



Modellierung und Entwicklung
von Pliable Objects
zum Aufbau dynamischer Informationssysteme
im medizinischen Fachgebiet der Anästhesie

DISSERTATION

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

angenommen durch die Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg

von Dipl.-Inf. Markus Preißner

geb. am 20.03.1970 in Stadthagen

Gutachterinnen/Gutachter

Prof. Dr. Gunter Saake
Prof. Dr. Reinhold Haux
PD Dr.-Ing. habil. Meike Klettke

Magdeburg, den 20.09.2013

E h r e n e r k l ä r u n g

Ich versichere hiermit, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; verwendete fremde und eigene Quellen sind als solche kenntlich gemacht. Insbesondere habe ich nicht die Hilfe eines kommerziellen Promotionsberaters in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Ich habe insbesondere nicht wissentlich:

- Ergebnisse erfunden oder widersprüchliche Ergebnisse verschwiegen,
- statistische Verfahren absichtlich missbraucht, um Daten in ungerechtfertigter Weise zu interpretieren,
- fremde Ergebnisse oder Veröffentlichungen plagiiert,
- fremde Forschungsergebnisse verzerrt wiedergegeben.

Mir ist bekannt, dass Verstöße gegen das Urheberrecht Unterlassungs- und Schadensersatzansprüche des Urhebers sowie eine strafrechtliche Ahndung durch die Strafverfolgungsbehörden begründen kann. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form als Dissertation eingereicht und ist als Ganzes auch noch nicht veröffentlicht.

Magdeburg, den 14.03.2013

meinem Vater,
der mir gezeigt hat, mit Ruhe und Gelassenheit seinem Ziel entgegenzustreben,

meiner Mutter,
die immer an mich geglaubt und mich gestärkt hat,

und meiner Frau,
die mir den Ruck gegeben hat, mein privates Ziel, die Fertigstellung dieser Arbeit,
nicht mehr zurückzustellen.

Zusammenfassung

Im medizinischen Anwendungsgebiet der Anästhesie ist ein hohes Informationsaufkommen zu dokumentieren. Im weiteren Sinne bedeutet Anästhesie die Schmerz- und Bewusstseinsausschaltung eines Patienten. Das Vorgehen und die Betreuung, die ein Anästhesist bei der Patientenbehandlung vornimmt, sind in Anästhesieprotokollen zu erfassen. Der Einsatz moderner Informationssysteme, die Unterstützung bei der Erfassung und dem „Bereitstellen“ von anästhesiespezifischen Informationen leisten, wird von Anästhesisten gefordert. Die Art und Anzahl der zu dokumentierenden anästhesiologischen Informationen orientieren sich an der fachlichen Ausrichtung der Klinik, z.B. Unfall-, Herz- oder Kinderklinik, der die Anästhesieabteilung angehört. Ferner werden die Informationen durch die Anästhesisten der Klinik selbst spezifiziert und besitzen einen hohen Grad an Individualität. Ein standardisiertes Informationssystem kann die individuelle, eventuell mobile Anästhesiedatenverarbeitung nicht dokumentieren, da ein nach objektorientierten Gesichtspunkten gestaltetes Anästhesiemodell nicht dynamisch formbar ist. Dieser Mangel an dynamischer Anpassbarkeit ist die Hauptmotivation dieser Arbeit.

Um diesen Mangel näher zu spezifizieren, wird zunächst aus Sicht der Informatik die Entwicklung eines Informationssystems betrachtet. Der Vorstellung verschiedener Modelle zur Anforderungsanalyse und zur Konzeption eines Softwaresystems, das die Anforderungen erfüllt, folgt eine kritische Auseinandersetzung, die den Mangel an dynamischer Anpassbarkeit offenlegt. Dabei ist dies weniger ein Problem in der Anwendungsmodellierung als vielmehr in der Anwendungsimplementierung.

Zur Überwindung dieser Hürde werden im zweiten Teil dieser Arbeit Pliable Objects entwickelt, die über die Dynamik verfügen, an anwenderspezifische, individuelle Anforderungen anpassbar zu sein. Diese Anpassbarkeit wird nicht nur im Modell selbst unterstützt, sondern auch von den realisierten Pliable Objects in einem implementierten Informationssystem.

Unter einem Pliable Object selbst wird eine Beschreibungsdomäne verstanden, die um eine Menge von Attributen, einer Menge von Aktionen und einer Menge von Regeln zu einem Tupel erweitert wird. Mittels eines (eher simplen) Datenbankschemas können Pliable Objects in einer Datenbank gespeichert und durch ein Datenbankmanagementsystem flexibel verwaltet werden. Aus dieser Flexibilität heraus kann die Dynamik gewonnen werden, die für den Aufbau von Informationssystemen im medizinischen Anwendungsgebiet der Anästhesie benötigt wird.

Abstract

Within the medical application area of anesthesia, a high density of information has to be documented. In the broad sense, anesthesia means the elimination of a patient's pain and awareness during an operation. The action and assistance an anesthetist gives to a patient during the medical treatment must be documented in anesthesia protocols. The application of modern information systems is required by anesthetists in order to support them by recording and providing anesthesia specific data. The kind and quantity of anesthetic information which have to be documented depend on the hospital's specialization the anesthesia department belongs to, e.g. emergency, cardiology or children's hospital. Additionally, the information are specified by the hospital's anesthetists themselves and have, therefore, a high degree of individuality and identity. A standardized information system is not able to document the individual and possibly mobile processing of anesthesia data as an anesthesia model which is based on object-oriented aspects is not changeable in a dynamic manner. This fault of dynamic adaptability is the main incentive of this paper.

In order to examine the fault more closely, the development of an information system from the point of computer sciences is reviewed first. The presentation of different models for requirement analysis and for the engineering of software systems fulfilling the requirements is followed by a critical discussion which discloses the fault of dynamic adaptability. This is, however, not a problem of the application's modeling, but a problem of the application's implementation.

To overcome this barrier, in the second part of this paper pliable objects are developed, which have the dynamic availability to be adaptable to user specific and individual requirements. This adaptability is not only supported in the model itself but also by the instantiated pliable objects in an implemented information system.

Basically, a pliable object is regarded as a description domain which is extended by a set of attributes, a set of actions and a set of rules. Using a (rather simple) database schema, pliable objects could be stored within a database and could be managed by the database management system in a very flexible manner. Due to this flexibility, a dynamic system can be realized which is needed for the creation of information systems within the medical application area of anesthesia.

Vorwort

Die vorliegende Arbeit präsentiert Ergebnisse, die im Rahmen einer mehrjährigen Beschäftigung rund um die Entwicklung medizinischer Informationssysteme gewonnen wurden. Die ersten Erfahrungen mit diesem Thema wurden im Rahmen des Drittmittelprojektes MC-MEDIS an der TU Clausthal bzw. Universität Hamburg gemacht. Ziel des Projekts war die Entwicklung eines Anästhesie-Dokumentationssystems, welches die Anästhesisten bei der Erfassung ihrer Anästhesie-Protokolle unterstützt und dabei die Sicherstellung der Plausibilität der Protokolle gewährleistet.

Innerhalb des Projektes wurde entsprechend eines Datenbank-Entwurf Prozesses (vgl. [V00]) ein Prototyp entwickelt. In Tausenden von Stunden hat das Projektteam die Anforderungen analysiert, ein Modell spezifiziert, dieses in ein Datenbankschema überführt und das Informationssystem schließlich erfolgreich implementiert.

Während der Entwicklung sind viele Probleme gelöst worden, die auch im Rahmen ähnlich gelagerter Forschungs- und Entwicklungsprojekte auftreten; angefangen von Zeitverzögerungen, durch z.B. Personalfluktuationen bis hin zur Behebung von Modellierungsfehlern. Die Lösung derartiger Probleme führt in der Regel zur Erweiterung des Erfahrungsschatzes, den sich die Projektmitarbeiter dabei aneignen. Eine Problembefehung, die die folgende Anekdote beschreibt, kann jedoch als ein Auslöser der vorliegenden Arbeit betrachtet werden.

Jedes medizinische Informationssystem muss in der einen oder anderen Form den Patienten verwalten. Für jeden zu erfassenden Patienten wird dabei dessen Geschlecht ermittelt und im System hinterlegt. Nach der erstmaligen Analyse der Anforderungen war für das Geschlecht die Auswahl zwischen den Werten *männlich* oder *weiblich* vorgesehen. Im Rahmen der ersten Implementierung ist diese Auswahl auf den Typ *boolean* abgebildet worden. Nachdem der erste Prototyp einen hohen Reifegrad erreicht hatte, ist von der Deutschen Gesellschaft für Anästhesiologie und Intensivmedizin (DGAI) die Auswahl für die Geschlechtsangabe eines Patienten auf die Auswahl *männlich*, *weiblich* oder *intersexuell* erweitert worden. Die interne Auflösung und Änderung der Typangabe von *boolean* auf einen Aufzählungstyp war recht aufwändig, da im internen System die Geschlechtsprüfung innerhalb von Plausibilitäten häufig zu finden war. Nach Abschluss der Fleißarbeit, die sich durch die Bereinigung ergeben hatte, ist der Prototyp einem durchreisenden, amerikanischen Arzt vorgestellt worden, der sich u.a. über aktuelle Forschungsvorhaben informieren wollte. Als er in der Bedienoberfläche die Auswahlmöglichkeiten für die Geschlechtsangaben sah, fragte er nur: „*Is this a joke?*“.

Bei der Beantwortung der Frage konnte man sich beruhigt auf die Festlegung der DGAI berufen. Allerdings weist die Anekdote auf ein Kernproblem hin. Unabhängig von der kulturellen Klassifizierung bestimmter Eigenschaften ist die Typfestlegung einzelner Eigenschaften in der Informatik nicht so ohne weiteres änderbar. Jede Eigenschaft, die erfasst werden soll, muss auch mittels einer geeigneten Funktionalität verarbeitet werden, d.h. für jeden zu verarbeitenden Wert gibt es mindestens eine Routine, die diese Verarbeitung realisiert. Eine Typänderung für einen Wert führt somit immer zu der Anpassung der Routinen, die den Wert verarbeiten. Die Anpassungen an sich werden durch manuelle Tätigkeiten vorgenommen, deren Umfang sich aus der Anzahl der den Wert verarbeitenden Routinen ergibt.

In der vorliegenden Arbeit wird eine Lösung angeboten, derartige Anpassungen technisch unterstützt, effektiv und effizient vornehmen zu können.

Danksagung

Beim Erstellen dieser Dissertation wurde ich von vielen Personen unterstützt. Ich danke allen, die mich kritisch und konstruktiv begleitet, mir in Diskussionen Grenzen, aber auch neue Ansätze aufgezeigt haben und die mir tatkräftig beim Ausräumen von Tipp- und Formulierungsfehlern geholfen haben.

Ich danke meinem Arbeitgeber, der Fa. Leuze electronic GmbH + Co. KG, der die Anfertigung meiner Dissertation parallel zu meinen Aufgaben tolerierte und mich ermutigte, dass Ziel der Fertigstellung weiter zu verfolgen.

Speziell danke ich allen Teilnehmern des 17. Workshops *Grundlagen von Datenbanken* und allen Teilnehmern des Workshops *Datenmanagement und Interoperabilität im Gesundheitswesen (DIG)*. Die fachlichen und konstruktiven Diskussionen waren eine echte Bereicherung und hatten Einfluss auf meine Dissertation.

Mein Dank gilt auch meinen ehemaligen Kollegen am Institut für Informatik der TU Clausthal und im Arbeitsbereich Technische Informatiksysteme von Prof. Dr. Möller am Fachbereich Informatik der Universität Hamburg, insbesondere den ehemaligen Clausthaler Kollegen Thorsten Gadmann, Dr. Christoph Grebe, Dr. Carsten Hörner, Dr. Björn Kesper, Frank Kroll und Bodo Volkmer sowie den ehemaligen Hamburger Kollegen Markus Bach, Stefan Bergstedt, Dr. Werner Hansmann, Simon Heß, Dr. Kai Himstedt, Birgit Koch, Christian Körber, Stefan Wiegrefe, Dr. Jochen Wittmann und Dr. Christian Zemke.

Ich danke den Mitarbeitern der Fa. Datapec GmbH für die Erfahrungen, die ich im Rahmen des Drittmittelprojektes machen durfte, insbesondere den mit der Anästhesie-Systementwicklung betrauten Mitarbeitern Dieter Damm und Ralf Single. Dem Geschäftsführer, Herrn Joachim Schweitzer, danke ich besonders für die Unterstützung und die Erlaubnis, Anästhesieprotokolle abdrucken zu dürfen.

Für den vor allem in der Endphase zu überprüfenden, nicht unerheblichen Anteil medizinischer Informationen danke ich herzlichst Herrn Dr. med. Heiko Bienengraber von der Klinik am Eichert Göppingen.

Den Mitarbeitern des Lehrstuhls für Datenbank- und Informationssysteme von Prof. Heuer am Institut für Informatik der Universität Rostock danke ich für kritische Diskussionen und Gespräche. Insbesondere Frau Dr. Klettke gilt mein Dank für ihre fachliche Betreuung, die äußerst hilfreich und lehrreich war.

Herrn Prof. Haux und seinen Mitarbeitern des Peter L. Reichertz Instituts für Medizinische Informatik der Technischen Universität Braunschweig danke ich für die konstruktive Kritik und für die Hinweise bei der medizinisch fachlichen Betrachtung.

Ohne die hohe Fachkompetenz und Unterstützung der Arbeitsgruppe Datenbanken der Fakultät Informatik der Otto-von-Guericke-Universität Magdeburg wäre diese Arbeit nie zustande gekommen. Ganz besonders danke ich Dr. Eike Schallehn für seine überaus geduldige Betreuung und Prof. Dr. Gunter Saake für die wissenschaftliche Begleitung.

Außerdem danke ich meinem Schwager Andreas Ott und meiner Frau Sibylle Preißner, ohne deren Hilfe die Tippfehler nie korrigiert worden wären. Ihr habt Herausragendes dabei geleistet. Vielen herzlichen Dank!

Last but not least danke ich meinen Eltern Helga und Willi Preißner, deren moralische Unterstützung und Glaube an mich immer eine Stärkung war. Beim Verfassen dieser Dissertation gab es immer wieder Momente, in denen es sehr schwerfiel, weiterzumachen. In diesen Momenten half der Rückhalt, den mir meine Eltern und meine Frau gegeben haben.

Kirchheim / Teck, März 2013

Markus Preißner

Inhalt

ABBILDUNGSVERZEICHNIS	XXI
TABELLENVERZEICHNIS	XXII
1 EINLEITUNG	1
1.1 Motivation	1
1.2 Zielsetzung	1
1.3 Aufbau der Arbeit	2
2 GRUNDLAGEN DER MODELLIERUNG VON INFORMATIONSSYSTEMEN	3
2.1 Einführung in das Leistungsspektrum von Informationssystemen.....	3
2.1.1 Relevante Dimensionen in der Entwicklung von Informationssystemen	4
2.1.2 Basisaspekte der Modellierung	6
2.2 Methode der Strukturierten Analyse	8
2.2.1 Datenflussdiagramme	8
2.2.2 Data Dictionary	10
2.2.3 Mini-Spezifikationen.....	10
2.2.4 Generelle Vorgehensweise der Strukturierten Analyse.....	10
2.2.5 Zusammenfassung.....	12
2.3 Methode des Entity-Relationship-Modells.....	13
2.3.1 Grundelemente	13
2.3.2 Semantische Elemente.....	15
2.3.3 Zusammenfassung.....	17
2.4 Methode der Objektorientierten Modellierung.....	18
2.4.1 Das Paradigma der Objektorientierung	19
2.4.2 Unified Modeling Language	21
2.4.3 Zusammenfassung.....	31
2.5 Methode der Ereignisgesteuerten Prozessketten.....	32
2.5.1 Grundelemente	32
2.5.2 Modellierung von Ereignisgesteuerten Prozessketten.....	33
2.5.3 Erweiterte Ereignisgesteuerte Prozessketten.....	33
2.5.4 Zusammenfassung.....	34
2.6 Konzept der Business Objects	35
2.6.1 Geschichtlicher Hintergrund	36
2.6.2 Business Object Definition der Object Management Group.....	36
2.6.3 Zusammenfassung.....	37
3 DOKUMENTATIONSANSPRÜCHE IM MEDIZINISCHEN FACHGEBIET DER ANÄSTHESIE	39

3.1	Was bedeutet Anästhesie?	39
3.1.1	Idealisierter Versorgungszyklus	40
3.1.2	Anästhesiologischer Arbeitsablauf.....	41
3.1.3	Intraoperativer Verlauf	42
3.1.4	Präoperativer Verlauf	43
3.1.5	Postoperativer Verlauf.....	44
3.1.6	Anästhesieprozess	45
3.2	Dokumentation der Anästhesie	45
3.2.1	Anästhesieprotokoll.....	46
3.2.2	Anästhesie-Dokumentationssystem	50
3.2.3	Beleglese-Systeme	50
3.2.4	Anästhesie-Informationen-Management-System	51
3.2.5	Stand der Forschung.....	52
3.2.6	Probleme von Anästhesie-Dokumentationssystemen	53
3.2.7	Anforderungen an Anästhesie-Dokumentationssysteme	56
3.3	Eignungsprüfung von Dokumenten-Management-Systemen	60
3.3.1	Bedeutung des Begriffs <i>Dokumenten-Management</i>	60
3.3.2	Klassisches Dokumenten-Management.....	61
3.3.3	Anästhesie-Dokumentation mittels Dokumenten-Management-Systemen.....	63
3.3.4	Fazit.....	65
3.4	Eignungsprüfung von Workflow-Management-Systemen	66
3.4.1	Prinzipielle Abbildung von Workflows in Workflow-Management-Systemen.....	67
3.4.2	Bestimmung eines anästhesiologischen Workflows	69
3.4.3	Fazit.....	72
3.5	Kritische Betrachtung des Entwicklungsprozesses von Informationssystemen...	74
3.5.1	Explizite Anforderungen vs. impliziter Erwartungshaltung	75
3.5.2	Systementwurf mit Berücksichtigung der Datenbedeutung.....	75
3.5.3	Unzulänglichkeiten bei der Implementierung des Modells.....	76
3.6	Schlussfolgerung	81
4	DAS MODELL PLIABLE OBJECT	83
4.1	Motivation für ein neues Objekt-Verständnis	83
4.1.1	Abstraktion eines Pliable Objects.....	84
4.1.2	Semantische Darstellung von Pliable Objects.....	86
4.2	Allgemeine Eingrenzung von Wertemengen	87
4.2.1	Menge aller Werte	87
4.2.2	Domäne	87
4.2.3	Mengen-Axiome	89
4.2.4	Beschreibungsdomäne.....	92
4.3	Formale Definition von Pliable Objects	92
4.4	Zustandsbeschreibung von Pliable Objects	93
4.5	Verhaltensbeschreibung von Pliable Objects	95

4.5.1	Anlehnung an die Algorithmus-Definition	96
4.5.2	Klassifikation von Zustandsänderungen	97
4.5.3	Sequenzierung von Abläufen mittels Statements	98
4.5.4	Aktionsbeschreibung durch kontextfreie Grammatiken.....	100
4.5.5	Spezifische Erweiterung über programmiersprachliche Konstrukte.....	100
4.5.6	Spezielle Zuweisungsanweisungen	103
4.6	Verhaltenskontrolle von Pliable Objects.....	104
4.7	Selbstbeschreibungsmerkmale von Pliable Objects.....	107
4.7.1	Sondereigenschaften.....	107
4.7.2	Methoden.....	111
4.7.3	Bedingungen.....	113
4.7.4	Spezifikation von Sonder-Pliable-Objects	114
4.7.5	Spezielle Pliable-Object-Klassifikation	116
4.8	Notationselement von Pliable Objects	119
4.8.1	Visualisierung der Sonder-Pliable-Objects	119
4.8.2	Visualisierung der Pliable-Object-Klassifikationen.....	120
4.9	Semantische Begriffsdefinitionen.....	121
4.9.1	Charakterisierende Definition	121
4.9.2	Äquivalenzrelationale Definitionen	121
4.9.3	Verallgemeinerung und Spezialisierung	124
4.10	Verknüpfung von Pliable-Object-spezifischen Informationen	128
4.10.1	Synthese von Pliable Objects	129
4.10.2	Bewertungsoptionen von Pliable Objects	134
5	INFORMATIONSSYSTEM FÜR PLIABLE OBJECTS	137
5.1	Klassische Realisierung von Informationssystemen	137
5.1.1	Bedeutung von Variablen.....	137
5.1.2	Verwendung von Datenstrukturen	139
5.2	Informationssystem-Architektur für Pliable Objects.....	143
5.2.1	Entity-Menge E.....	144
5.2.2	Entity-Domäne	144
5.2.3	Idee der Betrachtung eines Informationssystems als universales Pliable Object... 145	
5.3	Datenbankschema für Pliable Objects	146
5.3.1	Relationale Abbildung von Pliable Objects	146
5.3.2	Relation Domäne.....	148
5.3.3	Relation Attribut.....	149
5.3.4	Relation Pliable Object.....	149
5.3.5	Relationen der Pliable Objects zu eigenen Attributen	150
5.3.6	Relation Value.....	150
5.3.7	Relation Action	151
5.3.8	Relation Rule.....	152
5.4	Generische Verhaltensabbildung von Pliable Objects.....	154
5.4.1	Allgemeingültige Kontrollflussschleife mit generischer Ausprägung.....	155

5.4.2	Generische Prozesse	156
5.4.3	Verwaltung von Fehlerzuständen	165
5.4.4	Operationelle Attribute	166
5.4.5	Operationelle Aktionen	167
5.4.6	Unwiderrufliche Zustände	167
5.4.7	Abbildung von Werten	169
5.5	Basisarchitektur eines Informationssystems zur Verarbeitung von Pliable Objects	170
5.5.1	Zustandsabbildung	172
5.5.2	Transitionsbetrachtung	174
5.5.3	Workflowbasierte Betrachtung von Pliable Objects	176
5.6	Zusammenfassung	177
6	ANÄSTHESIE-MANAGEMENT MIT PLIABLE OBJECTS	179
6.1	Basisklassenmodell der Anästhesie	179
6.1.1	Anästhesie-Domänen	180
6.1.2	Unbestimmte Anästhesie-Domänen	181
6.1.3	Anästhesie-Pliable-Objects	182
6.2	Anästhesie-Prozess	184
6.2.1	Direkte Prozessabbildung	184
6.2.2	Indirekte Prozessabbildung	186
6.2.3	Nutzen der Erweiterbarkeit von Pliable Objects	191
6.3	Anästhesie-Dokumenten-Management	194
6.4	Fazit	196
7	ZUSAMMENFASSUNG UND AUSBLICK	199
7.1	Fazit	201
7.2	Ausblick	202
7.2.1	Ableitung intuitiver Bedienoberflächen	202
7.2.2	Geeigneter Mechanismus zur Bearbeitung von Fehlern und Fehlerzuständen	203
7.2.3	Integration eine rollenbasierten Bedienkonzeptes	203
7.2.4	Unterstützung von teamorientieren Arbeiten im Sinne von Groupware	204
7.2.5	Aufhebung der Beschränkung durch Schema-Pliable-Objects	205
8	ANHANG	207
8.1	SQL-Anweisungen eines Pliable-Object-Datenbankschemas	207
8.1.1	Create-Anweisungen für die Sonder-Pliable-Objects	207
8.1.2	Create-Anweisungen für Zustandsmaschinen	207
8.1.3	Create-Anweisung für die Relation Value	208
8.1.4	Create-Anweisung für die Own-Relationen	208
8.2	Initialisierung der Selbstbeschreibung	209
8.2.1	Beispiel für Domänen	209

8.2.2 Basisattribute	209
9 LITERATURVERZEICHNIS	211

Abbildungsverzeichnis

ABBILDUNG 1: HIERARCHIEKONZEPT DER STRUKTURIERTEN ANALYSE (VGL. [BA00]).....	11
ABBILDUNG 2: VISUALISIERUNG EINER EXKLUSIVE-ODER-BEZIEHUNG IM ERM (VGL. [KR05])	16
ABBILDUNG 3: BEISPIEL ANWENDUNGSFALLDIAGRAMM	22
ABBILDUNG 4: BEISPIEL KLASSENDIAGRAMM.....	23
ABBILDUNG 5: BEISPIEL SEQUENZDIAGRAMM.....	25
ABBILDUNG 6: BEISPIEL KOLLABORATIONSDIAGRAMM	26
ABBILDUNG 7: BEISPIEL ZUSTANDSDIAGRAMM	27
ABBILDUNG 8: BEISPIEL AKTIVITÄTSDIAGRAMM	28
ABBILDUNG 9: BEISPIEL KOMPONENTENDIAGRAMM	30
ABBILDUNG 10: IDEALISIERTER VERSORGUNGSZYKLUS.....	40
ABBILDUNG 11 TEILPROZESSE DER ANÄSTHESIE	42
ABBILDUNG 12: FORMULARSEITE EINES PRÄOPERATIVEN PROTOKOLLS.....	47
ABBILDUNG 13: GENERALISIERTES PROZESSVERSTÄNDNIS.....	67
ABBILDUNG 14: ABSTRAKTE ZEITLICHE FESTLEGUNG VON VORGÄNGEN	69
ABBILDUNG 15: ÜBERFÜHRUNG PROZESSRELEVANTER DATEN IN EIN PROTOKOLL.....	69
ABBILDUNG 16: SKIZZIERUNG EINER SEQUENZIERTEN DATENERFASSUNG	70
ABBILDUNG 17: AUSPRÄGUNGEN ANÄSTHESIOLOGISCHER TEILPROZESSE IN ABHÄNGIGKEIT ZU VORGELAGERTEN TEILPROZESSEN	71
ABBILDUNG 18: VISUALISIERUNG DER PROZESSSICHT MITTELS EINER PROZESS-STRUKTUR-MATRIX (PSM) NACH [PF01]	72
ABBILDUNG 19: SEMANTISCHE STRUKTUR EINES PLIABLE OBJECTS	86
ABBILDUNG 20: ACTION-KATEGORIEN	97
ABBILDUNG 21: STATEMENT-STRUKTURIERUNG	99
ABBILDUNG 22: ABLAUFBESCHREIBUNG EINER AKTION	99
ABBILDUNG 23: SIMPLES BEISPIEL FÜR SCHEMA-, SCHEMATISIERTE UND INSTANZIIERTE PLIABLE OBJECTS.....	117
ABBILDUNG 24: GRAPHISCHES NOTATIONSELEMENT FÜR PLIABLE OBJECTS.....	119
ABBILDUNG 25: VISUALISIERUNG DER SONDER-PLIABLE-OBJECTS	119
ABBILDUNG 26: VISUALISIERUNGSBEISPIEL EINES SCHEMA- UND EINES SCHEMATISIERTEN PLIABLE OBJECTS	120
ABBILDUNG 27: VISUALISIERUNGSBEISPIEL EINES INSTANZIIERTEN PLIABLE OBJECTS.....	120
ABBILDUNG 28: VISUALISIERUNG VON GEBUNDENHEIT UND VERKNÜPFUNG BEI PLIABLE OBJECTS.....	130
ABBILDUNG 29: VISUALISIERUNG EINER TRANSITIVEN GEBUNDENHEIT.....	132
ABBILDUNG 30: MEHRFACHE REFERENZIERUNG EINES PLIABLE OBJECTS IN EINEM VERKNÜPFUNGSZYKLUS	135
ABBILDUNG 31: PRINZIP DER ABBILDUNG EINER KLASSE IN SPEICHERBEREICHE	139
ABBILDUNG 32: AUSWIRKUNGEN EINER DATENTYPENÄNDERUNG	140
ABBILDUNG 33: BEISPIEL EINER RELATIONALEN DATENBANKABBILDUNG	141
ABBILDUNG 34: PRINZIP DER FREMSCHLÜSSELBEZIEHUNG.....	142
ABBILDUNG 35: SEMANTISCHES PLIABLE-OBJECT-MODELL NACH [PR11].....	145
ABBILDUNG 36: ER-MODELL FÜR PLIABLE OBJECTS	147
ABBILDUNG 37: UML-DIAGRAMM EINER ZUSTANDSMASCHINE.....	153
ABBILDUNG 38: INTEGRATION DER ZUSTANDSMASCHINE INS PLIABLE OBJECT KLASSENMODELL	153
ABBILDUNG 39: AKTIVITÄTSDIAGRAMM ZUR ALLGEMEINGÜLTIGEN VERARBEITUNG DER ZUSTANDSMASCHINEN, DIE DURCH PLIABLE OBJECTS AUSGEPRÄGT WERDEN.....	155
ABBILDUNG 40: DARSTELLUNG DES ERZEUGUNGSPROZESSES.....	157
ABBILDUNG 41: DARSTELLUNG DES LÖSCHUNGSPROZESSES	159
ABBILDUNG 42: DARSTELLUNG DES VERÄNDERUNGSPROZESSES.....	160
ABBILDUNG 43: DARSTELLUNG DES ERFASSUNGSPROZESSES	161
ABBILDUNG 44: DARSTELLUNG DES AUSFÜHRUNGSPROZESSES	162
ABBILDUNG 45: DARSTELLUNG DES VERARBEITUNGSPROZESSES.....	163
ABBILDUNG 46: DARSTELLUNG DES ÜBERWACHUNGSPROZESSES	165
ABBILDUNG 47: SPEZIELLE KLASSENHIERARCHIE VON PLIABLE OBJECTS	170
ABBILDUNG 48: ELEMENTARE SOFTWARE-ARCHITEKTUR ZUR VERARBEITUNG VON PLIABLE OBJECTS.....	171
ABBILDUNG 49: SKIZZIERUNG VON ZUSTANDSÜBERGÄNGEN	174
ABBILDUNG 50: BASISKLASSENDIAGRAMM DER ANÄSTHESIE	179
ABBILDUNG 51: BASISINFORMATIONEN DES KERNDATENSATZES DER DGAI.....	182
ABBILDUNG 52: HAUPTINFORMATIONEN DES KERNDATENSATZES DER DGAI.....	183
ABBILDUNG 53: DIREKTE PROZESSABBILDUNG AM ANÄSTHESIEOBJEKT	184
ABBILDUNG 54: ERWEITERUNG DER STEUERUNGSFUNKTIONALITÄT UM EINEN KELLERAUTOMATEN.....	186
ABBILDUNG 55: INDIREKTE PROZESSABBILDUNG DER ANÄSTHESIE	187
ABBILDUNG 56: DYNAMISCH ERWEITERTES PLIABLE OBJECT PRÄOPERATIV	193

Tabellenverzeichnis

TABELLE 1: GRAFISCHE SYMBOLE DER STRUKTURIERTEN ANALYSE	9
TABELLE 2: LOGISCHE VERKNÜPFUNGEN VON DATENFLÜSSEN	9
TABELLE 3: DATA-DICTIONARY-NOTATION	10
TABELLE 4: GRAFISCHE SYMBOLE DES ENTITY-RELATIONSHIP-MODELLS	14
TABELLE 5: GRAFISCHE SYMBOLE DER EREIGNISGESTEUERTEN PROZESSKETTEN	32
TABELLE 6: GRAFISCHE SYMBOLE DER ERWEITERTEN EREIGNISGESTEUERTEN PROZESSKETTEN	34
TABELLE 7: ABBILDUNG OBJEKTORIENTIERTER KONZEPTE AUF BUSINESS OBJECTS (NACH [KO01])	37
TABELLE 8: MoSCoW – PRIORISIERUNG FÜR REGELN	106
TABELLE 9: RELATION DOMÄNE	148
TABELLE 10: RELATION ATTRIBUT	149
TABELLE 11: RELATION PLIABLE OBJECTS.....	149
TABELLE 12: BEZIEHUNGSRELATIONEN: LINKS ATTRIBUTE, RECHTS AKTIONEN UND REGELN	150
TABELLE 13: RELATION VALUE.....	150
TABELLE 14: RELATION ACTION	152
TABELLE 15: REGELSPEZIFISCHE RELATIONEN	154
TABELLE 16: BEISPIEL ZUR ABBILDUNG VON DOMÄNEN AUF EINZELWERTE.....	169
TABELLE 17: BEISPIEL FÜR DIE REFERENZIERUNG VON EINZELWERTEN ABGEBILDETER DOMÄNEN.....	170
TABELLE 18: REGELN DES ANÄSTHESIEOBJEKTES	185
TABELLE 19: REGELN DES PLIABLE OBJECTS PATIENT.....	188
TABELLE 20: REGELN DES PLIABLE OBJECTS DIAGNOSE.....	188
TABELLE 21: REGELN DES PLIABLE OBJECTS ANÄSTHESIE	189
TABELLE 22: REGELN DES PLIABLE OBJECTS PRÄOPERATIV	190
TABELLE 23: REGELN DES PLIABLE OBJECTS INTRAOPERATIV.....	190
TABELLE 24: REGELN DES PLIABLE OBJECTS POSTOPERATIV.....	191
TABELLE 25: DYNAMISCH ERWEITERTE REGELN DES PLIABLE OBJECTS PRÄOPERATIV	193

1 Einleitung

1.1 Motivation

Das Gesundheitswesen ist heutzutage ohne eine umfassende und geplante Verarbeitung von Informationen nicht mehr denkbar (vgl. [Am03]). Eine Vielzahl an Daten muss bei der Versorgung von Patienten gespeichert und verarbeitet werden. Es handelt sich um administrative Stammdaten, um diagnostische und therapeutische Daten und um Kosten- und Leistungsdaten. Zur Verwaltung dieser Daten muss eine geeignete Informationsinfrastruktur etabliert werden, die eine optimale Unterstützung jeglicher Beteiligten bietet, ohne dass die Versorgung der Patienten darunter leidet. Beim Management dieser Daten leisten medizinische Informationssysteme technische Unterstützung.

Besonders stark ist die zu verarbeitende Informationsintensität (oder –dichte) im medizinischen Fachgebiet der Anästhesie. Die medizinischen Informationen in der Anästhesie besitzen einen hohen Grad an Individualität. Die Art und Anzahl der zu dokumentierenden anästhesiologischen Informationen orientieren sich an der fachlichen Ausrichtung der Klinik, z.B. Unfall-, Herz- oder Kinderklinik, der die Anästhesieabteilung angehört. So sind vielfältige Klassifikationen nicht nur möglich, sondern gängige Praxis.

Zur besseren Abgrenzung werden medizinische Informationssysteme in der Anästhesie als Anästhesie-Dokumentationssysteme bezeichnet. Deren Hauptaufgabe ist die Unterstützung bei der Dokumentation des Informationsaufkommens. Sie sollen die Anforderung erfüllen, die Individualität der anästhesiologischen Informationen und die damit verbundene Klassifikation zu visualisieren. Dadurch sind Transparenz und Nachvollziehbarkeit ärztlicher Entscheidungen gewährleistet. Diese Transparenz bzw. Nachvollziehbarkeit wird u.a. im Fachgebiet der Anästhesie teilweise vom Gesetzgeber, teilweise von den Anästhesisten gefordert. Einerseits, um Rechtssicherheit für den Fall zu bekommen, falls eine Operation ein juristisches Nachspiel hätte, andererseits, um die Qualitätssicherung der Anästhesie zu gewährleisten.

1.2 Zielsetzung

Eine Aufgabe der nationalen Gesundheitssysteme ist es, den medizinischen Fortschritt und die damit verbundenen neuesten Erkenntnisse schnell und flächendeckend umzusetzen. Für die schnelle Überführung des enormen Wissenszuwachses in die Praxis gibt es Selbstverwaltungsorgane, Fachgesellschaften und medizinische Versorgungszentren, die Lösungen für dieses Problem erarbeiten. Peter Haas (vgl. [Ha05]) nennt beispielsweise Disease Management, Managed Care und Evidence Based Medicine sowie die stetige bessere Verzahnung zwischen medizinischen Versorgungszentren. Ein gemeinsames Ziel der Lösungen ist die Sicherung bzw. das Anstreben einer stetigen Fortbildung des Personals sowie die Anpassung medizinischer Informationssysteme.

Offen bleibt jedoch die Frage, wie die medizinischen Informationssysteme derart angepasst werden können, dass neue Erkenntnisse und Behandlungsmethoden auch erfasst, technisch unterstützt und ausgewertet werden können - vor allem, wenn die neuen Erkenntnisse und Behandlungsmethoden zum Zeitpunkt der Inbetriebnahme der Informationssysteme noch gar nicht bekannt oder absehbar waren.

Die Beantwortung dieser Frage ist das Ziel dieser Arbeit.

1.3 Aufbau der Arbeit

In Kapitel 2 werden die Grundlagen und die Begriffswelt der Modellierung von Informationssystemen erläutert. Es wird insbesondere auf die objektorientierte Modellierung und auf das Verständnis und die Definition von „Business Objects“ nach der Object Management Group (OMG) eingegangen. Diese Definition war der Ideengeber für diese Arbeit und bildet die Grundlage für die Erarbeitung der Pliable Objects. Um eine klare Abgrenzung zum Begriff „Business Objects“, der im Bereich der Wirtschaftswissenschaften verwendet wird (vgl. [Kö03]), zu bekommen, wurde der Begriff „Pliable Object“ ausgewählt. Das Wort „Pliable“ ist dem englischen Sprachwortschatz entnommen und bedeutet biegsam, nachgiebig oder formbar (vgl. [La87]).

Kapitel 3 enthält einen erläuternden Abriss über die Anästhesie. Im Vordergrund steht die Anforderung, die Dokumentation von Anästhesien zu realisieren. Mögliche Lösungsansätze mittels Dokumenten-Management-Systemen werden betrachtet, wobei herausgearbeitet wird, dass diese Systeme keine ausreichende Lösung bieten. Nach Erörterung der Probleme, die mittels Informationssystemen zu lösen sind, wird gezeigt, dass ein standardisiertes Informationssystem die individuelle, eventuell mobile Anästhesiedatenverarbeitung nicht dokumentieren kann, da ein nach objektorientierten Gesichtspunkten gestaltetes Anästhesie-Informationssystem zur Laufzeit nicht dynamisch formbar ist.

In Kapitel 4 wird das Konzept der Pliable Objects entwickelt, die über die Dynamik verfügen, an anwenderspezifische individuelle Anforderungen anpassbar zu sein. Im Kern wird gezeigt, dass Pliable Objects eine besondere Form der Selbstbeschreibung bieten, mittels derer die geforderte Anpassbarkeit abgedeckt wird. Einige Ergebnisse, die das Kapitel enthält, sind teilweise in [Pr05] und [Pr11] veröffentlicht worden.

Kapitel 5 überführt das Konzept der Pliable Objects in eine relationale Darstellung. Darauf aufbauend wird ein allgemeines Informationssystem zur Verwaltung der Pliable Objects erarbeitet. Teil der Verwaltung sind realisierbare Zugriffsstrategien, anhand derer aufgezeigt werden kann, dass ein geeignetes Arbeiten mit Pliable Objects ermöglicht werden kann. Einige Ergebnisse, die das Kapitel enthält, sind teilweise in [Pr11] und [Pr12] veröffentlicht worden.

In Kapitel 6 wird die Spezialisierung des allgemeinen Informationssystems in ein Modell eines Anästhesie-Informationssystems vorgenommen. Nach der Modellierung und Entwicklung von Pliable Objects werden deren Möglichkeiten zur Unterstützung des Aufbaus von Informationssystemen im medizinischen Anwendungsgebiet der Anästhesie erarbeitet. Teilweise sind einige Ergebnisse, die das Kapitel enthält, in [Pr12] veröffentlicht worden.

Kapitel 7 bildet schließlich mit der Zusammenfassung und dem Ausblick den Abschluss der Arbeit. Es zeigt, dass mittels Pliable Object das Ziel erreicht wird, die Spezifikation eines Informationssystems der Anästhesie zu erlauben, welches primär die Erfassung und Dokumentation von Anästhesien unterstützt. Dieses Informationssystem erfüllt die Anforderungen der Anästhesie aus Kapitel 3.

2 Grundlagen der Modellierung von Informationssystemen

Bevor eine Lösung für die Anpassung medizinischer Informationssysteme an neue medizinische Erkenntnisse und Behandlungsmethoden vorgestellt werden kann, muss ein Grundverständnis über Informationssysteme erarbeitet werden. Im Folgenden werden Informationssysteme und verschiedene Modelle, die zu ihrer Entwicklung nutzbar wären, abstrakt betrachtet.

2.1 Einführung in das Leistungsspektrum von Informationssystemen

Historisch gesehen stand der Begriff *Informationssystem* anfänglich für ein Anwendungssystem, welches aus einer Softwareapplikation bestand, die auf einer Datenbank arbeitete. Solch ein Anwendungssystem übernahm Aufgaben zur Dokumentation und zur Archivierung. Weiterentwicklungen im Bereich der Informationssysteme ermöglichten die Erschließung neuer Tätigkeitsfelder, womit sich das Verständnis hinter dem Begriff *Informationssystem* gewandelt hat. Die historische Entwicklung wird umfangreich in [BMS06] dargestellt.

Heutige Informationssysteme realisieren soziotechnische („Mensch-Maschine-“) Systeme. Sie umfassen menschliche und maschinelle Komponenten und verfolgen das Ziel, Informationen optimal bereit zu stellen und die Kommunikation zwischen den Komponenten optimal zu unterstützen. Geprägt werden die Informationsbereitstellung und die Kommunikation von wirtschaftlichen Kriterien, weshalb mit Informationssystemen häufig betriebliche Informationssysteme bezeichnet werden, die nahezu alle Aspekte betrieblicher Abläufe unterstützen. Betriebliche Informationssysteme stellen einen Hauptforschungspunkt in der Wirtschaftsinformatik dar (vgl. [Kr00], [Ko01], [Sc02]).

Betriebliche Informationssysteme nehmen Einfluss auf die Aufgabenverrichtung des Einzelnen und die Betriebsorganisation insgesamt. Die Ausgangsbasis für die Spezifikation der Informationssysteme bildet das Modell eines Unternehmens oder des als relevant definierten Unternehmensausschnittes. Aus dem Modell werden zwei Kernbereiche konkretisiert; - zum einen die betrachtungsrelevanten Aufgaben und deren Verknüpfungen untereinander mit Berücksichtigung definierter betrieblicher Aufgabenbereiche, zum anderen die Aufgabenträger. Bei den Aufgabenträgern wird unterschieden zwischen den personellen Aufgabenträgern, die in der Unternehmenshierarchie verschiedene Rollen einnehmen, und den maschinellen Anwendungssystemen, die verschiedene Funktionen bereitstellen oder übernehmen.

Das maschinelle Anwendungssystem selbst ist ein computergesteuertes Informationssystem, welches als Subsystem des betrieblichen Informationssystems zu sehen ist. Seine Aufgabe ist es, jedem personellen Aufgabenträger die für seine Aufgabenerfüllung notwendigen Informationen zur Verfügung zu stellen bzw. verfügbar zu machen (vgl. [St07]). Um dies zu gewährleisten, müssen Operationen zur Eingabe, zur Verarbeitung und Ausgabe von Daten existieren. Die Eingabe von Daten umfasst neben der Entgegennahme der Daten auch die Speicherung inklusive einer Archivierung der Daten. Die Verarbeitung beinhaltet Funktionalitäten zur Änderung, Transformation und Löschung der Daten sowie zur Verknüpfung von Daten. Die Ausgabe beinhaltet neben der Ausgabe der Daten im Sinne einer Einsichtnahme auch die selektive Auswahl und gezielte Übertragung bestimmter Daten an angeschlossene Systeme. Alle Operationen müssen sicher, zuverlässig, robust und effizient sein. Hinsichtlich der Interaktion sind ergonomische Anforderungen zu erfüllen. Die

anspruchsvollste Anforderung ist allerdings die Flexibilität. Alle Operationen sollen flexibel an technische Änderungen und an geänderte Anforderungen anpassbar sein.

Während sich betriebliche Informationssysteme an die betrieblichen Unternehmensstrukturen anpassen, ist die technische Realisierung des maschinellen Anwendungssystems als Subsystem gleich geblieben. Die Kernfunktionalität des Anwendungssystems wird durch ein Datenbank-Managementsystem (DBMS) abgedeckt. Das Datenbank-Managementsystem übernimmt die Verwaltung der Daten, die von einer Softwareapplikation verwendet werden. Diese Softwareapplikation wiederum bildet eine Datenverarbeitungsebene, die das Interface bereitstellt, um mit Bedienern zu interagieren.

Maschinelle Anwendungssysteme selbst werden in der Informatik für sich betrachtet. Dort wird häufig die technische Realisierung im Zusammenhang mit dem Gebrauch des Begriffs Informationssystem verstanden. Aus Informatiksicht sollen Informationssysteme zur Lösung von Problemen beitragen, die in einem konkreten Umfeld auftreten. Dieses Problemumfeld stellt einen Ausschnitt der Realität dar und verfügt über Datenmaterial. Dieses Datenmaterial muss vom Informationssystem erfasst, verarbeitet und ausgewertet werden. Um dies zu gewährleisten, wird bereits vor der Entwicklung des Informationssystems das Problemumfeld analysiert. Beim Modellieren des Informationssystems wird das Datenmaterial aufbereitet, d.h. das schon existierende Datenmaterial wird erhoben, analysiert, strukturiert und letztlich in einem Modell abgebildet. Dieses Modell beschreibt den Ausschnitt der Realität, wie er ist bzw. wie er verstanden wird. Es dient als Grundlage zur Entwicklung eines Konzeptes für das Informationssystem. Unter dem Konzept wird ein Sollkonzept verstanden, welches eine vereinfachte Vorstellung darüber enthält, wie ein bestimmter Ausschnitt der Realität aussehen soll bzw. wie er aussieht, wenn die Vorstellung verwirklicht wird (vgl. [RP02], Seite 1046). Während der Entwicklung wird das Konzept in das Informationssystem überführt. Nach [RP02], Seite 1041 ist ein Informationssystem wie folgt definiert:

„Ein Mensch/Aufgabe/Technik-System zum Beschaffen, Herstellen, Bevorraten und Verwenden von Information, kurz: ein System zur Informationsproduktion und Kommunikation für die Deckung von Informationsnachfrage.“

Im Vordergrund dieser Definition steht der Informationszweck, d.h. ein Informationssystem soll die benötigte Information zur richtigen Zeit, am richtigen Ort und in der richtigen Qualität zur Verfügung stellen. Moderne Informationssysteme besitzen zusätzlich die Eigenschaften Offenheit, Dynamik, Kompliziertheit und Komplexität. Ergänzend sind auch die Eigenschaften für die Konstruktion des Managements und die Nutzung des Informationssystems von Bedeutung (vgl. [RP02], Seite 1042ff). Diese Eigenschaften *„bilden Objekte der Wirklichkeit mit Merkmalen ab, die für die definierten Zwecke und Ziele von Bedeutung sind“* (vgl. [RP02], Seite 1043).

2.1.1 Relevante Dimensionen in der Entwicklung von Informationssystemen

Auch wenn die technische Realisierung gleichgeblieben ist, hat sich im Laufe der Zeit die interne Architektur der Anwendungssysteme gewandelt. Die Hardware ist leistungsfähiger, Datenbanken sind effizienter geworden, Software ist effektiver und die Bediener sind anspruchsvoller geworden. Die gestiegenen Ansprüche der Bediener führen zu neuen Anforderungen an die Systeme, wobei im Kern die Ausnutzung der Leistungs-, Effizienz- und Effektivitätssteigerung steht. Das sich ergebende Problem der Entwickler ist, die Komplexität in den Griff zu bekommen (vgl. [AMA05], [Hi11], [LM98]).

Allgemein durchgesetzt hat sich das Bestreben, bei der Entwicklung von Informationssystemen planvoll vorzugehen (vgl. [Hü01], [IK08], [KKB08], [Ta06], [SW07]). In der Regel wird damit begonnen, das Problemumfeld zu analysieren und zu strukturieren, um die Komplexität der Probleme zu erfassen. Anschließend wird eine Lösung erarbeitet und umgesetzt. Abstrakt betrachtet werden dabei vier Phasen durchlaufen:

1. *Anforderungsanalyse*: Zuerst sind die genauen Anforderungen zu erfassen. Das eigentliche Ziel ist die Bereitstellung eines Informationssystems, das den Ansprüchen der Endanwender genügt. Um solch ein System realisieren zu können, müssen das Anwendungsgebiet bekannt und die konkreten Anforderungen verstanden sein.
2. *Systemanalyse*: Anschließend kann das System entworfen werden. Die konkreten Anforderungen werden analysiert und strukturiert, um so die Basis für einen Systementwurf zu bekommen, der die technische Realisierung beschreibt.
3. *Konstruktion*: Im dritten Schritt wird das System realisiert. Die technische Beschreibung dient als Vorlage für die Implementierung und das Informationssystem wird konstruiert.
4. *Abnahme*: Im letzten Schritt wird das realisierte System an den Kunden übergeben. Es werden Tests vorgenommen, ob das System alle Anforderungen erfüllt. Ist dies der Fall, wird das System abgenommen und der Entwicklungsprozess endet.

