinden: Datenbank &gt; Verbinden</li> <li>10. Problem untersuchen auswählen</li> </ol>
<b>Erwartung</b>	Die Verbindung zur Datenbank kann nicht hergestellt werden. Der Benutzer erhält die Möglichkeit, eine neue Datenbank zu wählen oder das Problem ermitteln zu lassen. Die gefundenen Inkonsistenzen werden angezeigt. Umbenannte Knoten werden zweimal gelistet (Er kann in der Datenbank und im BBN nicht gefunden werden). Normale Knoten werden nicht aufgelistet.
<b>Resultat</b>	Erwartungen sind eingetroffen

<b>Änderungen am BBN erkennen (Zustände)</b>	
<b>Beschreibung</b>	Die Software erkennt, falls Zustände von Input- und Output-Knoten seit der Datenbankeinstellung im BBN verändert wurden

<b>Vorbedingung</b>	<ul style="list-style-type: none"> <li>• Datenbank existiert</li> <li>• Die Knoten selbst wurden nicht verändert (sonst wird der notwendige Prüfung gar nicht gestartet)</li> </ul>
<b>Testablauf</b>	<ol style="list-style-type: none"> <li>1. BBN mit dem die Datenbank erstellt wurde in MSBNx öffnen</li> <li>2. Zustände von Input-Knoten verändern (umbenennen, neue hinzufügen, bestehende löschen)</li> <li>3. Zustände von Output-Knoten verändern (umbenennen, neue hinzufügen, bestehende löschen)</li> <li>4. Zustände von normalen Knoten verändern (umbenennen, neue hinzufügen, bestehende löschen)</li> <li>5. BBN speichern</li> <li>6. AMRA-BBN starten</li> <li>7. Verändertes BBN laden</li> <li>8. Mit Datenbank verbinden: Datenbank &gt; Verbinden</li> <li>9. „Problem untersuchen“ auswählen</li> </ol>
<b>Erwartung</b>	Die Verbindung zur Datenbank kann nicht hergestellt werden. Der Benutzer erhält die Möglichkeit, eine neue Datenbank zu wählen oder das Problem ermitteln zu lassen. Die gefundenen Inkonsistenzen werden angezeigt. Aus der Ausgabe ist ersichtlich, bei welchem Knoten Veränderungen durchgeführt wurden. Normale Knoten werden nicht aufgelistet.
<b>Resultat</b>	Erwartungen sind eingetroffen

### Berechnung: Untersuchung ohne Untersuchungsergebnisse

<b>Beschreibung</b>	Eine leere Untersuchung berechnen
<b>Vorbedingung</b>	<ul style="list-style-type: none"> <li>• Knoten sind markiert</li> <li>• Datenbank wurde erfolgreich erzeugt</li> <li>• Datenbank und BBN sind kompatibel zueinander</li> </ul>
<b>Testablauf</b>	<ol style="list-style-type: none"> <li>1. Datenbank mit Access öffnen</li> <li>2. Einen neuen Piloten erzeugen</li> <li>3. Dem Piloten eine Untersuchung zuordnen</li> <li>4. AMRA-BBN starten</li> <li>5. Mit Datenbank verbinden: Datenbank &gt; Verbinden</li> <li>6. Berechnungs Wizard öffnen: Datenbank &gt; Berechnen</li> <li>7. Vorher erstellte Untersuchung auswählen und berechnen</li> <li>8. Resultate in Datenbank mit manueller Abfrage im BBN vergleichen</li> <li>9. Export Wizard öffnen: Datenbank &gt; Export</li> <li>10. Vorher erstellte Untersuchung auswählen und exportieren</li> <li>11. Exportiertes DSC mit original DSC vergleichen</li> </ol>
<b>Erwartung</b>	Ein Pilot lässt sich einfach über ein Formular in die Datenbank eintragen. Eine Untersuchung mit einer eindeutigen Untersuchungsnummer kann dem Piloten zugeordnet werden. AMRA-BBN lässt sich ohne Probleme mit dem originalen BBN verbinden und die Berechnung sowie der Export durchführen. Es wird gewarnt, dass zu allen Knoten keine Eingabewerte in der Datenbank gefunden werden konnten. Die berechneten Resultate in der Datenbank sind identisch zu den Abfragen auf den jeweiligen Knoten des BBN. Das exportierte BBN ist identisch zum originalen BBN.
<b>Resultat</b>	Erwartungen sind eingetroffen

### Berechnung: Untersuchung mit Untersuchungsergebnissen

<b>Beschreibung</b>	Prüfen, ob Untersuchungsergebnisse richtig auf das Netzwerk projiziert werden
<b>Vorbedingung</b>	<ul style="list-style-type: none"> <li>• Knoten sind markiert</li> <li>• Datenbank wurde erfolgreich erzeugt</li> <li>• Datenbank und BBN sind kompatibel zueinander</li> </ul>
<b>Testablauf</b>	<ol style="list-style-type: none"> <li>1. Datenbank mit Access öffnen</li> </ol>