Das Hauptziel der ersten beiden Phasen ist eine Beschreibung der technischen Realisierung. Es handelt sich dabei um eine Lösungsspezifikation, die selbst wieder in vier informationstechnische Dimensionen unterteilt ist. Jede einzelne Dimension stellt eine Sicht dar, die in Bezug auf die Informationssystementwicklung entweder die Aufgaben, die Objekte, die Funktionen oder die Prozesse beschreibt (vgl. [Vo00], [Sp05]):

1. Die Aufgabensicht charakterisiert die zu verrichtenden Aufgaben an Objekten. Aufgaben sind zusammengehörige und relativ geschlossene Einheiten, die von Stellen bzw. Personen zu erledigen sind.
2. Die Objektsicht bestimmt die zu bearbeitenden und benötigten Objekte. Unter Objekten wiederum werden zusammenhängende Einheiten von Bearbeitungselementen verstanden.
3. Die Funktionssicht beschreibt verwandte und ähnliche Einheiten von Elementarfunktionen. Solch eine Einheit kann beispielsweise eine Programmroutine beschreiben, deren Zweck eine bestimmte Datenverarbeitung oder -operation ist.
4. Die Prozesssicht enthält Beschreibungen von Operationsfolgen, die als Einheiten Unterstützung bei der raum-zeitlichen Erfüllung der Aufgaben leisten.

Innerhalb der vier Sichten werden die Anforderungen an das System quantitativ und qualitativ analysiert und beschrieben (vgl. [PR09]). Die Ergebnisse werden in ein Modell des Informationssystems überführt. Dieses Modell beschreibt konzeptionell, d.h. neutral und hard- und softwareunabhängig, Aspekte der Anwendung (vgl. [St73]). Es definiert sich als Architektur, bestehend aus Daten-, Funktions- und Organisationsmodell und definiert ferner den Ist-Zustand sowie die Soll-Vorstellungen.

2.1.2 Basisaspekte der Modellierung

Verallgemeinernd ist ein Modell eine Abstraktion und Interpretation des Ausschnitts der Realität, die in der zugehörigen Sicht erfasst werden. Das Modell selbst kann aufgrund seiner Unabhängigkeit zu Hard- und Software einerseits verallgemeinert, andererseits verfeinert werden. Dies ermöglicht die Entdeckung von Optimierungspotenzialen, die im Rahmen der Implementierung der Anwendung genutzt werden könnten.

Das Modell selbst ist eine vereinfachte, strukturierte Darstellung des Gesamtkonzeptes und dient als Mittel zur Beschreibung der zu lösenden Aufgaben. Hierzu sollte das Modell eine formale oder semi-formale Strukturierungsvorschrift beinhalten. Das Modell wird so gesehen in der Strukturierungsvorschrift verfasst, indem die Anforderungen, die an das zu entwickelnde Informationssystem gestellt werden, in der Strukturierungsvorschrift erfasst werden.

Eine formale Strukturierungsvorschrift verfügt über einen vorgegebenen Sprachumfang. Dies bedeutet, dass die Konstrukte des Modells einer Syntax genügen und eine Semantik zur Festlegung ihrer Bedeutung besitzen. Semi-formale Strukturierungsvorschriften hingegen legen den Fokus auf die Strukturierung und Gewichtung von zu realisierenden Aufgaben. Häufig verfügen semi-formale Modelle über graphische Hilfsmittel zur Aufgabendefinition. Semi-formale Modelle unterstützen die genaue Analyse der Anforderungen, während formale Modelle auf die Bedürfnisse der Projektplanung zugeschnitten sind.

Wenn das Gesamtkonzept in einem Modell zum Ausdruck gebracht ist, bildet das Modell die Ausgangsbasis, welche mit geeigneten systematischen Mitteln und Werkzeugen in das zu entwickelnde Informationssystem zu überführen ist.

Um ein Modell zu erarbeiten, verfügt jedes Modell über eine Methodik, die beschreibt, wie ein Entwickler vorgehen soll, um zu dem Modell zu kommen. Die Methodik beinhaltet die systematische Anwendung verschiedener Beschreibungsmittel auf die Anforderungen. Die Anzahl der Beschreibungsmittel ist wiederum davon abhängig, wie die Sichten im Modell unterstützt werden. In Anlehnung an die Sichten werden Funktions-, Daten- und Prozessmodelle unterschieden.

Datenmodell

Ein Datenmodell dient nach [Vo00] allgemein der Herstellung einer Abstraktion zum Zwecke der Beschreibung eines „Problems“ im Sinne eines gegebenen „Realwelt-Ausschnitts“ und gleichzeitig von den Einzelheiten der physischen Speicherung. Auf Basis eines Datenmodells wird ein Informationssystem erarbeitet und erstellt. Das Datenmodell enthält die notwendigen Beschreibungen aller relevanten Daten, ihre Strukturen und Beziehungen zueinander. Diese Charakterisierung von Daten ist die Grundlage der Charakterisierung der Nutzung der Daten, d.h. der Definition der Verwendung der Daten durch spezifische Operationen. Die spezifischen Operationen sind ebenfalls Teil des Datenmodells und bilden Aktionen und Ereignisse des „Realwelt-Ausschnitts“ ab.

Nach [Ot09] lassen sich Datenmodelle grob in zwei Klassen einteilen:

„Zum einen in semantische Datenmodelle, die sich einfach beschreiben lassen und einen hohen semantischen Gehalt besitzen, der eine recht genaue Abbildung ermöglicht. Allerdings sind semantische Datenmodelle sehr schwierig zu implementieren.

Zum anderen in Standard-Datenmodelle, die einfach zu implementieren sind. Der Nachteil von Standard-Datenmodellen ist, dass sie ein sehr hohes Abstraktionsvermögen erfordern, die reale Situation in die vorgegebenen und von systembedingten technischen Besonderheiten geprägten Begriffs- und Konzeptkategorien gepresst werden.“

Unter den Standard-Datenmodellen werden Datenstrukturmodelle verstanden, die den Fokus entweder auf die Spezifikation der Datenstruktur oder der Datenoperationen legen. Stehen die Datenstrukturen im Fokus, enthält das Modell relativ komplexe, dafür jedoch spezifische Daten, auf denen mittels einfacher, relativ allgemeingültiger Zugriffsoperationen gearbeitet werden kann. Wenn die Datenoperationen im Fokus stehen, besitzen die Daten eine einfache, nahezu standardisierte Struktur, während die Operationen komplex sind und spezifische Algorithmen enthalten. Bei der Transformation in solche Daten geht im Allgemeinen die Bedeutung der Daten verloren, die den Beziehungen zwischen den abgelegten Daten entspringen.

Ein derartiger Verlust wird bei semantischen Datenmodellen vermieden. Bei semantischen Datenmodellen sind neben den Daten auch die Beziehungen der Daten untereinander Teil des Datenmodells.

Das hierarchische Datenmodell und das Netzwerkmodell sind Beispiele für Standard-Datenmodelle. Datenrelationsmodelle und objektorientierte Modelle sind zwei Vertreter der Klasse semantischer Datenmodelle. Datenrelationsmodelle fokussieren fachliche Informationen und überführen sie beispielsweise ins Entity-Relationship-Modell. Objektorientierte Modelle verwenden das objektorientierte Paradigma, um den Realitätsausschnitt, der von Informationssystemen abgebildet werden soll, durch Objektwelten zu modellieren.

Funktionsmodell

Neben der Modellierung der Daten ist auch die Funktionalität der Operationen auf diesen Daten zu spezifizieren. Hierzu werden funktionsorientierte Modellierungsansätze verwendet, die eine Funktionssicht und häufig auch eine Interaktionssicht auf das Informationssystem erarbeiten. In der Funktionssicht wird die funktionelle Aufgabenverrichtung im Sinne der Datenverarbeitung mittels mehrerer Bausteine des Informationssystems spezifiziert. Einfluss auf die Aufgabenverrichtung nehmen die Kommunikationskanäle, die die Basis der Interaktion zwischen personellen Aufgabenträgern und dem Anwendungssystem bilden. Häufig werden die Kommunikationskanäle in einer separaten Interaktionssicht spezifiziert.

Vom Anwendungssystem wird gefordert, dass es die personellen Aufgabenträger bestens unterstützt, damit diese ihre an sie übertragenen Aufgaben optimal erledigen können. Die Erfüllung der Anforderung erfordert eine ergonomische Interaktionsschnittstelle zu technischen Anwendungssystemen. Die Beschreibung geeigneter Ausführungsbedingungen in der Interaktionsschicht nimmt derart Einfluss auf die Funktionssicht, dass die funktionellen Datenverarbeitungsaufgaben so spezifiziert sein müssen, um eine optimale Unterstützung der Interaktion zu gewährleisten.

Beispiele für die funktionsorientierten Modellierungsansätze sind Strukturierte Analyse (SA), Structured Analysis and Design Technique (SADT), Object Modeling Technique (OMT) oder Object-Oriented Software Engineering (OOSE) (vgl. [Ba00], [RP02]).

Prozessmodell

Daten-, Funktions- und Interaktionssicht stellen keine vollständige Modellierung der Aufgabenstruktur zur Verfügung. Sie betonen Aufgabenmerkmale des Anwendungssystems aus statischer Perspektive. Betriebliche Informationssysteme bilden Unternehmensmodelle ab, die von einer starken Dynamik im Unternehmensumfeld geprägt sind. Diese Dynamik muss sich auch im Anwendungssystem widerspiegeln. Zur Modellierung der dynamischen Perspektive wird eine Vorgangssicht spezifiziert, die den dynamischen Ablauf von zielorientierten Verrichtungsaufgaben beschreibt. Dabei steht eine übergeordnete Funktionsdurchführung im Mittelpunkt. Geschäftsorientierte Ansätze modellieren einen ereignisgesteuerten Ablauf. Der Ablauf stellt verschiedene einzelne Verrichtungsaufgaben in eine Sequenz, um damit die Reihenfolge ihrer Abarbeitung zu spezifizieren. Indirekt nehmen geschäftsprozessorientierte Ansätze damit Bezug auf die aufgabenspezifischen Objekte in der Datensicht und in der Funktionssicht.

Ein Beispiel für einen geschäftsprozessorientierten Ansatz bilden Ereignisgesteuerte Prozessketten (EPK) (vgl. [Ko01], [Kr00], [RP02]).

Im Folgenden werden die Aufbereitungen von Daten-, Funktions- und Prozessmodellen in etablierten Analysemodellen vorgestellt.

2.2 Methode der Strukturierten Analyse

Die Strukturierte Analyse (SA) wurde Anfang der 1970er Jahre von Tom DeMarco entwickelt [De78]. Sie stellt eine Methode dar, die während der Systemanalyse eingesetzt wird und die Entwicklung von Modellen für Anwendungen auf verschiedenen Granularitätsstufen erlaubt. Die Strukturierte Analyse beinhaltet selbst kein neues Modellkonzept. In der SA werden verschiedene Basiskonzepte in einer Methodenklasse zusammengefasst und zur Erarbeitung eines Funktionsmodells verwendet (vgl. [Ki04], [Ba00]). Es handelt sich dabei um

- hierarchisch angeordnete Datenflussdiagramme (DFDs),
- *Data-Dictionary*-Einträge (DDs) und
- Mini-Spezifikationen (MiniSpecs).

2.2.1 Datenflussdiagramme

Die SA besitzt eine einfache Modellnotation. Sie basiert auf einer einfachen Syntax, um Anforderungen und Ideen über bestehende oder zu entwickelnde Systeme zu sammeln und zu strukturieren. Die Syntax definiert in erster Linie Datenflussdiagramme, die Systemaktivitäten und deren Beziehungen über Schnittstellen zur Umwelt darstellen. Die Notation innerhalb der DFDs verwendet die Symbole, die in Tabelle 1 gezeigt werden.

Prozesse

Die Prozesse haben die Aufgabe, Eingangsdaten in Ausgangsdaten zu transformieren. Dies geschieht durch einen Algorithmus, der durch den Prozess repräsentiert wird. Prozesse werden im Modell durch Kreise dargestellt. In den Kreis wird ein Name eingetragen, um den Prozess namentlich eindeutig zu kennzeichnen. Der Name wird ferner um eine Nummer ergänzt, um ein hierarchisch eindeutiges Schema zu erhalten (s.u.).

Datenspeicher

Datenspeicher stellen einen permanenten passiven Speicher für Daten dar. Sie nehmen Daten in Form von Werten entgegen, speichern sie und stellen die gespeicherten Werte zur Verfügung. Die Entgegennahme eines neuen Wertes überschreibt einen bereits gespeicherten Wert. Datenspeicher sind abstrakt, verzögerungsfrei und unendlich groß. Sie sind passiv, da die Daten von Prozessen aktiv abgeholt bzw. aktiv von Prozessen in den Speicher geschrieben werden müssen. Im Modell werden Datenspeicher durch zwei parallele Linien dargestellt.


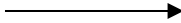

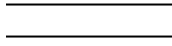
Logischer Systembestandteil	Symbol
Prozess	
Datenfluss	
Terminator	
Datenspeicher	

Tabelle 1: Grafische Symbole der Strukturierten Analyse

Datenflüsse

Datenflüsse kennzeichnen Pfade, auf denen Informationen zwischen

- Prozessen
- Prozessen und Datenspeichern oder
- Prozessen und Terminatoren

uni- oder bidirektional transportiert werden. Die Richtung des Datenflusses wird auch festgelegt. Datenflüsse werden mit eindeutigen Namen bezeichnet, die Rückschlüsse auf ihren Inhalt zulassen. Nur Datenflüsse von und zu Datenspeichern werden nicht beschriftet, da der Datentyp der Daten bereits durch den Datenspeichernamen gekennzeichnet ist. Bei Bedarf werden Datenflüsse prozedural verknüpft; Tabelle 2 zeigt die unterstützten Verknüpfungssymbole.

Symbol	Logische Verknüpfung
*	UND, Konjunktion
+	ODER, Disjunktion
⊕	Exklusiv ODER

Tabelle 2: Logische Verknüpfungen von Datenflüssen

Hierbei ist jedoch Vorsicht geboten, da das resultierende Diagramm zu komplex werden kann und dadurch seine Interpretation aufwendig wird. Im Modell werden Datenflüsse als gerichtete Kanten visualisiert. In Bezug auf Datenspeicher zeigt die Richtung an, ob aus dem Datenspeicher gelesen oder in ihn geschrieben wird. Ist beides möglich, wird ein Doppelpfeil verwendet.

Terminatoren

Terminatoren sind Objekte der Umwelt. Sie stehen für Personen, Organisationen oder ganze Systeme, die Daten in das beschriebene System senden. Der Empfang von Daten aus dem System ist Terminatoren auch gestattet. Terminatoren werden im Modell als Rechteck dargestellt. Bezogen auf das Modell werden Terminatoren als Datenquelle und/oder –senke verwendet. Der Aufbau der Terminatoren und deren interne Arbeitsweise haben keinen Einfluss auf das betrachtete Modell und werden nicht detaillierter wiedergegeben. Sie werden als Schnittstellen des Systems zu seiner Umwelt und als gegeben betrachtet.

2.2.2 Data Dictionary

Die Beschreibung der Zusammensetzung eines Datenflusses erfolgt im Data Dictionary (DD) bzw. dem Datenkatalog. Das Data Dictionary realisiert eine Art zentrales Verzeichnis, in welchem alle Informationen über die Struktur der im DFD entworfenen Daten und ihrer Verwendung gespeichert sind. Hierzu werden Datenelemente verwendet, die Daten darstellen, die nicht weiter zerlegt werden können. Die physikalische Repräsentation der Daten ist dabei nicht Bestandteil des Data Dictionary's. Die Datenelemente selbst werden in einer modifizierten Backus-Naur-Form beschrieben. Die Notation ist wie folgt festgelegt:

Symbol	Name
=	Zusammensetzung bzw. Äquivalenz
+	Verkettung bzw. Aufzählung
[]	Selektion
{}	Iteration
()	Option

Tabelle 3: Data-Dictionary-Notation

2.2.3 Mini-Spezifikationen

Mini-Spezifikationen beschreiben Prozesse, die nicht weiter verfeinert werden sollen. Sie stellen selbst eine primitive Beschreibung in einer Art Pseudocode dar. Im Pseudocode werden primitive Transformationen auf unterster Ebene ausgedrückt, die Eingabedaten oder Eingabedatenflüsse in Ausgabedaten bzw. Ausgabedatenflüsse überführen.

2.2.4 Generelle Vorgehensweise der Strukturierten Analyse

Das generelle Prinzip der Strukturierten Analyse ist, mittels eines Hierarchiekonzeptes eine Baumstruktur zu erarbeiten, die iterativ ein Datenflussdiagramm immer wieder verfeinert, bis die Mini-Spezifikationen vorliegen.

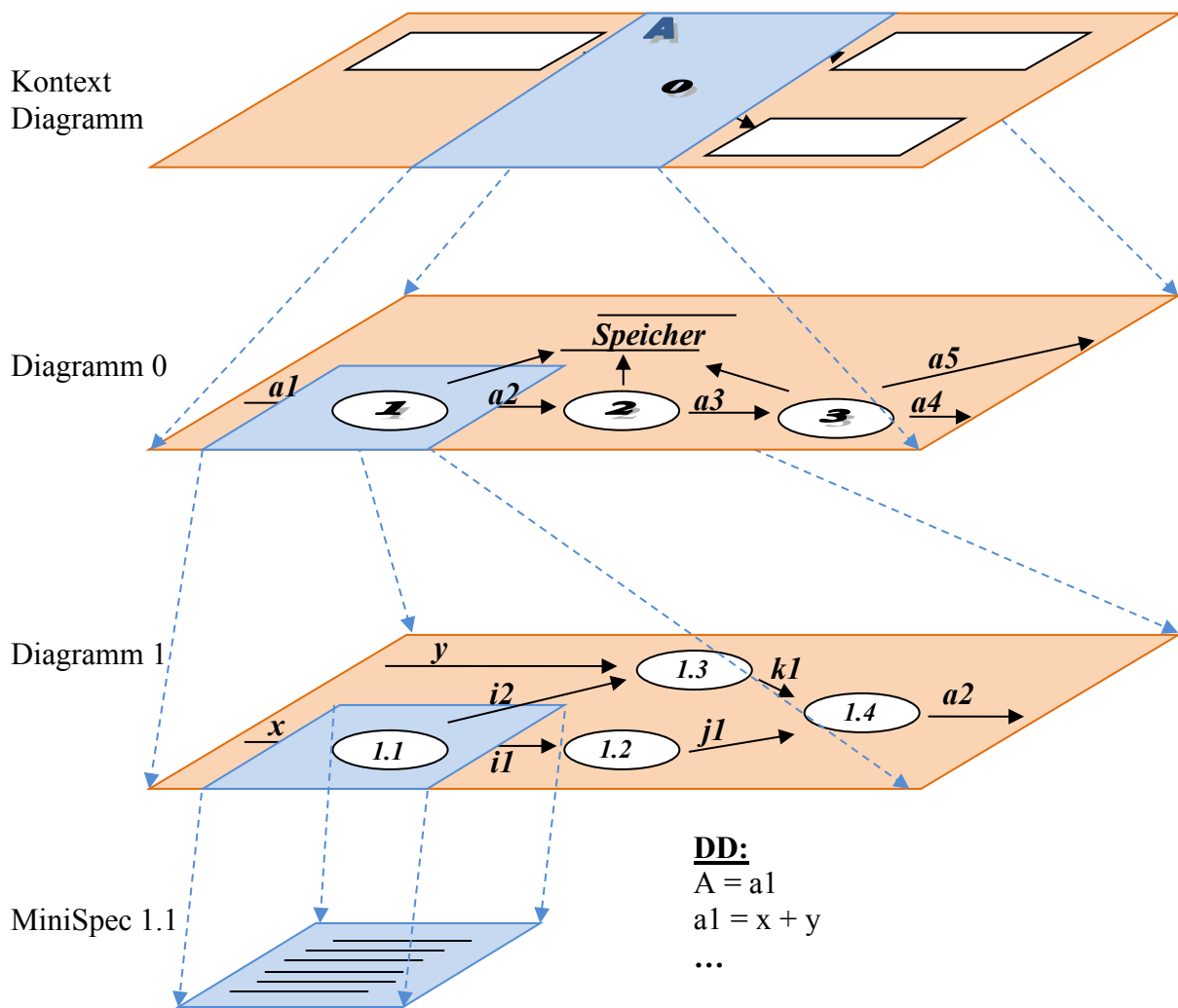


Abbildung 1: Hierarchiekonzept der Strukturierten Analyse (vgl. [Ba00])

Die oberste Ebene und somit die Ausgangsbasis der Datenflussdiagramm-Hierarchie bildet das Kontext-Diagramm (vgl. Abbildung 1). Es ist ein abstraktes Datenflussdiagramm, das die Schnittstellen des zu modellierenden Systems mit seiner Umwelt beschreibt. Alle Schnittstellen des Gesamtsystems zur Außenwelt werden ausschließlich im Kontextdiagramm definiert.

Das betrachtete System selbst wird in dem Kontext-Diagramm durch einen einzigen Prozess dargestellt, der mittels der Datenflüsse mit den Schnittstellen bzw. Terminatoren kommuniziert. Als einziges Datenflussdiagramm darf das Kontext-Diagramm keinen Datenspeicher beinhalten.

Über einen iterativen Prozess wird der Prozess des Kontext-Diagramms verfeinert; d.h. der Prozess wird durch ein Datenflussdiagramm modelliert. Ein Datenflussdiagramm (DFD) besteht aus Datenflüssen, die angeben, wie Daten im System fließen, und aus Prozessen, die Eingabedaten in Ausgabedaten umwandeln.

Dieses erste DFD ist das Diagramm 0. Es zerlegt den Prozess in Teilprozesse und verfeinert die Datenflüsse des Kontextdiagramms. Ergänzend können in diesem DFD Datenspeicher

definiert werden, die als Ablagemöglichkeit für Daten dienen. Die Datenflüsse, die mit einem Datenspeicher in Verbindung stehen, spezifizieren lesende und schreibende Zugriffe.

Die Verfeinerung von Datenflüssen wird im Data Dictionary hinterlegt, ebenso wie eingeführte Datenspeicher. Das Data Dictionary stellt den Zusammenhang zwischen den Datenflüssen der einzelnen DFDs her. Alle Datenflüsse eines untergeordneten DFDs müssen im übergeordneten DFD vollständig oder als Teilkomponente hinterlegt sein.

Nachdem das erste DFD erstellt worden ist, startet ein iterativer Vorgang über alle Teilprozesse des DFD. Jeder Teilprozess dieses DFDs wird durch ein einzelnes DFD verfeinert. Dabei ist darauf zu achten, dass alle Eingabe-/ Ausgabe-Datenflüsse des zu verfeinernden Prozesses auch im verfeinerten Prozess vorkommen.

Die verfeinerten, d.h. untergeordneten Diagramme werden als Kinder, die übergeordneten Diagramme als Eltern bezeichnet. Jedem Datenflussdiagramm wird eine hierarchisch gebildete Prozessnummer zugeordnet. Sie setzt sich zusammen aus einer Diagrammnummer, gefolgt von einem Dezimalpunkt und endet mit einer lokalen DFD-Nummer.

Wenn alle Teilprozesse des Diagramms 0 durch DFDs verfeinert wurden, wird der iterative Vorgang für jedes erstellte DFD erneut durchgeführt. Dieser Vorgang wird solange wiederholt, bis sich die Prozesse nicht weiter verfeinern lassen. Diese Prozesse sind nun durch Mini-Spezifikationen zu beschreiben. Die Beschreibung wird durch Pseudocodes, Entscheidungstabellen oder Entscheidungsbäume angegeben.

Wenn für keinen Prozess mehr eine Verfeinerung durchgeführt werden kann, folglich alle Mini-Spezifikationen die Blätter des Baumes realisieren, der durch die Strukturierte Analyse aufgespannt wird, ist die Strukturierte Analyse abgeschlossen. Gilt ferner für alle DFDs, dass ihre Datenflüsse in den ihnen übergeordneten DFDs hinterlegt sind, dann liegt Datenintegrität (Balancing) vor, d.h. die Datenflüsse zwischen Kind- und Elterndiagramm sind ausbalanciert.

2.2.5 Zusammenfassung

Die Strukturierte Analyse realisiert von der Form her einen strukturierten Entwurf, der von seiner Methodik Top-Down vorgenommen wird. Die Strukturierte Analyse hat weite Verbreitung gefunden, da sie bewährte Basiskonzepte kombiniert, leicht erlernbar ist und eine Verbesserung der Übersichtlichkeit durch die DFDs erlaubt sowie analytische Qualitätssicherungsmöglichkeiten bietet. Ein Problem stellt die mangelnde Verfeinerung von Datenspeichern und Schnittstellen dar. Aufgrund einer umfangreichen Anzahl an Schnittstellen können Darstellungsprobleme auftreten. Außerdem kann es zu einem Strukturbruch kommen, wenn man durch eine Transformation zu einem datenabstraktionsorientierten Entwurf kommt. Es entsteht implizit ein Funktionsbaum, der Funktionshierarchien beschreibt, die sich aus der Aufteilung der Prozesse ergeben.

Eine erfolgreiche Modellierung durch Datenflussdiagramme kann erzielt werden, wenn für das betrachtete System das funktionale Verhalten im Vordergrund steht und Daten- oder Steuerungsaspekte eher im Hintergrund stehen. Stehen bei der Modellierung die Beziehungen zwischen den Daten im Vordergrund und treten die Funktionen in den Hintergrund, so müssen als erstes Entity-Relationship-Diagramme erstellt werden.

Ein Vorteil der Strukturierten Analyse besteht darin, dass vielfältige Analysen zur Qualitätssicherung auf einem Modell vorgenommen werden können. Jede der verwendeten

Basiskonzepte kann für sich überprüft werden. Zusätzlich können mittels Querverweisen die Gültigkeiten von Modellelementen überprüft werden. Beispielsweise kann geprüft werden, ob

- alle Eingabeoptionen von Prozessen modelliert wurden,
- alle notwendigen Datenspeicher und Datenflüsse bei der Datenverarbeitung spezifiziert wurden,
- alle erzeugten Ausgaben im Modell definiert wurden,
- möglicherweise überflüssige Daten berücksichtigt wurden,
- Datendefinitionen umfangreich vorgenommen wurden, so dass eine detaillierte Strukturierung helfen könnte.

2.3 Methode des Entity-Relationship-Modells

Das Entity-Relationship-Modell (ER-Modell oder ERM) wurde in den 1970er Jahren von Peter Chen entwickelt [Ch76]. Es handelt sich dabei um eine allgemeine Methode zur strukturierten Modellierung von Daten. ER-Modelle dienen dazu, Zusammenhänge eines Realweltausschnitts auf einer abstrakten Ebene zu modellieren. Das ER-Modell an sich realisiert ein konzeptionelles Modell, d.h. es ist gegen Veränderungen der Funktionalität weitestgehend stabil, da vorrangig die Strukturen der Daten beschrieben werden.

2.3.1 Grundelemente

Üblicherweise wird ein ER-Modell graphisch notiert. Das sich ergebende Diagramm besteht aus graphischen Symbolen, die jedes für sich ein einzelnes Element des ER-Modells repräsentieren. Die Grundelemente des ER-Modells sind Entitäten (Entities), Beziehungen (Relationships) und Attribute (Attributes). Die graphischen Symbole der Hauptelemente zeigt Tabelle 4.

Entity

Ein Entity repräsentiert ein eindeutig identifizierbares und somit wohl unterscheidbares Datenobjekt, das in der realen Welt existiert, für das Problem Relevanz besitzt und eventuell in mehreren Ausprägungen vorkommt. Das Entity kann Informationen enthalten, die reale Dinge, Aktivitäten oder Ereignisse oder einen abstrakten Zusammenhang beschreiben.

Entity-Set

Eine Menge zusammengehöriger, vergleichbarer Entities wird durch ein Entity-Set repräsentiert. Zur Unterscheidung muss jedes Entity-Set einen eindeutigen und sprechenden Namen besitzen, der sich an der Fachterminologie des zugrundeliegenden Problems orientiert.

Attribut

Eine Eigenschaft, die zur Beschreibung einer Entität oder einer Beziehung verwendet wird, wird durch ein Attribut im Modell hinterlegt. Bei der Abbildung von Eigenschaften auf Attribute sind nur die für das Problemfeld relevanten Eigenschaften zu berücksichtigen. Attribute besitzen in der Regel einen beschreibenden Charakter, d.h. die beschreibenden Attribute können bei verschiedenen Entities eines Entity-Sets gleiche Werte annehmen.

Falls es Entities gibt, die bezüglich einer Eigenschaft eine gemeinsame Ausprägung, d.h. einen festen Wert haben, spricht man allgemeiner von einem *Value*. Attribute eines Entities, die keinen Wert annehmen (müssen), werden als optionale Attribute bezeichnet.

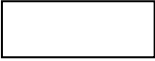

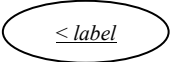
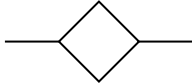
Element	Symbol
Entity, Entity Set	
Attribut	
Primärschlüssel	
Beziehung	

Tabelle 4: Grafische Symbole des Entity-Relationship-Modells

Schlüssel (Key)

Identifizierende Attribute bzw. identifizierende Attributkombinationen werden als Schlüssel (Key) bezeichnet. Es handelt sich dabei um Attribute, die zur Identifikation eines Entities dienen. Jeder Wert eines identifizierenden Attributes darf in diesem Fall in einem Entity-Set maximal einmal auftreten. Auf diese Art und Weise kann ein Entity innerhalb eines Sets eindeutig identifiziert werden. Handelt es sich bei dem identifizierenden Attribut bzw. der identifizierenden Attributkombination um ein minimales, so wird dieses Attribut bzw. diese Attributkombination als *Primärschlüssel (Primary Key)* des Entity-Sets bezeichnet.

Auf Basis dieser Definition werden folgende Regeln für das ER-Modell spezifiziert:

- Jedes Entity-Set muss einen Primärschlüssel definieren. Dies dient der Sicherstellung, dass jedes einzelne Entity des Sets eindeutig zu identifizieren ist.
- Ein Entity kann mehr als ein identifizierendes Attribut besitzen.
- Darüber hinaus sollte jedes Entity-Set durch problemrelevante beschreibende Attribute gekennzeichnet werden.
- Die Kombination aller Attribute eines Entity-Sets ist immer identifizierend.

Speziell sei erwähnt, dass der Name eines Attributs, das Teil eines Primärschlüssels ist oder das den Primärschlüssel realisiert, im Modell unterstrichen wird.

Beziehungen (Relationships)

Im Allgemeinen stehen im betrachteten Problemausschnitt die verschiedenen Entity-Sets einer Anwendung miteinander in Zusammenhang. Mögliche Zusammenhänge zwischen Entity-Sets werden im ER-Modell über *Beziehungen (Relationships)* beschrieben. Diese können um Kardinalitäten erweitert werden, die den jeweiligen Beziehungen einen mengenmäßigen Charakter verleihen. Eine detailliertere Betrachtung wird im Folgenden vorgenommen.

2.3.2 Semantische Elemente

Der Hintergrund, Zusammenhänge auf abstrakter Ebene zu modellieren, ist, dass von Menschen zusammengetragene Informationen nicht nur maschinell verarbeitet, sondern auch maschinell auswertbar sein sollen. Jede Information besitzt in der Realwelt eine Bedeutung. Diese Bedeutung muss soweit wie möglich eindeutig abgebildet werden, um dies zu gewährleisten. Eine Abbildung der Bedeutung an sich auf maschineller Ebene ist eine Abbildung auf die Ebene der Zeichen und betrifft den Bereich der Semantik. Die bisher vorgestellten Bestandteile des ER-Modells besitzen eine elementare Bedeutung in einem Sinne, dass sie, einzeln für sich betrachtet, existieren und fassbar sind. Eine Steigerung ihrer Bedeutung wird durch die Ausdrucksmöglichkeiten des Zusammenspiels der Grundelemente erreicht. Das ER-Modell nutzt an Ausdrucksmöglichkeiten Domains, Beziehungsarten und Integritätsbedingungen.

Domain

Die Zusammenfassung aller zulässigen Ausprägungen für ein Attribut wird als *Domain* bezeichnet. Diese definiert (folglich) einen Wertebereich. Hinsichtlich Entity-Sets und Entities sind der Name eines Entity-Sets und die Attribute eines Entities sowie die Zuordnung einer Domain zu einem Attribut nur selten Änderungen unterworfen. Einzig die Attributwerte eines Entities sowie die Anzahl der Entities eines Entity-Sets sind im Zeitablauf häufig Änderungen unterworfen. Zur eindeutigen Unterscheidung darf es keine zwei Entities geben, die bei allen Attributen identische Werte aufweisen. Ein konkretes Entity wird durch spezifische Zuordnung von Werten zu seinen Attributen definiert.

Domains bzw. Wertebereiche selbst werden im Modell nicht dargestellt. Stattdessen werden sie textuell notiert. Die Notation orientiert sich an einer mathematischen Aufzählung der zulässigen Werte in Form einer Menge, z.B. Schweregrade = {I, II, III, IV, V}.

Falls die Aufzählung selbst durch einen mathematischen Ausdruck beschrieben werden kann, ist die Notierung des mathematischen Ausdrucks ebenfalls zulässig, z.B. Gewicht einer Person = $\{x \mid 0 \leq x \leq 300\}$.

Beziehungsarten

Zwischen zwei Entities kann eine semantische Beziehung aufgebaut werden (vgl. [Ch76], [Go11], [SSH08], [Vo00]). Diese Beziehung wird im ER-Modell zwischen den zugehörigen Entity-Sets angebracht. Die Art der Beziehung legt dabei für jedes Entity-Set fest, wie viele Entities in einer konkreten Beziehung vertreten sein können bzw. müssen.

- **1 : 1 – Beziehung**

Zwischen zwei Entity-Sets ES1 und ES2 liegt eine 1:1-Beziehung vor, falls jedes Entity aus ES1 durch die Beziehung genau einem Entity aus ES2 zugeordnet ist.

Wenn jedem Entity aus ES1 entweder genau 1 oder gar kein Entity aus ES2 durch die Beziehung zugeordnet wird, wird die Beziehung als eine konditionale 1:1-Beziehung bezeichnet.

- **1 : n – Beziehungen**

Eine 1:n-Beziehung liegt vor, wenn zwischen zwei Entity-Sets ES1 und ES2 jedem Entity aus ES1 eine Anzahl von n Entities mit $n > 0$ aus ES2 zugeordnet sind. Gilt für die Anzahl n, dass $n \geq 0$ zulässig ist, spricht man auch hier vom konditionalen Fall. Aus Sicht des Entity-Sets ES2 gilt bei dieser Beziehung, dass jedes Entity aus ES2 zu genau einem Entity aus ES1 in Beziehung steht oder im konditionalen Fall zu höchstens einem Entity aus ES1.

- **m : n – Beziehungen**

Eine m:n-Beziehung liegt zwischen zwei Entity-Sets ES1 und ES2 vor, wenn jedem Entity aus ES1 m Entities mit $m > 0$ aus ES2 zugeordnet sind; $m \geq 0$ gilt für den konditionalen Fall. Aus Sicht des Entity-Sets ES2 sind jedem seiner Entities n Entities, mit $n > 0$ aus ES1 zugeordnet bzw. im konditionalen Fall $n \geq 0$ Entities.

Die obigen Beziehungsarten definieren die Beziehungen, die allgemein zwischen zwei Entity-Sets aufgebaut werden können. Besondere Beziehungsformen sind ergänzend möglich und werden wie folgt bezeichnet:

- Beziehungen zwischen 2 Entity-Sets werden als **binäre Beziehungen** bezeichnet.
- Bei **reflexiven Beziehungen** handelt es sich um Beziehungen zwischen Entities des gleichen Entity-Sets.
- **Mehrwertige Beziehungen** beschreiben Beziehungen, die mehr als zwei Entity-Sets umfassen.
- Wenn eine mehrwertige Beziehung vorliegt, jedoch bezogen auf ein Entity aus ES1 höchstens nur eine der Beziehungen gelten kann, wird dies als **Exklusive-Oder-Beziehung** bezeichnet (vgl. Abbildung 2).

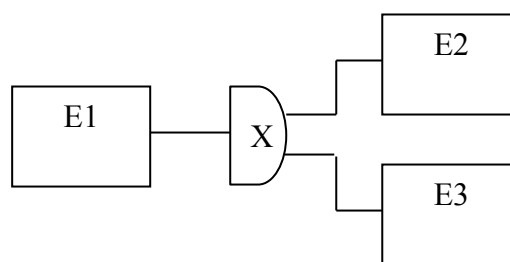


Abbildung 2: Visualisierung einer Exklusive-Oder-Beziehung im ERM (vgl. [Kr05])

Integritätsbedingungen

In der Praxis ergänzen oft einschränkende Bedingungen das ER-Modell. Diese Integritätsbedingungen unterstützen die Abbildung im ER-Diagramm, damit die abgebildeten Entity-Sets, Attribute und Beziehungen die Realität korrekt wiedergeben (vgl. [Ch76], [Go11], [SSH08], [Vo00]). Die Integritätsbedingungen beinhalten sogenannte *Part-of-* und *Is-a-*Beziehungen.

- **Part-of-Beziehung**

Eine Part-of-Beziehung definiert eine Verdichtung bzw. Aggregation. Im Rahmen der Beziehung werden mehrere Entities eines Entity-Sets ES 2 zu einem neuen Entity eines anderen Entity-Sets ES1 zusammengefasst bzw. aggregiert. Es wird zwischen einer Oder- und einer Und-Verdichtung unterschieden. Die Oder-Verdichtung verdichtet mehrere Entities eines Entity-Sets ES 3 entweder zu einem Entity eines Entity-Sets ES2 oder zu einem Entity des Entity-Sets ES1. Die Und-Verdichtung verdichtet mehrere Entities eines Entity-Sets ES2 und mehrere Entities eines Entity-Sets ES2 zu einem Entity eines Entity-Sets ES1. D.h. grundsätzlich wird ein neues Entity vom Typ ES1 durch die Verknüpfung von Entities anderer Entity-Sets geschaffen.

- **Is-a-Beziehung**

Eine Is-a-Beziehung spezifiziert eine Verfeinerung oder Verallgemeinerung. Bei einer Betrachtung der Entities eines Entity-Sets wird geprüft, ob es Entities gibt, die in Varianten, sogenannten Subtypen, aufgeteilt werden können. In diesem Fall werden die Entities in ein neues Entity-Set übernommen, welches eine Verfeinerung des betrachteten Entity-Sets darstellt.

Der gegensätzliche Fall tritt ein, wenn Entities zu einem neuen Entity-Set verallgemeinert werden können. In diesem Fall wird ein neues Entity-Set gebildet, welches zu einem Supertyp des betrachteten Entity-Sets wird.

Die gebildeten Subtypen sollten im Sinne der Klarheit der Modellierung paarweise disjunkt sein. Sind die Subtypen bezüglich des Supertypen vollständig, d.h. ergibt die Vereinigungsmenge der Subtypen die Menge des Supertypen, spricht man auch von 'Is-a'-Beziehungen.

In einem ER-Diagramm sind Integritätsbedingungen nur sehr schlecht darstellbar. Die Übersichtlichkeit des Diagramms würde leiden und ein hoher Formalisierungsgrad würde notwendig werden. Ähnlich wie Domain-Definitionen müssen Integritätsbedingungen daher schriftlich fixiert werden.

2.3.3 Zusammenfassung

Die ER-Diagramme sollen übersichtlich sein und den strukturierten Zusammenhang der Entity-Sets wiedergeben. Das ER-Diagramm ist die visuelle Beschreibung des Modells. Das Modell selbst besteht aus Entitäten und ihrer Relationships, d.h. den Daten und deren Beziehungen zueinander.

Eine Entität (Entity) beschreibt ein individuell identifizierbares Objekt des Problemumfeldes, wie z.B. einen konkreten Patienten. Jede Entität besitzt einen Entitätstyp, der zur Typisierung gleichartiger Entitäten, wie z.B. Ärzten oder Operationen, dient.

Eine Relationship (Beziehung) verknüpft zwei oder mehrere Entitäten miteinander. Beispielsweise operiert Arzt A den Patienten P in der Operation O. Auch Relationships wird ein Typ zugewiesen. Dieser Beziehungstyp dient zur Typisierung gleichartiger Beziehungen. Beispielsweise operiert ein Arzt einen Patienten während einer Operation.

Es ist erlaubt, Beziehungen zwischen Entitäten gleichen Typs aufzubauen. Dies erlaubt die Modellierung von Hierarchien. An Beziehungen können Konsistenzbedingungen geknüpft

werden. Diese werden als Kardinalitäten festgelegt; d.h. Zahlenpaare spezifizieren die Mindest- und Maximalanzahl an Entitäten eines Typs, die an einer Beziehung beteiligt sind.

Entitäten wie Beziehungen werden durch Eigenschaften, d.h. durch Attribute, beschrieben. Zur eindeutigen Identifikation einer Entität bzw. Beziehung werden Schlüssel definiert. Ein Schlüssel ist eine minimale Teilmenge der Attribute, die eine eindeutige Identifikation der Entität innerhalb ihrer Entitätenmenge bzw. der Beziehung innerhalb ihrer Beziehungsmenge zulässt. Im Fall, dass kein eindeutiger Schlüssel gefunden werden kann, wird ein künstliches Schlüsselattribut eingeführt, z.B. eine Patientenummer für die Menge der Patienten.

Das komplette ER-Datenmodell besteht aus einer graphischen Darstellung der Entity-Sets und ihrer Beziehungen, wobei folgende, textuell zu erfassende Beschreibungsinhalte die Darstellung ergänzen:

- Eine Beschreibung aller Entity-Sets,
- eine Beschreibung aller Attribute inklusive ihrer Wertebereiche und
- eine Beschreibung aller Beziehungen.

Das ER-Modell realisiert von der Form her einen strukturierten Entwurf, der von seiner Methodik Bottom-Up vorgenommen wird. Seinen Nutzen erfährt es durch die Schaffung einer gemeinsamen Kommunikationsbasis. Diese enthält eindeutige Definitionen, die durch die Darstellung der Beziehungen zwischen den Daten zum Ausdruck gebracht werden.

Hinsichtlich des Datei- und Datenbankentwurfs bietet das ER-Modell Unterstützung bei der Erstellung eines relationalen Datenbanksystems durch:

- Vorgabe der Relations- und Attributnamen,
- Vorgabe der Formate für die Attribute,
- Vorgabe der Schlüsselattribute und
- Vorgabe von Integritätsregeln für die Datenbank.

Das Hauptproblem des ER-Modells ist, dass für jedes Einzelproblem einer praktischen Anwendung ein Entwurf gemäß des ER-Modells anzufertigen ist. Alle Entwürfe werden anschließend in ein Gesamtschema zusammengebunden. Je nach Komplexität der Anwendung steigt der Umfang des Gesamtschemas. Erst wenn das Gesamtschema zur Verfügung steht, kann mit der Abbildung auf ein Datenbankschema und dem Aufbau des Informationssystems begonnen werden. Um dieses auf dem ER-Modell aufzubauen, ist die Spezifikation von Funktionen und Operationen nötig. Da diese nicht Teil des ER-Modells sind, kann nicht garantiert werden, dass das ER-Modell die geeignete Ausgangsbasis ist. Zunehmend ist daher das ER-Modell bereits durch eine objektorientierte Modellierung abgelöst worden.

2.4 Methode der Objektorientierten Modellierung

Die objektorientierte Modellierung beinhaltet eine Methodik, um mittels einer strukturierten Systemanalyse zu einem strukturierten Systementwurf zu kommen. Ausgehend von einer Problembeschreibung wird ein Analysemodell konzipiert, das die wichtigsten Merkmale der Problemstellung enthält. Anhand dieses Modells wird die globale Problemlösungsstrategie für die Implementierung des Systems entwickelt; d.h. die Systemarchitektur wird konzipiert, indem das Zielsystem in Teilsysteme zerlegt wird, die wiederum Hard- und

Softwarekomponenten zugeordnet werden. Unter Berücksichtigung der Systemarchitektur werden die Objekte spezifiziert, deren Interaktion untereinander die Lösung des Ausgangsproblems widerspiegelt. Die Objekte modifizieren bzw. erweitern das Analysemodell. Abschließend werden die konzipierten Objekte implementiert, wobei häufig eine objektorientierte Programmiersprache zum Einsatz kommt.

Die objektorientierte Modellierung entwickelte sich Ende der 1980er bzw. Anfang der 1990er aus drei Modellierungsansätzen in ihre heutige Form:

- Object Modeling Technique (OMT) nach Rumbaugh et al. ([Ru91]): OMT war ein Modellierungsansatz mit enger Verbindung zur Datenmodellierung. OMT sollte vor allem die Modellierung von komplexen Objekten im Sinne einer objektorientierten Erweiterung des Entity-Relationship-Modells unterstützen.
- Object-Oriented Software Engineering (OOSE) nach Jacobson et al. ([Ja92]): Der Modellierungsansatz OOSE stellte die Modellierung und die Simulation von Vorgängen der realen Welt in den Vordergrund. Ursprünglich wurde OOSE zur Modellierung von Telekommunikationssystemen entwickelt. Der Ansatz wies daher einen engen Bezug zu den dort eingesetzten Modellierungstechniken auf.
- Objektorientierte Entwicklung nach Booch [Bo94]: Der Modellierungsansatz nach der Booch-Methode hatte einen starken Bezug zur dynamischen Modellierung und zu Programmiersprachenkonzepten. Sie ist besonders zur Modellierung von Echtzeitsystemen und nebenläufigen Systemen geeignet, was sich aufgrund ihrer Adaptionen der Vergangenheit ergab.

Die Modellierungsansätze legen einen Fokus auf die Erhöhung der Qualität, Flexibilität und Verständlichkeit von Informationssystemen, indem sie objektorientierte Konzepte durchgängig sowohl für die Analyse als auch für den Entwurf und schließlich für die Implementierung eines Systems nutzen. Die zentralen Konzepte sind *Objekt*, *Klasse* und *Vererbung*. Diese Konzepte bilden das Paradigma der Objektorientierung oder auch das objektorientierte Paradigma. Im Folgenden werden diese Basiskonzepte vorgestellt. Auf eine ausführliche Darstellung wird verzichtet, da eine grundlegende Vertrautheit mit dem Paradigma vorausgesetzt wird. Ausführlichere Beschreibungen finden sich in [Ba00], [He92], [Ko01], [Lo94], [Mü97], [RP02].

2.4.1 Das Paradigma der Objektorientierung

Im Fokus der Objektorientierung steht das *Objekt*. Es ist das zentrale Konzept und damit namensstiftende Element der Objektorientierung. Das Objekt besteht aus Eigenschaften und Funktionalitäten, die dazu dienen, das Abbild eines individuellen Exemplars der realen Welt technisch zu beschreiben. Der Objektbegriff deckt ein weites Spektrum ab; prinzipiell wird unter einem Objekt jeder Gegenstand verstanden, der sich beschreiben lässt. Es kann sich dabei sowohl um konkrete Gegenstände wie ein Krankenhaus, einen Patienten oder einen Operationssaal handeln, als auch um abstrakte Gegenstände wie eine Kostenstelle, eine Leistungsabrechnung oder eine Überweisung.

Die Eigenschaften eines Objekts beschreiben dessen strukturelle Aspekte und werden durch eine Anzahl verschiedener Datenelemente repräsentiert, den *Attributen*. In den Attributen

werden Wertekombinationen hinterlegt, die jeweils den aktuellen *Zustand* des Objektes kennzeichnen.

Die Funktionalitäten eines Objekts beschreiben dessen dynamische, verhaltensbezogene Aspekte und werden durch eine Anzahl von Operationen repräsentiert, den *Methoden*. Eine Änderung oder Abfrage des Zustands eines Objektes ist nur über seine Methoden möglich. Das bedeutet, dass die Methoden die Attributwerte und Verbindungen verstecken.

Ferner gilt, dass Objekte einer Kapselung unterliegen. Die Kapselung versteckt die Attribute und Methoden eines Objektes hinter einer gemeinsamen Hülle. Die innere Struktur und das Verhalten eines Objektes sind folglich nicht direkt zugänglich (Prinzip des „information hiding“). Eine *öffentliche Schnittstelle* erlaubt die Kommunikation mit dem Objekt durch die Hülle. Ausschließlich über diese Schnittstelle werden Daten mit dem Objekt ausgetauscht. Der Datenaustausch selbst wird durch das Senden und Empfangen von *Nachrichten* über die Schnittstelle vorgenommen. Unterschiedliche Objekte wiederum interagieren miteinander über den Austausch von Nachrichten. Dabei entspricht dem Eingehen einer Nachricht die Ausführung einer auf die Nachricht verknüpften Methode.

Objekte mit exakt gleichen Eigenschaften und exakt gleichem Verhalten werden zu einer *Klasse* zusammengefasst. Die Klasse entspricht einer Schablone für die Instanziierung. Die Instanziierung bezeichnet den Vorgang zur Erzeugung neuer Objekte der Klasse und legt fest, dass die neuen Objekte über die Attribute und Methoden verfügen, die allen Objekten der Klasse gemein sind. Klassen erlauben damit eine einmalige Beschreibung, die für eine Menge von Objekten zutrifft.

Das Konzept der *Vererbung* beschreibt ein Abstraktionsprinzip, mit welchem es möglich ist, Klassen zueinander in eine gerichtete Beziehung zu setzen. Die Vererbungsbeziehung erlaubt einer untergeordneten Klasse sämtliche Struktur- und Verhaltenseigenschaften der übergeordneten Klasse mitzunutzen. Darüber hinaus können die Struktur- und Verhaltenseigenschaften aber auch erweitert oder überschrieben werden.

Mittels der *Spezialisierung* können aus einer Klasse spezielle Subklassen gebildet werden. Diese Subklassen erben alle Attribute und Methoden der Klasse und erweitern die Objekte der Subklasse um weitere Attribute und Methoden. Beim Erben besteht für die Subklasse die Möglichkeit, unter Beibehaltung der Schnittstelle, geerbte Methoden zu redefinieren.

Bei der *Generalisierung* wird aus einer Klasse eine allgemeinere Superklasse abgeleitet, die in der Regel aus kleineren Teilmengen an Attributen und Methoden besteht.

Die *Polymorphie* bezeichnet die Fähigkeit von Attributen, für Objekte verschiedener Klassen zu stehen, d.h. in einem Attribut kann die Referenz auf ein Objekt einer bestimmten Klasse hinterlegt werden. Die Polymorphie ermöglicht es dem Attribut auch Objekte von Klassen zu referenzieren, die von der ursprünglichen Klasse über eine Spezialisierung abgeleitet sind. Da durch die Polymorphie die Verknüpfung von Attribut und Objektreferenz nicht mehr statisch festgelegt werden kann, müssen objektorientierte Programmiersprachen die *Dynamische Bindung* unterstützen, d.h. die Klassenzugehörigkeit eines Objektes dynamisch zur Laufzeit ermitteln.

Ferner erlaubt das Konzept des Polymorphismus eine kontextabhängige Interpretation einer Nachricht. Verschiedene Objekte, die denselben Nachrichtentyp erhalten, interpretieren diesen in Abhängigkeit von ihrer Klassenzugehörigkeit. Dies kann zu unterschiedlichem Verhalten führen, das durchaus beabsichtigt ist (vgl. [CY94a], [CY94b], [Ja92], [Ru91], [Te97b]).

2.4.2 Unified Modeling Language

Alle objektorientierten Modellierungsansätze beinhalteten häufig eine eigene Notationsvorschrift für die Modellierung. Im Jahre 1996 hatte die Object Management Group (OMG), das Standardisierungsgremium für objektorientierte Entwicklung, einen Aufruf zur Spezifikation eines Modellierungsstandards erlassen. Dies verhinderte das Abgleiten der Diskussionen zur objektorientierten Modellierung in reine Notationsdiskussionen.

Zu diesem Zeitpunkt arbeiteten Booch, Jacobson und Rumbaugh bereits an einer Vereinheitlichung ihrer Ansätze. Deren Vorschlag wurde am 17.11.1997 als Unified Modeling Language (UML) in der Version 1.1 von der OMG als Modellierungsstandard akzeptiert. Es handelt sich bei der UML um eine einheitliche Modellierungssprache, deren Ausdrücke bei der objektorientierten Modellierung genutzt werden. Die UML ist der heutige Standard der objektorientierten Modellierung.

Softwaresysteme werden heutzutage mit der grafischen Modellierungssprache UML spezifiziert, konstruiert, visualisiert und dokumentiert. Ein solches System wird nicht en bloc entwickelt, sondern innerhalb eines Entwicklungsprozesses stufenweise verfeinert und aufgebaut (vgl. [GBB01]). Auf einem vorgegebenen Abstraktionsniveau beschreiben die dabei entstehenden grafischen Repräsentationen bzw. Diagramme einen bestimmten Aspekt des Problembereichs oder des zu realisierenden Systems. Die Diagramme unterstützen auch die Kommunikation zwischen Entwicklern und Kunden und den Entwicklern untereinander. Die international standardisierte Syntax und Semantik ist „für jedermann“ lesbar und beugt möglichen Missverständnissen vor.

Die UML unterscheidet in der Version 1.3 acht Diagrammartentypen, die sowohl für Modellierungszwecke als auch als grafische Repräsentation der Konstruktion objektorientierter Softwaresysteme dienen:

- Anwendungsfalldiagramm
- Klassendiagramm
- Sequenzdiagramm
- Kollaborationsdiagramm
- Zustandsdiagramm
- Aktivitätsdiagramm
- Komponentendiagramm
- Verteilungsdiagramm

Das Anwendungsfalldiagramm und das Klassendiagramm beschreiben die statische Struktur des zu entwickelnden Systems. Die vier Diagramme Sequenz-, Kollaborations-, Zustands- und Aktivitätsdiagramm werden zur Darstellung der dynamischen Struktur verwendet. Komponenten- und Verteilungsdiagramm repräsentieren Implementierungsdiagramme, da sie zur Visualisierung der Hardware- und Softwaretopologie auf der Ebene der Implementierung dienen.

Ob alle oder nur einzelne Diagramme bei der Entwicklung eines Systems verwendet werden, hängt vom zu entwickelnden System und vom zugrundeliegenden Entwicklungsprozess ab. Die UML schreibt keinen speziell einzuhaltenden Entwicklungsprozess vor. Sie ist vielmehr mit jedem beliebigen Entwicklungsprozess kombinierbar.

Die acht Diagramme der UML werden im Folgenden kurz vorgestellt, wobei sich die Vorstellung an [Kö03] anlehnt. Eine detailliertere Spezifikation der Diagramme kann man

u.a. [Ru05] und [Er00] entnehmen. [Ru05] beschreibt ferner die Erweiterungen der UML in der Version 2, die für die Betrachtung der Modellierung im Rahmen dieser Arbeit nicht ins Gewicht fallen.

Anwendungsfalldiagramm (Use Case Diagram)

Das Anwendungsfalldiagramm dient zur Beschreibung der geforderten Funktionalität des zu entwickelnden Gesamtsystems. Es besteht aus einer Menge von Anwendungsfällen und Schnittstellen zur Außenwelt in Form von kommunizierenden Akteuren.

Ein Anwendungsfall (*use case*) beschreibt ein bestimmtes Verhalten, das von dem zu entwickelnden System erwartet wird. Er wird durch eine mit dem Namen des Anwendungsfalls beschriftete Ellipse dargestellt. Alle Anwendungsfälle zusammen machen die Funktionalität des Gesamtsystems aus. Wer mit dem System interagieren soll, wird durch Akteure festgelegt.



Abbildung 3: Beispiel Anwendungsfalldiagramm

Akteure stehen klar außerhalb des Systems. Sie benutzen das System, indem sie die Ausführung von Anwendungsfällen initiieren, oder sie werden vom System benutzt, indem sie selbst Funktionalität zur Realisierung einzelner Anwendungsfälle zur Verfügung stellen.

Die UML kennt für Akteure zwei Darstellungsformen:

1. Ein Rechteck, das mit dem Namen des Akteurs und dem Schlüsselwort *actor* beschriftet ist oder
2. die piktografische Darstellung in Form eines beschrifteten Strichmännchens.

Auch wenn in der objektorientierten Entwicklung die identifizierten Objekte, ihre Beziehungen zueinander und ihre Interaktionen im Mittelpunkt stehen, so ist für den Auftraggeber und Benutzer eines Informationssystems die Funktionalität dieses Systems am wichtigsten, wie auch immer diese Funktionalität realisiert wird.

Klassendiagramm (Class Diagram)

Das Klassendiagramm stellt Klassen und deren Beziehungen untereinander dar. Um bestimmte Sachverhalte im Diagramm auszuprägen, können auch konkrete Instanziierungen, also Objekte und Objektbeziehungen, im Diagramm angezeigt werden.

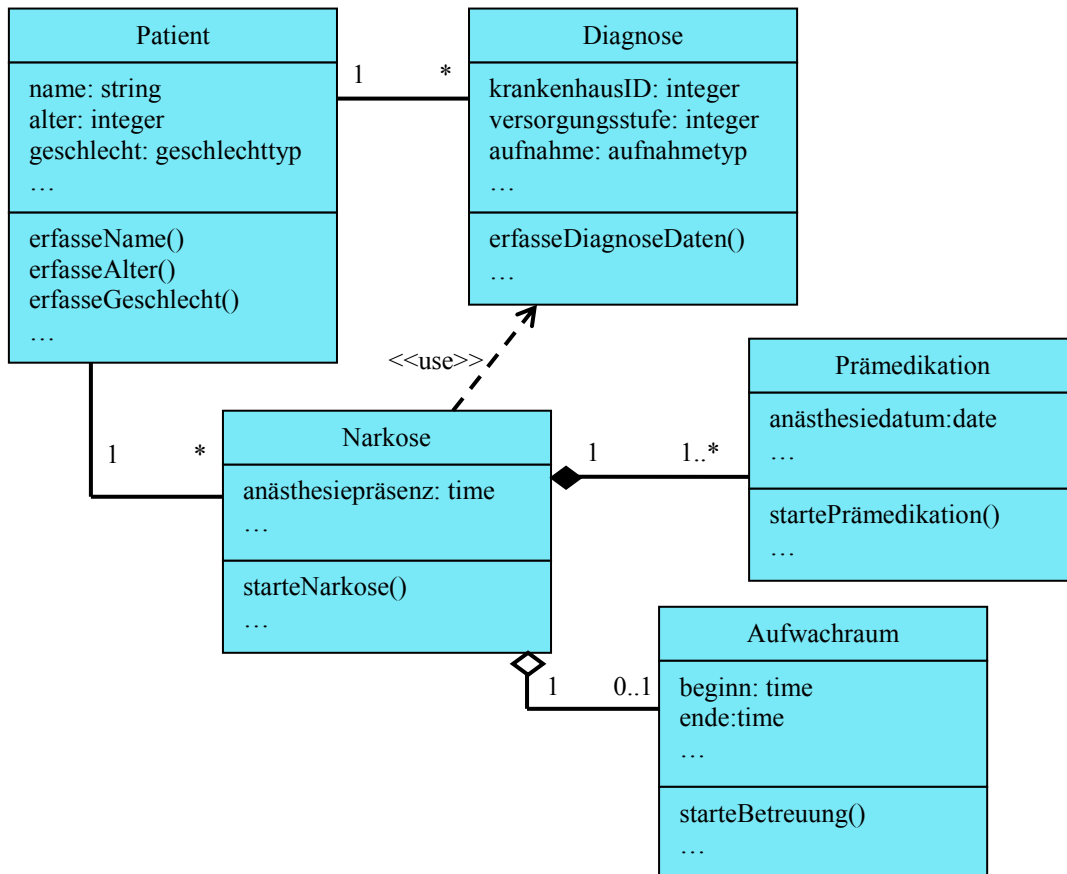


Abbildung 4: Beispiel Klassendiagramm

Eine Klasse (*class*) definiert die strukturellen Eigenschaften (*Attribute*) und das Verhalten (*Operationen*) einer Menge gleichartiger Objekte, den Instanzen der Klasse. Das Rechtecksymbol für eine Klasse ist in Abschnitte untergliedert. Mit Ausnahme des obersten Abschnittes dürfen alle anderen fehlen.

Im obersten Abschnitt sind der Name und optional allgemeine Charakteristika der Klasse vermerkt, der zweite Abschnitt enthält die Attribute und der dritte Abschnitt die Operationen.

Ein Attribut (*attribute*) ist ein Datenelement, über das jedes Objekt der entsprechenden Klasse verfügt. Weist ein Attribut für jedes Objekt einen individuellen Wert auf, wird dieses als Instanzattribut bezeichnet. Ein Klassenattribut dagegen trägt für alle Objekte einer Klasse im Attribut denselben Wert. Ein Attribut selbst wird durch seinen Namen definiert.

Eine Methode (*operation*) stellt eine Operation dar, die von einem anderen Objekt durch eine Nachricht angefordert werden kann. Eine Operation wird analog zu einem Attribut durch einen Namen definiert.

Eine Assoziation (*association*) beschreibt gleichartige Beziehungen zwischen Objekten als Beziehung zwischen den zugehörigen Klassen. Binäre Assoziationen werden durch einfache Verbindungslinien (Kanten) zwischen den beteiligten Klassen dargestellt. Eine Assoziation selbst wird ebenfalls wieder durch einen Namen definiert.

Einer Assoziation kann einer Assoziationsklasse (*association class*) zugeordnet sein. Die zugehörige Beziehung wird dann durch die Attribute der Assoziationsklasse näher

beschrieben. Eine Assoziationsklasse wird durch ein eigenes Klassensymbol notiert, das durch eine gestrichelte Linie mit der entsprechenden Assoziationskante verbunden ist.

Eine Beziehung zwischen mehr als zwei Objekten wird über eine mehrstellige Assoziation (*n-ary association*) zum Ausdruck gebracht. Eine Raute dient zur Darstellung, die mit allen Klassen, die an der Assoziation teilnehmen, durch eine Kante verbunden ist. Der Assoziationsname wird in der Umgebung der Raute notiert.

Die Aggregation (*aggregation, part-of relationship*) ist ein Spezialfall der allgemeinen Assoziation. Sie stellt eine asymmetrische Beziehung zwischen nicht gleichwertigen Objekten dar und realisiert eine „Teile-Ganzes-Beziehung“. Die Repräsentation einer Aggregation wird im Diagramm angezeigt, indem jenes Ende der Assoziationskante, das zur Aggregatklasse, also „zum Ganzen“ hinführt, durch eine kleine, nicht ausgefüllte Raute markiert wird.

Die Komposition (*composition*) ist eine spezielle, strengere Form der Aggregation. Sie wird durch eine ausgefüllte Aggregationsraute angezeigt. Innerhalb der Kompositionsbeziehung gelten folgende Einschränkungen für die beteiligten Objekte:

- Die Kompositionsklasse bzw. das „Kompositionsteil“ ist Teil der Klasse, die es aggregiert.
- Die Kompositionsklasse existiert höchstens so lange wie die Klasse, die es aggregiert.
- Operationen der Klasse, die aggregiert, werden an seine Kompositionsklassen propagiert.

Die Generalisierung (*generalization, Is-a-Beziehung*) stellt eine taxonomische Beziehung zwischen einer spezialisierten Klasse (Unter- oder Subklasse) und einer allgemeineren Klasse (Ober-, Basis- oder Superklasse) dar. Die Subklasse erbt dabei die Charakteristika (Attribute, Operationen, Assoziationen) der Superklasse (Eigenschaftsvererbung). Die erbende Klasse kann weitere Charakteristika hinzugefügt bekommen und geerbte Operationen anpassen. Eine Generalisierung wird durch einen Pfeil mit einem gleichseitigen Dreieck als Spitze notiert, der von der Subklasse zur Superklasse führt.

Zur Unterscheidung von Instanzen (Objekte, Objektbeziehungen (links) etc.) von den zugehörigen Typen (Klassen, Assoziationen etc.) steht in der UML ein einheitlicher Mechanismus zur Verfügung: Für die Instanz wird dasselbe grafische Symbol verwendet wie für den Typ. Zusätzlich werden der Name der Instanz und eine notwendige Typangabe unterstrichen. Objekten können darüber hinaus Attributwerte zugeordnet werden.

Ein Klassendiagramm, das nur Objekte und Objektbeziehungen darstellt, wird als Objektdiagramm (*object diagram*) bezeichnet.

Sequenzdiagramm (Sequence Diagram)

Das Systemverhalten wird in der UML mit Sequenz-, Kollaborations-¹, Zustands- und Aktivitätsdiagrammen dargestellt.

¹ Sequenz- und Kollaborationsdiagramme beschreiben das sogenannte Interobjektverhalten, das ist die Art und Weise, wie einzelne Objekte durch Nachrichtenaustausch interagieren, um eine bestimmte Aufgabe zu erfüllen. Sie werden daher auch als Interaktionsdiagramme (*interaction diagrams*) bezeichnet.

Ein Sequenzdiagramm modelliert i.d.R. ein bestimmtes Szenario, d.h. die exemplarische Darstellung eines Ablaufs (Anwendungsfall oder Operation). In der vertikalen Dimension des Sequenzdiagramms befindet sich eine ordinal skalierte Zeitachse. In der horizontalen Dimension werden die an dem Szenario teilnehmenden Objekte angeordnet. Sie werden durch eine gestrichelte Lebenslinie (lifeline) repräsentiert, die mit einem Objektsymbol beginnt.

Aktivierungsbalken (*activation bars*) auf den Lebenslinien markieren jene Zeiträume, in denen ein Objekt über den Kontrollfluss verfügt, also eine seiner Operationen ausgeführt wird. Aktive Objekte (*active objects*) besitzen einen durchgängigen Aktivierungsbalken und ein Objektsymbol mit fettem Rand.

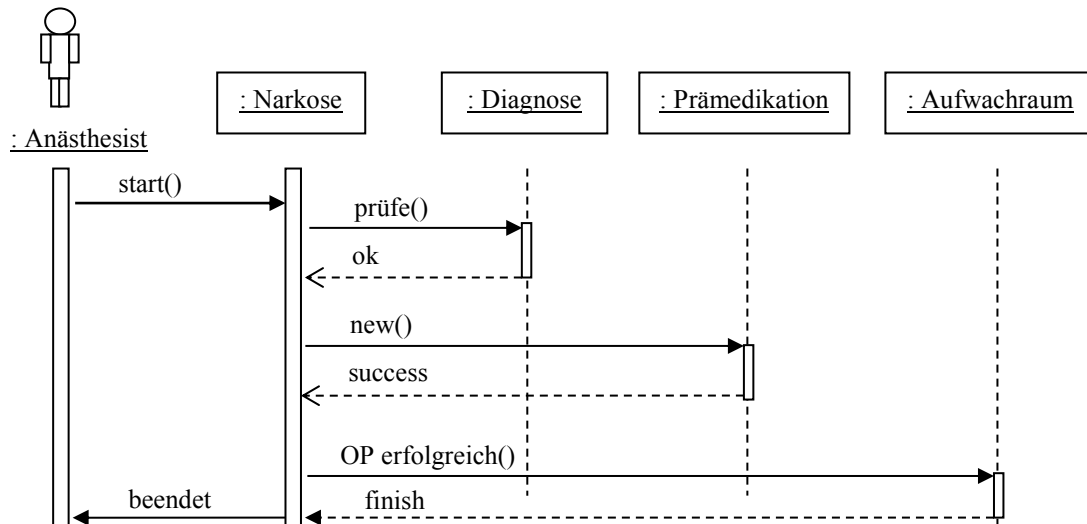


Abbildung 5: Beispiel Sequenzdiagramm

Zwischen den Objekten werden Nachrichten (*messages*) ausgetauscht. Diese sind als Pfeile zwischen den Lebenslinien angebracht und mit dem Namen und den Argumenten der Nachricht beschriftet.

Anhand verschiedener Pfeilspitzen wird zwischen synchroner (Prozeduraufruf), asynchroner (Signal) und unspezifizierter (nicht näher charakterisierter) Nachrichtenübermittlung unterschieden. Eine Rückgabe eines Ergebnisses kann über einen gestrichelten Pfeil sichtbar gemacht werden. Nachrichten, die Objekte erzeugen, werden direkt an das Objektsymbol geführt. Wird ein Objekt gelöscht, endet seine Lebenslinie in einem X.

Kollaborationsdiagramm (Collaboration Diagram)

Auch das Kollaborationsdiagramm zeigt die für einen bestimmten Zweck notwendigen Interaktionen zwischen Objekten. Im Gegensatz zum Sequenzdiagramm zeigt das Kollaborationsdiagramm zusätzlich zu den Interaktionen auch den ihnen zugrundeliegenden Kontext in Form von Objektbeziehungen (Kollaboration, s.u.). Dafür wird darauf verzichtet, die Zeit als eigene grafische Dimension zu modellieren. Um eine Reihenfolge von Nachrichten spezifizieren zu können, muss auf eine Nummerierung zurückgegriffen werden.

Eine Kollaboration (*collaboration*) in der UML definiert einen Ausschnitt der statischen Modellstruktur. Diese enthält genau jene Modellelemente, die zur Erreichung eines definierten Ziels miteinander kooperieren. Die Modellelemente selbst werden auf

Instanzebene (als Objekte und Objektbeziehungen) dargestellt. Der Nachrichtenaustausch zwischen den beteiligten Objekten wird durch die Beziehungskanten dargestellt. Eine Nachricht (*message*) wird durch einen Pfeil repräsentiert, der durch eine textuelle Spezifikation erweitert wird. Die Pfeilspitze gibt die Art des Nachrichtenaustausches an, wobei dieselben Typen wie im Sequenzdiagramm zulässig sind: synchron, asynchron und unspezifiziert.

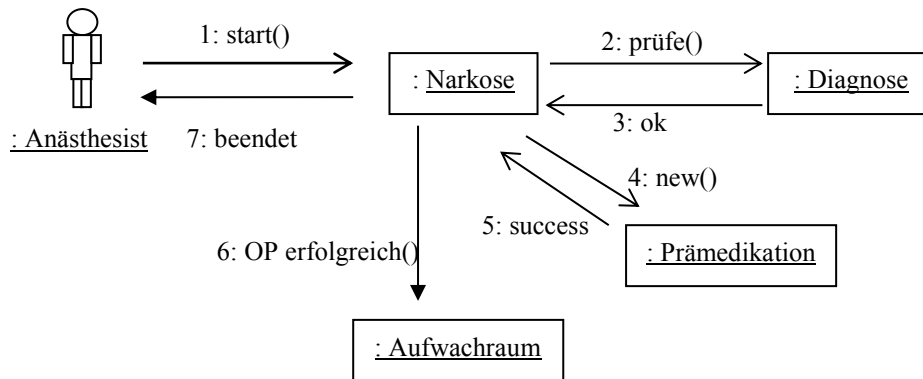


Abbildung 6: Beispiel Kollaborationsdiagramm

Aktive Objekte (*active objects*) werden im Diagramm durch die Eigenschaftsangabe *{active}* oder durch einen fetten Symbolrand markiert.

Zustandsdiagramm (State Chart Diagram)

Das interne Objektverhalten wird über Zustandsdiagramme beschrieben. Diese Diagramme spezifizieren die möglichen Auswirkungen, die ein einzelnes Objekt einer bestimmten Klasse hat, wenn es in einen bestimmten Zustand tritt. Das Zustandsdiagramm beschreibt alle Zustände, die das Objekt während seines „Lebenslaufs“, also von seiner Erzeugung bis zu seiner Destruktion und/oder während der Ausführung einer Operation der Reihe nach einnehmen kann.

Zustandsdiagramme selbst basieren auf Verallgemeinerungen von endlichen Automaten. Sie stellen einen Graph dar, dessen Knoten den Zuständen entsprechen, die von Objekten eingenommen werden können und dessen gerichtete Kanten die möglichen Zustandsübergänge angeben. Zustandsübergänge werden normalerweise von Ereignissen ausgelöst.

Ein Zustand (*state*) charakterisiert eine Situation, in der das betrachtete Objekt auf bestimmte äußere Ereignisse in situationsspezifischer Weise reagiert. Objekte verweilen solange in einem bestimmten Zustand, bis ein definiertes Ereignis (*event*) einen Zustandsübergang auslöst. Ein Zustand wird im Diagramm durch ein Rechteck mit abgerundeten Ecken dargestellt. Es kann in zwei Abschnitte unterteilt sein, wobei ein spezifizierender Name im oberen Abschnitt den Zustand charakterisieren kann. Unbenannte Zustände gelten grundsätzlich als voneinander verschieden. Im unteren Abschnitt können Angaben über Aktivitäten und innere Transitionen (s.u.) angegeben werden.

Zuständen können auch andauernde Aktivitäten (*activities*) zugeordnet werden. Diese werden solange ausgeführt, wie sich das Objekt in dem Zustand befindet, dem die Aktivität

zugeordnet ist. Eine Aktivität wird im Zustandssymbol nach dem Präfix *do/* angegeben und kann durch ein eigenes Zustandsdiagramm näher beschrieben werden. Transitionen hingegen können mit Aktionen assoziiert werden. Aktionen (*actions*) sind atomare, nicht unterbrechbare Tätigkeiten, die im Zuge des Zustandsübergangs durchgeführt werden und (konzeptuell) keine Dauer aufweisen.

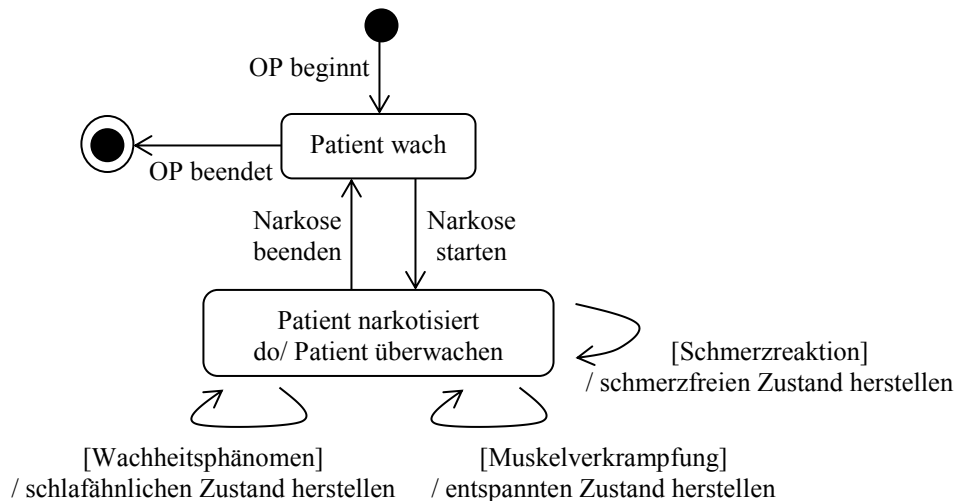


Abbildung 7: Beispiel Zustandsdiagramm

Eine Transition (*transition*) ist der Übergang von einem Zustand in seinen Folgezustand. Sie wird ebenfalls durch ein Ereignis ausgelöst und erfolgt, ohne selbst Zeit in Anspruch zu nehmen. Zur Darstellung einer Transition dient ein beschrifteter Pfeil. Dieser verbindet einen Zustand mit seinem Folgezustand. In der Beschriftung sind das auslösende Ereignis, eine optionale Bedingung und Aktionen angegeben:

- Auslösendes Ereignis: Sobald das Ereignis eintritt und die zugeordnete Bedingung erfüllt ist (s.u.), schaltet die Transition. Eine eventuell noch andauernde, dem Zustand zugeordnete Aktivität wird dabei unterbrochen.
- Optionale Bedingung (*guard condition*): Die Bedingung ist eine optionale Einschränkung, deren Einhaltung die Transition überwacht. Tritt das Ereignis ein und ist die Bedingung nicht erfüllt, so erfolgt kein Zustandsübergang.
- Aktionen: Die aufgeführten Aktionen sind im Zuge der Transition auszuführen. Bei mehreren Aktionen entspricht die Reihenfolge der Ausführung der Reihenfolge der Spezifikation.

Innere Transitionen (*internal transitions*) werden ebenfalls von Ereignissen ausgelöst, verlassen aber den aktuellen Zustand nicht. Ihre Spezifikation erfolgt im unteren Abschnitt des Zustandssymbols in der Form Ereignis/Aktion Ereignis. An die Stelle von echten Ereignissen können auch die beiden Pseudoereignisse *entry* und *exit* treten. Die ihnen zugeordneten Aktionen werden immer dann ausgeführt, wenn das Objekt in den Zustand eintritt bzw. ihn verlässt.

Aktivitätsdiagramm (Activity Diagram)

Aktivitätsdiagramme dienen zur Beschreibung von Abläufen. Abläufe treten auf gänzlich unterschiedlichen Detaillierungsebenen auf. Beispiele sind die Realisierung einer Operation, die einzelnen Schritte eines Anwendungsfalls oder das Zusammenspiel mehrerer Anwendungsfälle zur Realisierung (von Teilen) des Gesamtsystems.

Aktivitätsdiagramme stammen von den Datenflussdiagrammen der Strukturierten Analyse ab und ermöglichen die Beschreibung entsprechender Abläufe. Dabei kann spezifiziert werden, was die einzelnen Schritte des Ablaufs tun, in welcher Reihenfolge sie ausgeführt werden und wer für sie verantwortlich zeichnet. Die einzelnen Schritte eines Ablaufs werden als Aktionen oder Subaktivitäten bezeichnet:

- Aktionen (*actions*) sind nicht weiter zerlegbare Schritte eines Ablaufs.
- Subaktivitäten (*sub activities*) werden in eigenen, separaten Aktivitätsdiagrammen detaillierter spezifiziert.

Im Aktivitätsdiagramm beschreibt man beide durch Angabe von Zuständen, in denen die Aktionen bzw. Subaktivitäten ausgeführt werden. Ein Zustand (*state*) im Aktivitätsdiagramm wird durch ein Rechteck mit konvexen Vertikalen dargestellt. Darin enthalten ist der Name der auszuführenden Aktion bzw. Subaktivität.

Ein Großteil der Eigenschaften von Zustandsdiagrammen ist auch auf Aktivitätsdiagramme übertragbar. So werden Beginn und Ende eines Ablaufs wie in Zustandsdiagrammen durch einen eindeutigen Start- und möglicherweise mehrere Endzustände markiert.

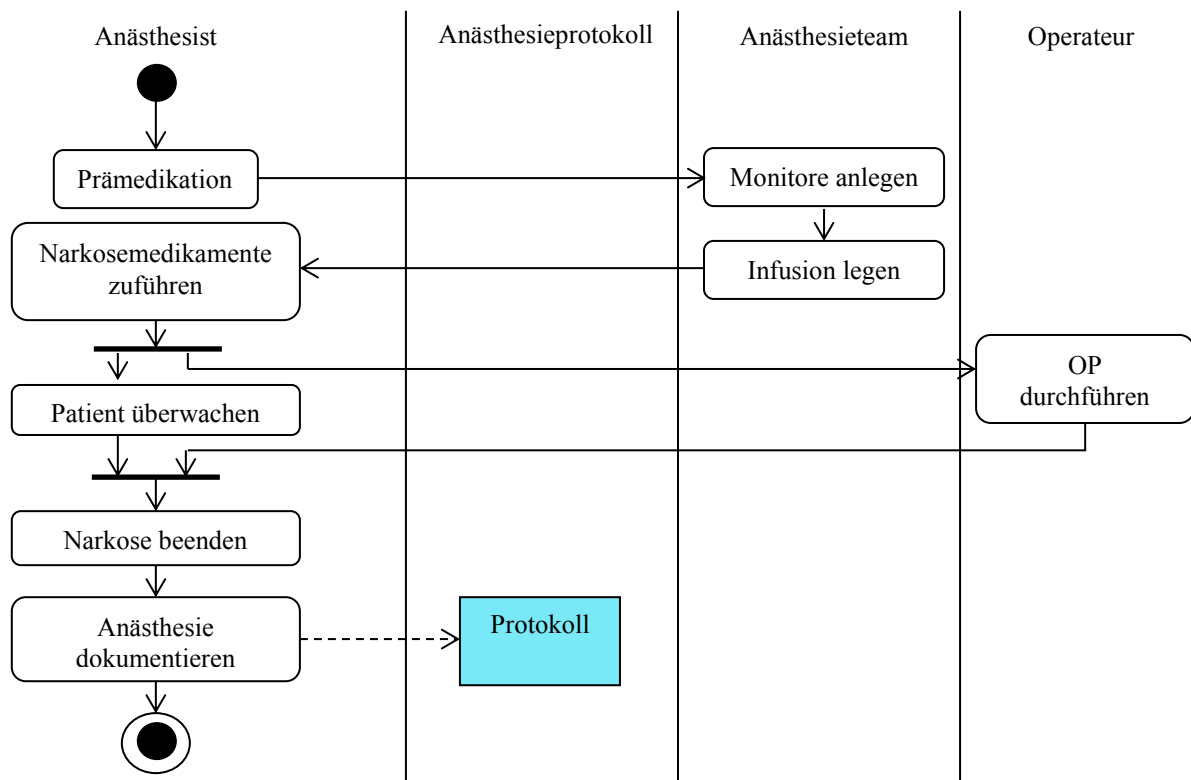


Abbildung 8: Beispiel Aktivitätsdiagramm

Aktivitäten werden untereinander durch Kontrollflusskanten (*control flow*) verbunden. Transitions Pfeile legen die Ausführungsreihenfolge der einzelnen Schritte fest. Der Abschluss einer Aktivität fungiert dabei als impliziter Trigger für die Ausführung der Folgeaktivität.

Eine Transition kann mit einer Überwachungsbedingung (*guard*) in eckigen Klammern und weiteren Aktionen beschriftet werden. In einem solchen Fall wird die Transition nur dann ausgeführt, wenn die vorangegangene Aktivität abgeschlossen und die Bedingung erfüllt sind. Ist dies der Fall, werden die angegebenen Aktionen ausgeführt, bevor die Aktivität, zu der die Transition führt, angestoßen wird.

Jeder Aktivität müssen mindestens eine ein- und mindestens eine ausgehende Transition zugeordnet sein. Mit Hilfe eines Entscheidungsknoten (*decision node*) genannten Pseudozustands können alternative Abläufe explizit modelliert und komplexe Entscheidungen in Form von Entscheidungsbäumen dargestellt werden. Einem Entscheidungsknoten sind eine oder mehrere eingehende Transitions und zwei oder mehrere ausgehende Transitions zugeordnet. Alle ausgehenden Transitions müssen einander wechselseitig ausschließende Bedingungen tragen, wobei auch die vordefinierte Bedingung [else] möglich ist.

Ein Kontrollfluss, der durch einen Entscheidungsknoten aufgespalten wurde, kann bei Bedarf auch durch dasselbe Symbol wieder vereinigt werden. Der Entscheidungsknoten wird in dieser Rolle dann als *Merge* bezeichnet. Einem Merge sind zwei oder mehrere eingehende Transitions und eine ausgehende Transition zugeordnet. Diese ausgehende Transition darf weder ein Ereignis noch eine Überwachungsbedingung tragen. Mit den Pseudozuständen Gabelung und Vereinigung können auch nebenläufige Abläufe, d.h. parallele Teilfolgen von Aktivitäten, dargestellt werden.

Die Gabelung (*fork*) zeigt den Beginn einer Nebenläufigkeit durch einen sogenannten Synchronisationsbalken an, von dem mehrere Transitions wegführen. Transitions, die eine Gabelung verlassen, können auch Überwachungsbedingungen tragen. Sollte eine solche Bedingung zum Zeitpunkt des Schaltens der Transition nicht erfüllt sein, wird der zugehörige Zweig im Aktivitätsdiagramm nicht aktiviert.

Analog zeigt die Vereinigung (*join*) das Ende einer Nebenläufigkeit durch einen Synchronisationsbalken an, in den mehrere Transitions münden. Die Vereinigung schaltet, sobald alle über die eingehenden Transitions verbundenen Aktivitäten beendet worden sind. Ist bei der zugehörigen Gabelung aufgrund einer Überwachungsbedingung ein bestimmter Zweig nicht aktiviert worden, so ist dieser auch für die Vereinigung irrelevant, d. h. es wird nicht „auf ihn gewartet“.

Aktivitätsdiagramme unterstützen in Anlehnung an Datenflussdiagramme die Modellierung von Objektflüssen (*object flows*), indem Objekte entweder als Eingabe oder als Ausgabe einer oder mehrerer Aktivitäten modelliert werden. Abhängig davon wird das entsprechende Objektsymbol, ein Rechteck mit dem unterstrichenen Namen des Objekts und seinem aktuellen Verarbeitungszustand in eckigen Klammern, als Eingabe zu oder als Ausgabe von einer Aktivität spezifiziert, indem eine gerichtete gestrichelte Kante vom Objektsymbol zur verarbeitenden Aktivität bzw. umgekehrt gezeichnet wird. Sofern der Objektfluss den Kontrollfluss zwischen zwei Aktivitäten vollständig spezifiziert, wird nur der Objektfluss dargestellt.

Für die Ausführung der Aktivitäten können Verantwortungsträger, d.h. verantwortliche Objekte oder Akteure, spezifiziert werden. Da Aktivitätsdiagramme meist relativ früh im

Entwicklungsprozess eingesetzt werden, können diese „Objekte“ aber auch sehr grob granulare Einheiten, z. B. die Abteilungen eines Unternehmens, dessen Geschäftsprozesse beschrieben werden, sein. Verantwortungsträger und die von ihnen ausgeführten Aktivitäten werden zu Verantwortungsbereichen (*swim lane*) zusammengefasst, das sind Regionen im Aktivitätsdiagramm, die mit den jeweils verantwortlichen Objekten beschriftet sind und durch vertikale Linien voneinander getrennt werden.

Komponentendiagramm (Component Diagram) und Verteilungsdiagramm (Deployment Diagram)

In den frühen Phasen der Software-Entwicklung werden die bisher vorgestellten Diagrammart der UML verwendet. Mit dem Komponentendiagramm und dem Verteilungsdiagramm bietet die UML aber auch diagrammatische Unterstützung zur Dokumentation des implementierten Systems an. Beide Diagramme stellen die implementierten Softwarekomponenten, ihre Abhängigkeiten untereinander und ihre Zuordnungen zu einzelnen Knoten einer Hardwaretopologie dar.

Ein Komponentendiagramm (*component diagram*) ist ein Graph, dessen Knoten Komponenten und dessen Kanten Abhängigkeiten zwischen diesen Komponenten repräsentieren.

Komponenten (*components*) existieren in der UML nur während der Implementierung bzw. Laufzeit des Systems und sind entweder Quellcode-Komponenten, Binärcode-Komponenten oder ausführbare Komponenten.

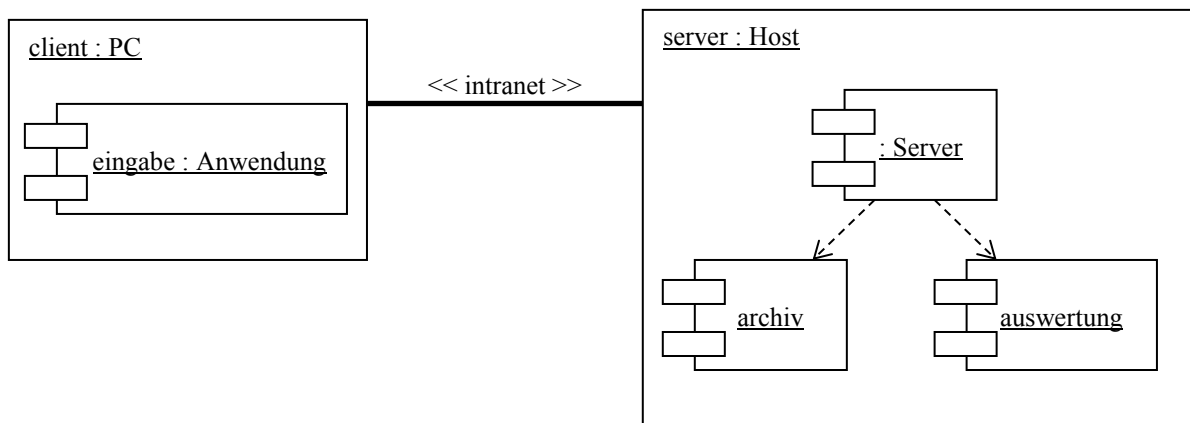


Abbildung 9: Beispiel Komponentendiagramm

Zur Darstellung einer Komponente dient ein Rechteck mit zwei kleinen, den linken Rand überlagernden Rechtecken. Innerhalb des Rechtecks wird der Name der Komponente definiert. Komponenten können sowohl auf der Typ- als auch auf der Instanzebene existieren. Eine Komponenteinstanz wird in Analogie zum Objekt durch das Komponentensymbol und mit Namen und Typbezeichner durch Doppelpunkt getrennt oder unbenannt mit Doppelpunkt, gefolgt vom Typbezeichner, dargestellt. Die gesamte Zeichenkette wird unterstrichen, um sie von einem Komponententyp zu unterscheiden.

Sinnvoll ist die Unterscheidung zwischen Typ und Instanz einer Komponente nur bei ausführbaren Komponenten. Alle anderen Komponentenarten existieren nur auf der Typebene.

Eine Komponente bietet eine Menge von Schnittstellen (*interfaces*) und deren Realisierung an. Jederzeit kann eine Komponente durch eine andere Komponente substituiert werden. Sie muss dabei alle Schnittstellen der ursprünglichen Komponente anbieten.

Mögliche Abhängigkeiten unter den Komponenten können mittels des bereits bekannten Abhängigkeitspfeils (gestrichelter Pfeil) dargestellt werden.

Komponentendiagramme kommen meist zum Einsatz, wenn das entwickelte System hinreichend komplex ist. Sie beschreiben physische Abhängigkeiten zwischen Komponenten, im Gegensatz zu logischen Beziehungen zwischen Klassen. Während der Modellierung stellen Komponentendiagramme wichtige Informationen für Konfigurationsmanagement und Versionierung der Software zur Verfügung. Komponentendiagramme existieren aber ausschließlich auf der Typebene und können daher keine Komponenteninstanzen beinhalten.

Diese können nur im Zusammenhang mit einem Verteilungsdiagramm gezeigt werden. Im Verteilungsdiagramm wird angegeben, auf welchem konkreten Knoten (Prozessor) die Komponenteninstanz ausgeführt wird. Eine Komponenteninstanz kann auch eine geschachtelte Struktur aufweisen, indem sie selbst wieder Komponenteninstanzen beinhaltet, die innerhalb dieser Komponente existieren und ausgeführt werden.

Ein Verteilungsdiagramm (*deployment diagram*) zeigt die eingesetzte Hardwaretopologie und das zugeordnete Laufzeitsystem in Form von Softwarekomponenten und eingebetteten Objekten. Das Verteilungsdiagramm ist ein Graph, dessen Knoten Verarbeitungseinheiten, i.d.R. Prozessoren, repräsentieren und dessen Kanten Kommunikationsbeziehungen zwischen diesen Verarbeitungseinheiten darstellen. Neben den Kommunikationsbeziehungen können auch die Abhängigkeitsbeziehungen zwischen den Komponenten dargestellt werden. Ein Knoten wird als Quader dargestellt, die Kommunikationsbeziehung zwischen den Knoten als durchgezogene Linie. Die Benennung von Knoteninstanzen erfolgt analog zu Komponenteninstanzen.

2.4.3 Zusammenfassung

Die objektorientierte Modellierung mit der UML hat sich zu einer standardisierten Notationssprache entwickelt. Sie dient als Architekturschema für Anwendungen und ermöglicht die Kommunikation zwischen Entwicklern.

In der UML werden Schwerpunkte der Datenmodellierung mit prozessorientierten Konzepten von Anwendungsfällen und funktionalen Entwurfsprinzipien kombiniert.

Die meisten Diagramme der UML bestehen aus Knoten und Kanten. Knoten beschreiben Klassen, Objekte oder Zustände, während Kanten Assoziationen, Abhängigkeiten oder Zustandsübergänge markieren. Damit bietet die UML als Vertreter der objektorientierten Modelle Beschreibungselemente an, die sowohl zur Beschreibung der statischen Struktur als auch des dynamischen Verhaltens von Systemen dienen können.

2.5 Methode der Ereignisgesteuerten Prozessketten

Das Modell ereignisgesteuerter Prozessketten (EPK) realisiert eine grafische Beschreibungstechnik zur Modellierung von Geschäftsprozessen. Ereignisgesteuerte Prozessketten sind gerichtete Graphen, die die Elemente „Funktion“ und „Ereignis“ beinhalten. Praktisch dienen ereignisgesteuerte Prozessketten der Beschreibung von Abläufen, d.h. dem Zusammenhang von Ereignissen, Aktivitäten und Ablaufreihenfolgen. Hierzu werden die beiden Elemente in einer alternierenden Folge angeordnet. Durch die Verknüpfung mittels „Konnektoren“ können auch nichtlineare Prozesse abgebildet werden (vgl. [Br09]).

2.5.1 Grundelemente

EPKs eignen sich besonders zur Modellierung ereignisgesteuerter Programmabläufe. Ereignisse lösen Aktivitäten aus, die wiederum als Ergebnis neue Ereignisse haben müssen. Die grafischen Beschreibungselemente einer Ereignisgesteuerten Prozesskette (EPK) werden in Tabelle 5 aufgelistet.


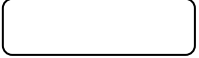
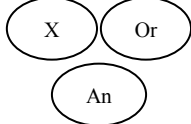
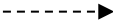
Element	Symbol
Ereignis	
Aktivität	
Konnektoren	
Kontrollfluss	

Tabelle 5: Grafische Symbole der Ereignisgesteuerten Prozessketten

Ereignis

Unter einem Ereignis einer ereignisgesteuerten Prozesskette versteht man einen eingetretenen Zustand, der den weiteren Ablauf des Geschäftsprozesses bestimmt. Die Zustandsänderung bezeichnet in der Regel das Auftreten eines Objektes oder die Änderung einer bestimmten Attributsausprägung.

Aktivitäten

Eine Aktivität stellt eine Tätigkeit bzw. einen betriebswirtschaftlichen Vorgang dar. Abgebildet werden die Tätigkeiten durch Programme, Funktionseinheiten oder auch Methoden, die aufgrund eines Auftretens von Ereignissen ablaufen sollen. Dabei soll es möglich sein, dass die ablaufbaren Aktivitäten selbst wieder untergliederbar sind und ebenfalls wieder durch ereignisgesteuerte Prozessketten ausgedrückt werden können.

Konnektoren

Um parallele Abläufe oder Handlungsalternativen abbilden zu können, stehen Konnektoren als Verknüpfungsoperatoren zur Verfügung. Man unterscheidet zwischen der UND-, der

ODER- und der exklusiven ODER-Verknüpfung. Die Bedeutung der einzelnen Konnektoren ergibt sich aus der Aussagenlogik.

Kontrollfluss

Die Abhängigkeiten bzw. der Zusammenhang zwischen Ereignissen und Aktivitäten wird graphisch durch den Kontrollfluss zum Ausdruck gebracht.

2.5.2 Modellierung von Ereignisgesteuerten Prozessketten

Die Beschreibungselemente werden genutzt, um Geschäftsprozesse über verschiedene Detaillierungsgrade hinweg zu veranschaulichen. Die verschiedenen Elemente stehen dabei in Beziehung zueinander.

Die Modellierung von EPKs verbindet Aktivitäten mit Ereignissen. Ausgehend von einem Startereignis liegt das Ziel darin, einen Ausgangszustand in einen Zielzustand zu transformieren, wobei dies über eine Kette von Aktivitäten geschieht, die sich nacheinander über Ereignisse triggern. Diese Kette spezifiziert somit die zeitlich-logischen Abhängigkeiten von Aktivitäten.

Das Modell der EPKs setzt die Aktivitäten in die Reihenfolge ihrer Ausführung. Ein Ereignis löst eine Aktivität aus. Das Ereignis ist als das Eintreten eines Zustands definiert, der eine bestimmte Folge bewirkt. Diese Folge wiederum ist die Ausführung einer Aktivität, die einen neuen Zustand herstellt, der ein weiteres Ereignis kennzeichnet.

Diese Beschreibung zeigt auf, dass Aktivitäten (und Ablaufelemente) als das zentrale Objekt eines EPK-Flusses angesehen werden können, da sie mit allen anderen Objekten in Beziehung stehen. Die Verbindungen zwischen Notationselementen per Konnektoren dienen nur einer Operationalisierung.

2.5.3 Erweiterte Ereignisgesteuerte Prozessketten

Zur Berücksichtigung weiterer Aspekte der realen Welt in der Modellierung dienen erweiterte Ereignisgesteuerte Prozessketten (eEPK). Die Erweiterung der ereignisgesteuerten Prozessketten umfasst die Visualisierung des Informationsflusses mittels Informationsobjekten. Dabei werden die beteiligten Organisationseinheiten und Prozesswegweiser in der Darstellung berücksichtigt. Tabelle 6 zeigt die grafischen Symbole, um welche die ereignisgesteuerten Prozessketten erweitert werden. Die Notation hierfür ist in der Fachliteratur nicht eindeutig festgelegt, weshalb die Notation aus [FG08] präsentiert wird.