	<ol style="list-style-type: none"> <li>2. Einen neuen Piloten erzeugen</li> <li>3. Dem Piloten eine Untersuchung zuordnen</li> <li>4. Untersuchungsergebnisse eintragen</li> <li>5. AMRA-BBN starten</li> <li>6. Mit Datenbank verbinden: Datenbank &gt; Verbinden</li> <li>7. Berechnungs Wizard öffnen: Datenbank &gt; Berechnen</li> <li>8. Vorher erstellte Untersuchung auswählen und berechnen</li> <li>9. Resultate in Datenbank mit manueller Abfrage im BBN vergleichen</li> <li>10. Export Wizard öffnen: Datenbank &gt; Export</li> <li>11. Vorher erstellte Untersuchung auswählen und exportieren</li> <li>12. Exportiertes DSC analysieren</li> </ol>
<b>Erwartung</b>	<p>AMRA-BBN lässt sich ohne Probleme mit dem originalen BBN verbinden und die Berechnung sowie der Export durchführen. Es wird gewarnt, falls zu einigen Knoten keine Eingabewerte in der Datenbank gefunden werden konnten. Die berechneten Resultate in der Datenbank sind identisch zu den Abfragen auf den jeweiligen Knoten des BBN. Das exportierte BBN enthält die in der Datenbank stehenden Untersuchungsergebnisse.</p>
<b>Resultat</b>	<p>Erwartungen sind eingetroffen</p>

## 7 Ergebnisse

### 7.1 Was wurde erreicht?

AMRA-BBN ist in der Lage die folgenden Aufgaben auszuführen:

Knoten können als Input- und/oder Output-Knoten markiert werden. Diese Eigenschaft wird beim Abspeichern ebenfalls persistiert und beim erneuten Laden des BBNs gleich übernommen.

Aus einem BBN mit den gesetzten Eigenschaften kann eine Microsoft Access Datenbank erzeugt werden. Es werden Tabellen für alle markierten Knoten erzeugt. Zusätzlich gibt es Tabellen für Piloten, Untersuchungen und die Knoten-Gruppen. Neben den Tabellen werden auch Abfragen erzeugt, mit deren Hilfe es auf einfache Weise möglich ist, ein Formular zu erzeugen.

Ein geladenes BBN kann mit einer Datenbank verbunden werden. Dabei wird automatisch geprüft, ob die beiden zueinander passen oder ob Unterschiede bestehen. Existieren Unterschiede, so hat man die Möglichkeit, eine andere Datenbank auszuwählen oder die Unterschiede zu ermitteln. Optional kann man auch noch überprüfen, ob die Datenbank gültige Werte enthält. Damit ist gemeint, dass überprüft wird, ob die Wahrscheinlichkeitssumme 0 (noch keine Einträge) oder 1 (eingetragene Untersuchung ist) sowie, dass sich alle Untersuchungswerte in der Range 0 -1 befinden.

Die Werte für die Output-Knoten können anhand der in der Datenbank eingetragenen Werten und des geladenen BBNs berechnet werden. Dabei kann ausgewählt werden, für welche Piloten und welche Untersuchung die Resultate berechnet werden sollen. Diese werden dann wieder in die Datenbank geschrieben und können über eine Abfrage betrachtet werden.

In der Datenbank eingetragene Untersuchungen können auch exportiert werden. Die Untersuchungswerte werden auf das BBN gemappt und anschliessend als DSC-Datei abgespeichert.

### 7.2 Was wurde nicht erreicht?

- Es ist keine graphischen Auswertung der berechneten Werte möglich
- Abspeichern sowie Exportieren ist nur im DSC-Format möglich
- Ein entdeckter Fehler im verwendeten BBN-Framework konnte mangels Zeit nicht behoben werden. Teilweise werden Kanten erst dann gespeichert, wenn man die Zustände bearbeitet. Da die Netzwerke mit MSBNX erstellt werden, ist dies vorerst nicht tragisch.
- Kleinere Benutzerhilfen wie: Tastenkürzel fürs Markieren, Mehrfachmarkierung, ...
- Automatisierte Synchronisation von Datenbank und BBN bei Änderungen des Netzwerks

Bei einer Fortführung oder Erweiterung des Projektes sollte geprüft werden, welche dieser oben genannten Punkte einen Gewinn an Funktionalität und Komfort für den Benutzer bedeuten. Diese sollten dann umgesetzt werden.

### 7.3 Schlussfolgerungen

Das Produkt erfüllt alle Muss-Kriterien der Aufgabenstellung sowie die von uns definierten Use Cases. Die Datenbankverbindung über JDBC funktioniert zuverlässig und wir konnten trotz Einschränkungen alles realisieren.