Informationsobjekt

Ein Informationsobjekt ist eine Abbildung eines Gegenstands der realen Welt. Häufig werden Dokumente visualisiert, die Schriftstücke repräsentieren, die im Rahmen des Informationsflusses weitergeleitet werden.

Organisatorische Einheit

Die Gliederungsstruktur eines Unternehmens wird mittels einer Organisatorischen Einheit abgebildet. Sie zeigen auf, von welchem Prozessbeteiligten eine Aktivität durchgeführt wird.


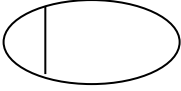

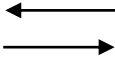

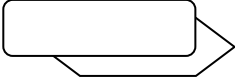
Element	Symbol
Informationsobjekt	
Organisatorische Einheit	
Anwendungssystem	
Informationsfluss	
Zuordnung	
Prozesswegweiser	

Tabelle 6: Grafische Symbole der erweiterten Ereignisgesteuerten Prozessketten

Anwendungssystem

Die Darstellung von Anwendungssystemen beinhaltet die Unterstützung von Prozessen, die die automatische Abwicklung von Aktivitäten mit Hilfe der IT veranschaulichen.

Informationsfluss

Der Informationsfluss beschreibt, ob Daten von einer Aktivität gelesen, geändert oder geschrieben werden. Die Darstellung der Verbindung der Elemente entspricht einem gerichteten Graphen.

Zuordnung

Mittels der Zuordnung wird der Sachverhalt zum Ausdruck gebracht, welche Einheit (z.B. Mitarbeiter) oder Ressource eine Aktivität bearbeitet.

Prozesswegweiser

Die Verbindung eines Prozesses von bzw. zu einem anderen Prozess wird durch den Prozesswegweiser visualisiert.

2.5.4 Zusammenfassung

Ereignisgesteuerte Prozessketten haben sich speziell zur Modellierung von betrieblichen Abläufen durchgesetzt (vgl. [Kr00]). Mittels EPKs können standardisierte Abläufe modelliert werden. Die Erfassung rein formaler Strukturen und Abläufe ist problematisch, genauso wie die Abbildung von System- wie auch von Datenbrüchen.

Die Grenzen von ereignisgesteuerten Prozessketten werden erreicht, sobald es sich bei den abzubildenden Geschäftsprozessen nicht mehr um vorhersagbare Abläufe handelt. Kreative und komplexe Tätigkeiten sind schlecht oder gar nicht darstellbar.

Geschäftsprozesse, deren Darstellung nur durch eine komplexe Visualisierung erreicht werden kann, leiden unter dem Problem, dass es der Darstellung an Übersichtlichkeit mangelt. Besonders zum Tragen kommt dieser Nachteil der ereignisgesteuerten Prozessketten, wenn im nächsten Abschnitt noch zusätzliche Elemente eingeführt werden. Abhilfe schafft die Gliederung von Prozessen in Teilprozesse.

Aus diesen Gründen sind ereignisgesteuerte Prozessketten insbesondere dafür geeignet, um sich einen Überblick über die Abläufe und Zusammenhänge der Prozesse des betrachteten Unternehmens zu verschaffen. Vorrangig dienen ereignisgesteuerte Prozessketten daher der Geschäftsprozessmodell-Analyse. EPKs erlauben die Schaffung eines Gesamtüberblicks, der dem Entdecken von Schwachstellen im Rahmen der Geschäftsprozessmodell-Analyse dient. Dies ist jedoch widersprüchlich, da zwischen nötiger Genauigkeit zur Schwachstellenfindung und nötiger Übersichtlichkeit bei der Abbildung das Verständnis der Abläufe und Zusammenhänge klar in Zusammenhang gebracht werden sollte. Zu Gunsten der Übersichtlichkeit wird meist auf die in den Funktionen verbrauchten Zeiten und Kosten der abgebildeten Tätigkeiten verzichtet, so dass die Effizienz der betrachteten Geschäftsprozesse nicht quantitativ erfolgen kann. Auch die Qualität der einzelnen Funktionen kann mit Hilfe der ereignisgesteuerten Prozessketten nicht geprüft werden.

Ein prinzipielles Problem bei der Lesbarkeit der ereignisgesteuerten Prozessketten stellt die Semi-Formalität der Beschreibungssprache dar. Verschiedene Autoren bilden die Elemente mit verschiedenen Notationen ab. Insbesondere tritt dieser Fall bei der Erweiterung der ereignisgesteuerten Prozesskette auf. Es werden neue Notationselemente spezifiziert, die in der Erstveröffentlichung gar nicht enthalten waren. Aufgrund der Semi-Formalität werden dann Bezeichnungen der Ereignisse und Aktivitäten mit Interpretationsspielräumen versehen, die es zulassen, dass Leser des Modells zu verschiedenen Verständnissen gleicher Geschäftsprozesse kommen.

2.6 Konzept der Business Objects

Eine Anforderung an betriebliche Informationssysteme ist die flexible Anpassung an sich ändernde betriebliche Abläufe. Unternehmen müssen sich heutzutage schnell und effektiv an veränderte Markt- und Umweltsituationen anpassen. Die Anpassung erzwingt eine Änderung in der Ausgestaltung von Unternehmensstrukturen und innerbetrieblicher Abläufe. Für die maschinellen Anwendungssysteme wird die Anpassbarkeit an die geänderten Verhältnisse zur Herausforderung. Das in den letzten Jahren populär gewordene Schlagwort *Business Objects* wird häufig im Bereich der Modellierung und Entwicklung von betrieblichen Informationssystemen verwendet (vgl. [Er01], [HS00], [Li98]). Es dient als Synonym für Ideen und Konzepte zur radikalen Neugestaltung bzw. Anpassung von Geschäftsprozessen in Unternehmen, die im Rahmen von *Business (Re)Engineering*² bzw. *Business Process Engineering* diskutiert werden (vgl. [ES98]).

² Unter Business Engineering wird die aufbau- und ablauforganisatorische Gestaltung einer Unternehmung verstanden.

2.6.1 Geschichtlicher Hintergrund

„Urheber“ dieses Begriffes ist nach [Pe99] Robert Shelton, der im Jahr 1993 der OMG die Einrichtung einer sogenannten „Business Object Management Special Interest Group“ (BOMSIG) vorgeschlagen hatte.

1994 definierte Oliver Sims in seinem Buch „Business Objects – Delivering Cooperative Objects for Client/Server“ [Si94] das Konzept der „*Cooperative Business Objects*“ (CBO). Sims versteht darunter auslieferbare und vermarktbar Softwareeinheiten, die als Endprodukte von Softwareentwicklungen zur Schaffung verteilter Objektsysteme entstehen.

Eine Vielzahl von Veröffentlichungen beschäftigt sich mit dem Begriff *Business Objects*. Eine gute Übersicht über die verschiedenen Interpretationen des Business-Object-Begriffs liefert [Pe99]. Dort werden sieben Bedeutungsebenen unterschieden, die jedoch nicht überschneidungsfrei sind. Auffällig am Zustandekommen der Bedeutungsebenen ist, dass die jeweiligen Autoren nicht notwendigerweise eine Definition für Business Objects liefern. Vielmehr beschreiben sie ein Konzept, welches als Business Object bezeichnet wird.

Anders verhält es sich bei der *Object Management Group* (OMG). Dort definierte 1995 Carol Burt den Begriff *Business Object* in einem Online-Artikel, der leider für die Öffentlichkeit nicht zugänglich ist. 1996 wurde ebenfalls bei der OMG ein Artikel veröffentlicht ([Ca96])³, der jedoch öffentlich zugänglich ist. In diesem Artikel ist die Definition von Business Objects angegeben.

Die allgemeinen Veröffentlichungen zur Thematik der Business Objects beziehen sich auf die im Artikel von Cory Casanave veröffentlichte Definition (vgl. [Ca97]). Diese Arbeit orientiert sich ebenfalls an der Definition der OMG. Um Verwechslungen mit der Firma Business Objects™ zu vermeiden, wird in dieser Arbeit der Begriff **Pliable Object** angewandt. Es läge zwar nahe, den deutschen Begriff *Geschäftsobjekt* zu nutzen, dieser wird jedoch in der Marketingbranche oder in Veröffentlichungen zur betrieblichen und volkswirtschaftlichen Wirtschaftslehre genutzt. Der Begriff *Geschäftsobjekt* wird dort außerdem mehrdeutig verwendet. Da das zu entwickelnde Modell letztlich auch in Bereichen der Wirtschaft eingesetzt werden könnte, dient die Unterlassung der Verwendung von *Geschäftsobjekt* der Vorbeugung zur Vermeidung von sprachlichen Verwirrungen bei zukünftigen Zusammenarbeiten. Im Folgenden wird die Definition der OMG von Business Objects näher betrachtet.

2.6.2 Business Object Definition der Object Management Group

In [Ca96] wird *Business Object* wie folgt definiert:

“A business object is defined as a representation of a thing active in the business domain, including at least its business name und definition, attributes, behavior, relationships, rules, policies and constraints. A business object may represent, for example, a person, place, event, business process or concept. Typical examples of business objects are: employee, product, invoice, payment.”

³ Auch wenn der Autor in diesem Artikel nicht explizit angegeben ist, referenziert sich Cory Casanave in [Su97] selbst, so dass der Autor bestimmt werden kann.

Mit dieser Definition betrachtet die OMG die Problemdomänen als Business Objects bzw. interpretiert die Problemdomänen als Teil eines Systems von Business Objects. Ein Business Object kann dabei in natürlicher Sprache, in einer Sprache, die einer Modellierung dient, oder in einer Programmiersprache verfasst sein. Die OMG nimmt real an, dass die Business-Object-Abstraktion, die eine reale Entität modelliert, durch ein oder mehrere Objekte in einem Informationssystem implementiert wird.

Die technologische Infrastruktur, die das „Plug and Play“ von Business-Applikationskomponenten unterstützt, wird als *Business Object Facility* bezeichnet. Diese Infrastruktur muss die Semantik besitzen, um Business Objects zu unterstützen.

2.6.3 Zusammenfassung

Allgemein versteht die OMG unter Business Objects Softwareapplikationen, die einem Bausteinprinzip genügen. Umfangreiche Anwendungssoftware, wie sie für die Abbildung von Geschäftsprozessen in wirtschaftlich orientierten Unternehmen benötigt werden, erhält man durch das Zusammensetzen von einzelnen Business Objects. Damit dieses Ziel erreicht werden kann, müssen die Schnittstellen, über die die Business Objects miteinander kommunizieren, eindeutig definiert sein. Dies hat zur Folge, dass die Business Objects eine Struktur beinhalten, die allen Business Objects gemeinsam ist. Über diese Struktur erfolgt der Datenaustausch zwischen den Business Objects, der auch dazu dient, die Verhaltenssemantik eines jeweiligen Business Objects den anderen Business Objects im System mitzuteilen.

Betrachtet man die Definition unter objektorientierten Gesichtspunkten, so fällt auf, dass die Business Objects durchweg mit objektorientierten Mitteln modelliert werden können. Prinzipiell bildet die OMG den Aufbau der Objektorientierung auf Business Objects ab (vgl. Tabelle 7).

Objektbestandteile nach den Grundkonzepten der Objektorientierung	Objektbestandteile nach dem OMG-Konzept
Name	Business Name
Attribute	Business Definition, Attribute, Business Rule
Methoden	Verhalten, Business Rule
Objektverbindungen	Beziehung

Tabelle 7: Abbildung objektorientierter Konzepte auf Business Objects (nach [Ko01])

Objekte besitzen einen eindeutig identifizierenden Namen, den Business Name. Attribute beschreiben den Objektzustand, wobei die Business Definition ein spezielles Attribut ist. Das Verhalten entspricht den Methoden, die das Objektverhalten repräsentieren. Beziehungen zwischen Objekten sind in der Objektorientierung ebenfalls bekannt. Business Rules kann man sowohl zur Abgrenzung des Zustandsraumes von Objekten als auch zur Festlegung von Verhaltensweisen, in denen das Objekt agiert und sich verändert, heranziehen (vgl. [Ko01]).

Die OMG abstrahiert die klassische Herangehensweise der Objektorientierung und führt neue Begriffe ein, um Probleme konkret zu benennen. Allerdings löst sie die Probleme selbst nicht, sondern bleibt beim alten objektorientierten Modell.

3 Dokumentationsansprüche im medizinischen Fachgebiet der Anästhesie

Ziel dieser Arbeit ist, eine Lösung für die Anpassbarkeit medizinischer Informationssysteme im Fachgebiet der Anästhesie zu präsentieren. Dazu ist es nötig, eine Anästhesie prinzipiell zu verstehen. Aus dem Verständnis heraus können schließlich die Objekte gewonnen werden, die zur Abbildung der Anästhesie in ein Informationssystem benötigt werden.

3.1 Was bedeutet Anästhesie?

Im weiteren Sinne bedeutet Anästhesie die Schmerz- und Bewusstseinsausschaltung eines Patienten. Der Begriff *Anästhesie* entstammt dem Griechischen und bedeutet etwa „Empfindungslosigkeit“ oder „Betäubung“. Früher wurde die Anästhesie meist als Narkose bezeichnet, womit der Umstand ausgedrückt wurde, dass sie zum Zweck einer Operation erfolgte. Heutzutage umfasst das medizinische Fachgebiet der Anästhesie auch die Behandlung von akuten oder chronischen Schmerzzuständen unterschiedlichster Art. Die Analgesie, d.h. die Schmerzlosigkeit bzw. Schmerzausschaltung, ist die wichtigste Teilkomponente der Anästhesie, die auch im Rahmen von Operationen eine wichtige Rolle spielt (vgl. [CC01], [KS01], [RT04], [USZ03], [Zw99]).

Drei Grundformen der Anästhesie werden unterschieden:

1. *Lokalanästhesie*: Unter Lokalanästhesie wird die umgangssprachlich als „örtliche Betäubung“ bezeichnete Anästhesie verstanden, bei der nur ein kleiner Bereich des Körpers schmerzfrei gemacht wird.
2. *Regionalanästhesie*: Im Rahmen der Regionalanästhesie werden ein größerer Körperabschnitt bzw. ganze Körperteile betäubt.
3. *Allgemeinanästhesie*: Umgangssprachlich wird unter der Allgemeinanästhesie die sogenannte „Vollnarkose“ verstanden. In der Allgemeinanästhesie werden die Patienten in einen schlafähnlichen, entspannten Zustand versetzt, in dem der gesamte Körper empfindungs- und schmerzfrei gemacht wird.

Lokalanästhesien werden bei kleineren Eingriffen und in der Regel vom behandelnden Arzt selbst, z.B. Chirurgen oder Zahnarzt, vorgenommen. In den Zuständigkeitsbereich von Anästhesisten gehören die Regional- und Allgemeinanästhesien. Die Aufgabe solcher Anästhesien ist, dafür zu sorgen, dass Patienten während der Operation bewusstlos und schmerzfrei sind. Als „funktionsbedingter“ Nebeneffekt der Anästhesie ist die Muskulatur des Patienten entspannt.

Die Durchführung von Anästhesien erfolgt in den meisten Fällen durch spezialisierte Fachärzte zusammen mit Anästhesiepflegefachkräften. Als Anästhesie-Team überwachen sie den Patienten. Die Teammitglieder haben generell eine spezielle Zusatzausbildung absolviert und verfügen über das notwendige Fachwissen. Außerdem besitzen sie die erforderliche Kenntnis von speziellen Techniken, um die Sicherheit der Patienten zu gewährleisten. Anästhesien schalten nicht nur Schmerzen aus, sondern haben auch andere Auswirkungen auf verschiedene wichtige Organsysteme, so dass anästhesierte Patienten engmaschig überwacht werden müssen. Dadurch können bei operativen Eingriffen Auswirkungen von Anästhesien auf lebenswichtige Organfunktionen vermieden oder sofort behandelt werden.

Die Aufgaben des Anästhesie-Teams gehen über die reine Durchführung der Anästhesie hinaus. Um der Verantwortung für die Überwachung und Aufrechterhaltung aller lebenswichtigen Organfunktionen gerecht zu werden, muss die Anästhesie eines Patienten individuell vorbereitet und koordiniert werden.

3.1.1 Idealisierter Versorgungszyklus

Die Anästhesie eines Patienten ist Teil der allgemeinen medizinischen Versorgungsleistungen, die dieser Patient erhält, wenn sein Leiden in einer medizinischen Versorgungseinheit⁴ medizinisch behandelt und dazu eine Operation notwendig wird. Der Anwendungsfall aus Sicht des Patienten skizziert sich als ein idealisierter Versorgungszyklus, wie ihn Abbildung 10 zeigt.

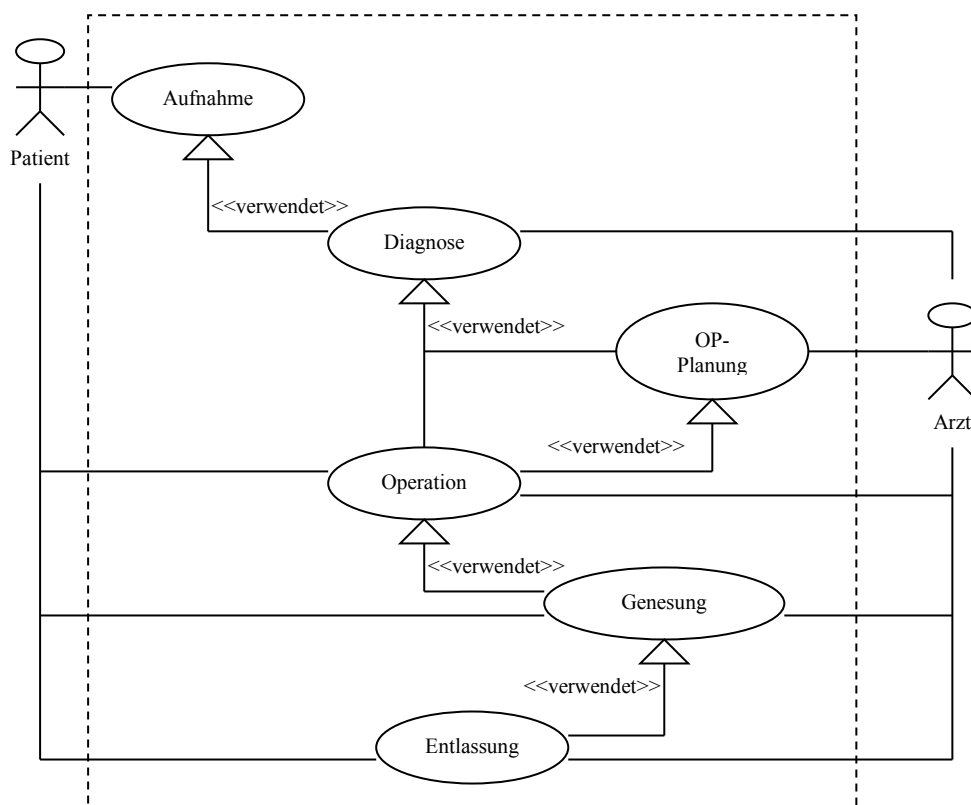


Abbildung 10: Idealisierter Versorgungszyklus

Der Anwendungsfall beginnt mit der Aufnahme des Patienten in einem Krankenhaus. Nach einer Untersuchung wird sein Leiden diagnostiziert, d.h. sein Krankheitszustand beurteilt. Diagnosespezifische Behandlungsrichtlinien werden auf Grundlage der individuellen Patientendaten spezifiziert und die notwendige Operation geplant. Entsprechend der Planung wird die Operation durchgeführt. Nach der Operation bleibt der Patient zur Genesung bis zu seiner Entlassung im Krankenhaus.

Explizit sei darauf hingewiesen, dass der eben skizzierte ideale Anwendungsfall nur als eine vereinfachte Darstellung gegeben ist. Es handelt sich gewissermaßen um einen roten Faden,

⁴ Umgangssprachlich wird eine medizinische Versorgungseinheit häufig als Krankenhaus bezeichnet.

der nicht allgemeingültig ist. Das hauptsächliche Ziel der Versorgung ist die vollständige Genesung des Patienten. Dies ist allerdings nicht garantiert, so dass wenigstens eine Linderung des Leidens versucht wird. Unter Umständen könnte aber selbst das Erreichen einer Linderung nicht möglich sein. Der Anwendungsfall des Versorgungszyklusses besitzt somit auch grundlegend andere Ausprägungen. Beispielsweise kann im Rahmen eines Notfalls ein Patient eingeliefert werden, der einer umgehenden Operation unterzogen werden muss. Zuerst wird die Aufnahme des Patienten mit „Dummy“-Daten vorgenommen. Auf Basis einer minimalen Diagnoseuntersuchung wird entsprechend eines Notfallplanes eine Notoperation durchgeführt. Bei einer minimalen Diagnose besteht die Gefahr, dass Symptome übersehen und erst während der Operation entdeckt werden (z.B. innere Blutung). Dadurch wird der weitere Ablauf der Operation beeinflusst. Die Diagnosedaten selbst werden parallel zur Operation erweitert. Im Anschluss an die Operation erfolgen schließlich die reguläre Aufnahme, die Genesung und die Entlassung des Patienten. Die Erfassung der Patientendaten bzw. die Aufnahme des Patienten erfolgen somit nachträglich. Die OP-Planung und die Diagnose besitzen dadurch eventuell eine grundlegend andere Ausprägung. Die einzelnen Ausprägungen können dabei einen Zustand repräsentieren, der bislang nicht absehbar war und deshalb auch nicht vorgesehen ist.

Möglich ist jedoch auch der Anwendungsfall, dass ein Krankenhaus über einen einzuliefernden Notfall informiert und daraufhin eine Operation vorbereitet wird; verstirbt der Patient leider zeitgleich mit oder vor der Einlieferung, so ist die Operation hinfällig. Dennoch kann je nach Verwaltungsvorgaben der medizinischen Versorgungseinheit eine Dokumentation vorzunehmen sein.

Für Abläufe in medizinischen Versorgungseinrichtungen muss angenommen werden, dass sie als allgemeine Anwendungsfälle spezifiziert werden, häufig allerdings individuelle und dynamische Ausprägungen besitzen. Die Bandbreite der unterschiedlichen Ausprägungen von allen möglichen Anwendungsfällen wird bei genauer Vergegenwärtigung der oben exemplarisch skizzierten Anwendungsfälle deutlich. Eine generalisierte einheitliche Sicht, die einen klaren eindeutigen Anwendungsfall beschreibt, ist nicht erhältlich. Anstelle eines Anwendungsfalles für die medizinische Versorgung eines Patienten muss ein Versorgungsszenario angenommen werden, welches alle Anwendungsfälle umfasst.

Die Anästhesie ist ein integrierter Bestandteil eines jeweiligen Versorgungsszenarios. Als Teilaspekt der medizinischen Versorgung realisiert die Anästhesie jedoch ein derart eigenständiges, unabhängiges Anwendungsgebiet, dass vielleicht spezifische Anästhesie-Anwendungsfälle betrachtet werden könnten. Aufgrund der Abhängigkeit der Anästhesie von der Diagnose, OP-Planung, Operation und Rehabilitation kann jedoch keine konkrete Spezifikation eines allgemeingültigen Anästhesieablaufes erarbeitet werden, der sämtliche anästhesiologischen Anwendungsfälle abdeckt, die möglicherweise auftreten können. Die Anzahl der möglichen Anwendungsfälle ist schlichtweg zu groß. Eine Anästhesie präsentiert sich somit nicht als ein allgemeingültig spezifizierbarer Anwendungsfall im medizinischen Versorgungsbereich, sondern als ein Anwendungsszenario, in welchem jede Szenarioausprägung einen Anwendungsfall darstellt.

3.1.2 Anästhesiologischer Arbeitsablauf

Das Behandlungsszenario der Anästhesie kann als ein aus verschiedenen Aktivitäten bestehender, komplexer Arbeitsablauf aufgefasst werden. Sämtliche Aktivitäten in der Anästhesie sind auf jeweils einen Patienten ausgerichtet. Ziel aller Aktivitäten ist die optimale Unterstützung einer Narkose, d.h. den Patienten zu Operationszwecken in einen

schlafähnlichen und schmerzfreien Zustand zu versetzen. Die Betreuung eines zu anästhesierenden Patienten ist grundsätzlich individuell und abhängig von dessen physischer und psychischer Befindlichkeit. Trotz der Individualität einer Betreuung ist die Narkose als ein grundsätzlicher Behandlungsprozess ermittelbar. Die Narkosebehandlung zeichnet sich dadurch aus, dass sie mit der Narkoseeinleitung einen definitiven Behandlungsbeginn und ein ausgezeichnetes Behandlungsende durch die Narkoseausleitung besitzt. Das medizinische Fachgebiet der Anästhesie ist somit eines der wenigen medizinischen Fachgebiete, das eine Kernbehandlung mit Start und Ende spezifizieren kann.

Allgemein kann angenommen werden, dass eine Anästhesiebehandlung aus drei Teilprozessen besteht. Diesen Teilprozessen können die verschiedenen Aktivitäten zugeordnet werden, die den komplexen Anästhesiearbeitsablauf beschreiben. Die verschiedenen Aktivitäten sind entweder Bestandteil eines Kernprozesses oder Bestandteil eines Prozesses, der dem Kernprozess entweder vor- oder nachgelagert ist. Der anästhesiologische Arbeitsablauf wird in einen präoperativen, einen intraoperativen und einen postoperativen Verlauf unterteilt.

Als Analyse sei der Prozess bezeichnet, der die dem Kernprozess vorgelagerten Aktivitäten enthält und als Validierung der Prozess, der die dem Kernprozess nachgelagerten Aktivitäten enthält. Das Anästhesieszenario kann somit als Prozessfolge, bestehend aus Analyse, Kernprozess und Validierung, verstanden werden. Dabei beschreibt ausschließlich der Kernprozess die maßgeblichen Aktivitäten der Anästhesie (vgl. Abbildung 11).

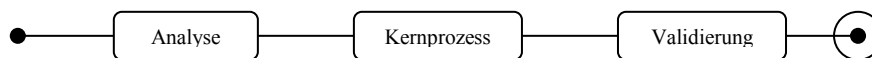


Abbildung 11 Teilprozesse der Anästhesie

3.1.3 Intraoperativer Verlauf

Der intraoperative Verlauf beinhaltet den Kernprozess, in dem die eigentliche Narkose durchgeführt wird. Die Narkose wiederum entspricht dem Durchlaufen einer Phasensequenz, die gegeben ist als die Aktivitäten Narkoseeinleitung, Narkoseführung und Narkoseausleitung.

Narkoseeinleitung

Die Phase der Narkoseeinleitung wird vom verantwortlichen Anästhesisten absolviert. In dieser Phase wird der Patient in den schlafähnlich entspannten Zustand versetzt. Die Vornahme der Narkoseeinleitung beginnt entweder auf dem Operationstisch oder das Krankenhaus verfügt hierfür über einen separaten Einleitungsraum. Im Rahmen der Narkoseeinleitung werden alle wichtigen Daten überprüft. Zur Überwachung von Herz, Kreislauf und Atemfunktion während der Operation werden an den Patienten ein Blutdruckmessgerät, ein EKG und ein Fingerclip zur Messung der Sauerstoffsättigung des Blutes angeschlossen. Eine kleine Kunststoffkanüle wird zusätzlich an Hand oder Arm des Patienten platziert. Über die Kanüle wird eine Infusion, d.h. eine elektrolythaltige Lösung infundiert. Nach Abschluss aller notwendigen Vorbereitungen beginnt die Narkose. Dem Patienten werden über den venösen Zugang die Medikamente zum Einschlafen verabreicht.

Der Patient sollte müde werden, einschlafen und erst wieder wach werden, wenn die Operation beendet ist.

Narkoseführung

In der Phase der Narkoseführung wird der Patient in seinem Zustand gehalten. Der Anästhesist steuert dazu die Narkosetiefe. Während der chirurgischen Eingriffe kommt es zu Phasen unterschiedlicher Schmerz- und Stimulationsintensität. Der Anästhesist muss die Anästhetikadosierung an den jeweiligen Grad der Stimulation anpassen, um eine Intoxikation⁵ oder um Wachheitsphänomene zu vermeiden. Der Anästhesist übernimmt während der Narkose die Verantwortung für den Patienten und tritt für die Wahrung seiner Interessen und Grundbedürfnisse ein, die der Patient im narkotisierten Zustand nicht selbst wahrnehmen und befriedigen kann.

Narkoseausleitung

In der Phase der Narkoseausleitung wird der Patient aus seinem narkotischen Schlaf geholt. Er wird praktisch aus seinem Zustand „geweckt“. Der Anästhesist stabilisiert die Vitalfunktionen des Patienten und überwacht dessen Atem- und Kreislauffunktionen.

3.1.4 Präoperativer Verlauf

Der präoperative Verlauf ist ein Anästhesieteilprozess, der der Analyse des Patientenzustandes dient. Er dient zur Ermittlung von Informationen, auf denen eine Spezifikation des intraoperativen Verlaufes vorgenommen werden kann. Solch eine Analyse ist im Allgemeinen hilfreich, aber nicht zwingend notwendig für den Kernprozess, so dass ein präoperativer Verlauf entfallen kann, z.B. bei einem Notfall. Der präoperative Verlauf kann die Aktivitäten einer Prämedikationsvisite und einer Prämedikation enthalten.

Prämedikationsvisite

Der Zweck der Prämedikationsvisite ist die Durchführung einer anästhesiologischen Anamnese. Dies bezeichnet die medizinische Zustandsbeurteilung des Patienten, auf deren Basis die Anästhesie geplant sowie geeignete Narkoseverfahren bestimmt werden. Real entspricht die Prämedikationsvisite einem Besuch des Anästhesisten beim Patienten auf dessen Station. Der Anästhesist führt mit dem Patienten ein sogenanntes Narkosegespräch. Er lernt dabei den Patienten kennen, baut eine vertrauensvolle Beziehung auf, informiert ihn über mögliche Narkoseverfahren und –risiken und befragt ihn zu seinem Lebenswandel und eventuellen Vorerkrankungen. Häufig wird den Patienten vor ihrem Narkosegespräch auch ein Informationsbogen ausgehändigt, den der Patient gründlich zu lesen und auszufüllen hat. Anhand des ausgefüllten Fragebogens informiert sich der Anästhesist über den Gesundheitszustand des Patienten. Zum Abschluss des Gespräches erfolgt eine körperliche Untersuchung durch den Anästhesisten. Aus rechtlichen Gründen ist es für die Anästhesie wichtig, dass der Patient eine Einwilligungserklärung unterschreibt, da die Durchführung der

⁵ Das Narkosestadium Intoxikation bezeichnet die Lähmung des Hirnstammes und den Ausfall der Atem- und Kreislaufregulation (vgl. [RT04], Seite 10).

Narkose den Tatbestand der Körperverletzung erfüllt. Nach der Prämedikationsvisite bespricht sich der Anästhesist mit dem operierenden Arzt, um Informationen über die geplante Operation zu bekommen. Der Anästhesist sollte das OP-Verfahren und dessen Ablauf kennen, um ein optimales Narkoseverfahren auswählen zu können und um somit eine korrekte Narkose zu gewährleisten.

Prämedikation

Die Prämedikation dient der Vorbereitung der Narkose. Meist erhält der Patient im Rahmen der Prämedikation zur Beruhigung Medikamente in Tablettenform. Die Prämedikation richtet sich nach dem Patienten, d.h. nach dessen Vorerkrankungen, Körpergewicht, Alter, etc. Sie kann bereits am Abend vor der Operation beginnen, um eine bessere Stressabschirmung zu erreichen; spätestens am Morgen des Operationstages wird sie vorgenommen. Die Beruhigung des Patienten dient der Erholung des Körpers und bietet Schutz vor z.B. hohem Blutdruck aufgrund von Angst bzw. Nervosität vor der Operation. Eine Operation ist eine körperliche Anstrengung, die besser gelingt, wenn der Patient ausgeruht und entspannt eine Operation beginnt. Jegliche Form von Aufregung sollte vermieden werden. Der Patient wird daher in seinem Bett in den Operationsbereich gefahren, wo das OP-Team beim Umsteigen auf den Operationstisch hilft.

3.1.5 Postoperativer Verlauf

Der postoperative Verlauf ist ein Anästhesieteilprozess, der zur Validierung und zur Überprüfung der Narkose dient. Der postoperative Verlauf kann entfallen, falls z.B. der schlimmste Fall eintritt und der Patient während der Operation verstirbt. Der postoperative Verlauf kann die Aktivitäten Aufwachraum und postanästhesiologische Visite enthalten.

Aufwachraum

Nach Abschluss der Operation wird der Patient in den Aufwachraum verlegt. In diesem Raum wird das weitere „Wachwerden“ bzw. die Erholung des Patienten nach der Operation überwacht. Seine Atem- und Kreislauffunktionen werden weiter kontrolliert und bei Bedarf medikamentös stabilisiert; eventuell werden ihm Schmerzmedikamente verabreicht, falls es notwendig sein sollte. Ferner werden Komplikationen medikamentös kompensiert. Ist der Patient wach, geht es ihm gut und hat er keine Schmerzen, folgt seine Verlegung zurück auf die Station. Während der gesamten Zeit im Aufwachraum wird der Patient von einem Anästhesie-Team betreut. Bei Bestehen der Notwendigkeit wird die Betreuung auf der Intensivstation fortgesetzt.

Postanästhesiologische Visite

Die postanästhesiologische Visite definiert einen Besuch, den der Anästhesist nach der Narkose beim Patienten auf der Station vornimmt. Vornehmlich dient der Besuch der Erkundigung des Patientenbefindens. Der Anästhesist überprüft dabei, ob der Patient die Anästhesie unbeschadet überstanden hat oder ob der Patient durch Übelkeit, Schwindel etc. beeinträchtigt wird.

3.1.6 Anästhesieprozess

Aufgrund des hohen Grades an Individualität verschiedener Anästhesien kann kein eindeutiges Vorgehen für Planung, Koordination und Durchführung standardisiert vorgegeben werden. Ausschließlich eine grobe Skizzierung des Anästhesieablaufes ist möglich. Die Skizzierung besitzt einen Richtliniencharakter, der eingehalten werden sollte, häufig jedoch nicht eingehalten werden kann. Um die ordnungsgemäße Durchführung einer Anästhesie zu bestätigen, hat der für die Operation eingesetzte Anästhesist die Pflicht, die Anästhesie zu dokumentieren.

3.2 Dokumentation der Anästhesie

Die Anästhesie ist mit Risiken verbunden - wie prinzipiell alle ärztlichen Maßnahmen. Dennoch konnte die anästhesiebedingte Sterblichkeit drastisch reduziert werden und auch schwerwiegende Komplikationen sind selten geworden. Die Ursachen für diese Reduktion sind vielfältig. Einerseits konnte in der Krankenversorgung generell die Qualität gesteigert werden, andererseits hat sich auch die theoretische und praktische Ausbildung von Ärzten und Pflegepersonal verbessert. Entscheidenden Anteil hatte auch die Einführung technischer und organisatorischer Sicherheitsstandards (vgl. [Le05]). Ein wesentlicher Bestandteil ist die Dokumentationspflicht einer ärztlichen Behandlung. Die Dokumentation muss alle Feststellungen, Ergebnisse und Maßnahmen umfassen, die für die Behandlung des Patienten und ihren Verlauf von Bedeutung sind (vgl. [LGH97], [OW99]). Die Dokumentation dient der

- Erfüllung der Nachweispflicht, da die Diagnostik nachvollziehbar sein muss,
- Erfassung der Leistungen, zur Überprüfung der getroffenen Behandlungsmaßnahmen,
- Sicherung der Qualität,
- Informationsweitergabe,
- Transparenz der Verantwortung, die gegenüber dem Patienten besteht, der seine Interessen und Grundbedürfnisse in narkotisiertem Zustand nicht selbst befriedigen kann.

Die Deutsche Gesellschaft für Anästhesiologie und Intensivmedizin (DGAI) hat für die Qualitätssicherung in der Anästhesie allgemeine Richtlinien verabschiedet (vgl. [DGAI]). Diese allgemeinen Rahmenrichtlinien definieren einen „Quasi“-Standard, der einen Kerndatensatz für eine Protokollierung einer Anästhesie vorgibt. Der Kerndatensatz umfasst Empfehlungen, welche Daten im Rahmen einer Anästhesie zu erfassen und zu protokollieren sind.

Die Anästhesieärzte einer Anästhesieabteilung legen im Konsens fest, welche Basisdaten für die Anästhesie in ihrer Anästhesieabteilung als wichtig erachtet werden und erfasst werden müssen. Neben der Festlegung der Basisdaten, die in der Regel den Kerndatensatz beinhalten, wird auch deren Struktur festgelegt. Über die Anzahl an Basisdaten definiert sich die Aussagekraft der abteilungsspezifischen Anästhesie-Dokumentation. Die definierte Menge der Basisdaten wird verwendet, um ein Standardformular zu definieren. Dieses Standardformular wird als Anästhesieprotokoll bezeichnet und es wird verwendet, um eine Anästhesie zu dokumentieren. Die Dokumentation einer Anästhesie in einem Anästhesieprotokoll unterliegt in vollem Umfang der ärztlichen Schweigepflicht sowie datenschutzrechtlichen Bestimmungen.

3.2.1 Anästhesieprotokoll

Ein Anästhesieprotokoll ist ein Dokument, welches – auch zur Qualitätssicherung – den gesamten Anästhesieablauf eines Patienten protokolliert und somit eine einzige Anästhesie dokumentiert. Nach herkömmlicher Auffassung sind Dokumente „*Mitteilungsmittel mit unbefristet gespeicherter, schwer veränderbarer authentischer und sichtbarer Mitteilung in Gestalt von Texten oder Festbildern mit gerichtlicher Beweiskraft*“ (vgl. [K101]). Das Anästhesieprotokoll präsentiert sich als ein Formular, welches die Richtlinien der DGAI schematisch aufbereitet. Es gliedert sich in die drei Teilprotokolle Präoperatives Protokoll, Intraoperatives Protokoll und Postoperatives Protokoll.

1. Im Präoperativen Protokoll werden die prämedikativen Daten dokumentiert, die im Rahmen der Analyse (vgl. 3.1.4) ermittelt werden. Abbildung 12 zeigt eine Formularseite eines Präoperativen Protokolls. Fixiert werden verschiedene patientenspezifische operationsrelevante Angaben, wie Größe und Gewicht des Patienten. Speziell werden Angaben zum Lebenswandel des Patienten dokumentiert, d.h. ob der Patient Raucher, Diabetiker etc. ist, wie sein Verhältnis zu Alkohol ist, ob er regelmäßig Medikamente einnimmt, ob eine Schwangerschaft besteht, ob eine Drogenabhängigkeit existiert, ob ein Hörgerät verwendet wird, etc. Wichtig ist die Klärung des Zahnstatus und des Venenstatus, um Intubationsprobleme zu vermeiden. Ferner ist wichtig, wie der Patient frühere Anästhesien vertragen hat und ob Beschwerden an Organsystemen vorliegen, insbesondere an Herz, Kreislauf, Gefäßen, Lungen und Atemwegen, Speiseröhre, Magen, Darm, Nieren und Harnwegen, Stoffwechsel, Schilddrüse, Augen, Bewegungsapparat, Blut, Nerven, etc. Ebenfalls wichtig ist, ob Allergien vorliegen und wie der Gemütszustand des Patienten ist.

Aufgrund der erhobenen Informationen nimmt der Anästhesist eine Risikoeinschätzung bzgl. möglicher Komplikationen vor und dokumentiert diese ebenfalls im Präoperativen Protokoll. Das Komplikationsrisiko ist wesentlich vom Gesundheitszustand des Patienten abhängig. Erkrankungen und Beeinträchtigungen lebenswichtiger Organsysteme, z.B. der Atmungsorgane und der Kreislauforgane, können mit einem erhöhten Risiko verbunden sein. Krankheiten und Verletzungen, frühere Operationen und Anästhesien, Dauermedikationen und Allergien liefern wichtige Hinweise, die zur Risikoeinschätzung beitragen. Die Befragung wird durch eine gezielte körperliche Untersuchung und gegebenenfalls weiterführende Abklärungen ergänzt. Diese können Laboruntersuchungen, EKG (Herzstromkurve) oder ein Röntgenbild des Brustkorbes sein, welche Zusatzinformationen über den Zustand von Herz und Lungen geben. Die Daten spiegeln den präoperativen Zustand des Patienten wider. Auf Grundlage der Informationen des Präoperativen Protokolls wird für jeden Patienten ein Anästhesieverfahren maßgeschneidert. Ferner wird der präoperative Check-Up protokolliert, d.h. die Auswahl von Geräten und das Instrumentarium sowie Medikamente und Infusionslösungen.

2. Im Intraoperativen Protokoll werden anästhesiespezifische Daten zu Narkoseverfahren und -ablauf dokumentiert. Es werden die Daten dokumentiert, die innerhalb des Kernprozesses (vgl. 3.1.3) zwischen der Narkoseeinleitung und der Narkoseausleitung anfallen. Das Intraoperative Protokoll gibt Aufschluss über den tatsächlich stattgefundenen Ablauf der Narkose. Die Angaben umfassen u.a. Angaben zu den benötigten Schmerz- und Schlafmitteln, die Art und Weise, wie und wann die Medikamente zugeführt wurden, z.B. mit der Beatmung ("inhalativ") oder per Infusion direkt in den Kreislauf. Abschließend werden Überwachungsinformationen bzgl. Blutdrucks, EKG und Sauerstoffsättigung notiert. Wichtig zu dokumentieren sind auch

sogenannte interoperative Anästhesieverlaufsbeobachtungen⁶, die während der Narkose auftraten.

AN-DOK DAS ANÄSTHESIEDOKUMENTATIONSSYSTEM VON DATA-TEC
 Telefon: (07127) 970-000 www.data-tec.de
 Copyright 2000 Henn, Martin, Messelken
 RIECO (05359) 69-400 0400

Name: <input type="text"/>										Geburtsdatum: Tag <input type="text"/> Monat <input type="text"/> Jahr <input type="text"/>										Wahlleistung nein <input type="checkbox"/> ja <input type="checkbox"/>									
Vorname: <input type="text"/>										Geburtsdatum: <input type="text"/>										Geschlecht m <input type="checkbox"/> w <input type="checkbox"/>									
Geburtsdatum: <input type="text"/>										Straße: <input type="text"/>										Gravidität nein <input type="checkbox"/> ja <input type="checkbox"/>									
PLZ: <input type="text"/>										Ort: <input type="text"/>										Aufnahmeort stationär <input type="checkbox"/> teilstationär <input type="checkbox"/> ambulanz <input type="checkbox"/> vorstationär <input type="checkbox"/> nachstationär <input type="checkbox"/> amb. operiert <input type="checkbox"/>									
Station: <input type="text"/> Zi.-Nr.: <input type="text"/>										Diagnose/ geplante Operation: <input type="text"/>										Dringlichkeit elektiv (>24h) <input type="checkbox"/> innerhalb 24 h <input type="checkbox"/> Notfall (innerh. 2h) <input type="checkbox"/> sofort <input type="checkbox"/>									
geplante Anästhesie: <input type="text"/>										Anästhesie relevantes Risiko* keines <input type="checkbox"/> n. beurteilbar <input type="checkbox"/> Schock/Hypovol. <input type="checkbox"/> Über/Untergew.30% <input type="checkbox"/>										Frühgeburt <input type="checkbox"/> MCV <input type="checkbox"/> Sepsis <input type="checkbox"/> ARDS <input type="checkbox"/> sonstiges <input type="checkbox"/>									
Anamnese und klin. Befund o.B. <input type="checkbox"/> Akutversorgung <input type="checkbox"/> Wiederhol. Narkose <input type="checkbox"/>										Zähne* locker/besch. <input type="checkbox"/> o.B. <input type="checkbox"/> Prothese <input type="checkbox"/>										Mallampati I <input type="checkbox"/> II <input type="checkbox"/> III <input type="checkbox"/> n. beurteilb. <input type="checkbox"/>									
EKG* normal <input type="checkbox"/> nicht vorh. <input type="checkbox"/> Tachykard <input type="checkbox"/> Bradykardie <input type="checkbox"/>										Röntgen-Thorax o.B. <input type="checkbox"/> nicht vorhanden <input type="checkbox"/> path. <input type="checkbox"/>										ICD-10 A <input type="checkbox"/> N <input type="checkbox"/> B <input type="checkbox"/> O <input type="checkbox"/> C <input type="checkbox"/> P 0 <input type="checkbox"/> D <input type="checkbox"/> Q 1 <input type="checkbox"/> E <input type="checkbox"/> R 2 <input type="checkbox"/> F <input type="checkbox"/> S 3 <input type="checkbox"/> G <input type="checkbox"/> T 4 <input type="checkbox"/> H <input type="checkbox"/> V 5 <input type="checkbox"/> I <input type="checkbox"/> W 6 <input type="checkbox"/> J <input type="checkbox"/> X 7 <input type="checkbox"/> K <input type="checkbox"/> Y 8 <input type="checkbox"/> L <input type="checkbox"/> Z 9 <input type="checkbox"/>									
Bewusstsein normal <input type="checkbox"/> somnolent <input type="checkbox"/> komatös <input type="checkbox"/> sediert <input type="checkbox"/>										Myokard o.B. <input type="checkbox"/> n. beurteilb. <input type="checkbox"/> kompensiert <input type="checkbox"/> Bei Insuff. <input type="checkbox"/> dekompens. <input type="checkbox"/>										Koronarien* normal <input type="checkbox"/> n. beurteilb. <input type="checkbox"/> KHK <input type="checkbox"/> Stabile AP <input type="checkbox"/> Instabile AP <input type="checkbox"/>									
Herzinfarkt n.bekannt <input type="checkbox"/> <3 Monate <input type="checkbox"/> <6 Monate <input type="checkbox"/> >6 Monate <input type="checkbox"/>										Blutdruck Normotonie <input type="checkbox"/> n. beurteilb. <input type="checkbox"/> Hypertonie beh. <input type="checkbox"/> Hypert. unkontr. <input type="checkbox"/> Hypotonie <input type="checkbox"/>										Gefäße* o.B. <input type="checkbox"/> n. beurteilb. <input type="checkbox"/> Varicosis <input type="checkbox"/> AVK <input type="checkbox"/> Thromb/Emb. <input type="checkbox"/> zereb. Dblst. <input type="checkbox"/> sonstiges <input type="checkbox"/>									
Atmung* o.B. <input type="checkbox"/> n.b. <input type="checkbox"/> Obstruktion <input type="checkbox"/> Restriktion <input type="checkbox"/> Infekt <input type="checkbox"/> Beatmung <input type="checkbox"/>										Leber o.B. <input type="checkbox"/> n.b. <input type="checkbox"/> path. <input type="checkbox"/>										Niere o.B. <input type="checkbox"/> n.b. <input type="checkbox"/> komp. NI <input type="checkbox"/> term. NI <input type="checkbox"/> akute NI <input type="checkbox"/>									
Neurologie* o.B. <input type="checkbox"/> path. PNS <input type="checkbox"/> path. ZNS <input type="checkbox"/>										Muskelerkrankung n. bekannt <input type="checkbox"/> n. beurteilb. <input type="checkbox"/> gesichert <input type="checkbox"/> MH Disposition <input type="checkbox"/>										Blutungsneigung ASS bis <input type="checkbox"/> NSAR bis <input type="checkbox"/>									
Cor: <input type="text"/>										Diabetes kein <input type="checkbox"/> n.b. <input type="checkbox"/> insulinpfl. <input type="checkbox"/> oral eing. <input type="checkbox"/> nicht eing. <input type="checkbox"/>										ASA I <input type="checkbox"/> II <input type="checkbox"/> III <input type="checkbox"/> IV <input type="checkbox"/> V <input type="checkbox"/>									
Pulmox: <input type="text"/>										Labor* o.B. <input type="checkbox"/> Hb: <input type="text"/> Hk: <input type="text"/>										nicht vorhanden <input type="checkbox"/> Thr: <input type="text"/>									
Fremdblut <input type="checkbox"/> Eigenblut <input type="checkbox"/> Eigenplasma <input type="checkbox"/>										kleines BB path. <input type="checkbox"/> Quick: <input type="text"/>										Gerinnung path. <input type="checkbox"/> K: <input type="text"/>									
Blutgruppe <input type="text"/> AK-Suchtest <input type="checkbox"/>										Elektrolyte path. <input type="checkbox"/> BZ: <input type="text"/>										BGA path. <input type="checkbox"/> Krea: <input type="text"/>									
Präoperative Anordnungen: <input type="text"/>										and. path. Werte <input type="text"/> γGT: <input type="text"/>										Puls: <input type="text"/> RR: <input type="text"/> Größe: <input type="text"/> Gewicht: <input type="text"/>									
Blutkonserven: <input type="text"/>										Dauermedikation: <input type="text"/>										Cor: <input type="text"/>									
Prämedikation Vorabend <input type="text"/>										Medikament <input type="text"/>										Applikation <input type="text"/>									
Prämedikation OP-Tag <input type="text"/>										Uhrzeit/Unterschrift <input type="text"/>										Name/Unterschrift <input type="text"/>									
AnästhesistIn 0 <input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5 <input type="checkbox"/> 6 <input type="checkbox"/> 7 <input type="checkbox"/> 8 <input type="checkbox"/> 9 <input type="checkbox"/>										Dienstart* FD <input type="checkbox"/> BD <input type="checkbox"/>										Zeitaufwand 5 <input type="checkbox"/> 10 <input type="checkbox"/> 15 <input type="checkbox"/> 20 <input type="checkbox"/>									
H <input type="checkbox"/> Z <input type="checkbox"/> E <input type="checkbox"/>										25 <input type="checkbox"/> 30 <input type="checkbox"/> 35 <input type="checkbox"/> 40 <input type="checkbox"/>										45 <input type="checkbox"/> 50 <input type="checkbox"/> 55 <input type="checkbox"/> 60 <input type="checkbox"/>									
Wochenende/Feiertag nach 20 Uhr <input type="checkbox"/>										Anästhesieambulanz <input type="checkbox"/>										nach Prämedikation keine OP <input type="checkbox"/>									
AN-DOK Art. Nr. 8010 BS 6/0/1 Prämedikation										P.-Nr.: <input type="text"/>																			

Abbildung 12: Formulareseite eines präoperativen Protokolls

⁶ Eine Anästhesieverlaufsbeobachtung ist ein technischer Defekt, eine Überschreitung der geplanten OP-Dauer oder ein Ereignis während einer Anästhesie, welches den Anästhesisten zum Eingreifen zwingt, um den weiteren Schlaf und die weitere Schmerzfreiheit des Patienten zu gewährleisten und sicherzustellen.

3. Im Postoperativen Protokoll werden die begleitenden Maßnahmen dokumentiert, die den Patienten aus seinem narkotischen Schlaf holen. Die Dokumentation startet mit der Verlegung des Patienten in den Aufwachraum. Sie endet spätestens mit dem Abschluss der gegebenenfalls stattfindenden postanästhesiologischen Visite. Protokolliert werden, wie auch während der Anästhesie, die Vitalparameter, Angaben zu verabreichten Medikamenten, die fortgesetzte Überwachung von Blutdruck, EKG und Sauerstoffsättigung, und aufgetretene postoperative Anästhesieverlaufsbeobachtungen. Generell werden die Daten im Postoperativen Protokoll erfasst, die während der Validierung (vgl. 3.1.5) auftreten.

Neben den primären Informationen, d.h. den patientenspezifischen Angaben, werden in den drei Teilprotokollen noch sekundäre und tertiäre Informationen erfasst. Sekundäre Informationen stellen behandlungsspezifische Angaben dar, die später anästhesiespezifisch ausgewertet werden können, um als Erfahrungen aus der Patientenversorgung Eingang in medizinische Versorgungsverfahren zu finden. Tertiäre Informationen sind verwaltungsspezifische Daten. Es handelt sich dabei um Angaben zum jeweils durchführenden Anästhesisten und unterstützenden Anästhesiepflegefachkräften, Zeitangaben, dienstspezifische Angaben sowie die Erfassung der Krankenkasse des Patienten.

Real präsentiert sich ein Anästhesieprotokoll als ein mehrseitiges Formular, welches nach der schriftlichen Erfassung einer Anästhesie durch die Unterschrift des verantwortlichen Anästhesisten bei der Operation den Dokumentcharakter erhält. Die Anfertigung und Lagerung dieses schriftlichen Protokolls besitzt Vor- und Nachteile, die im Folgenden kurz umrissen werden.

Die Vorteile eines Protokolls zur Anästhesie-Dokumentation sind:

- Das Standardformular beschreibt und vereinheitlicht den krankenhauspezifischen Anästhesieablauf. Das Anästhesieprotokoll liefert so eine relativ lückenlose Dokumentation aller stattgefundenen prä-, intra- und postoperativen medizinischen und pflegerischen Beobachtungen und Maßnahmen bezüglich einer Anästhesie.
- Die Standardisierung der Anästhesiedokumentation erlaubt die Standardisierung und die Optimierung von Anästhesieabläufen.
- Jede fachlich ausgebildete Person kann sich anhand der patientenspezifischen Anästhesiedokumentation sofort über den Zustand des Patienten informieren, orientieren und zielgerecht agieren.
- Falls Behandlungsfehler zu juristischen Streitfällen führen, kommen den Anästhesieprotokollen tragende Bedeutungen zu. Anhand der dokumentierten Komplikationen, die während der Operation auftraten, kann der Anästhesist seine ergriffenen Maßnahmen rechtfertigen.
- Mehrere Anästhesieprotokolle können hinsichtlich Gemeinsamkeiten ausgewertet werden, um behandlungsspezifische Abhängigkeiten zu suchen, die Rückschlüsse für eine verbesserte Behandlung zulassen und letztlich Eingang in neue Versorgungsverfahren haben.

Die Anästhesie-Dokumentation mittels eines Anästhesieprotokolls beinhaltet allerdings folgende Nachteile:

- Ein Anästhesist muss sich in erster Linie um den Patienten kümmern, d.h. eine zeitgleiche Anästhesiedokumentation seiner Tätigkeiten kann nicht erfolgen. Die Protokollierung erfolgt somit immer nachträglich.
- Mit einem Standardformular sind Probleme bzgl. dessen Aktualität zu berücksichtigen. Ständige Neuerungen in Medizin und Pflege ziehen häufig regelmäßige Überarbeitungen des Standardformulars nach sich.
- Bereits eine Personalfuktuation kann eine Überarbeitung des Standardformulars zur Folge haben, da sich der Konsens über die Anzahl der zu erhebenden Basisdaten in der neuen personellen Struktur verändert.
- Ein oder mehrere Anästhesisten nehmen die zeitaufwändigen Korrekturen des Standardformulars während ihrer Arbeitszeit vor. Deren Ausfallzeit muss von den weiteren Anästhesisten der Anästhesieabteilung kompensiert werden.
- Auf dem Standardformular sind sehr viele Informationen untergebracht, um eine möglichst große Bandbreite von Fällen abzudecken. Im Standardformular wird somit versucht, das gesamte Anästhesieszenario in all seinen Ausprägungen abzudecken. Häufig nehmen die tatsächlich einzutragenden Informationen eines tatsächlichen Anwendungsfalles nur einen kleinen Teil ein.
- Anästhesisten stellen fest, dass immer wieder Informationen zu erfassen sind, die nicht ins (Standardformular-) Schema passen; d.h. im Standardformular sind keine Angabemöglichkeiten vorgesehen, die sich auf diese zu erfassenden Informationen beziehen. Beispielsweise hat ein Patient eine ganz spezielle Allergie. Im Standardformular ist diese spezielle Allergie nicht aufgeführt. Der Anästhesist behilft sich anschließend mit dem schriftlichen Eintrag in einem Feld für sonstige Angaben. Das ursprüngliche Allergiefeld ist obsolet.
- Das Standardformular kann zu einer Fixierung auf den zu erhebenden Informationsumfang der Anästhesie führen, d.h. die Erhebung weiterer medizinischer Daten erfolgt nicht oder die Durchführung medizinischer Leistungen wird nicht vorgenommen, weil entsprechende Vermerke nicht im Standardformular stehen.
- Das gesamte Anästhesiepersonal verliert an Flexibilität, da teilweise versucht wird, den Patienten an den Standard anzupassen statt umgekehrt.
- Die Anästhesieprotokolle müssen archiviert werden. Zu Archivierungszwecken müssen entsprechende Lagerkapazitäten geschaffen werden.
- Die inhaltliche Korrektheit eines Anästhesieprotokolls kann nicht sichergestellt werden. Daher bestätigt der Anästhesist die Richtigkeit des Anästhesieprotokolls durch das Setzen seiner Unterschrift auf dem Dokument. Er übernimmt die Haftung für den Inhalt des Anästhesieprotokolls. Die inhaltliche Korrektheit liegt folglich ganz im Ermessen des haftenden Anästhesisten.

3.2.2 Anästhesie-Dokumentationssystem

In einer Klinik, die die übliche medizinische Grundversorgung der Bevölkerung sicherstellt, fallen schätzungsweise 25.000 Operationen⁷ pro Jahr an. Die damit verbundene Menge an Anästhesieprotokollen führt zu der Anforderung, diese Dokumente hinsichtlich auffälliger Anästhesieverlaufsbeobachtungen auszuwerten, um Hinweise auf mögliche Verbesserungen in der Behandlung von Patienten zu erhalten. Bis vor wenigen Jahren wurde solch eine Auswertung unter hohem personellem Aufwand manuell vorgenommen. Die Reduzierung des hohen Aufwands versuchte man bereits in den 1980er Jahren durch Einführung automatischer Anästhesie-Dokumentationssysteme zu erreichen. Das grundlegende Ziel eines Anästhesie-Dokumentationssystems ist die optimale Unterstützung des Anästhesisten bei der Dokumentation seiner Anästhesie. Die Unterstützung soll weitestgehend durch die automatische Verarbeitung der Daten gegeben sein. Eingang in Anästhesieabteilungen fanden automatische Anästhesie-Dokumentationssysteme erst mit der zunehmenden Verbreitung grafischer Bedienoberflächen. Allgemein existiert eine Unterteilung von Anästhesie-Dokumentationssystemen in die beiden Klassen Offline- und Online-System.

3.2.3 Beleglese-Systeme

Es gibt drei Kategorien von Offline-Systemen (vgl. [Qi03]). Die erste ist die ausschließliche manuelle Erfassung auf einem Papierformular. Diese Kategorie entspricht der Beschreibung zum Anästhesieprotokoll (siehe 3.2.1) und kennzeichnet die Lösung, die durch eine technische Lösung abgelöst werden soll. Die zweite Kategorie ist gegeben durch eine manuelle Erfassung des Anästhesieprotokolls und einer manuellen computergestützten Nacherfassung an einem PC. Bei der Nacherfassung handelt es sich in der Regel um eine Teilerfassung von bestimmten administrativen Daten, wie z.B. Arbeitszeiterfassungen. Auch diese Kategorie soll durch eine technisch umfassendere Lösung abgelöst werden. Die dritte Kategorie umfasst eine manuelle Dokumentation der Anästhesie auf einem speziellen Belegformular, welches nach Abschluss der Dokumentation automatisiert über ein Beleglese-System in ein Anästhesie-Informationssystem eingelesen wird.

Beleglese-Systeme sind die aktuelle Standardlösung für automatische Datenerfassung von Anästhesieprotokollen (vgl. [KS06]). An vielen Kliniken sind aktuell die Produkte ANDOC und MALENA II im Einsatz, die zwei klassische Vertreter der Beleglese-Systeme sind. Da eine ausführliche Beschreibung von Beleglese-Systemen in [Qi03] gegeben ist, wird im Folgenden nur das Verständnis für Beleglese-Systeme geklärt.

Der Einsatz von Beleglese-Systemen führte zu einer Layoutumgestaltung der Anästhesieformulare in eine Belegform. Das Layout eines solchen Beleges ähnelt dem Layout eines Multiple-Choice-Formulars (vgl. Abbildung 12). Ein Beleg hat die Vorteile, dass der Anästhesist seine Dokumentationspflicht auf ein „schnelles Ankreuzen“ reduzieren kann und die Belege mittels eines Beleglesers maschinell verarbeitet werden können. Diese Lösung setzt notwendigerweise voraus, dass der Anästhesist seine Anästhesie auf den vorgesehenen Belegen protokolliert. Nach Abschluss der Anästhesie ist auch die Protokollierung abgeschlossen und die Belege sind ausgefüllt. Die ausgefüllten Belege werden vom Anästhesie-Dokumentationssystem mittels Belegleser eingelesen und maschinell verarbeitet.

⁷ Der Wert basiert auf den Erfahrungen aus dem Drittmittelprojekt MC-MEDIS des Fachbereichs Informatik – AB Technische Informatiksysteme der Universität Hamburg mit der Fa. DATAPEC GmbH aus dem Jahre 2001.

Bei der maschinellen Verarbeitung durch den Belegleser werden die markierten Daten vom Beleg extrahiert und an das angeschlossene Informationssystem übermittelt. Das Informationssystem speichert die Daten in seiner Datenbank. Das interne Datenbankschema bildet in der Regel den vollen Umfang aller Informationen im Beleg ab, d.h. die Datenbanken wurden auf Grundlage der in den Belegen zu erfassenden Daten entwickelt. Informationssysteme solcher Art werden als Anästhesie–Dokumentationssysteme bezeichnet.

Die Anästhesie-Dokumentationssysteme beinhalten intern vorhandene Prüfredeln. Diese werden zu einer Prüfung herangezogen, um die Plausibilität der Daten sicherzustellen. Die Prüfredeln, im Folgenden als Plausibilitäten bezeichnet, spiegeln heuristische Erfahrungen aus konkreten Behandlungen wider, die als Richtlinien Eingang in ärztliche Verfahrensweisen gefunden haben. Die Plausibilitäten besitzen einen dynamischen Charakter. Sie erheben keinen Anspruch auf Allgemeingültigkeit und werden durch den Anästhesisten vorgegeben. Ein Anästhesie–Dokumentationssystem stuft anhand der Plausibilitäten Anästhesie–Protokolle entweder als korrekt ein oder bemängelt sie als fehlerhaft.

Nach dem Einlesen der Belege per Belegleser ins System und der Plausibilitätsprüfung besteht die Möglichkeit der Nachbearbeitung der Daten. Die Nachbearbeitung geschieht am Bildschirm mit Hilfe der Bedienoberfläche des bestehenden Anästhesie-Dokumentationssystems und dient zur Korrektur von Protokollen, die die Plausibilitätsprüfung nicht bestanden haben. Die Bildschirmmaske dient nicht nur der Nachbearbeitung. Alternativ bieten Anästhesie-Dokumentationssysteme häufig die Möglichkeit der bildschirmgestützten Erfassung einer Anästhesie an, so dass auf eine schriftliche Protokollierung verzichtet werden könnte. Aus rechtlichen Gründen muss allerdings ein schriftliches Protokoll grundsätzlich angefertigt werden.

Neben der Einsichtnahme der gespeicherten Datenbestände über die Bedienoberfläche bieten aktuelle Anästhesie-Dokumentationssysteme die Möglichkeit zur statistischen Auswertung der gespeicherten Datenbestände. Die Auswertungsergebnisse sind für die Anästhesisten von Bedeutung.

3.2.4 Anästhesie-Information-Management-System

Die Klasse der Online-Systeme ist durch Anästhesie-Information-Management-Systeme (AIMS / AMS) geprägt. Die historische Entwicklung der Online-Systeme ist in [Qi03] beschrieben, so dass erneut nur das Verständnis für Anästhesie-Information-Management-Systeme im Folgenden geklärt wird.

Nach [Br07] sind unterschiedliche Synonyme für diese Art von Systemen im Gebrauch, z.B. Patienten-Daten-Management-System (PDMS). Bei Systemen dieser Klasse handelt es sich um spezielle klinische Anästhesie-Arbeitsplatzsysteme, die eine direkte computergestützte Erfassung der Anästhesie unterstützen. Sie sind in der Regel mit anderen Subsystemen gekoppelt, wie z.B. Laborsysteme oder OP-Dokumentationssysteme oder mit dem Krankenhaus-Informationssystem (KIS) der Klinik. Durch die Kopplung besteht die Möglichkeit, Daten mit den angeschlossenen Systemen auszutauschen und die Datenbasis automatisiert zu vervollständigen.

Durch den Einsatz von solchen Online-Anästhesie-Dokumentationssystemen wird angestrebt, den Arbeitsaufwand, den ein Anästhesist bei der Protokollierung hat, zu vermindern. Ein Anästhesie-Dokumentationssystem soll unter Verantwortung des Anästhesisten einen so

großen Dokumentationsaufwand wie möglich übernehmen. Die Erfassung einer Anästhesie wird nicht mehr nachträglich im System vorgenommen, sondern begleitend zur Anästhesie.

Zwei Vertreter solcher Systeme sind MedAktIS und NarkoData. Die Abkürzung MedAktIS steht für Medizinisches Aktivitätsbasiertes Informationssystem und bezeichnet ein medizinisches Informationssystem, welches als Lehrreponat zu Lehr- und Studienzwecken von der Firma MEDACTIS GmbH im Rahmen eines Projektes an der Fachhochschule Dortmund entwickelt wurde (vgl. [Ha05]). NarkoData ist ein kommerziell erhältliches System. Beide Systeme sind selbst Gegenstand aktueller Forschung und befinden sich noch in der (Weiter-) Entwicklung. Exemplarisch wird der Stand der Forschung am System NarkoData vorgestellt.

3.2.5 Stand der Forschung

NarkoData ist gemäß [NDO05] ein Online-Dokumentationssystem für die Anästhesie. Es bietet Unterstützung bei der Erfassung und Bearbeitung von prä-, intra- und postoperativen Anästhesiedaten. Diese Daten werden in einer relationalen Datenbank gespeichert. Zum Einsatz kommt eine Oracle-Datenbank. Die Applikation NarkoData realisiert eine Hauptkomponente des Anästhesie-Informations-Management-Systems und arbeitet auf der Datenbank. Sie ermöglicht administrative, statistische und wissenschaftliche Auswertungen auf dem Datenbestand. Die Applikation ist in der Programmiersprache C++ implementiert worden.

Das Narkosedokumentationssystem NarkoData selbst muss stark in die Infrastruktur des umgebenden Krankenhausinformationssystems (KIS) integriert werden, um den Anwendern einen Nutzen zu bringen. Alle prä- und intraoperativ anfallenden Daten werden online am Arbeitsplatz des Anästhesisten in elektronischer Form dokumentiert (vgl. [MBR04]). Bei einem solchen Arbeitsplatz handelt es sich um einen Standard-PC. Je nachdem, wieviele Narkosearbeitsplätze eingerichtet werden, sind entsprechend viele Rechner mit der Applikation auszustatten und mit dem KIS zu vernetzen.

Mittels einer graphischen Bedienoberfläche werden sowohl die manuell im Verlauf der Narkose erhobenen Daten (z.B. Arzneimittel, OP-Zeiten, etc.) als auch die automatisch übernommenen Monitoringdaten (z.B. für Blutdruck, Herzfrequenz, etc.) analog zum bekannten Papierprotokoll dargestellt.

Diese Vorgehensweise ermöglicht eine weitestgehend vollelektronische Erfassung und Dokumentation von Narkosen. Von Vorteil ist eine sich ergebende Arbeitersparnis, da der automatische Import vieler Daten die notwendige manuelle Erfassung auf die wenigen Daten reduziert, die nicht importierbar sind, z.B. eine ungewöhnliche Anästhesieverlaufsbeobachtung. Außerdem bietet das System die Möglichkeit eines automatischen Datenexports, womit die Notwendigkeit gesonderter Datenerfassung zur Leistungsdokumentation entfällt.

Der unterstützte Datenaustausch mit weiteren Teilsystemen des KIS bietet ferner den Vorteil, dass aufgrund der direkten Kommunikation Fehler durch Falscheingaben weitgehend ausgeschlossen werden. Maschinell kommunizierte Daten führen zu einer erhöhten Datenqualität, da sie nicht Ursache von manuellen Übertragungsfehlern sein können.

3.2.6 Probleme von Anästhesie-Dokumentationssystemen

Die hohe Anzahl an anfallenden Anästhesieprotokollen legt eine elektronische Verarbeitung der Anästhesieprotokolle und der darauf fixierten Daten nahe. Die bestehenden Anästhesie-Dokumentationssysteme erlauben

- die kontinuierliche Dokumentation mit hoher Qualität,
- die Vergleichbarkeit der Daten aufgrund der Verfügbarkeit in einer Datenbank und
- die Verfügbarkeit zusätzlicher Stammdaten⁸, die zur Validierung herangezogen werden.

Doch der Einsatz bestehender Anästhesie-Dokumentationssysteme führt auch zu Problemen. Die wichtigsten Nachteile werden im Folgenden kurz umrissen.

- **Kosten:** Die Anschaffungskosten für ein Anästhesie-Dokumentationssystem sind sehr hoch. Die hohen Kosten spiegeln den Aufwand wider, den ein Dienstleister hat, um eine zufriedenstellende Lösung zu entwickeln. Leider sind die hohen Anschaffungskosten nicht vereinbar mit dem Kostendruck, der in sämtlichen medizinischen Bereichen existiert. Von den Sparmaßnahmen sind u.a. auch die Ausgaben für die EDV betroffen, d.h. es wird sowohl beim Kauf von Hard- als auch beim Kauf von Software gespart. Bzgl. der Hardwareausgaben kann das dazu führen, dass kostengünstige Rechner gekauft werden, z.B. von Lebensmitteldiscountern; bzgl. der Softwareausgaben führt deren Limitierung zum Erwerb von Softwarepaketen mit geringerem Leistungsumfang. Darauf reagieren die Softwareanbieter, indem teure Entwicklungen für Individuallösungen gar nicht erst in Angriff genommen, sondern Standardlösungen angeboten werden.
- **Standardsoftware:** Jede Anästhesieabteilung bräuchte ihre eigene, spezifische Systemlösung. Applikationen, die spezifische Lösungen realisieren, müssen gezielt für eine Anästhesieabteilung entwickelt werden. Eine gezielte Entwicklung ist in den meisten Fällen nur mit hohem Kostenaufwand möglich, die, wie oben erwähnt, das Budget einer Klinik übersteigen. Ein Softwareanbieter aus der IT-Branche kann folglich nur ein Einzelsoftwareprodukt etablieren, welches als ein Standardsystem für mehrere Anästhesieabteilungen in Frage kommt. Bei den Anforderungen hält sich der Softwareanbieter an die Rahmenrichtlinien der DGAI. Bezüglich dieser Rahmenrichtlinien (vgl. [DGAI]) steht es jeder Anästhesieabteilung frei, ob sie die Rahmenrichtlinien in Gänze in ihr eigenes Anästhesieprotokoll übernimmt oder nur in Teilen. Die Entscheidungshoheit darüber obliegt der Abteilung. Bezogen auf die Entscheidung für ein Softwareprodukt ist die Anästhesieabteilung ausschließlich auf die gesamten Rahmenrichtlinien eingeschränkt, welche die Basis des kleinsten gemeinsamen Nenners bilden und in dem Standardsystem realisiert sind. Die Standardsoftware genügt somit generell nicht den Ansprüchen der Anästhesisten. Hinsichtlich der Offline-Dokumentation ist die Mehrseitigkeit der Anästhesiebelege eine Herausforderung für viele Beleglese-Systeme. Es besteht in der Praxis die Möglichkeit, dass einige Seiten eines Belegs nicht ausgefüllt werden und beim Einlesen nicht zur Verfügung stehen. Aus Systemensicht ergibt sich eine Inkonsistenz, da die Systeme generell voraussetzen, dass alle Seiten eines Belegs vorhanden sein müssen.

⁸ Unter Stammdaten werden krankenhausspezifische Angaben verstanden, die in dem Informationssystem hinterlegt sind. Es handelt sich um Informationen zu Personal, Operationssälen, Dienstplänen, etc.

- **Bedienoberflächen:** Die Bedienoberfläche einer Anästhesie–Dokumentation ist mit zu vielen Fenstern überfrachtet. Eine reale 1:1-Anzeige eines Beleges in einer einzelnen Bildschirmmaske ist aus rein technischen Gründen nicht möglich, da die Auflösung der Bildschirme zu gering ist. Erst Bildschirme neuester Generation könnten sich als nützlicher erweisen. Aufgrund der technischen Restriktion bzgl. der Auflösung und der damit verbundenen Aufbereitung der Informationen wurden die Informationen eines Beleges gemäß einer medizinisch fachlichen Gruppierung in einzelnen Dialogmasken untergebracht. Ein Anwender muss sich in die Software einarbeiten, um eine gewisse Vertrautheit mit der Anwendung zu erlangen. Erst eine Vertrautheit ermöglicht die zügige Bearbeitung über die Bedienoberfläche. Liegt diese Vertrautheit nicht vor, kann das Auffinden der Eingabefelder zu bestimmten Informationen in eine Odyssee über sämtliche Fenster der Anwendung hinweg ausarten. Erschwerend ist die Abbildung des Layouts der Anästhesieprotokolle in die Bedienoberflächen, da so die Bildschirmmasken das starre Schema des Beleges zeigen. Über einen Wiedererkennungseffekt gestaltet sich die Bedienung für den Anästhesisten zwar einfacher, aber die Bedienoberflächen übernehmen dadurch die formularbasierten Nachteile, dass einerseits eventuell nicht benötigte Informationen visualisiert werden, und dass andererseits Informationen, die nicht ins (Beleg-)Schema passen, schwierig zu erfassen sind. Eine dynamische Anpassungsfähigkeit an patientenindividuelle Daten fehlt den Systemen.
- **Zeitaufwand:** Die Dokumentation per Formular erfolgt nachträglich (siehe Nachteile unter 3.2.1). Eine belegleserunterstützte Erfassung im Anästhesie-Dokumentationssystem am Bildschirm erfolgt ebenfalls nachträglich. Anwenderseitig wird bemängelt, dass diese Art der EDV-basierten Bearbeitung zeitaufwändiger ist als die Bearbeitung eines Formulars.
- **Plausibilitäten:** Anästhesieabteilungen, denen die Standardsoftware für ein Beleglese-System bzgl. des Beleglayouts genügt, haben den Anspruch, die Plausibilitäten, die sie als wichtig erachten, je nach Meinungsbildung der Anästhesisten spezifisch definieren zu können. Anästhesie–Dokumentationssysteme integrieren die Plausibilitätsanforderungen meist codiert im Quellcode, so dass eine dynamische Änderung nicht unterstützt werden kann. Dieser Nachteil wiegt schwer, da gerade im medizinischen Bereich häufig neue Forschungsergebnisse aktuell vorhandene Informationen ersetzen bzw. erweitern und in den Systemen hinterlegt werden müssten. Mangels dynamischer Änderbarkeit können neue Erkenntnisse nicht schnell genug in die Systeme integriert werden.

Besonders bei Belegleser-Systemen muss eine Plausibilitätsprüfung korrekt durchzuführen sein, trotz eventuell existierender belegseitenübergreifender Plausibilitäten. Fehlt eine Belegseite, so soll der Fehler vom System automatisch bemängelt werden, aber auf eine Plausibilitätskontrolle darf nicht verzichtet werden.

Plausibilitäten bedingen die Eingabemöglichkeiten der Bedienoberflächen der Anästhesie-Dokumentationssysteme. Über die Bedienoberfläche kann ein Anästhesieprotokoll am Bildschirm erfasst werden. In die Bildschirmmasken sind die Plausibilitäten als Eingabebedingungen integriert. So reagiert eine Bildschirmmaske bei der Erfassung auf Falscheingaben, wenn eine bestimmte Plausibilität definiert ist, und gibt eine vorgegebene Fehlermeldung aus. Einer eventuell durch die Eingabe ermöglichte Manipulation des Datenbestandes wird so entgegengewirkt. Allerdings kann eine Eingabe trotz Plausibilitätsverletzung korrekt sein. Das System verhindert in diesem Fall die eigentlich korrekte Eingabe.

- Fehlende Entscheidungsgewalt: Auf dem papiergestützten Formular bestätigt der dokumentierende Arzt durch seine Unterschrift die Korrektheit des Protokolls. In Anästhesie-Dokumentationssystemen fehlt das Angebot eines entsprechenden Mechanismus. Die Entscheidungsgewalt, ob ein fehlerhaft bemängeltes Anästhesieprotokoll tatsächlich Fehler aufweist, liegt nicht mehr beim Anästhesisten, sondern beim Anästhesie-Dokumentationssystem.

Zur Ermöglichung der Eingabe von aus Systemsicht inkonsistenten Daten werden Plausibilitätskontrollen auf ein Minimalmaß einiger weniger Prüfungen reduziert. Die Verantwortung der Korrektheit der Daten obliegt wieder dem Anwender. Das Anästhesie-Dokumentationssystem kann keine Garantie mehr geben, dass die Daten nach einer interaktiven Bearbeitung bzw. Erfassung am Bildschirm korrekt vorliegen. Der Anwender erhält somit keine rechnergestützte Kontrollunterstützung.

- Statistik: Anästhesie-Dokumentationssysteme bieten jeweils eine Auswertungs-Komponente an, die statistische Auswertungen auf Basis des gesamten Datenbestandes erlauben. Die Aussagekraft von Auswertungen, die auf nicht plausiblen Daten vorgenommen werden können, ist zweifelhaft. Andererseits bietet sich durch den Zugriff auf als inkonsistent klassifizierte Daten die Möglichkeit, den Datenbestand hinsichtlich seiner Korrektheit auszuwerten.
- Warten in vernetzten Systemen: Daten sollten nur einmalig erfasst werden. Auf den Anästhesieprotokollen werden z.B. patientenspezifische Daten erhoben, die ebenfalls vom Operateur, z.B. Chirurgen, und ebenfalls bei der Aufnahme im Krankenhaus erhoben werden. Da alle medizinischen Fachabteilungen mittlerweile über Systeme verfügen, die sich vernetzen lassen, werden i.d.R. bereits Daten ausgetauscht. Dabei sieht die systemseitige Realisierung die Erhebung von bestimmten Datensätzen gezielt in bestimmten Fachabteilungen vor. Dies hat zur Folge, dass Anästhesisten von den Eingaben Dritter abhängig sind. Beispielsweise darf in einem mündlich berichteten Fall nur ein Operateur eine Operation im System anlegen. Diese Daten bekommen schließlich Anästhesisten, die die anästhesiespezifischen Operationsdaten daraufhin ergänzen. Falls der Operateur seine Eintragungen nicht vornehmen kann, weil er terminlich keine Zeit findet oder es schlichtweg vergisst, sind alle weiteren betroffenen Mediziner bzgl. ihrer Dateneingabe blockiert; d.h. die Informationen laufen in Papierform auf und die Betroffenen sehen sich gezwungen in einer Gewaltaktion einen ganzen Papierstapel an Daten dann einzugeben, wenn der Operateur seiner EDV-Aufgabe endlich nachgekommen ist.

Ein anderes Problem ist, dass bereits erfasste Daten nicht rechtzeitig an die Stationen weitergeleitet werden, die die Daten ebenfalls benötigen. So erfahren Anästhesisten beispielsweise, dass Patienten auf einer Station liegen und sie die Visite durchführen können. Die Aufnahmedaten des Patienten liegen in diesem Moment dem Anästhesisten noch nicht vor, obwohl sie im System bereits erfasst wurden. Da der Anästhesist den Patienten nicht warten lassen soll, nimmt er dennoch zusätzlich die Patientendaten bei seiner Visite auf.

3.2.7 Anforderungen an Anästhesie-Dokumentationssysteme

Trotz der Nachteile zeichnet sich ein Trend ab, der von der papiergestützten Erfassung wegführt. Im Kern werden weiterhin die Daten erfasst, die es im Anästhesieprotokoll zu erfassen gilt. Die Art der Erfassung ändert sich hin zu einer rechnergestützten Erfassung, wobei eine Überführung des Papierdokumentes in ein elektronisches Dokument stattfindet. An Informationssysteme, die eine derartige Unterstützung bieten sollen, werden Anforderungen gestellt, die im Folgenden besprochen werden. Die Besprechung entspricht dabei der schematischen Vorlage, wie sie in [Br07] angegeben ist.

Grundlegende Anforderungen

- Die Dokumentation erfolgt in einem einheitlichen, lesbaren und vollständigen elektronischen Dokument. Der zu erfassende Datensatz des Dokumentes umfasst neben dem Kerndatensatz der DGAI Daten über den Patientenzustand, zu dessen Lagerung während der Narkose, zu patientenspezifischen Messungen und Befunden, über die Verabreichung von Medikamenten und Infusionen sowie deren Verbrauch, zu invasiven Maßnahmen und intraoperativen Zusatzmaßnahmen, zu eingesetzten Techniken und verwendeten Anästhesieverfahren. Ergänzend sind administrative Daten zu erfassen, die Informationen zu Arbeitszeiten und zum zuständigen Personal liefern. Den Umfang des zu erfassenden Datensatzes legt die jeweilige Anästhesieabteilung fest, die das System einsetzt. Das System unterstützt dies bei der Festlegung des Datensatzes.
- Jedes Datum eines Datensatzes ist derart ausprägbar, dass individuell eingestellt werden kann, welche Werte für die Angabe zulässig sind.
- Für betriebswirtschaftliche, wissenschaftliche und qualitätssichernde Zwecke müssen verschiedene Datenaggregationen unterstützt werden. Dazu zählen verschiedenste Plausibilitäts- und Vollständigkeitskontrollen sowie die Unterstützung der Kodierung aus der klinischen Dokumentation heraus, wie z.B. die Verwendung des passenden ICD-Codes. Gängige Kodierhilfen, wie der ICD-Code müssen ins System integriert werden können, eventuell sogar mittels einer eigenen Katalogverwaltung.

Funktionsspezifische Anforderungen

- Unabhängig von individuellen Vorgaben müssen auch Eingaben erlaubt sein, die Werte setzen, die nicht im zulässigen Wertebereich eines Datums enthalten sind; d.h. die Eingabe einer Information durch den Anästhesisten hat Vorrang vor der Bewertung des Systems, so dass eine Zurückweisung der Eingabe nicht erfolgen darf. Ein solcher Fall wäre z.B. durch die Dokumentation der Verabreichung eines neuen Medikamentes gegeben, welches noch nicht in der zulässigen Werteliste hinterlegt ist.
- Plausibilitäten müssen dynamisch zur Laufzeit eingegeben und bearbeitet werden können; d.h. Plausibilitäten definieren die Anästhesisten selbst. Die Definition einer Plausibilität geschieht am System, durch das System unterstützt und wird im System gespeichert. Die Definition der Plausibilität schließt mit ein, dass der Anästhesist auch die Reaktion festlegt, die auf ein Ergebnis einer Plausibilitätsprüfung zu erfolgen hat. Die Steuerung

der Durchführung einer Plausibilitätsprüfung liegt ebenfalls beim Anästhesisten. Plausibilitätskontrollen können wahlweise automatisch oder nach Anforderung durchgeführt werden. Wahlweise kann der Anästhesist einen kontrollierbaren Automatismus bzw. eine automatische Triggerung für die Plausibilitätsprüfung zuschalten. Der Automatismus muss wiederum de- und aktivierbar sein, ebenso auch die Reaktion auf das Ergebnis einer Plausibilitätsprüfung.

- Interoperative Daten sollen unter Kontrolle des Anästhesisten automatisch übernommen werden, sofern die Patientenmonitore über entsprechende Schnittstellen ans Dokumentationssystem angeschlossen sind. Andernfalls müsste der Anästhesist die Daten von den Patientenmonitoren ablesen, gegebenenfalls schriftlich fixieren und manuell ins Dokumentationssystem übertragen. Ein Medienbruch ist nach Möglichkeit zu vermeiden (s.u. Technische Anforderungen).
- Der Aufbau einer Kommunikation zu einem übergeordneten Krankenhausinformationssystem (KIS) muss möglich sein, um die Übernahme von Befunden, Leistungen, Diagnosen und Prozeduren aus dem KIS und das Senden von Leistungen, Diagnosen und Prozeduren in das KIS zu unterstützen.
- Neue medizinische Erkenntnisse erfordern häufig eine Anpassung der Dokumentation, entweder durch eine Änderung am zugrundeliegenden Datensatz oder am Erfassungsablauf. Eine zügige und zeitnahe Umsetzung der Änderungen muss vom System unterstützt werden. Die Änderungen sollen maximal innerhalb eines Arbeitstages bewältigt werden können, wobei die Software dazu nicht erweitert werden muss, sondern sich automatisch getätigten Änderungen anpasst.
- Alle Daten müssen revisionssicher gespeichert werden. Dies schließt mit ein, dass aufgrund der Änderungen der Parametrierung alte Daten(-Felder) unabhängig von der Parametrierung weiter einsehbar sein müssen. Außerdem ist eine Migrationssicherheit zu gewährleisten, also der Transfer von Daten aus einer (Software-)Umgebung in eine andere.
- Nach Abschluss einer Anästhesie ist im Idealfall deren Dokumentation ebenfalls abgeschlossen. Eine mögliche nachträgliche Änderung von Informationen wird nicht in Anästhesie-Dokumentationssystemen protokolliert. Mittels einer Änderungskennzeichnung sind Änderungen im System zu hinterlegen, um nachträgliche Anpassungen transparent und nachvollziehbar zu halten.

Technische Anforderungen

- Die Erfassung des Dokuments muss den anästhesiologischen Arbeitsablauf vollständig abdecken (siehe 3.1.2), d.h. die zeitlich und räumlich getrennte Teildatenerfassung ist zu unterstützen. Zur Vermeidung eines Medienbruches durch eine zwischenzeitlich papiergestützte Erfassung muss eine mobile Rechnerunterstützung gewährt werden können. So wird z.B. die Möglichkeit einer Erfassung der Prämedikationsdaten auf Personal Digital Assistants (PDA) oder Tablet-PCs gefordert, die zusätzlich per WLAN-Technik das Übermitteln von bereits im System hinterlegten Patientendaten auf den mobilen Arztrechner erlauben, während der Anästhesist beim Patienten auf Visite ist.

Bedienungsorientierte Anforderungen

- Ein Dokumentationssystem soll sich an die Arbeitsabläufe der Anästhesisten anpassen. Momentan passen die Anästhesisten eher ihre Arbeitsabläufe an die aktuellen Dokumentationssysteme an.
- Beim handschriftlichen Ausfüllen eines Anästhesieprotokolls findet eine Bewusstmachung des anästhesiologischen Verlaufs statt. Es besteht die Gefahr, dass durch die automatische Registrierung die mentale Verarbeitung der Daten durch den Anästhesisten derart eingeschränkt wird, dass Qualitätseinbußen in der Betreuung zu befürchten sind. Um dies zu verhindern, müssen "intelligente" Alarmsysteme entwickelt und verbesserte grafische Datenpräsentationen in den Bedienoberflächen genutzt werden, um die volle Aufmerksamkeit des Anästhesisten auf die Anästhesieprotokollierung zu lenken.
- Die Bedienoberflächen sollen sich personalisieren lassen und Zugriff auf verschiedene Sichten bieten. Eine patientenbezogene Sicht enthält alle Informationen zu einem Patienten, wobei eine Filterung auf spezifisch gruppierte Informationen möglich ist, z.B. durch eine Gegenüberstellung der Vital- und Beatmungsparameter mit verabreichten Medikamenten. Eine Bereichssicht gibt beispielsweise einen Überblick über die Belegung des OP- oder des Stationsbereichs, um dadurch Planungen zu unterstützen.
- Die Dokumentation soll zur Laufzeit eine patientenindividuelle Ausrichtung bekommen; d.h. transitiv abhängige Informationen, die bedeutungslos geworden sind, sind automatisch auszublenden.
- Nach Abschluss der Dokumentation muss ein vollständiger Datenexport in einen Bericht möglich sein, der gegebenenfalls ausgedruckt werden kann. Alle Berichte liegen im Zugriff.
- Der Zugriff auf alle Dokumente soll durch eine Suche unterstützt werden. In dieser Suche sind alle Angaben, über die ein Dokument verfügt, als Suchkriterium heranziehbar.
- Betriebswirtschaftliche und wissenschaftliche Auswertungen müssen über frei wählbare statistische Methoden gewährleistet werden.

Betriebswirtschaftliche Anforderungen

- Die Anästhesie-Dokumentationssysteme müssen günstig sein.
- Die Anforderungen an die Hardwareausstattung müssen minimal sein, um vorhandene Hardware nutzen zu können.

Optionale Anforderungen

- Offene Netzchnittstellen: Die Netzwerktopologie des jeweiligen Krankenhauses sollte das technische Design von Anästhesie-Dokumentationssystemen unterstützen. Unter Einhaltung entsprechender Sicherheitsmaßnahmen ist eine Verbindung zum Internet sinnvoll. Die Verbindung soll einen zügigen Datenaustausch zwischen den verschiedenen Fachabteilungsservern sowie entfernten Servern, die patientenspezifische Daten bereithalten, unterstützen.
- Prozessdokumentation: Neben der Möglichkeit, Prozesse anzustoßen und zu steuern, muss für diese Möglichkeit auch deren Dokumentation zwecks Nachvollziehbarkeit gegeben sein. Neue Arten von Informationen, die zu speichern sind, vergrößern den Datenbestand. Zur Sicherung muss ein geeignetes Archivierungskonzept gefunden werden, welches auch zur Versionierung geeignet ist, um eine dynamische Anpassung zu erlauben. Weil bislang in sich abgeschlossene Vorgänge in sequentiell hintereinander abzuarbeitende Prozesse untergliedert werden, öffnen sich Planungsmöglichkeiten, die zu einer übersichtlicheren Gestaltung realer Vorgänge beitragen können. Die Möglichkeit der dezentralen Erfassung von Daten sollte existieren. In der Praxis werden die realen Daten zusammengetragen und komplett an einer zentralen Eingabestelle des Systems einheitlich eingegeben.
- Flexibilität: Zum optimalen Gelingen der Operation aus anästhesiespezifischer Sicht trägt entscheidend die Prämedikation bei. Für die Prämedikation gilt nach [KS01], dass dabei Flexibilität besser ist als ein starres Schema. Dies zieht zwar ein erhöhtes Dokumentationsaufkommen nach sich, dient aber dem Wohl des Patienten. Die Einschränkung der Flexibilität der Anästhesisten durch die unbeweglichen Anästhesie-Dokumentationssysteme muss aufgehoben werden.
- Im Rahmen der elektronischen Erfassung besteht (vorerst prinzipiell mangels entsprechender Konzepte) der Wunsch einer multimedialen Erfassung. Dokumentationsbegleitend könnten audiovisuelle Sequenzen dem Anästhesieprotokoll beigelegt werden, wie z.B. die Röntgenbilder des Patienten. Diesbezüglich ist die Frage zu klären, ob ein bestehendes Anästhesie-Dokumentationssystem um die Schnittstellen erweitert wird, die es erlauben, das System mit weiteren Fachabteilungsprotokollsystemen einer Klinik zu verknüpfen oder ob das Anästhesie-Dokumentationssystem zusammen mit den anderen Fachabteilungsprotokollsystemen in ein Klinik-Informationssystem überführt wird.
Beide Varianten würden eine Betrachtung der Prozessabläufe in den jeweiligen Krankenhäusern erlauben. Eine anschließende Optimierung ermöglicht, dass automatische Abläufe realisiert werden, die weitere Systeme, wie z.B. Aufnahme-, OP-Planungs- und Abrechnungssysteme, mit den notwendigen Daten versorgen und darauf aufbauende Prozesse starten.

3.3 Eignungsprüfung von Dokumenten-Management-Systemen

Ähnliche Anforderungen, die an ein Anästhesie-Dokumentationssystem gestellt werden, existieren bereits im weiten Umfeld von Dokumenten-Management-Systemen. Dokumenten-Management-Systeme (DMS) sind selbst allerdings noch Gegenstand wissenschaftlicher Forschung. Im Folgenden werden die Ansprüche, die an ein DMS gestellt werden, erörtert und deren bisherige Erfüllung diskutiert. Es wird sich zeigen, dass DMS die Probleme, die sich im Bereich der Anästhesie-Dokumentation ergeben, nicht bewältigen können.

3.3.1 Bedeutung des Begriffs *Dokumenten-Management*

Bei dem Begriff *Dokumenten-Management*⁹ handelt es sich um einen Anglizismus. Im angloamerikanischen Sprachraum bezeichnet man mit dem Begriff „*Document Management*“ lediglich die Verwaltung von Dateien zur Überwindung der Probleme des hierarchischen Dateimanagements. In Deutschland versteht man unter *Dokumenten-Management* allgemein die Erfassung, Bearbeitung, Verwaltung und Speicherung von Dokumenten unter Sicherstellung von Genauigkeit, Performance, Sicherheit und Zuverlässigkeit - unabhängig davon, wo und in welchem Format die Dokumente gespeichert sind (vgl. [He96], [KM99], [KI01], [Li01]).

Dokument

Im angloamerikanischen Sprachraum bezeichnet der Begriff *Document* eine beliebige Datei. Im Deutschen dagegen hat der Begriff *Dokument* einen konkreten Bezug zu papiergebundenem Schriftgut. Darunter wird häufig ein Schriftstück von hoher inhaltlicher Qualität und rechtlicher Bedeutung verstanden.

In der Informationsverarbeitung wurde die Bedeutung von Dokument als *Urkunde* oder *Beweisstück* dahingehend ausgedehnt, dass in diesem Kontext auch jede Art von permanenter Aufzeichnung als Dokument bezeichnet wird. Ein Dokument ist demnach eine Kombination aus gespeicherten Informationen, Informationsträgern (z.B. Papier) und der für das jeweilige Dokument typischen Struktur. Das Dokument wird grundsätzlich als Einheit interpretiert und dargestellt, auch wenn es Informationen unterschiedlicher Typen, wie z.B. Daten, Text, Grafik oder Animation, enthält. Damit bekommt der Begriff **Dokument** eine elektronische Bedeutung, da sich der Begriff im Prinzip auf alle Arten von unstrukturierten Informationen bezieht, die als geschlossene Einheit als Datei in einem Datenverarbeitungssystem vorliegen. Um den Charakter einer solchen geschlossenen Einheit zu erhalten, sollte somit eine Datei, die ein Dokument repräsentiert, theoretisch neben den relevanten Informationen auch die Strukturinformationen enthalten.

In der Praxis enthalten die Dateien nur die relevanten Informationen - beispielsweise handelt es sich bei einer solchen Datei um ein gescanntes Papierformular, ein digital übermitteltes Fax, eine Datei aus einem Textverarbeitungssystem oder einen Datenbankauszug. Die zugehörigen Strukturinformationen solcher Dateien sind hingegen in den jeweiligen Datenverarbeitungssystemen derart eingebunden, dass alle Strukturinformationen in generalisierter Form das zugrundeliegende Datenmodell der Datenverarbeitungssysteme bilden. Die Datenverarbeitungssysteme zu den eben genannten Beispielen wären ein Image

⁹ Die einzelnen Begriffe *Dokumenten* und *Management* werden durch einen Bindestrich voneinander getrennt geschrieben. So lautet die korrekte Schreibweise des Begriffes *Dokumenten-Management*, auch wenn in der Literatur häufig die beiden Begriffe zu einem Wort zusammengefasst werden.

Viewer für das eingescannte Papierformular bzw. für das digital übermittelte Fax, ein Textverarbeitungssystem oder ein Datenbankmanagementsystem.

Alle gespeicherten digitalen Objekte repräsentieren demnach Dokumente, wobei ein Dokument (meist) für jegliche diskrete mediale Einheit mit Informationsgehalt steht. Der Eindruck der geschlossenen Einheit der Dokumente wird in Verbindung mit dem zugehörigen Datenverarbeitungssystem vermittelt.

Um eine Verarbeitung von Dokumenten jeweils als geschlossene Einheit vornehmen zu können, ist für das Datenverarbeitungssystem allgemein die Kenntnis des physischen, formalen und inhaltlichen Aufbaus wichtig. Damit ergibt sich, dass jedes Dokument grundsätzlich folgende Merkmale hat (vgl. [KM99]):

- Ein Dokument besitzt physische Eigenschaften (Papier, Datei o.ä.),
- ein Dokument besitzt formale Eigenschaften (Aufbau, Gestaltung o.ä.),
- ein Dokument hat eine Ordnung (fachliche Zugehörigkeit, Reihenfolge, Version o.ä.),
- ein Dokument besitzt einen Inhalt (inhaltlicher Bezug o.ä.),
- ein Dokument besitzt einen Charakter (Archivierungswürdigkeit, Rechtscharakter, Bearbeitungsmöglichkeiten o.ä.),
- ein Dokument verfügt über Zeitangaben (Erzeugungsdatum, Verfallsdatum, letzte Benutzung o.ä.),
- ein Dokument hat einen Erzeuger (Absender, Ersteller o.ä.),
- ein Dokument hat mindestens einen Nutzer (Empfänger, berechtigter Bearbeiter, Leser, letzter Bearbeiter o.ä.).