## 8 Glossar

- AMRA-BBN: Air Medical Research Academy - BBN. Die neue Software, welche auf dem Programm FAI-BBN basiert und um eine Datenbankanbindung erweitert wurde.
- FAI-BBN: Fliegerärztliches Institut - BBN. Software von Remo Siegwart, welches Bayesian-Belief-Netzwerke darstellen und bearbeiten kann.
- BBN: Bayesian-Belief-Netzwerk - Entscheidungs-Netzwerk zur Modellierung von unsicherem Wissen
- DSC: Format, in welchem die Microsoft-Software MSBNx ihre BBNs abspeichert.
- I18N: Internationalisierung. Es wird die Möglichkeit geboten, auf eine einfache Art und Weise die Software in eine andere Sprache zu übersetzen.
- JDBC: Java Database Connectivity. Schnittstelle für Java-Anwendungen, um auf eine Datenbank zuzugreifen.

## 9 Referenzliste

- Christoph Stamm, Samuel Huber: Informatik Projektarbeit P5 (I1121) Bayesian-Belief-Netzwerke
- Remo Siegwart: Projektdokumentation: Software zur Modellierung von Bayesian-Belief Netzwerken; S. 83

## 10 Anhang

### 10.1 Neue Properties in den DSC-Parser einfügen

Neue Properties lassen sich nicht ohne weiteres in das DSC-File schreiben. Der Parser stolpert ansonsten darüber, wenn er die Datei einliest. Daher muss die Grammatik des Parser in der Datei DSC.jj angepasst und die Parserklassen neu kompiliert werden. Dazu ist der Parsergenerator JavaCC (Java Compiler Compiler) notwendig.

#### Notwendige Änderungen:

Am einfachsten orientiert man sich an einem bestehenden Property. Dazu sucht man die Textstellen in der Parserdatei und fügt seinen eigenen Code an diesen Stellen an.

Für das Einfügen des Properties DB\_Settings sind folgende Ergänzungen notwendig:

1. In der Sektion "Token" muss ein neues Schlüsselwort definiert werden:

```
| <NODEDBSETTINGS: "DB_Settings" >
```

2. In der Methode "VariableContent(String name)" müssen diese Anweisungen hinzugefügt werden:

```
| s=DBSettings()  
{ properties.addElement(s); }
```

3. Schliesslich muss noch die neue Methode "DBSettings()" definiert werden:

```
/* DB_Settings definition */  
String DBSettings() :  
{  
String input;  
}  
{  
<NODEDBSETTINGS> "=" input=getFullString() ";"  
  { return("DB_Settings = " + input); }  
}
```

Nach den Anpassungen müssen die Parserklassen neu erzeugt werden. Dies erledigt die Bat-Datei, welche in jedem Parser-Package bereits vorhanden ist. Sie ruft lediglich JavaCC auf und übergibt die DSC.jj-Datei als Parameter. Nach erfolgreichem Durchlauf sind die neuen Parser-Klassen verfügbar.

#### Nacharbeiten:

Nach dem Generieren der Parser-Klassen sind jeweils folgende zwei Änderungen notwendig:

1. In die Datei DSCTokenManager.java muss am Ende folgende Methode eingefügt werden:

```
public String getProperties() {  
    if(properties != null) {  
        return properties.toString();  
    } else {  
        return null;  
    }  
}
```

2. Die Klasse ParseException muss die Klasse IFException erweitern (nicht Exception):

```
public class ParseException extends IFException {...
```

### 10.1.1 SimpleJdbcTemplate

Lesende Operation:

```
public Person findPerson(long id) {
    String sql = "select id, name, age from people where id = ?";
    ParameterizedRowMapper<Person>
        mapper = new ParameterizedRowMapper<Person>() {
        public Person mapRow(ResultSet rs, int rowNum) throws SQLException {
            Person person = new Person();
            person.setId(rs.getLong("id"));
            person.setName(rs.getString("name"));
            person.setAge(rs.getInt("age"));
            return person;
        }
    };
    return this.jdbcTemplate.queryForObject(sql, mapper, id);
}
```

Schreibende Operation:

```
...
sql = "insert into demotable (id, company, employees) values (?, ?, ?)";
Object[] search = new Object[] {1, "37signals", 200};
this.jdbcTemplate.update(sql, search);
...
```

Effektiv muss ein Entwickler nur noch den Query-String zusammenbauen und diesen ausführen. Bei einer schreibenden Operation, wie z.B. Insert, müssen Parameter nicht direkt in das Query eingesetzt werden. Es ist möglich, den Platzhalter „?“ im Query zu verwenden und die Parameter später beim Ausführen des Querys anzugeben. Lesende Operationen werden durch einen RowMapper unterstützt. Der Mapper wird als anonyme Klasse implementiert und für jede Zeile im Resultset ausgeführt.

- **Vorteile**
  - Nahezu kein technischer Code mehr
  - Sauberes Ressourcenhandling
  - Spring verpackte alle Fehler in DataAccessExceptions
  - Einfacher austauschen der DataSource Implementierung möglich
  - Transparente Teilnahme an Spring Transaktionen möglich