Die Dokumentmerkmale stellen Informationen dar, die auch für eine Verwaltung von Dokumenten benötigt werden. Die Verwaltung kontrolliert den Umgang mit den Dokumenten und versucht von vornherein, diesen mittels Zugangskriterien zu regeln. Dafür ist ein breites Spektrum von Verwaltungsabläufen notwendig (vgl. [Br97a], [Br97b]). Einerseits sind wohlstrukturierte Prozessabläufe zu installieren, die die Archivierung und den Zugriff der Dokumente regeln, andererseits Entscheidungsprozesse, die die Steuerung der Prozessabläufe beeinflussen, wie z.B. im Erkenntnisfall eines inkorrekt ausgefüllten Formulars. Solch eine Verwaltung von Dokumenten wird in dem Gebiet des klassischen Dokumenten-Management realisiert.

3.3.2 Klassisches Dokumenten-Management

Unter Dokumenten-Management im klassischen Sinne wird die dynamische Verwaltung von Informationen verstanden. Die für Anwender relevanten Informationen sind als Schriftgut auf Dokumenten niedergeschrieben. Die Verwaltung dieser Informationen verlagert sich dadurch auf eine geeignete Verwaltung der Dokumente. Solch eine Dokumentverwaltung realisieren Anwender (z.B. Verwaltungsangestellte), unterstützt durch (je nach Nutzungsgrad) Datenverarbeitungssysteme. Diese Unterstützung der Datenverarbeitungssysteme beschränkt sich meist auf das Speichern von Informationen, die dem Wiederfinden bzw. dem Verbleib der zugeordneten Dokumente dienen.

Die typischen Merkmale eines solchen Verwaltungssystems sind:

- Dynamische Ablage (kurz- und mittelfristige Aufbewahrung und Verwaltung von Dokumenten zum Zweck des schnellen und einfachen Zugriffs),
- kontrollierter Änderungsdienst mittels Versionsmanagement,

- Verwaltung von Informationen aus DV-Systemen,
- kooperative Bearbeitung von Dokumenten durch mehrere Anwender,
- Zugriff auf Dokumente mittels Datenbank über Suchmerkmale oder Strukturen, ähnlich denen eines Dateimanagers,
- Speicherung von Dokumenten und Verwaltungsinformationen auf magnetischen Speichermedien,
- Verwendung von Check-in- und Check-out-Mechanismen zum kontrollierten Dokumentenaustausch.

Dokumenten-Management-System im engeren Sinn (DMS i.e.S.)

Die geeignete Unterstützung im klassischen Dokumenten-Management leisten Archivsysteme. Diese werden im Allgemeinen als Endablage eingesetzt und dienen der revisionssicheren, unveränderbaren Speicherung von Informationen. Ein Archivsystem wird als *Dokumenten-Management-System (DMS)*¹⁰ bezeichnet, wenn es sich bei dem Archivsystem um ein Computersystem zur Erfassung, Bearbeitung, Verwaltung, Recherche und Speicherung von Dokumenten handelt.

Ein Dokumenten-Management-System im engeren Sinn speichert Dokumente unter Sicherstellung von Konsistenz (d.h. inklusive Versionskontrolle), Eindeutigkeit und Sicherheit. Wo und in welchem Format sie gespeichert sind, ist davon unabhängig. Allgemein werden zur Speicherung Dokumentengruppen in sogenannten Containern zusammengefasst. Bei einem Container handelt es sich um ein Objekt, das aus verschiedenen inhaltlich strukturierten und unstrukturierten Informationen, elementaren und zusammengesetzten Daten mit internen und externen Referenzen und Zugriffsinformationen besteht. Über ein aufgesetztes Versionsmanagement werden Veränderungen verwaltet und die Zugriffe im Rahmen elementarer Prozesse abgebildet. Ziel von DMS i.e.S. ist die Bildung selbstbeschreibender Dokumentenobjekte (Self contained Objects). Darunter werden Objekte verstanden, die ihre Verwaltungsinformationen mit sich tragen.

Real verwaltet ein DMS i.e.S. Dateien in Netzwerken. Der Zugriff und die Verwaltung sind dokumentenorientiert, d.h. sie erfolgen auf Basis von wenigen Dokumentenmerkmalen wie Schlagworte, Autoren oder Aktenzeichen. Organisatorische Gesichtspunkte wie das gemeinsame Arbeiten mit Dokumenten, Einbinden in Prozesse (im Sinne von Vorgängen), etc. spielen keine Rolle.

Dokumenten-Management-System im weiteren Sinn (DMS i.w.S.)

Dokumenten-Management-System im weiteren Sinn ist ein Sammelbegriff für die sich zunehmend mischenden Systemkategorien Dokumenten-Management-Systeme im engeren Sinn, Document-Imaging, Groupware, elektronische Archivsysteme mit digitalen optischen Speichern, elektronischen Formularen (eForms) und Workflow (vgl. [KM99]).

Document Imaging bezeichnet die computergestützte Erfassung, Speicherung, Suche, Änderung und Ausgabe von Images, die das digitale Abbild eines jeweils realen Dokumentes darstellen. In diesem Zusammenhang werden Informationen nach den unterstützten

¹⁰ Als Dokumenten-Management-Systeme (DMS) werden in der freien Wirtschaft generell Softwarepakete bezeichnet, die den Bereich des Dokumenten-Managements eventuell auch nur teilweise abdecken.

Datenformaten des DMS i.w.S. klassifiziert. Ein Image liegt in einem Bildformat vor und enthält *Non Coded Information (NCI)*, d.h. die Informationen sind nicht in einer Weise kodiert, die direkt von einem Rechner verarbeitet werden können. Informationen, die direkt von einem Rechner verarbeitet werden können, liegen als *Coded Information (CI)* vor, d.h. ihr Codierungsformat erlaubt eine direkte Interpretation und Verarbeitung. Beispielsweise werden Informationen, die als Zeichenfolgen im ASCII-Format vorliegen, generell direkt verarbeitet.

Groupware bezeichnet die Unterstützung zur gemeinsamen Verwendung und Organisation von Informationen in einer verteilten Mehrbenutzerumgebung (vgl. [SSU01]). Ein entsprechendes System bietet aufgabenorientierte Werkzeuge zur Kooperation, Kommunikation und Koordinierung von Arbeitsgruppen.

Die elektronische Archivierung ersetzt die klassische herkömmliche Archivierung von Papierdokumenten in Aktenschränken durch eine digitale Form.

Elektronische Formulare dienen rein der Eingabe, Anzeige, Ausgabe und Verwaltung variabler Informationen. Generell handelt es sich dabei um fensterorientierte Bedienoberflächen, die ein formularähnliches Erscheinungsbild besitzen.

Der Begriff *Workflow* stellt einen Anglizismus dar. Im angloamerikanischen Sprachraum wird damit der Umstand des „Routens“ von Dokumenten durch Netzwerke bezeichnet. Im Deutschen wird mit diesem Begriff eine voll integrierte Vorgangsbearbeitung assoziiert.

Die Workflow-Unterstützung, die in Dokumenten-Management-Systemen angeboten wird, versteht sich dagegen als computerunterstützte Automatisierung von Geschäftsprozessen oder Vorgängen auf minimaler Ebene. Eine regelbasierte Steuerung kontrolliert strukturierte Abläufe, inklusive einer Status- und Aktionsüberwachung sowie einer Verarbeitung von CI- und NCI-Dokumenten. Die Verarbeitung beinhaltet eine kontrollierte Weiterleitung sowohl von Dokumenten als auch von Vorgängen. Die Vorgangsteuerung wird derart realisiert, dass lediglich die Verbindung einzelner Bausteine zu vordefinierten Dialoganfragen geschaltet wird.

3.3.3 Anästhesie-Dokumentation mittels Dokumenten-Management-Systemen

Bestehende Anästhesie–Dokumentationssysteme stellen in der Regel ein datenbankgestütztes Archivsystem dar. Die einzelnen Anästhesieinformationen werden in Relationen oder gekapselten Objekten einer entsprechenden Datenbank gespeichert. Die Zugriffsberechtigungen werden durch das zugrundeliegende Datenbank–Managementsystem (DBMS) geregelt. Der Anästhesist wird beim Datenzugriff über eine GUI–orientierte Anwendung unterstützt, die die angeforderten Daten über das DBMS aus der Datenbank anfordert. Das Erscheinungsbild der Bedienoberfläche ähnelt dem eines elektronischen Formulars, welches der Eingabe, Anzeige, Ausgabe und Verwaltung variabler Informationen dient. Die Nachbearbeitung von erfassten Anästhesie–Protokollen oder die generelle Erfassung der Anästhesie wird über die Bedienoberfläche ermöglicht.

Die Funktionalität, die ein Anästhesie–Dokumentationssystem bietet, geht somit bereits konform mit dem Funktionsumfang eines DMS i.w.S. Genauer betrachtet bieten Anästhesie–Dokumentationssysteme sogar einen größeren Funktionsumfang, da die Suche spezieller Dokumente nach all den spezifischen Kriterien ermöglicht wird, die als Attribute im

Datenbankschema hinterlegt sind. Bei einem DMS i.w.S. ist die Suche auf die angegebenen Schlagworte beschränkt.

Einige Anbieter von Anästhesie–Dokumentationssystemen erweitern bereits ihre Systeme um eine Document-Imaging–Komponente, wie sie aus einem Dokumenten–Managementsystem bekannt ist. Ein Teil des Document-Imaging wird durch eine Scan–Komponente realisiert, die einen Scanner unterstützt. Diese Komponente ermöglicht das Einscannen und das Speichern von Anästhesieprotokollen als Bilddateien. Das durch seine drei Teilprotokolle für prä-, intra- und postoperativen Verlauf gegebene Anästhesieprotokoll wird dadurch zu einem digitalen Dokument, welches je nach Format in einer oder mehrerer Bilddateien abgespeichert wird. Den weiteren Zugriff und das sichere Wiederherstellen muss das Anästhesie–Dokumentationssystem garantieren. Zur Unterstützung werden die oben angesprochenen Container benötigt, in denen die Bilddateien hinterlegt werden. Container werden meist in einer Datenbank als Binary Large Objects (BLOBs) verwaltet. Den BLOBs sind bestimmte Attribute im Sinne einer Verschlagwortung beigelegt. Über die Schlagworte steht ein Dokument im Zugriff. Falls ein Anwender genauere Informationen über den Inhalt des Dokumentes benötigt, liest er diese Informationen nach, indem er sich die Bilddatei am Bildschirm anzeigen lässt.

Die Ablage eines Anästhesieprotokolls in einem Container muss weiterhin den gleichen Gegebenheiten bestehender Anästhesie–Dokumentationen genügen und z.B. Suche über alle Informationen ermöglichen; d.h. der volle Funktionsumfang der Anästhesie–Dokumentationen muss beibehalten werden. Um dies zu gewährleisten, werden bestehende Lösungen um Optical-Character-Recognition-(OCR)- und Intelligent-Character-Recognition-(ICR)-Module erweitert:

- OCR–Module bieten Methoden zur Umwandlung von Texten, die als NCI vorliegen, in CI an.
- ICR–Module bieten Methoden zur Texterkennung, die die rein optische Zeichenerkennung (OCR) durch Methoden ergänzen, wodurch sich die Erkennungsrate erhöht (z.B. durch Wahrscheinlichkeiten, Kontextanalysen, Rechtschreibung, etc.).

In der Praxis wird ICR zum Lesen von handgeschriebenen, OCR zum Lesen von maschinengeschriebenen Informationen eingesetzt. Nach dem Einscannen der Anästhesieprotokolle werden automatisch die darauf enthaltenen Informationen extrahiert und analog zum bisherigen Vorgehen in den entsprechenden Datenbanktabellen gespeichert.

Die Speicherung eines Anästhesieprotokolls in einem Container eines Dokumenten-Management-Systems erlaubt ferner eine Erweiterung des Containers um multimediale Informationen. Dies eröffnet die Möglichkeit, ergänzende Bildinformationen, wie z.B. Röntgenbilder, zum Anästhesieprotokoll aus anderen angeschlossenen Systemen einzupflegen.

Einem Anästhesisten kann nun eine Nachbearbeitung angeboten werden, bei der er neben der erforderlichen Bedienoberfläche auch das Anästhesieprotokoll als Bildinformation in Gänze angezeigt bekommen kann, um so die notwendigen Informationen selbst aus dem Bild extrahieren zu können, die bei der automatischen Extraktion fehlerhaft oder gar nicht erkannt wurden.

3.3.4 Fazit

Die Idee, eine Anästhesie-Dokumentation über Dokumenten-Managementsysteme vorzunehmen, ist charmant, führt aber nicht zum gewünschten Ergebnis. In der klassischen Verwaltung stehen die Archivierung und die Zugriffsverwaltung auf Basis der Verwaltungsinformationen, unabhängig vom Inhalt des Dokumentes, im Vordergrund. Das Dokumenten-Management-System orientiert sich generell in einer abstrakten Weise an den Dokumenten, die es verarbeiten soll. Die inhaltlichen Bezüge der Dokumente werden nur oberflächlich im Rahmen einer Gruppierung durch die Verschlagwortung betrachtet. Die inhaltlichen Detailinformationen auf den Dokumenten bleiben dem System verborgen.

Ein berechtigter Anwender, der spezifische Informationen auf Dokumenten sucht, muss alle Dokumente einzeln einsehen, um diejenigen zu finden, die seinen Suchkriterien entsprechen. Dokumenten-Management-Systeme können viele gleichartige Dokumente mit wenigen Prozessen nach gleichem Schema verarbeiten. Normale Dokumenten-Management-Systeme sind Content-Management-Systeme, die Contents verarbeiten, ohne auf den Inhalt einzugehen. Sie speichern nur den Inhalt der Dokumente in der Art und Weise, dass er hundertprozentig wiederhergestellt werden kann. Dieser Rahmen ist zu oberflächlich, wenn detaillierte Informationen gespeichert werden müssen.

Bei Anästhesieprotokollen stehen allerdings besonders deren inhaltliche Informationen bei der weiteren Verarbeitung im Vordergrund. Die individuelle Gestaltung eines Anästhesieprotokolls setzt die Zusammenführung einer ausgiebigen Eigenschaftssammlung voraus, die die inhaltlichen Informationen beschreiben. Die Eigenschaftssammlung soll dabei derart minimal sein, dass alle möglichen Eventualitäten abgedeckt werden. Andererseits soll die Eigenschaftssammlung maximal sein; d.h. die ausgiebige Eigenschaftssammlung muss auf das Wesentliche komprimiert werden können, darf aber an Exaktheit und Präzision nichts verlieren.

Dokumenten-Management-Systeme unterstützen Eigenschaftssammlungen nur im Rahmen einer Verschlagwortung. Diese Eigenschaftssammlungen sind zu klein für die Anästhesieinformationen. Eine Anpassung dahingehend, dass Dokumenten-Management-systeme größere Eigenschaftssammlungen verarbeiten können, konfrontiert diese Systeme mit ähnlichen Problemen, wie sie bei Anästhesie-Dokumentationssystemen vorliegen. Mit einer Änderung der Rahmenbedingungen einer Anästhesie würde eine Änderung des Anästhesieprotokolls nötig. Dies zieht die Anpassung des Softwareproduktes, welches die Anästhesie-Dokumentation realisiert, derart nach sich, dass das Datenbankschema per Schemaevolution und die Anwendung durch Update-Installation angepasst werden muss.

Die Aktualisierung der Anwendung beinhaltet auch eine Aktualisierung der Document-Imaging-Methode, da Informationen mittels OCR- oder ICR-Methoden von anderen Positionen im Dokument extrahiert werden müssen. Aus Kompatibilitätsgründen müssen auch ältere Versionen von Anästhesieprotokollen weiterhin verarbeitet werden können. Dies führt zu der Anforderung einer Unterstützung eines Versionsmanagements, welches die Verwaltung und konsistente Speicherung der unterschiedlichen Versionen von Dokumenten im Rahmen einer Anästhesie-Dokumentation umfasst. Diesen Anspruch werden Dokumenten-Managementssysteme nicht unterstützen.

Im Fokus von Dokumenten-Managementsystemen steht eine Vorgangsunterstützung. Konkrete Vorgänge werden durch Prozesse im System abgebildet. Die rudimentäre Umsetzung der Vorgangsteuerung in Dokumenten-Managementsystemen realisiert konkret

definierte, einfache Abläufe, die auf Basis der Container arbeiten. Hier können auch Kontroll-Automatismen, wie z.B. die Wiedervorlage oder Spezialbehandlung von fehlerhaft bemängelten Dokumenten definiert werden.

Der Container bzw. das gescannte Dokument ist die grundlegende Gemeinsamkeit der Prozesse in Dokument-Managementsystemen. Das Dokument wird von einem Prozess an den nächsten verknüpften Prozess gereicht. Die Prozessabfolge ist konkret vorgegeben und wird grundsätzlich eingehalten. Innerhalb der Prozesse wird das Dokument bzw. der Container bearbeitet, wobei verschiedene anwenderspezifische Aktivitäten zur Auswahl stehen. Mehrere parallele und/oder serielle Aktivitäten, die zur Erreichung eines gemeinsamen Zieles miteinander verbunden sind, werden in einem Vorgang zusammengefasst. Es kann sich um manuelle und/oder automatisierte Aktivitäten handeln, die den Inhalt des Dokumentes verändern. Der Inhalt des Dokumentes selbst hat keinerlei Einfluss auf den Prozessablauf. Der Prozessablauf besitzt dadurch einen statischen Charakter. Dies ist der entscheidende Unterschied zu Anästhesie-Dokumentationssystemen. Hier hat der Inhalt Einfluss auf den Prozessablauf. Der Prozessablauf in Anästhesie-Dokumentationssystemen besitzt einen dynamischen Charakter.

Zukünftige Entwicklungen im Anästhesie-Dokumentationsbereich sehen sich der Anforderung gegenüber, eine Verwaltungsfunktionalität anzubieten, wie sie Dokumenten-Management-Systeme zur Verfügung stellen. Die Integration einer Verwaltungsfunktionalität ist machbarer als der umgekehrte Fall, die Anästhesie-Dokumentation in Dokumenten-Management-Systeme einzubetten.

Von einem prozessorientierten Standpunkt aus betrachtet durchläuft die Anästhesie einen Prozess (vgl. 3.1.2), in dessen Rahmen die Anästhesie-Dokumentation anzufertigen ist. Der Prozess ist jedoch nicht derart standardisierbar, dass eine eindeutige Prozessabbildung vorgenommen werden kann, die innerhalb der rudimentären Vorgangsteuerung eines DMS bearbeitet werden kann. Jede Anästhesie ist einzigartig und kaum mit anderen Anästhesien vergleichbar.

Aufgrund der Relevanz des Anästhesieprozesses liegt es nahe, Workflow-Managementsysteme dahingehend zu prüfen, ob sie eine Lösung zur Abhilfe bieten.

3.4 Eignungsprüfung von Workflow-Management-Systemen

Prägend für Workflow-Management-Systeme ist der Begriff *Workflow*. Allgemein wird darunter die Koordination von Informationen von einem prozessorientierten Ansatz aus verstanden (vgl. [KM99]). Es ist dabei zwischen den Begriffen *Arbeitsablauf* und *Workflow* zu differenzieren. Arbeitsabläufe, deren Tätigkeiten zur Umsetzung eines Geschäftsprozesses dienen, werden als *Arbeitsvorgänge* bezeichnet; *Workflows* bezeichnen dagegen Arbeitsvorgänge, die von einem Workflow-Management-System gesteuert werden. In den Fokus der Betrachtung rückt somit eine genauere Untersuchung der Bedeutung des Begriffs *Workflow*.

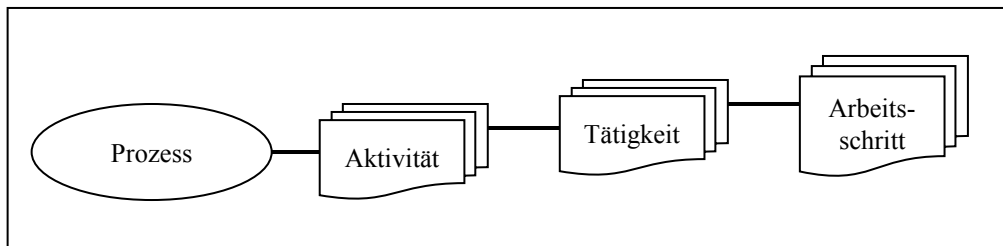


Abbildung 13: Generalisiertes Prozessverständnis

3.4.1 Prinzipielle Abbildung von Workflows in Workflow-Management-Systemen

Die ursprüngliche Bedeutung von Workflow war die Vorgangsteuerung und -kontrolle. Unter einem Vorgang wird ein generalisierter Prozess verstanden. Dieser generalisierte Prozess untergliedert sich in eine oder mehrere Aktivitäten (vgl. Abbildung 13). Eine Aktivität wiederum setzt sich aus einer oder mehreren Tätigkeiten zusammen. Eine Tätigkeit besteht aus einem oder mehreren Arbeitsschritten.

Geprägt sind derartige Prozesse von einem strengen Sequenzgedanken. Die Arbeitsschritte, die eine Tätigkeit definieren, werden nacheinander abgearbeitet. Im Anschluss an die Abarbeitung ist die Tätigkeit erfüllt und die nachgeschaltete Tätigkeit wird angestoßen. Nach Erfüllung aller Tätigkeiten ist die Aktivität, die durch die Tätigkeiten definiert ist, vollbracht und die nachfolgende Aktivität wird angegangen. Sind alle Aktivitäten, die den Prozess definieren, vollbracht, ist der Prozess bzw. der dadurch abgebildete Vorgang erledigt.

Die Bedeutung von Workflow hat sich gewandelt und meint heutzutage die Integration von Daten, Dokumenten und Applikationen zur Ausführung von Arbeitsschritten, zur computerunterstützten Automatisierung von Geschäftsprozessen oder Vorgängen. Eine Definition des Begriffes lautet nach [JBS97] wie folgt:

„Ein Workflow ist eine zum Teil automatisiert (algorithmisch) – von einem Workflow-Management-System gesteuert – ablaufende Gesamtheit von Aktivitäten, die sich auf Teile eines Geschäftsprozesses oder andere organisationelle Vorgänge beziehen. Ein Workflow besteht aus Abschnitten (Subworkflows), die weiter zerlegt werden können. Er hat einen definierten Anfang, einen organisierten Ablauf und ein definiertes Ende. Ein Workflow-Management-System steuert die Ausführung eines Workflows. Workflows sind überwiegend als ergonomische (mit Menschen als Aufgabenträgern) und nicht als technische (z.B. Einsatz von Maschinen) Prozesse zu sehen.“

Workflow-Management-Systeme dienen der Automatisierung und dem Management von Geschäftsprozessen über Abteilungs- und Funktionsgrenzen hinweg. *„Ein Workflow-Management-System ist ein (re-)aktives Basissoftwaresystem zur Steuerung des Arbeitsflusses (Workflows) zwischen beteiligten Stellen nach den Vorgaben einer Ablaufspezifikation (Workflow-Schema). Zum Betrieb eines Workflow-Management-Systems sind Workflow-Management-Anwendungen zu entwickeln. Ein Workflow-Management-System unterstützt mit seinen Komponenten sowohl die Entwicklung (Modellierungskomponente) von Workflow-Management-Anwendungen als auch die Steuerung und Ausführung (Laufzeitkomponente) von Workflows“* (vgl. [JBS97]).

Eine Workflow-Management-Anwendung ist dabei „eine implementierte und eingeführte Lösung zur Steuerung von Workflows mit einem Workflow-Management-System“.

Charakteristisch für solche Anwendungssysteme ist nach [Bö00], dass die Steuerung von Kontroll- und Datenfluss zwischen den Arbeitsvorgängen dem Workflow-Management-System obliegt, während die Ausführung der einzelnen Arbeitsschritte bei den Prozessbeteiligten verbleibt.

[HS96] versteht unter einem Workflow die Implementierung eines Geschäftsprozesses. Ein Geschäftsprozess ist gemäß [JBS97] „ein Vorgang in Wirtschaftseinheiten (Unternehmen, Verwaltungen etc.), der einen wesentlichen Beitrag zu einem nicht notwendigerweise ökonomischen Unternehmenserfolg leistet. Er läuft in der Regel funktions-, hierarchie- und standortübergreifend ab, kann die Unternehmensgrenzen überschreiten und erzeugt einen messbaren, direkten Kundennutzen“.

Um Workflow-Management-Anwendungen unabhängig von den Geschäftsprozessen wieder verwenden zu können, wird die Anwendungslogik der Geschäftsprozesse in sogenannten Workflow-Typen gekapselt, welche einen wesentlichen Bestandteil der Anwendungssysteme darstellen (vgl. [Bö00]). Änderungen von Geschäftsprozessen haben dann nur die Änderung der Workflow-Typen zur Folge.

Die Konstruktion eines Workflow-Typs beginnt mit der Schematisierung eines Geschäftsprozesses, in der Regel durch Ereignisgesteuerte Prozessketten (EPK)¹¹. Das erhaltene Schema wird in eine formale Notation überführt, aus welcher mittels funktionaler Dekomposition die sogenannten Aufgabentypstrukturen extrahiert werden. Aufgabentypstrukturen sind semantisch angereicherte Darstellungen der funktionalen Dekomposition. Visualisiert werden sie mit den Ausdrucksmitteln der Unified Modeling Language (UML) als gerichtete azyklische Graphen. Über eine Formalisierung der Inhalte der Aufgabentypstrukturen wird der Workflow-Typ des Geschäftsprozesses konfiguriert. Aus der Konfiguration wird mittels einer gewählten Workflow-Sprache das Workflow-Schema erstellt, auf welchem wiederum die Workflow-Management-Anwendung arbeitet.

Die Konstruktion von Workflow-Typen stellt ein Verfahren dar, welches generell in der Phase des Systementwurfs einer Workflow-Management-Anwendung angesiedelt ist. Die Entwicklung einer Workflow-Management-Anwendung, welche die geforderte Funktionalität bietet, verläuft gemäß einem klassischen Phasenkonzept, wie z.B. nach dem sogenannten Wasserfallmodell¹² (vgl. [Kr00], [Ba00]); d.h. im Anschluss an die Definition der Anforderungen an die Workflow-Management-Anwendung wird ein vorläufiges Konzept aufgestellt. Nach Prüfung hinsichtlich der Anforderungen wird das Konzept solange entsprechend modifiziert und verfeinert, bis ein Entwurf entsteht. Sobald dessen Design abgeschlossen ist, beginnt die Implementierung, die fast parallel um eine Phase des Testens zur Fehlerfindung und –ausräumung erweitert wird, und schließlich mit einer gefundenen Lösung endet. Die Inbetriebnahme und Wartung der Workflow-Management-Anwendung bildet die letzte Phase.

Das Verfahren zur Workflow-Typ-Konstruktion verfeinert dabei den Ablauf des Systementwurfs und orientiert sich am abzubildenden Geschäftsprozess. Dieser beinhaltet

¹¹ Abbildungen von Geschäftsprozessen werden aktuell vermehrt mittels der Business Process Modeling Notation (BPMN) vorgenommen (vgl. [FG08]).

¹² Anstelle des Wasserfallmodells werden auch Spiralmodell, V-Modell u.ä. Modelle verwendet (vgl. [Sp05]).

eine Ablauflogik, die sich über eine Menge an zusammenhängenden Aufgaben definiert. Diese einzelnen Aufgaben sind als Folge von logischen Einzelfunktionen definierbar.

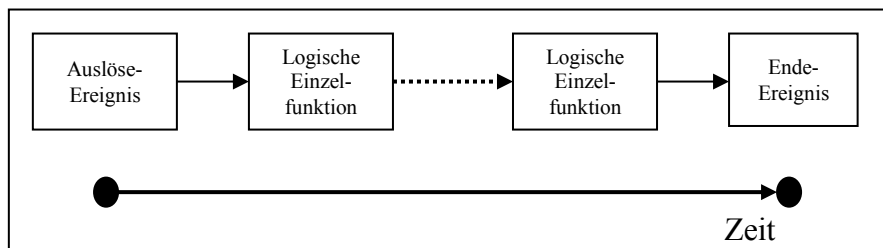


Abbildung 14: Abstrakte zeitliche Festlegung von Vorgängen

Allgemein werden Prozesse von einem Ereignis ausgelöst und mit einem Ereignis beendet. In diesem Rahmen definiert eine Ablauflogik die zur Aufgabenerfüllung erforderliche zeitliche und räumliche Festlegung von Vorgängen, wie sie in Abbildung 14 gegeben ist.

3.4.2 Bestimmung eines anästhesiologischen Workflows

Der Anästhesieprozess beschreibt den anästhesiologischen Arbeitsablauf (vgl. 3.1.2). Dieser besteht, als einzelner Vorgang betrachtet, aus drei Teilprozessen. Die Teilung des Anästhesievorgangs in die bereits reale Betrachtung der prä-, inter- und postoperativen Phase ermöglicht theoretisch die Unterstützung des Anästhesievorgangs durch ein Workflow-Management-System. Der Gedanke, die drei Teilprozesse durch eigene Workflow-Anwendungen in ein System abzubilden, ist nicht nur charmant, sondern kommt den Wünschen der Anwender entgegen, die eine besser automatisierte, EDV-basierte Phasenunterstützung wünschen.

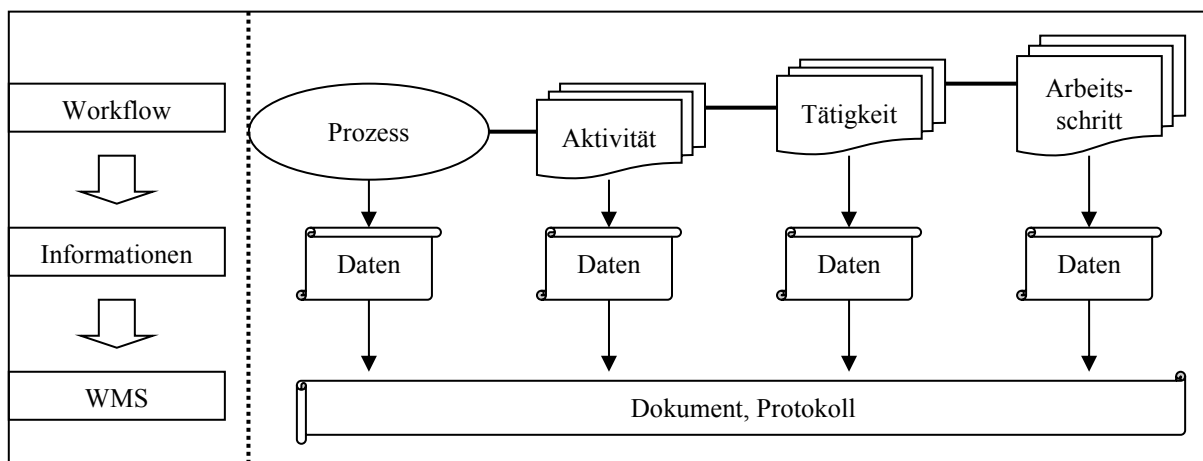


Abbildung 15: Überführung prozessrelevanter Daten in ein Protokoll

Grundsätzlich wäre dazu die Anästhesie als Workflow zu erarbeiten und in ein Workflow-Management-System abzubilden. Aus dem Workflow wären die anästhesiologisch relevanten Daten zu extrahieren und durch das Workflow-Management-System im Anästhesieprotokoll

zusammenzuführen. Abbildung 15 skizziert die Extraktion prozessrelevanter Daten und deren Überführung bzw. deren Zusammenfassung in einem Dokument, welches Protokollcharakter besitzt.

Bezogen auf den anästhesiologischen Workflow regelt und kontrolliert das Workflow-Management-System den termin- und arbeitsplatzspezifischen Ablauf der Anästhesieteilprozesse. Die dazu notwendigen Software-Ressourcen werden automatisch zugewiesen und verteilt. Die Ausführung der jeweiligen Vorgänge werden durch das System protokolliert und derart in den entsprechenden Dokumenten hinterlegt, dass die notwendigen Daten von den Stellen abgerufen werden können, welche in die Vorgangsbearbeitung eingebunden sind.

Das Workflow-Management-System steuert so den Arbeitsfluss zwischen definierten Teilnehmern gemäß definierter Prozesse, die aus verschiedenen Aktivitäten und Tätigkeiten bestehen. Es koordiniert Anwender, Anwendungen und Geräte um definierte Ziele zu festgelegten Schlussterminen zu erreichen. Alle zur Ausführung der Prozesse erforderlichen Dokumente, Daten und Informationen werden automatisch bereitgestellt. Abbildung 16 präsentiert die Abfolge der Erfassung von anästhesiologisch wichtigen Daten.

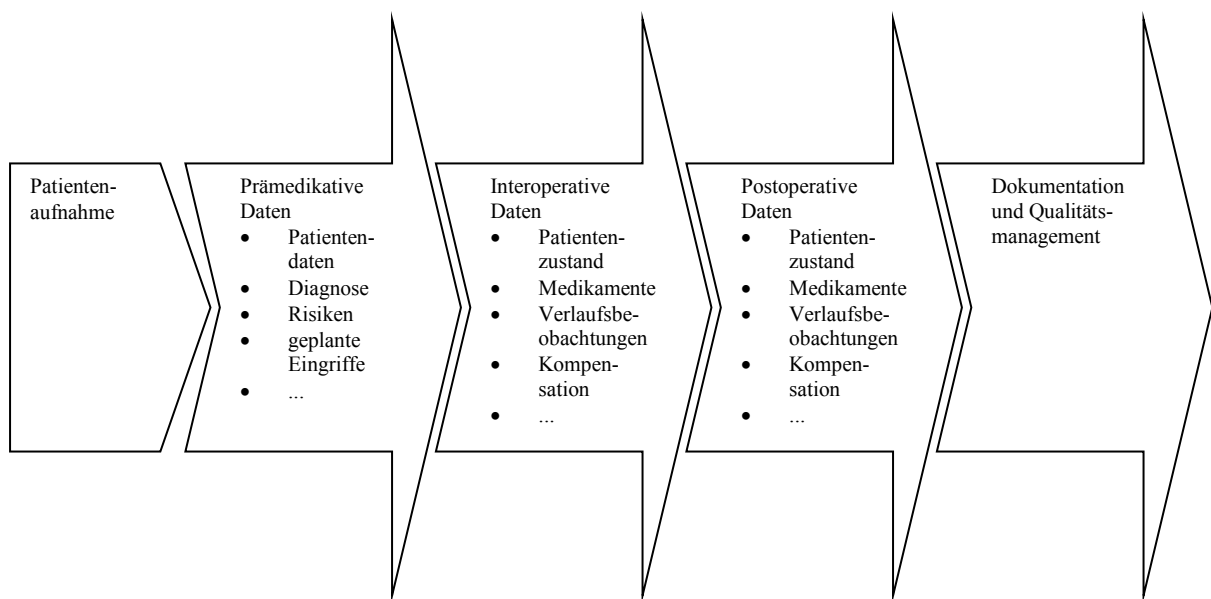


Abbildung 16: Skizzierung einer sequenzierten Datenerfassung

Die Patientenaufnahme startet den Workflow. In fester Reihenfolge werden prämedikative, interoperative und postoperative Daten erfasst. Die verschiedenen Aufgaben und Arbeitsabläufe koordiniert und kontrolliert das Workflow-Management-System. Über die abschließende Dokumentation im Rahmen des Qualitätsmanagements können die Entscheidungen und der spezifische Arbeitsablauf nachvollzogen werden. Notwendige Informationen werden für jeden Teilprozess aufgabenorientiert zur Verfügung gestellt. Hierzu wäre jeder Teilprozess der Anästhesie durch eine Workflow-Management-Anwendung abzubilden. Für diese muss zunächst ein entsprechender Workflow-Typ definiert werden.

Jede Anästhesie stellt ablauforientiert eine spezielle Ausprägung des Anästhesievorgangs dar, wobei sich der nächste Schritt eines Anästhesieprozesses aus dem vorherigen ergibt. Wie in Abbildung 11 dargestellt ist, kann der Anästhesievorgang als Kombination verschiedener Teilprozesse angegeben werden. Die skizzierte Kombination der Teilprozesse stellt allerdings einen „roten Faden“ dar, von dem durchaus abgewichen werden kann. Innerhalb der Teilprozesse werden verschiedene Informationen erfasst, die nach Kategorien geordnet oder in Gruppen zusammengefasst werden können, wie z.B. die Gruppierung von patientenspezifischen Laborwerten im prämedikativen Protokoll. Allerdings ist weder die Anzahl der Informationskategorien oder -gruppen noch die Anzahl der Informationen je Kategorie bzw. Gruppe fest vorgegeben.

Hinsichtlich der Ausprägungen der Teilprozesse einer Anästhesie sind diese Ausprägungen abhängig von den existierenden Vor- und Nachbedingungen. Beginnend mit dem Zustand eines Patienten, dessen Erfassung die notwendigen Daten für die bewusste Beeinflussung des Intraoperativen Verlaufes bereitstellt, erfährt der präoperative Verlauf durch die Patientenzustandsbeurteilung eine Ausprägung, deren Natur generischer Art ist.

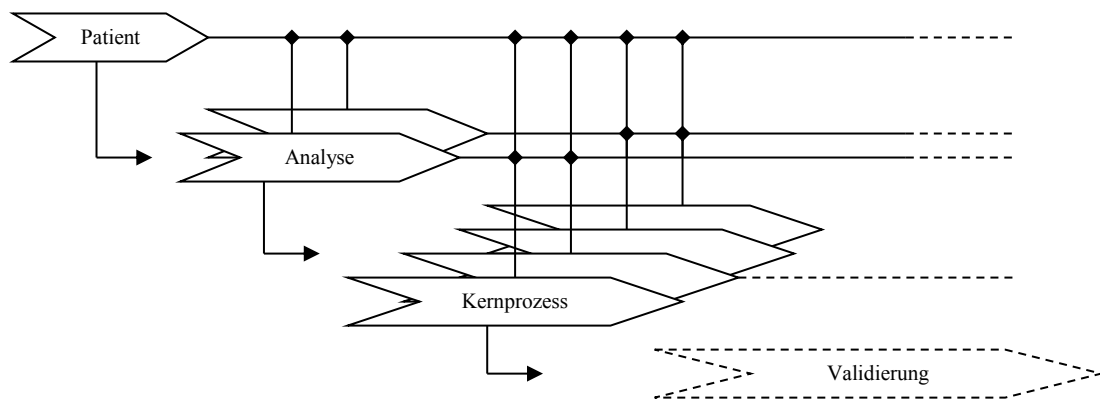


Abbildung 17: Ausprägungen anästhesiologischer Teilprozesse in Abhängigkeit zu vorgelagerten Teilprozessen

Die Informationen des präoperativen Verlaufes, die zur Narkose in Zusammenhang mit den Patienteninformationen weitergeleitet werden, prägen den intraoperativen Verlauf. Dessen Ausprägung ist generisch zu bewerten. Abschließend ist der postoperative Verlauf ebenfalls generisch zu betrachten. Der postoperative Verlauf bietet die größte Bandbreite an Ausprägungen, da sämtliche Zustandsinformationen vorheriger Teilprozesse in den postoperativen Verlauf einfließen. Abbildung 17 skizziert die Ausprägungsabhängigkeiten und kennzeichnet dabei, dass vorgesehene Teilprozesse, wie z.B. die Validierung, nicht durchlaufen werden können. Dies könnte beispielsweise dann eintreten, wenn der Patient während der Operation verstirbt. Die Ursache ist dabei nicht in der Anästhesie zu suchen, sondern in den Umständen, die den Fall des Patienten begleiten. Die Anästhesie hat diese Umstände nach Möglichkeit, eventuell in einem engen Zeitrahmen, zu erkennen und sollte darauf reagieren. Für diese Reaktion kann kein starrer Prozess definiert werden, da sich der Prozess an den Entscheidungen zu orientieren hat, die der verantwortliche Anästhesist trifft. Die gesamte Anästhesie realisiert folglich eine generische Aktivität.

3.4.3 Fazit

In Workflow-Management-Systemen werden Geschäftsprozesse abgebildet und computerunterstützt automatisiert, d.h. Workflow-Management-Systeme realisieren ein Prozessmanagement. Ausgangsbasis für die Realisierung ist die Modellierung der Geschäftsprozesse als Use-Case-Diagramm. Ein Use-Case-Diagramm soll auf einem hohen Abstraktionsniveau die Funktionalität des Geschäftsprozesses beschreiben. Ein hohes Abstraktionsniveau ist durch eine starke Generalisierung des Geschäftsprozesses möglich. Um eine Generalisierung zu erreichen, müssen Informationen ausgeklammert werden. Bei den auszuklammernden Informationen handelt es sich meist um Detailinformationen, die für die generelle Betrachtung als unnötig erachtet werden.

Dies ist im medizinischen Umfeld ein entscheidender Nachteil, da es gerade diese Detailinformationen sind, die so manchem medizinischen Geschäftsprozess sein „Gesicht“ geben. Zur Vergegenwärtigung mögen sie vielleicht zu vernachlässigen sein, zur detaillierten Betrachtung allerdings sind sie unabdingbar.

Der Geschäftsprozess der Anästhesie stellt sich als die Anästhesie, die zu protokollieren ist, selbst dar. Das dabei auftretende Hauptproblem offenbart sich bereits in der Definition der System-Anforderungen. Die Anwender, d.h. die Anästhesisten, müssten detaillierte Angaben machen, damit die Anästhesie als Workflow modelliert werden kann. Da sich die Anästhesie allerdings als Anwendungsszenario offenbart, können Anästhesisten keine eindeutige Beschreibung liefern. Einzig möglich ist die Angabe, dass sich der Anästhesieprozess aus mehreren aufeinanderfolgenden Teilprozessen aufbaut. Diese Teilprozesse setzen sich wiederum aus verschiedenen Aktivitäten zusammen. Es kann angenommen werden, dass jede Aktivität aus verschiedenen Aufgaben besteht, die je nach real vorliegender Gegebenheit entsprechend wahrgenommen werden. Die Aufgabenvielfalt präsentiert sich dabei so, dass bestehende Aufgaben durch neue Aufgaben ergänzt oder abgelöst werden können.

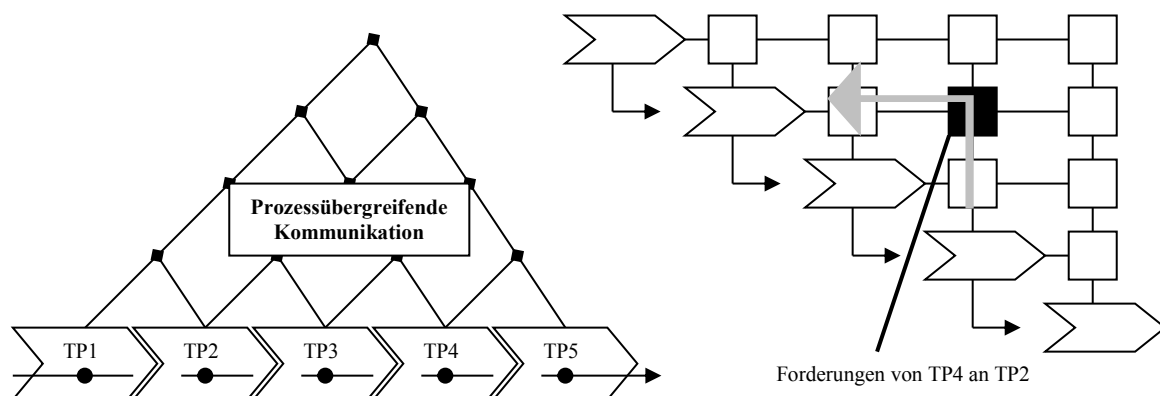


Abbildung 18: Visualisierung der Prozesssicht mittels einer Prozess-Struktur-Matrix (PSM) nach [Pf01]

Reiht man diese Aufgaben gemäß der Reihe ihrer Abarbeitung aneinander, ergibt sich eine Aufgabenkette, die nach [Pf01] als eine Prozesskette aufgefasst werden kann. Prozessketten bilden eine Grundlage für ein prozessorientiertes Qualitätsmanagement. Aus der Analyse effizienter Prozessketten gewinnt [Pf01] die Erkenntnis, dass Informationsbeziehungen zwischen allen Teilprozessen der Prozesskette bestehen.

Gemäß [Ho96] kann man aus der Prozessorientierung erkennen, dass arbeitsteilige Vorgänge nicht mehr in funktionale abgeschlossene Einheiten zerteilt werden. Vielmehr steht ein integrativer Gedanke im Vordergrund, der im Prozess Zuständigkeiten, Anwender, Anwendung und Ausführung vereint. Dadurch werden die Abläufe transparenter und die Grundlage dafür geschaffen, dass die Abläufe optimaler gestaltet und ausgeführt werden können. Ferner kann ermittelt werden, welche Daten zwischen Prozessbeteiligten ausgetauscht werden. Visualisiert werden diese in einer Prozesssicht (vgl. Abbildung 18).

Die Prozesssicht erhält man nach [Pf01], indem die beteiligten Teilprozesse (TP) aufeinander folgend aufgereiht werden. Gemäß einer Matrix werden die Informationsbeziehungen abgebildet, so dass eine Art „Dach“ über den Prozessen entsteht. Dieses „Dach“ stellt die Kommunikationsschnittstellen zwischen allen Prozessbeteiligten dar. Aus der Prozesssicht gewinnt man die Prozess-Struktur-Matrix (PSM), indem man - bildlich gesprochen - die Grafik der Prozesssicht im Uhrzeigersinn um 45 Grad dreht. Aus dem „Dach“ wird eine Matrix, in der die einzelnen Kommunikationsschnittpunkte die Forderungen darstellen, die die vorgelagerten Prozesse erfüllen müssen, damit der nachgelagerte Prozess seiner Bestimmung nachkommen kann. Die Prozess-Struktur-Matrix (PSM) wird im Qualitätsmanagement zur prozessorientierten Absicherung von Geschäftsprozessen herangezogen, die einen hohen Wiederholungscharakter besitzen.

Der Anästhesieprozess an sich besitzt ebenfalls einen hohen Wiederholungscharakter. Wie der Geschäftsprozess einer Unternehmung soll die Anästhesie auch optimal und reibungslos funktionieren. Der Unterschied liegt dabei in der Anpassbarkeit. Ein Geschäftsprozess ist auf ein bestimmtes Ziel hin ausgerichtet, z.B. auf ein konkretes Produkt einer Unternehmung.

Die Anästhesie hat zwar auch immer nur ein Ziel, der Weg zu diesem Ziel orientiert sich jedoch an den individuellen Bedürfnissen des Patienten. Werden zwei Patienten jeweils erfolgreich narkotisiert und aus ihrem narkotischen Schlaf geholt, so sind die zugehörigen Anästhesien in der Regel unterschiedlich.

Während die Teilprozesse eines Geschäftsprozesses einer Unternehmung grundsätzlich identische Ausprägungen bei erneuter Durchführung des Geschäftsprozesses besitzen, existieren die Teilprozesse einer Anästhesie in verschiedenen Ausprägungen.

Der Anästhesieprozessablauf, der von einem Workflow-Management-System verwaltet werden soll, kann als Vorgang folglich nur grob skizziert werden. Er besitzt einen dynamischen Charakter und ist abhängig von den Zustandsinformationen des Patienten, die generell im Anästhesieprotokoll zu erfassen sind. Ein allgemeingültiges, standardisiertes Ablaufschema kann nicht aus dem skizzierten Vorgang abgeleitet werden. Generische Aspekte kommen bei der Bestimmung eines Arbeitsschrittes zum Tragen, wobei sich die Aspekte durch die Gesamtheit der Ausprägungen aller vorherigen Schritte ergeben; d.h. die tatsächliche Struktur der vorherigen Schritte, verknüpft mit deren tatsächlichen Werten, bestimmt den nächsten vorzunehmenden Schritt. Somit bestimmt die reale Ausprägung einer zu protokollierenden Aktivität teilweise die Ausprägung der nachfolgenden Aktivitäten. Eventuell laufen einige Aktivitäten parallel ab oder eine Aktivität muss abgebrochen und durch eine andere ersetzt werden. Diese Form der Anästhesiegestaltung orientiert sich an der Individualität, die eine Anästhesie meist besitzt.

Die Vielfalt solcher Anästhesieausprägungen im Anästhesieumfeld hätte bei der Abbildung auf Geschäftsprozesse eine Vielfalt an Geschäftsprozessausprägungen zur Folge. Diese Vielfalt mündet in zu vielen Konzepten, die zu erarbeiten und zu implementieren wären. Dies

ist aus heutiger Sicht nicht machbar, auch weil die Logik zur Auflösung der komplexen Abhängigkeiten innerhalb eines Prozesses dessen genauerer Definition entgegensteht. Workflow-Management-Systeme erfordern aber genau definierte Prozesse, um Schritte optimal aufeinander abzustimmen.

Workflow-Management-Systeme eignen sich hervorragend für Anwendungsgebiete, in denen eine derartige Abstraktion möglich ist. Beispiele dafür wären vor allem Bereiche von Produktionsabläufen im ingenieurwissenschaftlichen Umfeld oder verwaltungstechnische Bereiche. Dort ermöglichen abstrakte Betrachtungsweisen auch eine Bewertung unter Qualitätsgesichtspunkten, die zu einer Optimierung führen. Diese Anwendungsgebiete zeichnen sich ferner durch die Anforderung einer hohen Reproduzierbarkeit aus; d.h. die Abläufe, die einmal erarbeitet sind, besitzen grundsätzlich den Charakter eines hohen Wiederholungsgrades. Generell gut strukturierte bzw. vorstrukturierte Abläufe lassen sich auf wiederholbare Vorgänge abbilden, die durch Workflow-Management-Systeme gesteuert werden können. Solche Vorgänge finden sich bei Behörden (z.B. Einwohnermeldeamt), Banken (z.B. Kreditwesen) und Versicherungen (z.B. Schadensabwicklung).

Im Ingenieur- und auch im Verwaltungsbereich ist die Wiederholbarkeit wichtig, im Bereich der Medizin wird jedoch Flexibilität verlangt. Folglich können Workflow-Management-Systeme nicht für die Anästhesie-Dokumentation herangezogen werden.

3.5 Kritische Betrachtung des Entwicklungsprozesses von Informationssystemen

Existierende Informationssysteme können zur Anästhesie-Dokumentation nicht verwendet werden. Anästhesie-Dokumentationssysteme sind somit separat zu entwickeln; d.h. zusammengefasst lautet die Aufgabe, ein Informationssystem zu entwickeln, welches im Kern die Dokumentation der Anästhesie im Sinne der Protokollierung unterstützt, ebenfalls jedoch auch Planungs- und Gestaltungsoptionen im Sinne von Workflow-Management-Systemen unterstützt und darüber hinaus mehrere unterschiedliche Datenschnittstellen (z.B. zur Online-Abfrage vernetzter Datenquellen, zum Datenimport von hardwaregestützten Erfassungssystemen wie PDAs, etc.) unterstützt.

Aus den bisherigen Betrachtungen zur Anästhesie und verschiedener Realisierungsoptionen wird deutlich, dass die Entwicklung von Anästhesie-Dokumentationssystemen aufwändig und umfangreich sein dürfte. Die Entwicklung selbst orientiert sich an einer Entwurfsmethode, die einem Vorgehensmodell genügt (vgl. [Vo00], [ZGK04]). Generell besteht eine Entwurfsmethode aus einer Abfolge der Arbeitsschritte Anforderungsanalyse, Systemanalyse, Konstruktion und Abnahme (vgl. 2.1.1).

Im ersten Arbeitsschritt werden die Anforderungen an die Software erstellt. Diese Anforderungen werden im zweiten Arbeitsschritt mit dem Ziel analysiert, dass ein Analyse-Modell, welches fachlich die Anforderungen beschreibt, entsteht. Die Erarbeitung dieses Modells dient der Vereinfachung des Entwicklungsprozesses. Im dritten Schritt wird die Architektur der Software entworfen. Dabei wird das Analyse-Modell weiterentwickelt und in eine konkrete Softwarearchitektur umgeformt. Diese Softwarearchitektur enthält Informationen über technische Details und dient als Vorlage für die Implementierung. Nach Abschluss der Implementierung folgt die Inbetriebnahme der Software, welche durch den letzten Arbeitsschritt signalisiert wird.

Innerhalb des Entwicklungsprozesses tauchen verschiedene Probleme auf, die im Folgenden aufgezeigt und erörtert werden sollen.

3.5.1 Explizite Anforderungen vs. impliziter Erwartungshaltung

Zur Erarbeitung des abstrakten Modells ist eine Anforderungsspezifikation notwendig. Die Qualität der Anforderungsspezifikation korreliert mit der Qualität des Modells. Der Informatiker versteht unter der Anforderungsspezifikation eine formale Beschreibung sowohl der vom System zu verarbeitenden Daten als auch der funktionalen Eigenschaften der Verarbeitung im System selbst.

Die Anästhesisten können solch eine Anforderungsspezifikation nicht liefern. Deren Anforderungsspezifikation ist informeller Natur. Somit müssen ausgiebige Gespräche geführt werden, um beim Informatiker das Verständnis der Anforderungen zu erarbeiten. Erschwerend kommt hinzu, dass Anästhesisten ihre Sicht auf Informationssysteme haben und sich aufgrund eines informatik-technischen Kenntnismangels bestimmter Information nicht bewusst sind. Dies führt schließlich dazu, dass Fehler im Modell in der Besprechung zur Validierung der Anforderungen nicht erkannt werden. Beispielsweise gehen Anästhesisten stillschweigend davon aus, dass ein im System erfasstes Anästhesieprotokoll auch ausgedruckt werden kann. Der Informatiker hingegen nimmt den Anwendungsfall für das Drucken nicht im Modell auf, weil diese Anforderung nicht formell notiert wurde.

Dies bedeutet, dass eine akzeptierte Anforderungsspezifikation für ein Anästhesie-Dokumentationssystem selten vollständig ist, da die Bedeutung des Modells für die Systementwicklung den Anästhesisten nicht bekannt ist. Somit kann den in der Spezifikation genannten expliziten Anforderungen eine hohe implizite Erwartungshaltung gegenüberstehen.

3.5.2 Systementwurf mit Berücksichtigung der Datenbedeutung

Mit Vorliegen der Anforderungsspezifikation beginnt die Systemanalyse. Die Spezifikation wird zunächst einer Datenanalyse unterzogen, die verschiedene Probleme im Datenbereich zu bewältigen hat. In der Regel existiert keine hundertprozentig klare und abgestimmte Begriffswelt. Vielmehr gibt es für bestimmte Informationsdaten mehrere Begriffe, die synonym oder homonym verwendet werden. Dadurch besteht die Gefahr einer hohen Redundanz der Datenbestände, gefolgt von der Gefahr von Dateninkonsistenzen. Es gibt auch keinen detaillierten Überblick über die verfügbaren Datenbestände. Existierende und verwendete Programme, die zusätzlich zu migrieren wären, sind eng mit den Daten verknüpft. Es liegt eine logische Datenabhängigkeit vor, d.h. Änderungen der Programme führen zu Änderungen an den Daten und umgekehrt.

Das Anästhesie-Dokumentationssystem soll die logische Datenabhängigkeit auflösen und letztlich nur Daten und Zusammenhänge zwischen Daten speichern. Unabhängig von einer Erweiterung des Systems soll das System die Daten und Zusammenhänge wieder zur Verfügung stellen. Hinter dieser Forderung steht die Idee, dass die eigentliche Information dem Erkenntnisgewinn beim Nutzer entspricht. Der Informationsgehalt wird vom Nutzer subjektiv bestimmt und hängt maßgeblich von dessen Vorwissen ab. Auch der Informationswert ist subjektiv und von den Interessen bzw. Bedürfnissen des Nutzers abhängig. Aufgrund dieser Subjektivität sind folgende informelle Eigenschaften von Informationssystemen zu berücksichtigen:

- **Datenangebot:** Ein Informationssystem sollte Unterstützungsvielfalt für Daten, Aussagen und Zusammenhängen anbieten. Aus dem zur Verfügung stehenden Datenangebot bedient sich der Anwender gemäß seiner Interessen. Ein aktives Zuschicken von bestimmten Daten an einen Nutzer sollte nur in wichtigen und/oder offensichtlichen Fällen geschehen. Dies bedeutet, selbstdefinierbare „Alert“-Funktionen wären zusätzlich anzubieten.
- **Selbstbestimmung:** Die Auswahl und Darstellung von Daten muss dem Anwender selbst überlassen werden. Eine größtmögliche Unterstützung ist zu gewährleisten, um den Zugriff einfacher zu gestalten.
- **Flexibilität:** Der Anwender befindet sich in einer Situation, in der sich sein Vorwissen und seine Interessen sowie seine organisatorischen Aufgaben permanent ändern können. Das Informationssystem muss in der Lage sein, die Anpassung daran so leicht, einfach und schnell wie möglich zu erlauben.

Diese Eigenschaften werden selten zur Verfügung gestellt.

3.5.3 Unzulänglichkeiten bei der Implementierung des Modells

Moderne Softwareapplikationen realisieren Anwendungssysteme, die objektorientiert im Sinne des objektorientierten Paradigmas entwickelt wurden (vgl. 2.4). Sie arbeiten in der Regel auf relationalen Datenbanksystemen.

Objektorientierter Entwurf

Das objektorientierte Paradigma hat auch Einfluss auf die Entwurfsmethodik von Software genommen. Erfolgt die Erstellung von Software nach objektorientierten Gesichtspunkten, spricht man vom objektorientierten Entwurf. Hierbei werden die Anforderungen nach objektorientierten Gesichtspunkten analysiert, um zu einem objektorientierten Analyse-Modell zu kommen. Dieses beschreibt die Anforderungen fachlich mit objektorientierten Konzepten (vgl. [CY94a]). Der Architekturentwurf der Software wird als objektorientiertes Design bezeichnet (vgl. [CY94b]). Üblicherweise wird im objektorientierten Design das Ziel verfolgt, welches die Vorlage der Softwarearchitektur aller Details bereits in der Syntax der Programmiersprache beschreibt, die in der Implementierungsphase eingesetzt wird. Die Verwendung einer objektorientierten Programmiersprache vereinfacht die Implementierung, wenn Analyse und Design des Entwurfs nach denselben objektorientierten Gesichtspunkten geschehen, die in der Programmiersprache ebenfalls vorhanden sind.

Objektorientierte Programmierung

„Eine **Programmiersprache** ist“ nach [Lo94] als „ein notationelles System zur Beschreibung von Berechnungen in durch Maschinen und Menschen lesbarer Form“ definiert. Für die Berechnungen, die jegliche Verarbeitung von Informationen durch Computer beschreiben, gilt, dass sie einfach strukturiert sein müssen, um durch Maschinen lesbar zu sein. Diese Strukturierung wiederum basiert auf einer bestimmten Sichtweise auf Daten und Operationen auf diesen Daten. Je nach Sichtweise ist eine Programmiersprache eine Ausprägung eines

Programmierparadigmas. Neben den prozeduralen, funktionalen und logischen Programmierparadigmen hat sich das objektorientierte Programmierparadigma etabliert.

Im objektorientierten Paradigma werden alle Informationen, die zur Lösung eines Problems notwendig sind, als Objekte aufgefasst. Die Objekte kommunizieren durch Nachrichten und Methoden miteinander. Objektorientierte Programmiersprachen, die dem objektorientierten Programmierparadigma genügen, beinhalten ein Klassenkonzept, welches den Aufbau einer Vererbungshierarchie unterstützt. Ferner bieten sie den Polymorphismus in der Form an, wie es im objektorientierten Paradigma beschrieben ist.

Der Vorläufer der objektorientierten Programmierung ist die strukturierte Programmierung. Diese zählt zur Klasse des prozeduralen Programmierparadigmas, wobei als Besonderheit der strukturierten Programmierung auf bedingte Sprünge verzichtet wird (vgl. [HV04]). Innerhalb des prozeduralen Programmierparadigmas werden Programme in kleinere Teilprogramme, den Prozeduren, zerlegt. Daten liegen in einem homogenen Speicherbereich und die Prozeduren arbeiten auf den Daten, die ihnen übergeben wurden. Daraus resultiert eine Trennung von Daten und Routinen, die das prozedurale Programmierparadigma kennzeichnet. In der strukturierten Programmierung werden die Daten nicht mehr als homogener Speicherbereich betrachtet. Durch die Definition von Typen ist die Verwendung von Variablen möglich, die dynamisch alloziert werden können und Dateninhalte aufnehmen. Bei den Inhalten handelt es sich um definierte Strukturen, wie z.B. Typen, Zeiger, Records, Arrays, Listen, Bäume oder Mengen (vgl. [LR09]).

Problematisch ist, dass Routinen direkten Zugriff auf Datenstrukturen nehmen können, womit die Sicherstellung der Konsistenz der Daten schwer zugesichert werden kann. In der objektorientierten Programmierung sind solche Zugriffe unterbunden. Dort gehören Daten explizit einem Objekt und können nur über klar definierte Schnittstellen gelesen und geändert werden; d.h. in der objektorientierten Programmierung besitzt das Objekt das alleinige Recht, seine Daten zu ändern. Das Prinzip der Datenkapselung erlaubt eine einfachere Sicherstellung der Konsistenz von Daten.

Implementierung von Datenmodellen und Klassen

Ein erarbeitetes, objektorientiertes Modell enthält spezifizierte Datenmodelle. In die Datenmodellbildung fließt die technische Restriktion mit ein. Der Einfluss der technischen Restriktion ist dabei so stark, dass viele Aspekte ausgeblendet werden, was am Beispiel der Klassenbildung im objektorientierten Datenmodell deutlich wird. Nach [CY94a] versteht man unter einer Klasse allgemein eine Gruppe von Menschen oder Dingen, die aufgrund einer bestimmten Ähnlichkeit oder aufgrund bestimmter Gemeinsamkeiten zusammengehören. In einer objektorientiert beeinflussten Systemanalyse erfolgt ausgehend vom Klassenbegriff eine Klassifikation der zu erfassenden Objekte. Zu dieser Klassifikation werden ausschließlich die gemeinsamen Eigenschaften herangezogen und betrachtet. Aus den Gemeinsamkeiten der Objekte werden die konkreten Klassen modelliert (vgl. [HK06b]).

Die Umsetzung der modellierten Klassen in ein System überführt diese Klassen jedoch in unveränderbare Schemata. Die Schemata beschreiben die Menge der ihnen zugeordneten Klassenobjekte. Die Klassenobjekte erfüllen dabei die Gesetzmäßigkeit des Schemas, d.h. die Klassenobjekte genügen dem Schema im laufenden System. Klassenobjekte, die vom Schema abweichen, sind nicht zugelassen. Damit ist die Gruppierung von Objekten zu einer Klasse, die „einiges gemeinsam haben“, durch eine Zuordnung in eine Klasse, deren Objekte „alles

gemeinsam haben“ ersetzt worden. Die nicht-gemeinsamen Eigenschaften werden dabei ignoriert. Der Begriff „aufgrund“ wird dabei quasi aus dem Verständnis einer Klasse gestrichen. Die Verständniserklärung wird zu einer Art Definition, die zahlreiche Aspekte der Realität ignoriert. Um solch eine Ignoranz abzumildern, wäre eine Anpassung der Schemata mittels einer Schemaevolution möglich; allerdings reagieren darauf viele Systeme empfindlich (vgl. [SH96]).

Bei zahlreich zu modellierenden Klassen enthält das objektorientierte Modell entsprechend viele Klassenmodelle, die im Zusammenspiel komplexe Gegebenheiten widerspiegeln. Ein Nachteil ist, dass die Klassen des Modells direkt in Klassen einer objektorientierten Programmiersprache abgebildet werden. Dadurch wird die Implementierung der Klassen sehr zeitintensiv. Ferner besteht die Gefahr, dass die Entwurfsmethode des objektorientierten Entwurfs bei zu vielen Klassen den Schwerpunkt auf die zu bearbeitenden Objekte legt. Dies geht zu Lasten der Funktionen, die ein System realisieren soll (vgl. [HK06a]).

Für ein Objekt gilt, dass es einen bestimmten Zustand besitzt und mit einem definierten Verhalten auf seine Umgebung reagiert. Abgebildet und somit für Rechner zugänglich werden Objekte durch die Zuweisung eines Speicherplatzes. Der Zustand wird durch lokale Variablen dargestellt, auf die wiederum nur mittels Funktionen und Prozeduren aus einer dem Objekt zugehörigen Menge zugegriffen werden kann (vgl. [Lo94]). Bedingt durch den sorgfältigen rechnerinternen Umgang mit Objekten (damit Informationen durch falsche Speicherplatzplatzierung nicht verloren gehen) ist eine genaue Definition und Spezifizierung der Objektstrukturen erforderlich.

Objektstrukturen zwingen Objekte allerdings in ein Korsett. Dieses Korsett dient zwar hervorragend dazu, verschiedene Objektausprägungen bzw. -instanzen zu verwalten, erlaubt aber keine im Nachhinein definierbare Erweiterung des Modells durch neue Objekte, die weitere Zwecke und Ziele beinhalten. Bezogen auf die Gesamtheit aller zu modellierenden Objekte repräsentiert ein Informationssystem ebenfalls ein Korsett.

Nach [Sa93] besteht ein Informationssystem im Wesentlichen aus einer Datenbank (den gespeicherten Informationen) und Anwendungsprogrammen. Eine Erweiterung im Datenbankbereich kann einzig durch aufwändige Schemaevolutionen vorgenommen werden, während bei Anwendungsprogrammen, die häufig die Realisierung eines komponentenorientierten Ansatzes sind, eine Erweiterung ausschließlich durch neue Komponenten erlaubt sind. Jede Erweiterung kostet Zeit. So ist sowohl für eine neue Komponente als auch für ein neues Datenbankschema ein Modell, dessen Abbildung sowie Implementierung zu erarbeiten und letztlich in ein bestehendes System per Update einzuspielen.

Mangelnde Transparenz der Methoden

Die objektorientierte Programmierung an sich besitzt nach [BS02] weitere Schwächen. Der Objektorientierung mangelt es an Transparenz bei der Fernwirkung von bzw. in Methoden. Es treten dadurch Seiteneffekte auf, deren Kompensation einen Overhead beschreiben.

In objektorientierten Modellen sind Methodenabläufe bzw. deren Existenz ersichtlich, der genaue inhaltliche Ablauf, das „Wie“, ist jedoch nicht transparent dargestellt. Die Funktionalität von Methoden wird (meist) in Zustands- oder Aktivitätsdiagrammen beschrieben. Diese Diagrammtypen stellen allerdings nur eine idealisierte Form der

Funktionalität abstrakt dar. Häufig kann man die Bedingungen und Abhängigkeiten eines tatsächlich implementierten Ablaufs nur am Quellcode erkennen (vgl. [Ho08]).

Anhand der tatsächlichen objektorientierten Implementierung sind somit nur Ähnlichkeiten mit dem zugehörigen objektorientierten Modell erkennbar, d.h. die objektorientierte Programmierung entspricht nicht dem objektorientierten Modell. Dieser Kritikpunkt ist relativ schwach, da die Methoden nach dem objektorientierten Paradigma generell dem Geheimnisprinzip (Information Hiding) unterliegen. Das Geheimnisprinzip widerspricht dem Transparenzgedanken. Besonders in der Modellierung von Prozessen wird dies deutlich. Prozesse müssen einsehbar sein, um sie zu verstehen. Blendet man den Prozessablauf aus, ist eine eventuell gewünschte Optimierung nicht möglich.

Fehlende Semantik bei der Abbildung eines Entwurfsmodells in ein Implementierungsmodell

Mit dem Klassenbegriff beginnt eine fatale Fehlinterpretation der Objektorientierung. Als Modell steht die Objektorientierung für Wiederverwendbarkeit und Flexibilität. Durch objektorientierte Analyse und Design ergibt sich eine Strukturierung, die bei der Abbildung des OO-Modells in die implementierte Softwarearchitektur auf vereinfachende Art und Weise Unterstützung liefern soll. Allerdings fehlen die semantischen Vorgaben, wie einzelne Modellaspekte implementiert werden sollten.

Die Klassifikation per Klasse führt beispielsweise zu starren (Daten-)Strukturen, die dynamisch nicht ausgelöst werden können. Bei der Überführung eines OO-Modells in eine Implementierung mittels einer Programmiersprache präzisiert man die Eigenschaften und ordnet sie einer Klasse starr zu. Es können zwar viele unterschiedliche Objekte der Klasse als Instanz angelegt werden, aber Unterschiede sind ausschließlich auf Unterschiede in den Ausprägungen der Eigenschaften, d.h. auf Wertunterschiede, festgelegt.

Das Problem ist, dass die Semantik der Objektorientierung bei der Abbildung eines objektorientierten Modells auf die dynamische Datenstrukturen einer Programmiersprache verloren geht. Einer Programmiersprache wird eine objektorientierte Dynamik zugesprochen, wenn sie Vererbungsmechanismen unterstützt und dadurch Wiederverwendbarkeit ermöglicht. Allerdings tritt diese Wiederverwendbarkeit nur während der Entwicklungs- bzw. Implementierungsphase hilfreich auf. Zur Laufzeit bilden die Klassen das gleiche starre Schema, wie es von den Datenstrukturen her bekannt ist. Instanzen einer Klasse sind jeweils in einem Speicherbereich abgelegt und die Werte der Eigenschaften der Instanzen jeweils mit einem Offset zur Startadresse des Speicherbereichs abgelegt. Zwei Instanzen unterscheiden sich nur in ihren Werten. Eine Unterscheidung auf Basis von Eigenschaften ist nicht möglich. Eine Hinzunahme zu oder die Entfernung einer Eigenschaft von einem Objekt ist nicht möglich, obwohl es vom Modell der Objektorientierung her nicht verboten ist. Die Hinzunahme von Eigenschaften kann allerdings über den Vererbungsmechanismus erreicht werden, da die damit mögliche Spezialisierung von Objekten erlaubt wird, was einer Eigenschaftenerweiterung entspricht.

Eine Eigenschaftsverringering im Implementierungsumfeld einer Programmiersprache ist dynamisch nicht möglich. Indirekt ist eine statische Eigenschaftsverringering möglich. Setzt man die Existenz eines komponentenbasierten Frameworks voraus und spezifiziert im Vorfeld sein Anwendungsgebiet, so kann durch die Auswahl geeigneter Komponenten eine Einschränkung auf die zur Verfügung stehenden Eigenschaften genommen werden. Ob solch

eine Anpassung als dynamische Verringerung bezeichnet werden kann, ist individuell jedem selbst überlassen. Nach Verständnis des Autors ist es das nicht, da die Spezifizierung im Vorfeld eine starre Festlegung auf fixierte Tatsachen ist und dem Akt einer Standardisierung der Softwarearchitektur für das Anwendungsfeld gleichkommt, die zu diesem Zeitpunkt eher nicht gewünscht ist.

Ein weiteres Problemfeld der Objektorientierung ist die mangelnde Spezifizierung von Methoden. Methoden werden zwar herausgearbeitet und ihr Zweck derart beschrieben, dass das Verhalten des Objektes verständlich ist, aber eine explizite Aufschlüsselung der Verhaltenslogik wird nicht angegeben. Aufgrund der fehlenden Kenntnis ist eine konkrete Aussage unbestimmbar. Im Modell wird nur das „Was“, nicht jedoch das „Wie“ erarbeitet. Die Kodierung der Verhaltenslogik liegt somit im Ermessensspielraum des Programmierers. Dabei ist eine genaue inhaltliche Methodenspezifizierung ebenso wichtig wie die genaue Eigenschaftenspezifizierung. In den Methoden wird eine eigene Berechnungslogik in Form von Algorithmen bei der Implementierung realisiert. Die Algorithmen bestimmen letztlich das spezifische Verhalten, das wiederum einer gewissen Eigendynamik nicht entbehren kann. Hier steckt eine erhebliche Diskrepanz zwischen Modell und Implementierung, zumal die namentliche Methodenbenennung individuell interpretierbar ist.

Die Vererbung der Objektorientierung umfasst auch das Vererben von Algorithmen in Form von Methoden. Methoden werden entweder in Gänze vererbt oder nur die Benennung in Form der Schnittstelle, wobei der Methodenrumpf neu implementiert, d.h. überladen wird. Das Verhalten ist allein durch die Berechnungslogik der Methode gegeben. Mangels der genauen Festlegung im Modell ist diese unbestimmbar. Es bleibt nur, das Verhalten direkt im Quellcode nachzuvollziehen, wenn dieser denn zur Verfügung steht.

Ein weiteres Manko der Objektorientierung ist die Kommunikation von Objekten mittels Nachrichten. Ein ausgewogenes Nachrichtenkonzept enthält das Modell nicht. In der Implementierung ist es üblich, dass die Nachrichtenkommunikation dahingehend aufgelöst wird, dass die Methoden der Objekte als wohldefinierte Schnittstellen zur Verfügung stehen. Ein Objekt kann in einer seiner Methoden durch einen Methodenaufruf, ähnlich der prozeduralen Programmierung, die Methode des Objektes einbinden. Um das Einbinden zu ermöglichen, ist die genaue hundertprozentige Kenntnis der Schnittstelle von Nöten. Dies erklärt, warum eine Schnittstelle wohldefiniert sein muss. Einmal wohldefiniert, ist die Schnittstelle standardisiert und darf nicht mehr verändert werden. Das bedeutet wiederum, dass ein Objekt bzw. eine Klasse starr fixiert wird und seinen dynamischen Charakter verliert.

Da es den Methoden an einer eindeutigen Beschreibung mangelt, gehen Verknüpfungen, wie sie durch eingebettete Methodenaufrufe gegeben sind, nach außen hin verloren. Dies wird auch noch durch das Prinzip der Kapselung begünstigt.

Komplexe Änderungen führen zu neuen Applikationen

Der Objektorientierung mangelt es an einer klaren Semantik, wie verschiedene Beschreibungstechniken zu implementieren wären. Da derartige Konzepte als Basis einer Softwarearchitektur fehlen, kann gefolgert werden, dass komplexe Veränderungen im Anwendungsbereich zu komplexen Veränderungen in der Implementierung führen. Diese komplexen Veränderungen setzen sich weiter fort und führen schließlich zu der Erstellung einer neuen Applikation.

Ergänzend sei bemerkt, dass auch einfach erscheinende Änderungen im objektorientierten Modell einen erheblichen Aufwand durch Änderungen in der Programmierung nach sich ziehen können. Das Problem liegt dann in den Methoden des objektorientierten Paradigmas. Diese werden in der Regel durch einen entsprechend zu implementierenden Algorithmus angegeben. Das Finden des korrekten und effizienten Algorithmus zu einer gegebenen Problemstellung ist eine der Hauptaufgaben der Informatik wie auch der Mathematik (vgl. [BI68], [OW02]). Ändert sich die Problemstellung, ist in der Regel der bisherige Algorithmus an die neue Problemstellung anzupassen. Es kann keine Angabe zu den sich ergebenden Kosten gemacht werden, da jeder Algorithmus, und sei es auch nur die Änderung eines bestehenden Algorithmus, neu ist. Kosten und Implementierungsaufwand sind für neue Algorithmen schwer zu bestimmen.

Die Lösungsstrategien zur Anpassung an die Veränderungen sind in der Regel ebenfalls recht komplex. In Anbetracht solcher Komplexitäten ist es schwierig, relativ zeitnah Anpassungen von Applikationen vorzunehmen. Aus diesem Grund werden neue Entwicklungen benötigt, die Ansprüche zur dynamischen Anpassung erfüllen.

Forderung nach dynamischer Anpassung

Die Anwender fordern softwarebasierte Lösungen, die sich zeitnah an dynamische Vorgänge anpassen bzw. anpassen lassen. Objektorientierte Systeme können den Anspruch dynamischer Anpassung allerdings nicht erfüllen. Übersehene Aspekte in der Konzeptionsphase bringen einen gerade noch überschaubaren Aufwand an Anpassungen mit sich. Veränderungen, wie sie in der Anästhesie dagegen auftreten, bei der nicht vorhersehbare Daten zu erfassen sind, bringen tendenziell komplett neue Aspekte in den Tätigkeitsbereich der Anwender. Diese sind von relativ komplexer Natur, so dass das Systemmodell nicht mehr angepasst werden kann, sondern schlichtweg durch ein neues ersetzt werden muss. Stets ist der individuelle Kontext der Anästhesie zu berücksichtigen, so dass es ein Idealkonzept zur Gestaltung von Modellen nicht geben kann. Vielmehr ergibt sich die Erfordernis, ständig die realen Gegebenheiten zu überprüfen und die Modelle des Systems derart weiterzuentwickeln, dass reale Veränderungen interne und externe Anpassungen zur Folge haben. Nur auf relativ hoher Abstraktionsstufe lassen sich Gestaltungsempfehlungen angeben.

3.6 Schlussfolgerung

Die Objektorientierung bietet Flexibilität und theoretisch auch Wiederverwendung. Das ist durch praktische Beispiele auch belegt. Allerdings begründet sich die Flexibilität auf die mangelnde Präzisierung bzw. Ausarbeitung von Methoden. Im Modell werden Methoden nur namentlich aufgeführt und mittels verschiedener Diagrammtypen beschrieben. Es liegt am Programmierer ihnen Leben einzuhauchen, indem er den zugehörigen Algorithmus erarbeitet und in einer Programmiersprache implementiert. Sein Erfolg begründet den Erfolg des Systems, welches es gemäß dem Modell zu entwickeln gilt. Dabei stellt sich die Frage, warum der Erfolg eines Systems von dessen Implementierung abhängt? Warum gibt es keine Systeme, die den Implementierungsaufwand minimieren oder gar aufheben, indem sie Alternativen anbieten?

Die Anästhesisten wünschen sich ein Informationssystem, in dem sie vieles oder sogar alles selber einstellen können. Die Bedienung sollte so intuitiv wie möglich gehalten werden. Falls eine intuitive Bedienung nicht möglich wäre, lassen sich Ärzte auch schulen, vorausgesetzt, nach der Schulung ist der jeweilige Arzt dazu befähigt, alle gewünschten Einstellungen am System selbst vornehmen zu können. Dies bedeutet somit, dass ein Krankenhaus-Angestellter die Verantwortung für die Systempflege übernimmt und jegliche Anpassung im Haus eigenständig vornimmt. Die Kliniken versprechen sich dadurch die Unabhängigkeit von Systemhäusern und deren Softwarelösungen. Dies beinhaltet auch, dass keine teuren Serviceleistungen mehr in Anspruch genommen werden müssen und keine Bevormundung durch externe Servicedienstmitarbeiter stattfindet, da die Ärzte eigenständig ihre Freiheitsgrade und sich selbst auferlegte Restriktionen verwalten können. Eventuell werden auch für die Systeme Stellen geschaffen, auf die entsprechend geschultes Personal eingestellt wird.

Unter einem Anästhesie-Dokumentationssystem verstehen Anästhesisten im Prinzip ein datenbankgestütztes Textverarbeitungssystem, welches die Möglichkeit bietet, auf den protokollierten Daten statistische Auswertungen fahren zu können. Ein Textverarbeitungssystem offeriert in der Regel die Möglichkeit, Vorlagen zentral anzulegen und den Anwendern zur Verfügung zu stellen. Jeder Anwender hat die Möglichkeit, die Vorlage individuell im Rahmen der Bearbeitung zu verändern. Im Rahmen solch einer Vorlage wünschen die Ärzte ein Anästhesieprotokoll selbstständig zu gestalten. Aus einem gestalteten Anästhesieprotokoll sollen anschließend automatisch die notwendigen Strukturen und Verfahren zur Erfassung, Verwaltung, Archivierung und Auswertung tatsächlicher Anästhesieprotokolle gewonnen werden.

Im Rahmen der intuitiven Gestaltung wollen die Anästhesisten unabhängig von der technischen Repräsentation der Daten sein, d.h. die Anästhesisten wünschen keine Beschäftigung mit Datentypen oder Typsystemen. Diese Analyse soll das Informationssystem automatisch durchführen. So können Anästhesisten Daten angeben wie sie es gewohnt sind. Zahlen werden als Zahlen und Texte als Texte erfasst. Plausibilitätsregeln wollen sie so angeben können, wie sie in einer Tabellenkalkulation Formeln definieren können.

Zur Erfüllung dieser Wünsche werden im Folgenden Pliable Objects vorgestellt. Diese Objekte sollen direkt nach Anwenderangaben umgesetzt werden können. Diese Umsetzung geschieht im Rahmen eines Informationssystems, welches die Erarbeitung der Pliable Objects nicht nur unterstützt, sondern diese auch in einer Datenbasis hinterlegt. Die Datenbasis legt dabei den Funktionsumfang des Informationssystems fest. Die Erweiterung der Datenbasis durch neue Pliable Objects ermöglicht dem Informationssystem neue Zustandsinformationen und neue Zustände anzunehmen. Unter Berücksichtigung der Anwenderbedürfnisse entwickelt sich das Informationssystem zu dem System, welches sich die Anwender wünschen.

4 Das Modell Pliable Object

Anästhesisten wünschen sich ein Informationssystem, in welchem Änderungen an zu erfassenden, medizinischen Daten und an der Datenerfassung selbst schneller einzuspeisen sind, wobei nicht notwendigerweise neue Binärdateien bei einem Softwarehaus in Auftrag zu geben sind (vgl. [Ha05]). Sie erheben den Anspruch, dass Erweiterungen am System selbst und durch das System unterstützt vorgenommen werden können (vgl. 3.2.7). Die Erfüllung dieser Anforderung soll mit Hilfe von sogenannter Pliable Objects geschehen. Pliable Objects werden in dieser Arbeit ähnlich zu Business Objects der OMG definiert (vgl. 2.6). Die von der OMG vertretene Auffassung und Interpretation der Definition als Grundlage für Komponenten wird nicht vertreten. Stattdessen wird die Theorie entwickelt, dass Pliable Objects nicht auf einer Ebene oberhalb des objektorientierten Modells aufsetzen, sondern dass es sich bei den Pliable Objects um ein Modell handelt, welches mittels des objektorientierten Modells ausgedrückt wird. Ein Objekt bleibt dabei gewissermaßen ein Objekt, allerdings wird der Objektbegriff ein wenig „gedehnt und verbogen“. Das genaue Verständnis wird im Folgenden zunächst näher erörtert. Anschließend wird das neue Verständnis auf das Verhältnis übertragen, dass Objekte einer Objektstruktur genügen; ähnlich dem objektorientierten Verhältnis zwischen einer Instanz und einer Klasse. Dabei wird die Idee verfolgt, dass auch eine Objektstruktur als ein Objekt verstanden wird. Es eröffnet sich dadurch die Möglichkeit, wenn allgemein ein Objekt mittels einer definierten Methodik verändert werden kann, dass infolge auch eine Objektstruktur über die definierte Methodik verändert werden kann. Das Kapitel endet mit der Spezifikation eines Modells, welches letztendlich die Basis für ein entsprechendes Informationssystem bildet, das nicht mehr ausschließlich auf Objektausprägungen arbeitet, sondern auch auf den Objektstrukturen Änderungen zulässt. Einige Ergebnisse, die das Kapitel enthält, sind teilweise in [Pr05] und [Pr11] veröffentlicht worden.

4.1 Motivation für ein neues Objekt-Verständnis

Das objektorientierte Paradigma hat in den letzten Jahrzehnten Eingang in Modellierungskonzepte, Datenbanken und Programmiersprachen gefunden, und objektorientierte Systeme sind weit verbreitet (vgl. [Ba00], [He92], [Ko01], [Lo94], [RP02]). Damit wird das Ziel verfolgt, eine direkte Beziehung zwischen einem Objekt im Modell und dessen Entsprechungen in der Datenbank und in der Programmiersprache herzustellen. Dieses Ziel wird auch erreicht. Problematisch wird es erst, wenn ein Objekt im Modell eine Änderung erfährt.

Beispielsweise sei der Systementwurf auf Abstraktionsebene bzw. auf der Schemaebene betrachtet. Dort ist es möglich, einer existierenden Klasse *Person* dynamisch neue Attribute zuzuordnen. Geschieht dies, muss die Veränderung am Schema unmittelbar und aufwändig auf die existierenden Instanzen übertragen werden. Logisch betrachtet stellt sich die Frage, warum ein derartiger Aufwand vorgenommen werden muss. Angenommen, in der medizinischen Forschung gelänge eine bahnbrechende Erfindung, die das Gesundheitswesen revolutionieren würde. Diese Entdeckung würde es verlangen, auf der Krankenversicherungskarte eines jeden Bürgers den Quotienten aus Atem- und Herzfrequenz anzugeben. Die Informationssysteme, die die Bürgerdaten verwalten, wären durch eine Schemaevolution der zugrundeliegenden Datenbanken sowie durch Aktualisierungen der Anwendungen schnell an die geänderten Anforderungen angepasst. Die Datenbestände selbst müssten zunächst mit Nullwerten umgehen, wie z.B. *Nicht Erfasst*. Die Anpassung der Daten, d.h. die Ergänzung des neuen Datums, würde relativ lange dauern, da individuell jeder Bürger

diese zusätzliche Angabe seiner Krankenkasse liefern müsste. Dies dürfte erst dann geschehen, wenn die Daten bei der Ausstellung einer neuen Krankenversicherungskarte neu erfasst werden, da erst dabei die neue Eigenschaft automatisch erfragt werden würde. Solange keine neue Karte ausgegeben wird, besitzt die existierende Krankenversicherungskarte weiterhin ihre Gültigkeit, auch wenn ihr diese Angabe fehlt.

Aus Modellierungssicht betrachtet, existieren im obigen Beispiel zwei Objekte der gleichen Klasse *Krankenversicherungskarte*. Beide Objekte unterscheiden sich darin, dass die neue Krankenversicherungskarte ein Attribut mehr besitzt als die alte. Mit dem Modell der Pliable Objects wird das Ziel verfolgt, Instanziierungen derart zu ermöglichen, dass Instanzen einer Klasse eine unterschiedliche Anzahl an Attributen besitzen, wobei die Vereinigung aller Attribute durch die Klasse selbst gegeben ist. Die Attribute einer Klasse stehen im Zugriff der Methoden der Klasse. Die unterschiedliche Anzahl an Attributen hat auch Rückwirkungen auf die Methodendefinition. Pliable Objects sollen ebenfalls mit unterschiedlichen Methoden zu Recht kommen.

Wie bereits erwähnt, war die Definition von Business Objects der OMG Ideengeber dieser Arbeit. Nach der Interpretation der OMG schließt die Definition von Business Objects auch Business Rules derart ein, dass im objektorientierten Konzept die den Klassenbegriff definierenden Attribute und Methoden durch Business Rules nach außen transparenter werden. Im Gegensatz dazu werden im Modell der Pliable Objects Business Rules als eigenständige Objekte verstanden, die allgemein als Regeln bezeichnet werden. Diese Regeln erweitern eine aus Attributen und Methoden bestehende Klasse zu einem Pliable Object.

Es soll ein Informationssystem erarbeitet werden, welches Pliable Objects klar auf objektorientierte Konzepte abbildet und dadurch verarbeiten kann. Dieses System soll allerdings auch die Erstellung neuer Pliable Objects zur Laufzeit und deren Verknüpfung mit bereits existierenden Pliable Objects erlauben. Neue Pliable Objects könnten beispielsweise auch so angelegt werden, dass sie in einer Hierarchie mit existierenden Pliable Objects stehen. So sollten neue „Klassen“, wie z.B. *Arzt* oder *Pflegekraft*, in Beziehung zum Pliable Object *Person* gebracht werden.

Ferner werden Pliable Objects so ausgelegt, dass sie das „Leben“ eines Geschäftsobjektes nachvollziehbar gestalten. Eine Person beispielsweise erhält und verliert teilweise auch im Verlauf ihres Lebens Fähigkeiten, die ihre Eigenschaften erweitern bzw. beschränken. Das zu entwickelnde System soll ein solches freies Maß an Flexibilität zulassen, nach Möglichkeit sogar skalierbar, dass ein Objekt Eigenschaften bekommen und verlieren kann, wobei sich sein Verhalten jeweils daran anpasst.

4.1.1 Abstraktion eines Pliable Objects

Allgemein gilt, dass eine reale Entität in der realen Welt nicht den technischen Anforderungen genügt, um direkt in Rechneranwendungen bzw. Informationssystemen eingebunden werden zu können (vgl. [RP02]). Zur Abbildung der realen Entität muss ein Modell gefunden werden, welches die Wesentlichkeit der Entität abstrahiert um diese in einem Informationssystem erfassen und verarbeiten zu können (vgl. [Re91], [Vo00], [Ve91]). Dieses Modell wird mit und durch Pliable Objects entwickelt. Das Modell soll das Verständnis der Pliable Objects transparent zeigen und keinerlei Restriktion für die technische Umsetzung in einer Software beinhalten.

Mit der Existenz eines Modells wird entschieden, ob die Umsetzung direkt über den zu implementierenden Code geschieht oder ob ein System entwickelt werden kann, welches die Modellbeschreibung der Pliable Objects verarbeitet und ein Verhalten zeigt, welches über die Modellbeschreibung der Pliable Objects realisiert wird.

Zuerst soll eine detaillierte Betrachtung der Abbildung realer Entitäten vorgenommen werden, bevor deren Verarbeitung im Rechner erfolgen wird. Bei der Erarbeitung des Modells bietet die objektorientierte Modellierung eine sinnvolle Unterstützung (vgl. [Ba00]). Die Abbildung des objektorientierten Modells in eine Anwendung ist die eigentliche Problematik, die es zu lösen gilt. Aufgrund des objektorientierten Paradigmas wird die Abbildung des Modells in Anwendungen mittels objektorientierter Programmiersprachen vorgenommen. Klassen des Modells werden in Klassen der Programmiersprache abgebildet und Methoden des Modells in Programmroutinen implementiert. Die den Methoden zugrundeliegenden Algorithmen werden dabei in eine konkrete Programmlogik übersetzt. Beim übersetzten Code handelt es sich häufig um einen Binärcode, der nicht mehr verändert werden kann.

Fatal an dieser Stelle ist die Möglichkeit der Wiederverwendung bei der objektorientierten Entwicklung. In einem Modell sind objektorientierte Elemente abstrahierbar und lassen sich im Rahmen der Wiederverwendung mit anderen Elementen in neuen Modellen wieder verwenden. Für die anschließend aus dem neuen Modell resultierenden Systeme bedeutet dies, dass prinzipiell die Wiederverwendung des bereits implementierten Codes gemeint ist. Die Wiederverwendung von implementierten Codes bedeutet die Wiederverwendung von Programmlogik. Da die Programmlogik eventuell nicht fehlerfrei ist, bedeutet dies eine Vererbung der Fehler ins neue System. Ebenfalls könnte der Fall auftreten, dass die bereits entwickelte Programmlogik nicht ins neue System passt und dadurch Fehler verursacht. Zur Fehlervermeidung müssten die Algorithmen der Systeme, die der Programmlogik zugrunde liegen, miteinander verglichen werden. Diese sind häufig in der Programmlogik verschlüsselt und nicht transparent dargestellt, was die Nachvollziehbarkeit deutlich erschwert. Die Folge wäre eine zeitintensive Wartung des neuen Systems.

Für Pliable Objects wird angenommen, dass sie ein Konstrukt sind, welches keine direkte Umsetzung als Konzept in einer Programmiersprache bildet. So wird das Konzept der „Klasse“ in objektorientierten Programmiersprachen z.B. dazu verwendet, eine 1:1-Abbildung der entsprechenden Klasse eines Modells zu realisieren. Solch eine Abbildung für Pliable Objects vom Modell in eine Programmiersprache ist nicht vorgesehen.

Pliable Objects sind allein von ihrer Definition her dermaßen komplex, dass andere Entwicklungskonzepte zur realen Implementation von Pliable Objects gefunden werden müssen. Um zu solch einem Konzept zu gelangen, müssen die einzelnen Merkmale von Pliable Objects näher betrachtet und die typischen Charakteristika von Pliable Objects bestimmt werden. Im Folgenden werden Pliable Objects als ein Datenmodell erarbeitet und formalisiert betrachtet.

4.1.2 Semantische Darstellung von Pliable Objects

Ein Pliable Object besteht aus Attributen, Aktionen und Regeln. Diese Struktur orientiert sich an der Definition von Business Objects der OMG (vgl. 2.6) und kann wie folgt skizziert werden:

- Ein **Attribute** (*Eigenschaft*) charakterisiert eine Eigenschaft, die für das Pliable Object relevant ist, um seine Geschäftsaufgaben erfüllen zu können.
- Eine **Action** (*Aktion*) definiert ein bestimmtes Verhalten, welches ein Pliable Object dazu befähigt, seine Wirkung durchzuführen.
- Eine **Rule** (*Regel*) beschreibt eine Struktur, die das Verhalten, die Beziehungen und die Attribute des Pliable Objects verwaltet. In den Regeln werden Geschäftsregeln, Geschäftspolitiken und -beschränkungen zusammengefasst. Die **Rules** „regieren“ quasi das Verhalten, die Beziehungen und die Attribute eines Pliable Objects. Außerdem integrieren sie die Verbindungen des Pliable Objects, die die Interaktion im Rahmen seines Zweckes widerspiegeln. Dies schließt die Erkennung von Ereignissen in seiner Umgebung, die Änderung seiner Attribute und die Interaktion mit anderen Pliable Objects mit ein.

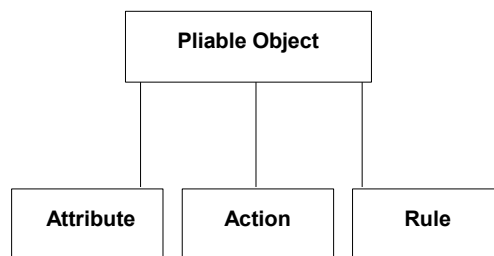


Abbildung 19: Semantische Struktur eines Pliable Objects

Das Aggregat **Pliable Object** besitzt eine Menge von Attributen, Aktionen und Regeln (vgl. Abbildung 19). Im Sinne objektorientierter Vorstellungen kapselt ein Pliable Object seine Eigenschaften und sein Verhalten. Dies impliziert, dass der Wert der Attribute lediglich durch Pliable-Object-Methoden veränderbar ist.

Zur Entwicklung von Pliable Objects muss ein Mengenbegriff konkretisiert werden, damit eine Mengenbildung realisierbar wird. Die Formulierung einer Menge ermöglicht die Betrachtung der Menge aller Attribute sowie die Zuweisung einer mengenwertigen Eigenschaft an ein Pliable Object. Die Werte der Attribute werden innerhalb einer Aktion ausgelesen und im Rahmen des Aktionsalgorithmus weiterverarbeitet. Diese Algorithmen arbeiten die Methodik ab, die durch Prozessstrategien ins Spiel kommt.

Vor der formalen Definition eines Pliable Objects werden zunächst einige grundlegende Definitionen vereinbart, die die Formalisierung ermöglichen.

4.2 Allgemeine Eingrenzung von Wertemengen

Pliable Objects dienen zur Modellierung realer Entitäten. Eine Unterscheidung von Entitäten wird auf der Basis von Werten vorgenommen. Werte werden als existent und gegeben betrachtet. Sie sind einzeln voneinander unterscheidbar. Alle Werte werden in einer universalen Wertemenge zusammengefasst.

4.2.1 Menge aller Werte

Informationen werden durch Werte repräsentiert, die wiederum digital durch Zeichen eines Zeichenvorrats dargestellt werden (vgl. [RP02]). Als Ausgangsbasis wird eine Menge definiert, die alle Werte umfasst, die Informationen repräsentieren können.

Definition 4.1 Universum

Das Universum U sei die Menge aller Werte.

Beispiel:

Das Universum U ist gegeben durch

$$U = \{ \text{„Zeichen“}, \infty, \langle \text{alle Zahlen} \rangle, \langle \text{alle Buchstaben} \rangle, \langle \text{alle Kombinationsfolgen} \rangle, \dots \}$$

4.2.2 Domäne

Domänen dienen dazu, bestimmte Werte in einzelne Bereiche zusammenzufassen. Sie realisieren verschiedene Gruppierungen von Werten. Zur besseren Unterscheidung der einzelnen Gruppierungen wird jede eindeutig bezeichnet. Domänen bestehen aus einer eindeutigen Bezeichnung und einer Wertemenge; formal werden sie wie folgt definiert:

Definition 4.2 Domain

Sei $\mathbf{T} \in P(U)$ eine *Typbezeichnungsmenge* und $\mathbf{W} \subset P(U)$ eine *Wertemengenmenge*¹³. Die Typbezeichnungsmenge stellt eine Menge von Bezeichnungen dar, die nach natürlichem Verständnis *etwas* bezeichnen können. Unter der Wertemengenmenge sei einfach eine Menge von Wertemengen verstanden. Die Relation

$$\Delta \subseteq \mathbf{T} \times \mathbf{W}$$

definiert eine bijektive Funktion, die jeder Menge $\Omega \in \mathbf{W}$ eine Bezeichnung $\tau \in \mathbf{T}$ zuordnet. Eine **Domain** (*Domäne*) ist gegeben als ein Tupel $(\tau, \Omega) = \delta \in \Delta$.

Zur Kennzeichnung einer speziellen Domäne wird die Typbezeichnung tiefgestellt, gemäß

$$(\tau, \Omega_\tau) = \delta_\tau \in \Delta.$$

¹³ Es sei darauf hingewiesen, dass es sich um eine echte Teilmenge handelt. Dies dient dazu, dass generell neue Domänen angelegt werden können. Bei Gleichheit zur Potenzmenge wäre dies nicht möglich.

Bemerkungen zu Wertemengen und ihren Typbezeichnungen

Angenommen, die Potenzmenge $P(U)$ ist gegeben durch

$$P(U) = \{ .., \{Zahlen, Zeichenketten, ..\}, \{1, 2, 3, .., \infty\}, \{a, b, abc\}, .. \}$$

Dann sind folgende Domänen möglich:

$$\begin{aligned} \mathbf{T} &= \{ \text{Zahlen, Zeichenketten, ...} \} \text{ und} \\ \mathbf{W} &= \{ \{1, 2, 3, .., \infty\}, \{a, b, abc\}, .. \} \text{ mit} \end{aligned}$$

$$\Delta = \{ (\text{Zahlen, } \{1, 2, 3, .., \infty\}), (\text{Zeichenketten, } \{a, b, abc, ..\}), .. \}$$

Für die Wertemengen von Domänen wird angenommen, dass jedes Element der Wertemengen einen konkreten Wert darstellt. Zur Notation einer Domäne wird vereinbart, dass deren Wertemenge durch die Aufzählung der enthaltenen Elemente angegeben wird. Falls die Aufzählung zu umfangreich werden sollte, kann die Elementangabe, sofern dies möglich ist, alternativ durch einen regulären Ausdruck beschrieben werden. Speziell wird darauf hingewiesen, dass es sich bei Domänen um willkürliche Zusammenstellungen von Werten handelt. Der Umstand, dass die Wertemenge einer Domäne mit Hilfe der Ausdrucksmöglichkeiten eines regulären Ausdrucks beschrieben werden kann, ist eine nützliche Eigenschaft. Es wird jedoch keine Forderung daraus abgeleitet, dass jede Domäne durch einen regulären Ausdruck beschrieben werden muss.

Bei der Notation wird die Typbezeichnung der Domäne der Wertemenge vorangestellt. Die Typbezeichnung einer Domäne soll durch einen prägenden Fachausdruck gegeben sein, der konkret als Synonym für die Wertemenge verwendet wird.

Einschränkend für die Relation Δ gelten die folgenden grundlegenden Vereinbarungen:

- Der Leerstring („“) ist nicht Element der Typbezeichnungsmenge:

$$\forall \tau \in \mathbf{T} \mid \tau \neq \text{„“}$$

Somit existiert keine Domäne, deren Wertemenge durch den Leerstring bezeichnet wird.

- Es gilt $\mathbf{T} \in \mathbf{W}$, da die Elemente von \mathbf{T} , die als Typbezeichnungen dienen, zusammen selbst eine Wertemenge bilden. Damit existiert die Domäne (τ, \mathbf{T}) , für die τ die Bezeichnung *Typbezeichnungsmenge* als Wert annimmt, d.h.

$$\tau = \text{Typbezeichnungsmenge} \in \mathbf{T}$$

Die Typbezeichnungsmenge enthält also ihre eigene Bezeichnung als Element:

$$(\text{Typbezeichnungsmenge, } \{ \text{Zahlen, Zeichenketten, ..., Typbezeichnungsmenge, ...} \}) \in \Delta$$

- Wenn sich eine alternative Typbezeichnungsmenge finden lässt, die äquivalent zu \mathbf{T} ist, dann dient diese als „Übersetzung“.

$$\text{z.B.: } \{ \text{Zeichen, Zahlen, ...} \} \sim \{ \text{STRING, INTEGER, ...} \} \sim \{ \text{Signs, Numbers, ...} \}$$

4.2.3 Mengen–Axiome

Domänen verwalten konkrete Werte. Die Angabe eines konkreten Wertes geschieht durch die Verwendung einer eindeutigen logischen Bezeichnung. In der Realität existiert der Wert nur rein imaginär. Erst durch das Vergeben und Verknüpfen des Wertes mit einer eindeutigen Bezeichnung wird der Wert für einen Menschen logisch fassbar.

Die logische Wertbezeichnung erlaubt es dem Menschen, die Bezeichnung mittels symbolischer Zeichen (z.B. unser Alphabet) niederschreiben zu können. Theoretisch ist es einem Menschen möglich, alle seine ihm bekannten Werte schriftlich zu fixieren. Die Erkenntnisfähigkeit eines Menschen ermöglicht die Entdeckung neuer Werte, die sein persönliches Wertuniversum erweitern. Solch eine Erweiterung bedingt die Erweiterung von Domänen oder Beschreibungsdomänen oder gar die Schaffung neuer Domänen.

Mit der Relation Δ sind Domänen gegeben. Aus Sicht der EDV basieren die Domänen auf den diskreten Werten, die im System hinterlegt sind. Da von vornherein nicht alle Werte einem System mitgegeben werden können, müssen Verfahren zur Verfügung gestellt werden, um neue diskrete Werte zu ergänzen, bestehende anzupassen und obsolet gewordene Werte zu löschen, d.h. einem Anwender soll es möglich sein, eigene Domänen zu definieren und bestehende zu verändern. Eine Anforderung an das System ist die Unterstützung der Domänenanpassung durch entsprechende Verfahren. Formal werden diese Verfahren durch die folgenden Mengenaxiome beschrieben. Die Axiome geben Aufschluss über die als existent betrachteten Operationen bzgl. Mengen. Die Werte einer Domäne bilden eine Menge und alle (bekannten) Domänen bilden eine Menge, die als Domänenmenge \mathbf{D} bezeichnet sein soll. Durch axiomatisch und allgemein vereinbarte Operationen soll die Bearbeitung aller möglichen Mengen gewährleistet sein.

Einfügeoperationen

- Hinzufügen
Zwecks der Erweiterbarkeit einer Menge existiere eine Hinzufüge–Operation **INSERT** dergestalt, dass darüber die gegebene Menge X um ein neues Element x erweitert wird, d.h.

$$\mathbf{INSERT}(x, X): X \rightarrow X \cup \{x\}.$$

Bzgl. der Domänen sei vereinbart, dass die Erstellung einer neuen Domäne dadurch geschieht, dass jeweils eine Einfügeoperation auf die Typbezeichnungsmenge \mathbf{T} , die Wertemengenmenge \mathbf{W} und die Domänenmenge \mathbf{D} vorgenommen wird.

- Entfernen:
Zwecks der Reduzierung einer Menge existiere eine Entferne–Operation **DELETE** dergestalt, dass darüber aus der gegebenen Menge X ein bestimmtes Element x entfernt wird, d.h.

$$\mathbf{DELETE}(x, X): X \rightarrow X \setminus \{x\}.$$

Bzgl. Domänen sei vereinbart, dass die Entfernung einer Domäne dadurch getätigt wird, dass jeweils eine Entfernungsoperation auf die Typbezeichnungsmenge \mathbf{T} , die Wertemengenmenge \mathbf{W} und die Domänenmenge \mathbf{D} getätigt wird.

Potenzmengenbildung

- Es existiere eine Potenzmengenbildungs-Operation **POWERSET**, die zu einer gegebenen Menge deren Potenzmenge bildet, d.h.:

$$\mathbf{POWERSET}(X): X \rightarrow P(X)$$

Über die Potenzmengenbildung soll später die Mehrwertigkeit abgebildet werden.

Anfrageoperationen

- Die Anzahl der Elemente einer Menge X ist über eine Kardinalitätsoperation **COUNT** ermittelbar, d.h.

$$\mathbf{COUNT}: |X| \rightarrow n \in \mathbb{N}_0$$

- Die Elemente einer Menge X sind über eine Ausleseoperation **ALL** erfragbar, d.h.

$$\mathbf{ALL}: X \rightarrow (x_1, x_2, x_3 \dots x_n)$$

listet die Elemente von X auf. Mathematisch sei diese Operation durch den Allquantor \forall wiedergegeben.

Prüfoperationen

- Wertgleichheit:
Zur Prüfung, ob zwei Elemente, die jeweils einer beliebigen Wertemenge angehören, gleich sind, existiere eine Gleichheitsoperation **EQUAL**. Voraussetzung dafür ist die Definition einer Domäne *Wahrheitswerte* gemäß

$$\begin{aligned} \delta_{\text{Wahrheitswerte}} &= (\tau_{\text{Wahrheitswerte}}, \Omega_{\text{Wahrheitswerte}}) \\ &= (\text{Wahrheitswerte}, \{ \text{TRUE}, \text{FALSE} \}) \end{aligned}$$

Die Gleichheitsoperation bildet das Tupel (w_1, w_2) , die das Paar der zwei zu vergleichenden Werte bildet, auf die Menge der Wahrheitswerte ab, gemäß

$$\begin{aligned} \mathbf{EQUAL}: (w_1, w_2) &\rightarrow \text{TRUE}, \text{ wenn } w_1 \text{ und } w_2 \text{ gleich sind, und} \\ \mathbf{EQUAL}: (w_1, w_2) &\rightarrow \text{FALSE}, \text{ wenn } w_1 \text{ und } w_2 \text{ ungleich sind.} \end{aligned}$$

- Existenz:
Zur Prüfung, ob ein Element w in einer Menge W enthalten ist, existiere eine Existenzoperation **EXIST**. Die Existenzoperation bildet das Tupel (w, W) auf die Menge der Wahrheitswerte ab, gemäß

$$\begin{aligned} \mathbf{EXIST}: (w, W) &\rightarrow \text{TRUE, wenn } w \text{ in } W \text{ enthalten ist, und} \\ \mathbf{EXIST}: (w, W) &\rightarrow \text{FALSE, wenn } w \text{ nicht in } W \text{ enthalten ist.} \end{aligned}$$

Es wird angenommen, dass diese Existenzprüfung auf die **ALL**-Anfrage zurückgeführt wird und jedes Element der Menge mit dem Vergleichswert einzeln über die **EQUAL**-

Prüfung verglichen wird. Liefert ein einzelner Wertvergleich TRUE, so liefert EXIST ebenfalls TRUE; im anderen Fall liefert EXIST den Wert FALSE. Mathematisch sei wie folgt formalisiert:

$$\begin{aligned}\mathbf{EXIST} \rightarrow \text{TRUE} &\rightarrow x \in X \\ \mathbf{EXIST} \rightarrow \text{FALSE} &\rightarrow x \notin X\end{aligned}$$

Diese Prüfoperation soll auch Anwendung finden, wenn es sich bei x um eine Menge handelt.

- Mengengleichheit:
Zur Prüfung, ob zwei Mengen gleich sind, wird die Gleichheitsoperation **EQUAL** auf eine „Verarbeitung“ von Mengen erweitert. Dabei soll die Gleichheitsoperation wie folgt vereinbart sein:

$$\begin{aligned}\mathbf{EQUAL}: (W_1, W_2) \rightarrow \text{TRUE} &\leftrightarrow \forall x \in W_1 | x \in W_2 \wedge \forall y \in W_2 | y \in W_1 \\ \mathbf{EQUAL}: (W_1, W_2) \rightarrow \text{FALSE} &\leftrightarrow \exists x \in W_1 | x \notin W_2 \vee \exists y \in W_2 | y \notin W_1\end{aligned}$$

Spezial-Operationen

- Bzgl. konkreter Domänen sollen Spezialoperationen **SPECIALS** definiert sein. So kann bei Zahlen z.B. die Addition, Subtraktion, Multiplikation, etc. jeweils als eigene Operation definiert sein, die die mit dem jeweiligen Begriff verbundenen speziellen Operationen ausführt.
Da die Definition solcher Spezialoperationen zu umfangreich erscheint, wird auf die Spezifikation an dieser Stelle verzichtet. Wichtig ist die Feststellung, dass Domänen spezielle Operationen definieren können.

Es bleibt anzumerken, dass die Axiome auf Operationen abzubilden sind, die in einem Informationssystem zur Verwaltung von Domänen zur Verfügung gestellt werden. Da die Domänen als Mengen betrachtet werden, für diese Mengen jedoch nicht angenommen werden kann, dass die Elemente der Mengen einheitlich verarbeitet werden können, wird im Folgenden vorausgesetzt, dass die die Axiome abbildenden Operationen generischer Natur sind. Dies bedeutet, dass die Operationen bei Aufruf vorgenommen werden, unabhängig davon, um welche Mengenmanipulation es sich handelt.

So wird beispielsweise bei `INSERT(<Spezielle Person>, <Personenmenge>)` die gewünschte Operation durchgeführt, genau wie für `INSERT(<Aufzählungswert>, <Aufzählungsliste>)`. Nach Abschluss der Einfügeoperationen sind beide Wertemengen jeweils um einen Wert erweitert, unabhängig davon, ob die zwei Wertemengen gleich oder ungleich behandelt werden. Wenn beide Wertemengen gleich behandelt werden können, genügt eine Einfügeoperation für beide Wertemengen. Wenn beide Wertemengen unterschiedlich aufgebaut sind, muss die Einfügeoperation diesen Umstand bei der Mengenerweiterung berücksichtigen. Eventuell sind zwei unterschiedliche Realisierungen der Einfügeoperationen zur Verfügung zu stellen.

4.2.4 Beschreibungsdomäne

In der bisherigen Betrachtung ist eine Domäne als eine *ausgezeichnete* Wertemenge gegeben, d.h. die Wertemenge enthält konkrete Werte, und ihre Kardinalität ist konstant. Wenn die Kardinalität der Wertemenge einer Domäne variabel ist, wird sie als Beschreibungsdomäne bezeichnet. Die Angabe einer Beschreibungsdomäne unterscheidet sich von der Angabe einer Domäne dahingehend, dass anstelle der Wertemenge eine Definition angegeben wird.

Definition 4.3 Characterization Domain

Sei $\tau \in \mathbf{T}$ eine Typbezeichnung, die als Fachausdruck, und $Definition \in \mathbf{W}$ eine Beschreibung, die als umgangssprachliche Definition gewertet wird. Eine **Characterization Domain** (*Beschreibungsdomäne*) ist gegeben als ein Tupel:

$$\gamma = (\tau, Definition)$$

1. Die Typbezeichnung τ einer Beschreibungsdomäne wird abstrakt mit *Name* bezeichnet. Der Name ist die Charakteristik, die als eine Zeichenkette die Bezeichnung aufnimmt, welche die Beschreibungsdomäne γ unter Fachleuten eindeutig kennzeichnet. Es handelt sich bei der Bezeichnung um einen Begriff, der die Beschreibungsdomäne in einer Art und Weise „*sprechend*“ macht. Experten interpretieren den Begriff als (Fach-)Ausdruck und erhalten durch ihn die korrekte abstrakte Vorstellung und eine Grundlage für die korrekte Klassifikation der Werte, die durch die Beschreibungsdomäne beschrieben werden. Beispiele für *sprechende* Bezeichnungen sind Tisch, Stuhl, Auto, Computer, Person, usw.
2. Falls der Name der Beschreibungsdomäne als Fachausdruck nicht für eine klare Verständigung bzw. Klassifizierung ausreicht, erklärt die optionale *Definition* die Bedeutung und den Zweck der Beschreibungsdomäne. Die (umgangssprachliche) Erklärung entspricht einer Beschreibung, wie sie Experten zur Erläuterung eines Sachverhaltes geben. Die *Definition* erläutert zusätzlich, welche Ziele mit der Beschreibungsdomäne erreicht werden sollen.

Beispielsweise kann eine Beschreibungsdomäne *Adresse* wie folgt angegeben werden:

$$\gamma_{\text{Adresse}} = (\text{Adresse}, \text{„Eine Adresse ist die formatierte Angabe einer Anschrift auf Postsendungen“})$$

4.3 Formale Definition von Pliable Objects

Beschreibungsdomänen stellen Werte eines Wertebereiches in einen kontextuellen Zusammenhang. Falls alle Werte der Beschreibungsdomäne eine derartig komplexe Natur besitzen, dass sie formal einzeln bei einer möglichen Wertemengenaufzählung als mehrwertige Objekte in Tupelform notiert werden, dann wird von Pliable Objects gesprochen. Dabei wird davon ausgegangen, dass alle Werte Gemeinsamkeiten bei Charakteristika und bei Verhalten haben. Formal wird ein Pliable Object wie folgt definiert:

Definition 4.4 Pliable Object

Ein *Pliable Object* π ist eine Beschreibungsdomäne γ , die um eine Menge von *Attributes* A , einer Menge von *Actions* X und einer Menge von *Rules* P zu einem Tupel

$$\pi = (\gamma, A, X, P)$$

erweitert wird.

Die Definition eines Pliable Objects realisiert folgenden dualen Sachverhalt:

1. Zum einen kann unter einem Pliable Object ein Sachverhalt verstanden werden, der eine Menge gleichartiger Werte verwaltet.
2. Zum anderen kann unter einem Pliable Object auch ein konkreter Wert verstanden werden, der in der zugeordneten Wertemenge der Beschreibungsdomäne enthalten ist, die selbst zu einem Pliable Object erweitert wird.

Die Wertemenge eines Pliable Objects enthält generell zwar konkrete Werte, doch zeichnen sich diese Werte durch eine Mehrwertigkeit aus. Diese Mehrwertigkeit kann als eine Art Vektor verstanden werden, der selbst aus konkreten Werten aufgebaut ist. Diese konkreten Werte werden durch die Attribute ausgedrückt, d.h. die Werte sind den Attributen zugeordnet. Dadurch stellen die Attribute eine Brücke zwischen der Wertemenge des Pliable Objects und eines konkreten Elementes der Wertemenge her. Die Betrachtung eines Pliable Objects differenziert sich anhand der Frage, ob ein Pliable Object inklusive der Wertausprägung seiner Attribute oder unabhängig von dieser Wertausprägung vorgenommen wird. Um die Doppeldeutigkeit aufzulösen sei bei einem Pliable Object zwischen einem Schema und einer Instanz zu unterscheiden:

1. Ein Pliable Object sei ein Schema, wenn die Betrachtung auf das Pliable Object ohne die Wertausprägung der Attribute vorgenommen wird.
2. Ein Pliable Object sei eine Instanz, wenn die Wertausprägung der Attribute mit in die Betrachtung einfließt.

4.4 Zustandsbeschreibung von Pliable Objects

Jedem Pliable Object sind bestimmte Charakteristika zu eigen. Es handelt sich dabei um die Eigenschaften, die das Pliable Object beschreiben und für dieses auch relevant sind, um seinen Zweck erfüllen zu können. Die Menge der Eigenschaften dient dazu, das Pliable Object zu strukturieren. Auf Basis einer solchen Struktur kann der Zustand eines Pliable Objects „zum Ausdruck gebracht“ werden. Als Zustand wird die wertmäßige Erfassung einer Eigenschaft bezeichnet, d.h. einer Eigenschaft wird ein Wert zugewiesen. Über die Einzelzustände der Eigenschaften wird schließlich der (Gesamt-)Zustand des Pliable Objects beschrieben. Unter einer Eigenschaft selbst wird ein Attribut des Pliable Objects verstanden.

Die Attribute sind als eine Menge von Eigenschaften gegeben, die das Pliable Object besitzen kann. In den Attributen werden die wertmäßigen Daten der Pliable Objects gehalten, die ein konkretes Pliable Object beschreiben und zu dessen informativem Wert beitragen. In der

Regel wird man unter Attributen von Pliable Objects dasselbe verstehen können wie unter Attributen in der klassischen Objektorientierung.

Definition 4.5 Attribute

Ein *Attribute* α dient der Angabe einer Zustandsinformation und sei gegeben als das Tripel

$$\alpha = (v_\alpha, \delta_\tau, \omega_\alpha).$$

- Die Benennung *Name* v_α ist eine Bezeichnung, die eine eindeutige Kennzeichnung darstellt. Bildet man die Menge aller Attribute durch Vereinigung aller Attribute sämtlicher Pliable Objects, so kennzeichnet diese Bezeichnung das Attribut innerhalb der Menge eindeutig und macht es für Fachleute identifizierbar¹⁴.
- Die *Domäne* δ_τ kennzeichnet allgemein einen Wertebereich, der die zulässigen Werte des Attributes enthält. Der Wertebereich eines Attributes ist in der Regel nicht veränderlich; er ist grundsätzlich gegeben durch die Wertemenge Ω_τ der Domäne δ_τ . Im Attributtupel wird die Domäne durch ihre Typbezeichnung τ oder alternativ durch die Wertemenge Ω_τ selbst angegeben.
- Der *Wert* ω_α eines Attributes ist ein Element aus der Wertemenge der Attributsdomäne. Anhand des Wertes wird der Zustand des Attributes definiert. Der Wert des Attributes ist in der Regel veränderlich. Eine Wertänderung entspricht einem Zustandswechsel des Attributes. Beim Einsatz von Attributen wird vorausgesetzt, dass der Wert des Attributes grundsätzlich aus der Wertemenge stammt.

Default-Wert

Bei der Verarbeitung ist geplant, dass der Anwender den Wert eines Attributes durch die Dateneingabe bestimmt. Vereinbarung wird, dass ein Attribut vor der Dateneingabe einen Wert enthält, der kennzeichnet, dass das Attribut noch keinen Wert zugewiesen bekommen hat. Dieser Wert sei als der **Defaultwert** ε gegeben. Bei diesem Defaultwert handelt es sich um einen fest definierten Wert, der generell die Wertvorbelegung eines jeden Attributes realisiert.

Für die *Domäne* δ_τ eines Attributes folgt aus der Vereinbarung, dass der Wertebereich, der die zulässigen Werte des Attributes enthält, auch den Defaultwert enthält. Der Wertebereich eines Attributes ist in der Regel nicht veränderlich; er ist grundsätzlich gegeben durch die Wertemenge Ω_τ der Domäne δ_τ . Ist der Defaultwert $\varepsilon \notin \Omega_\tau$, so muss der Defaultwert durch die Vereinigung mit der Wertemenge der Domäne hinzugefügt werden. In diesem Fall erhält man die *Domäne* δ_τ gemäß:

$$\delta_\tau = (\tau, \Omega_\tau) \text{ mit } \Omega_\tau = \Omega_\tau \cup \{ \varepsilon \}$$

¹⁴ Vergleicht man Attribute verschiedener Pliable Objects miteinander, so können Bezeichnungsgleichheiten auftreten, die in Kombination mit dem zugehörigen Pliable Object eindeutig aufgelöst werden. So ist z.B. *Nummer* allgemein nicht eindeutig. Die Nummer des Pliable Objects *Konto* und die Nummer eines Angestellten, d.h. die *Personalnummer*, sind jedoch unterscheidbar.

Wenn gilt, dass der Wert $\varepsilon \in \Omega_\tau$, so muss ein Wert aus der Wertemenge der Domäne als Defaultwert gekennzeichnet sein. Im Attributtupel wird die Domäne durch ihre Typbezeichnung τ oder alternativ durch die Wertemenge Ω_τ selbst angegeben.

Beispiele:

Mit der Attributdefinition können Attribute folgendermaßen angegeben werden:

$$\alpha_{\text{Geschlecht}} = (\text{Geschlecht, Geschlechtsarten, } \varepsilon)$$

$$\alpha_{\text{Herzinfarkt}} = (\text{Herzinfarkt, Herzinfarktsarten, } \varepsilon)$$

Vorausgesetzt werden die Domänen:

$$\delta_{\text{Geschlechtsarten}} = (\text{Geschlechtsarten, } \{\text{männlich, weiblich, intersexuell}\})$$

$$\delta_{\text{Herzinfarktsarten}} = (\text{Herzinfarktsarten, } \{\text{n.bekannt, } < 3 \text{ Monate, } < 6 \text{ Monate, } > 6 \text{ Monate}\})$$

Beispielsweise wäre ein konkretes Attribut Herzinfarkt¹⁵ mit der entsprechenden Domäne wie folgt anzugeben:

$$\alpha_{\text{Herzinfarkt}} = (\text{Herzinfarkt, } \delta_{\text{Herzinfarktsarten, n.bekannt})$$

mit

$$\delta_{\text{Herzinfarktsarten}} = (\text{Herzinfarktsarten, } \{< 3 \text{ Monate, } < 6 \text{ Monate, } > 6 \text{ Monate}\} \cup \{\text{n.bekannt}\})$$

4.5 Verhaltensbeschreibung von Pliable Objects

Ein Pliable Object besitzt nicht nur einen Zustand, sondern kann viele voneinander verschiedene Zustände annehmen. Zu einem konkreten Zeitpunkt hat ein Pliable Object nur einen Zustand inne. Von diesem Zustand aus kann es in andere Zustände wechseln. Die Art und Weise, wie ein Zustandswechsel vollzogen wird, steht in einer Art „Vorschrift“, die dem Pliable Object zugeordnet ist. Alle einem Pliable Object zugeordneten Vorschriften definieren das Verhalten des Pliable Objects.

Eine „Vorschrift“ bündelt verschiedene Tätigkeiten, die das Pliable Object jeweils in einen bestimmten Zustand überführen. Eine Überführung kann eventuell über mehrere Zustandswechsel geschehen. Um solch eine Überführung beschreiben zu können, muss die Vorschrift genauer spezifiziert werden. Die Vorschrift besitzt ihr eigenes Verhalten. Die Ausprägungen des Verhaltens werden abstrakt durch eine Struktur beschrieben. Eine konkrete Ausprägung einer Struktur ist eine Aktion, d.h. eine konkrete Vorschrift, die eine bestimmte Tätigkeit beschreibt. Formal soll eine Aktion wie folgt definiert sein:

¹⁵ Anmerkung: Die Reinfarktrate ist innerhalb des ersten halben Jahres nach dem Infarkt besonders hoch. Liegt ein Infarkt maximal ein halbes Jahr zurück, wird empfohlen, einen neuen Operationstermin festzulegen, wenn die Operation aufschiebbar ist. Andernfalls sollten zusätzliche subtile Überwachungsmethoden zum Einsatz kommen.

Definition 4.6 Action

Eine *Action* χ ist die Definition eines Zustandswechsels und sei gegeben als das Tripel

$$\chi = (v_\chi, A_\chi, \Sigma_\chi) \text{ mit } A_\chi \neq \emptyset.$$

- Die *Benennung* v_χ ist – analog zur Attributdefinition – eine Bezeichnung, die eine eindeutige Kennzeichnung darstellt. Bildet man die Menge aller Aktionen durch Vereinigung aller Aktionen sämtlicher Pliable Objects, so kennzeichnet diese Bezeichnung die Aktion innerhalb der Menge eindeutig und macht es für Fachleute identifizierbar¹⁶.
- Die *Aktionsattribute* A_χ stellen allgemein eine Art Wertemenge der Aktion dar. Diese Wertemenge sollte in der Regel nicht veränderlich und darf auch nicht leer sein. Die Wertemenge enthält die Attribute, deren Zustand im Rahmen der Ausführung der Aktion verändert bzw. die zur „Verarbeitung“ herangezogen werden. Durch die Namen und Domänen der Attribute wird die Grammatik der Aktionsdefinition konkretisiert (s.u.).
- Die *Aktionsliste* Σ_χ der Aktion ist eine nichtleere Menge von Anweisungen. Diese Anweisungen stehen in einer geordneten Reihenfolge und beschreiben die Zustandsänderungen. Die Anweisungen genügen einer kontextfreien Grammatik G , die durch die Namen und Domänen der Aktionsattribute zu einer aktionsspezifischen Grammatik K_G konkretisiert werden. Es ist anzunehmen, dass Aktionen definiert werden, die nur eine einzige Anweisung enthalten. In diesem Fall wird ein möglicher Zustandswechsel des Pliable Objects durch diese Anweisung beschrieben.

4.5.1 Anlehnung an die Algorithmus-Definition

In der Informatik wird eine Zustandsänderung durch Ausführung eines Algorithmus herbeigeführt. Der Begriff des Algorithmus ist dabei intuitiv gegeben, und man versteht darunter gewöhnlich eine allgemeine Methode, ein allgemeines Verfahren zur Lösung einer Klasse von Problemen (vgl. [BI68]). Im Informatik-Handbuch [RP02] wird der Algorithmus wie folgt definiert:

„Ein Algorithmus ist eine präzise, d.h. in einer festgelegten Sprache abgefasste, endliche Beschreibung eines schrittweisen Problemlösungsverfahrens zur Ermittlung gesuchter Größen aus gegebenen Größen, in dem jeder Schritt aus einer Anzahl ausführbarer eindeutiger Aktionen und einer Angabe über den nächsten Schritt besteht. Ein Algorithmus heißt abbrechend, wenn er die gesuchten Größen nach endlich vielen Schritten liefert, andernfalls heißt der Algorithmus nicht abbrechend.“

Die Algorithmus-Definition baut den Algorithmus selbst durch *eindeutige Aktionen* auf, wobei die Aktionen sowie deren Eindeutigkeit intuitiv gegeben sind. In Verwendung bei Pliable Objects wird durch eine Menge von Aktionen die Gesamtheit des Verhaltens des Pliable Objects ausgedrückt. Eine Aktion ist dabei ein „Verfahren“, das eine Zustandsänderung auf einem oder mehreren Attributen des Pliable Objects herbeiführen kann. Unter einer Aktion wird daher ein Algorithmus verstanden, der speziell auf bestimmte

¹⁶ Vergleicht man Aktionen verschiedener Pliable Objects miteinander, so können Bezeichnungsgleichheiten auftreten, die in Kombination mit dem zugehörigen Pliable Object eindeutig aufgelöst werden; so ist z.B. „Fahren“ nicht eindeutig. Aber das „Fahren“ des Pliable Objects *Schiff* und das „Fahren“ des Pliable Objects *Auto* sind unterscheidbar.

Zustandsänderungen eines Pliable Objects zugeschnitten ist. Eine Aktion muss dazu einerseits Kenntnis über die Attribute des Pliable Objects besitzen, deren Zustand die Aktion ändern soll, andererseits eine Beschreibung dessen darstellen, wie die Zustandsänderung vorzunehmen wäre.

4.5.2 Klassifikation von Zustandsänderungen

Aktionen dienen der Herbeiführung eines Zustandswechsels auf Pliable Objects. Der Zustand eines Pliable Objects wird durch seine Attribute ausgedrückt. Unabhängig von der genauen Kenntnis eines konkreten Pliable Objects kann angenommen werden, dass ein Pliable Object je nach Anzahl seiner Attribute eine entsprechend große Anzahl an möglichen Zuständen annehmen kann. Unter der weiteren Annahme, dass jeder Zustand nur durch eine bestimmte Aktion erreicht werden kann, ergibt sich eine mutmaßlich große Anzahl verschiedener Aktionen. Aus der Existenz einer Aktion folgt die Existenz von mindestens einem Attribut, welches die Zustände erhalten kann, die mit der Ausführung der Aktion bewirkt werden können. Aktionen sind somit von Attributen abhängig. Umgekehrt sind Attribute nur von Aktionen abhängig, wenn der Zustand, der über das Attribut gegeben ist, durch die Aktion geändert werden soll.

Die Art einer Zustandsänderung ist überschaubar, so dass trotz einer großen Anzahl an möglichen Aktionen eine überschaubare Klassifikation angegeben werden kann. Die Klassifikation unterteilt Aktionen in elementare Kategorien (vgl. Abbildung 20).

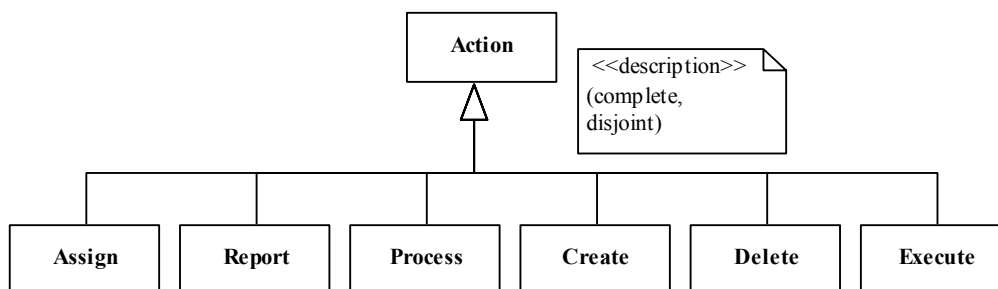


Abbildung 20: Action-Kategorien

Report und Assign

Aktionen der Kategorie Report und Assign realisieren eine Abfrage bzw. eine Zuweisung von Werten auf Attributen. Eine **Report**-Aktion fragt den Zustand eines Attributes ab. Sie ermittelt den Attributwert und übergibt ihn an ein Ausgabemedium. Eine **Assign**-Aktion dagegen nimmt einen Wert von einem Eingabemedium entgegen und weist diesen Wert einem spezifizierten Attribut zu, d.h. eine Assign-Aktion versetzt ein Attribut in einen Zustand.

Process

Unter Process werden Aktionen gruppiert, die eine elementare Verknüpfung auf den Werten von Attributen realisieren. Eine Verknüpfung entspricht auf maschineller Ebene einer Verarbeitung. Darunter können z.B. die Bildung einer Summe, einer Konkatenation etc. verstanden werden. Eine **Process**-Aktion verwendet intern die Report- und Assign-Aktionen der Attribute, deren Werte für die Verknüpfung benötigt werden. Die Berechnungen, die letztlich auf Basis der Werte der Attribute durchgeführt werden, sind zwischen dem Abfragen und Zuweisen des/der jeweiligen Werte(s) positioniert.

Create und Delete

Die bisherigen Kategorien arbeiten auf einer fest vorgegebenen Menge von Werten bzw. Attributen. Um die Menge verändern, d.h. erweitern oder einschränken zu können, dienen die Aktionen Create und Delete. Die Erzeugungsaktion **Create** fügt einer gegebenen Situation „Neues“ hinzu, d.h. mit Hilfe der Erzeugungsaktion können neue Werte oder neue Objekte erzeugt werden. Die Löschkaktion **Delete** entfernt aus einer gegebenen Situation „Bekanntes“, d.h. bekannte Werte oder bekannte Objekte werden gelöscht.

Execute

Die bisherigen allgemeinen Aktionsbeschreibungen stellen „Mächtigkeiten“ dar, die „etwas ermöglichen“. Das „Mögliche“ muss allerdings in das „Konkrete“ überführt werden. Dieser Umstand wird so zum Ausdruck gebracht, dass jede Aktion selbst zwei Zustände besitzt:

1. Der erste Zustand ist der wartende Zustand, d.h. die Aktion kann ausgeführt werden und
2. der zweite Zustand ist der ausführende Zustand, d.h. der Zustand, in dem die Aktion gerade ausgeführt wird.

Die Ausführungsaktion **Execute** dient dazu, dass Aktionen von dem einen Aktionszustand in den anderen wechseln. Sie veranlasst, dass eine nicht ausgeführte Aktion ausgeführt wird und anschließend wieder in den Urzustand zurückkehrt. Auf diese Weise wird die Ausführungsfunktionalität in die anderen Aktionen integriert.

4.5.3 Sequenzierung von Abläufen mittels Statements

In allen sechs Kategorien müssen Beschreibungen existieren, die den Ablauf dessen festlegen, was durch die jeweilige Aktion erreicht werden soll. Während in den Kategorien Report, Assign, Create, Delete und Execute die Ablaufbeschreibungen relativ klar und eindeutig vorhanden sein müssen, wird in der Process-Kategorie durch variable Ablaufbeschreibungen eine große Vielfalt ermöglicht. Die Grundlage von Ablaufbeschreibungen sind Anweisungen, die unter **Statements** zusammengefasst werden.

Ein Statement (vgl. Abbildung 21) besitzt in der Regel eine Menge an Statementparametern. Die Statementparameter stellen die Schnittstelle des Statements nach außen dar. Statementparameter unterteilen sich in Argumente und Results. Ein Argument dient zur

Informationshinzunahme, während ein Result der Informationsveröffentlichung dient. Das Statement nutzt die Information, die ein Argument enthält, um darauf eine Verknüpfung aufzubauen. Aus der Verknüpfung heraus kann eventuell ein Ergebnis bestimmt werden. Falls dieses Ergebnis „nach außen“ geliefert werden soll, wird ein Statementparameter des Typs Result dazu verwendet.

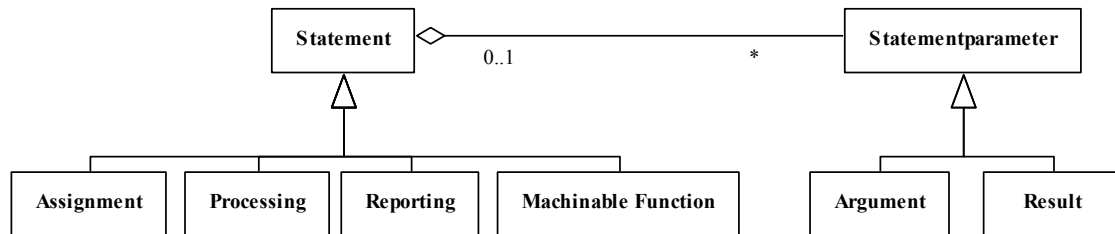


Abbildung 21: Statement-Strukturierung

Statements selbst sind maschinennahe ausführbare Anweisungen. Sie gliedern sich in Assignment-, Processing- und Reporting-Statements sowie Machinable Functions:

- Ein Assignmentstatement entspricht einer Zuweisung eines Wertes in einen Speicherbereich.
- Das Reporting-Statement wird zum Lesen eines Wertes aus dem Speicherbereich verwendet.
- Ein Processing-Statement kann eine zusammengesetzte Statementfolge realisieren, wobei die Einzelstatements durch elementare und allgemeingültige Gegebenheiten der Maschinenhardware gelöst werden. So können z.B. sämtliche Summenberechnungen durch eine einzelne entsprechende integrierte Schaltung (IC) durchgeführt werden.
- Machinable Functions sind Hardwarelösungen, die nicht allgemeingültig sind. Es handelt sich um Hardware, die eine Sonderfunktionalität realisiert (vgl. [Te97a]).

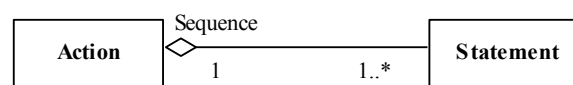


Abbildung 22: Ablaufbeschreibung einer Aktion

Mit obigem Verständnis von Statements sollen Ablaufbeschreibungen, jeweils bestehend aus einer Sequenz von Statements, ermöglicht werden. Eine bestimmte Ablaufbeschreibung spezifiziert dann eine konkrete Action (vgl. Abbildung 22).

4.5.4 Aktionsbeschreibung durch kontextfreie Grammatiken

Ein Statement soll generell sowohl vom Menschen als auch von einer Maschine lesbar sein. Ein Statement kann dadurch analog zu einer Anweisung in einer Programmiersprache verstanden werden. Gemäß [Lo94] ist eine Programmiersprache definiert als „*ein notationelles System zur Beschreibung von Berechnungen in durch Maschinen und Menschen lesbarer Form*“. Die hierarchische Struktur von Programmiersprachen wird auf recht natürliche Weise mittels einer Grammatik beschrieben (vgl. [ASU99], [Er08]). Häufig wird jeweils in Backus-Naur-Form (BNF) als kontextfreie Grammatik die Syntax von Programmiersprachen ausgedrückt. Dies erleichtert das Schreiben von Übersetzern für die Sprache, da die Stufe des Parsens automatisiert werden kann (vgl. [BH98]).

Eine kontextfreie Grammatik besteht aus einer Reihe grammatikalischer Regeln, die sich in Komponenten aufteilen (nach [ASU99], [Lo94]):

1. Eine Menge von Symbolen, auch Terminale genannt, die sich in keine weiteren Strukturen zerlegen lassen.
2. Eine Menge von Nichtterminalen, die Strukturen beschreiben, die sich in Terminale und Nichtterminale zerlegen lassen.
3. Eine Menge von Produktionen, die die Grammatikregeln zum Ausdruck bringen und unter Verwendung von Ableitungen die Zeichenfolgen „produzieren“, die der Grammatik genügt. Die Produktionen enthalten nur Metasymbole¹⁷. Dabei gliedert das Metasymbol „::=“ eine Produktion in zwei Teile. Der linke Teil nimmt die Nichtterminalsymbole und der rechte Teil Kombinationen sowohl aus Nicht- als auch aus Terminalsymbolen der Grammatik auf.
4. Ein ausgezeichnetes Nichtterminal als Startsymbol.

Mit Hilfe einer geeigneten Grammatik ist es möglich, die Statements einer Aktion derart zu formulieren, dass eine Maschine die Anweisungen ausführen kann. Die Ausführungsreihenfolge der Statements geschieht gemäß der Ordnungsreihenfolge, die wiederum die Sequenz von Zustandswechseln widerspiegelt, die durch die Ablaufbeschreibung einer Aktion ausgedrückt wird.

Damit ergibt sich eine direkte Analogie zu Ausdrücken in Programmiersprachen. Es liegt daher nahe, Erkenntnisse und Erfahrungen aus der Entwicklung von Programmiersprachen zu nutzen, um Aktionen abzubilden. Im Fokus steht dabei das Finden von Gemeinsamkeiten und eventuell das Zusammenführen von optionalen Ergänzungen, die Vorteile bieten könnten. Es sei darauf hingewiesen, dass in dieser Arbeit keine Diskussion geführt wird, ob Aktionen explizit durch eine Programmiersprache ausgedrückt werden oder ob Programmiersprachen die Einbettung von Aktionen erlauben sollen.

4.5.5 Spezifische Erweiterung über programmiersprachliche Konstrukte

Programmiersprachen besitzen Elemente wie arithmetische Ausdrücke, Bedingungsausdrücke etc., die zur Beschreibung von Rechenverfahren herangezogen werden, um diese Rechenverfahren von einer Maschine ausführen zu lassen. Programmiersprachen sind dabei derart universell gehalten, dass der Anspruch erfüllt werden kann, verschiedene Lösungsverfahren für jeweils verschiedene Aufgaben programmieren zu können (vgl. [Se05], [SM09]).

¹⁷ BNF-Metasymbole: „::=“, „|“, „<“, „>“ bzw. EBNF-Metasymbole: „::=“, „|“, „<“, „>“, „{“, „}“

Aktionen sollen im Gegensatz zu Programmiersprachen nicht derart universell gehalten werden, vielmehr sind Aktionen Pliable Objects zugeordnet. Das Ziel von Aktionen ist die Zustandsänderung von Attributen. Eine Zustandsänderung kann allerdings auch unter Verwendung eines bestimmten Rechenverfahrens bestimmt werden. Es ist daher sinnvoll, die bereits existierenden Lösungen zur Unterstützung von Rechenverfahren derart zu nutzen, dass sie im Rahmen der Zustandsänderung wieder genutzt werden können.

Die Attribute, deren Zustand durch eine Aktion geändert werden kann, sind die Attribute des Pliable Objects, dem die Aktion zugeordnet ist. Um Elemente der Programmiersprache zu verwenden, muss die Grammatik einer Aktion derart formbar sein, dass die von der Aktion verwendeten Attribute die Grammatik spezifisch für die Aktion erweitern; d.h. die Attributbezeichnungen und Attributdomänen der verwendeten Attribute werden direkt in die Grammatik der Aktion aufgenommen. Damit ist es möglich, Zustandsänderungen der Attribute mit der Grammatik auszudrücken. Die Grammatik muss drei Anweisungsarten spezifizieren.

Zuweisungen

In einem Ausdruck muss das Attribut, dessen Zustand geändert werden soll, angegeben werden können, d.h. in der Grammatikdefinition muss der Zugriff auf die Attribute integriert sein. Daraus folgt, dass die Domäne eines Attributes auch in die Mächtigkeit der Grammatik mit einfließt. Die Einbindung der Domäne konkretisiert die Grammatik auf aktionsspezifische Weise, um so auch semantische Informationen zu nutzen. Für jedes Attribut des Pliable Objects, welches in einer Aktion verwendet wird, ist ein Zuweisungsausdruck zu generieren, der jede Zuweisung eines Wertes auf das aktuell bearbeitete Attribut abdeckt.

$$\langle \text{Attributzuweisung} \rangle ::= \langle \text{Attributname} \rangle = \langle \text{Attributwert} \rangle$$

Für den $\langle \text{Attributwert} \rangle$ gilt dabei, dass er einem Terminalsymbol entspricht, welches eindeutig ein Element aus der *Attributdomäne* des Attributs identifiziert, d.h. die Grammatik der Aktion muss sich dynamisch dahingehend anpassen, dass jeder Domänenwert durch ein eindeutiges Terminalsymbol repräsentiert wird.

Operationen bzw. Berechnungen

Sind speziellere Operationen wie z.B. Berechnungsoperationen gewünscht, so muss die Grammatik diese in Abhängigkeit von der Domäne der verwendeten Attribute zur Verfügung stellen. Die Domäne stellt Operationen zur Verfügung, die auf der Domäne möglich sind. Die Grammatik muss die zur Verfügung gestellten Operationen durch eine entsprechende Ausdrucksformalität einbinden können.

Beispiel:

Auf der Domäne **Integer** sind so z.B. die Addition und Multiplikation erlaubte Operationen. Eine Addition in Infix Notation der Form „Zahl = Zahl + 5“ sollte von der Grammatik akzeptiert werden. Eine mögliche Erweiterung ist wie folgt angegeben:

```
<Ausdruck> ::= <Term> {+<Term>}
<Term>      ::= <Faktor> {*<Faktor>}
<Faktor>    ::= (<Ausdruck>) | Zahl | Attributname
<Zahl>      ::= <Ziffer> { <Ziffer>}
<Ziffer>    ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Auf der Domäne *Geschlechter* hingegen ist keine geeignete Addition bzw. Multiplikation vorstellbar, d.h. ein Ausdruck der Form „Geschlecht = Geschlecht + 5“ müsste bemängelt werden. Die jeweiligen Grammatiken sind diesbezüglich domänenspezifiziert.

Im Rahmen von verarbeitenden bzw. berechnenden Anweisungen können temporäre Attribute nötig werden. Temporäre Attribute werden als Statementattribute bezeichnet. Statementattribute realisieren Variablen, die für die Verarbeitung einer Anweisung benötigt werden. Reine Statementattribute dienen als interne temporäre Zwischenspeicher. Für die 'Kommunikation nach außen' existieren die spezielleren Statementattribute, die als Argumente oder Resultate verstanden werden.

Ein Argument ist eine Variable, deren Wert von außen, d.h. außerhalb des Statements, gesetzt wird. Es dient als Input bzw. Funktionsargument für das Statement. Ein Resultat ist eine Variable, deren Wert nach außen gereicht wird; d.h. außerhalb des Statements wird der Parameter ggf. weiterverarbeitet. Ein Resultat dient als Output bzw. Funktionswert bzw. Funktionsergebnis für das Statement. Sie ergeben zusammen mit den Attributen, die eine Zustandsänderung erfahren sollen, die Menge der Aktionsattribute.

Ausführungen

Generell soll die Ausführung einer Aktion einen Zustandswechsel zur Folge haben. Bei einem Zustandswechsel handelt es sich um eine Wertzuweisung¹⁸. Um dies zu gewährleisten, muss die Aktion nicht nur „schriftlich fixierbar“, sondern auch ausführbar sein. Unter der Ausführung einer Aktion wird die sequentielle Abarbeitung der Anweisungen der Aktion verstanden. Jede einzelne Anweisung muss derart interpretierbar sein, dass die Anweisung zum hinterlegten Ziel führt. Gewährleistet und festgelegt wird dies durch die Mächtigkeit der Grammatik. Eventuell muss berücksichtigt werden, dass in der Aktion eine fest definierte Arbeitsabfolge existiert. Eine mögliche Beschreibung der Ausführungserweiterung der Grammatik kann wie folgt angegeben werden:

```
<Executionstatement> ::= Execute <ActionBezeichnung>
<ActionBezeichnung> ::= <Actionname> |<Pliable Object>.<Actionname>
```

¹⁸ Bei einer Zuweisung kann generell der aktuell zugewiesene Wert erneut zugewiesen werden. Prinzipiell verbleibt das „Objekt“ dann im gleichen Zustand. Somit ist mit Zustandswechsel nicht unbedingt eine Wertänderung verbunden.

Schleifen- und Bedingungskonstrukte

Die Aktionsliste kann mehrere Anweisungen enthalten. Damit weist die Aktionsliste eine theoretische Ähnlichkeit zu Prozeduren und Funktionen auf, die in einer Programmiersprache geschrieben werden können. Innerhalb von Prozeduren können Abhängigkeiten zwischen verschiedenen einzelnen Anweisungen der Prozedur existieren. Die Abhängigkeiten werden vom Abarbeitungsablauf berücksichtigt. Die zugehörige Grammatik einer Programmiersprache stellt hierzu Schleifen- und Bedingungskonstrukte zur Verfügung, um die Abhängigkeiten zu realisieren. Ein Bedingungskonstrukt erlaubt eine Verzweigungsmöglichkeit; ein Schleifenkonstrukt erlaubt die mehrfache Ausführung bestimmter Anweisungen.

Die Aktionsliste einer Aktion soll keine Bedingungskonstrukte enthalten. Die Bedingungskonstrukte dienen bei Programmiersprachen als Kontrollstrukturen, die den Kontrollfluß eines Programms beeinflussen. Solch eine Beeinflussung ist bei Aktionen nicht gewünscht. Mit einer Aktion soll ein bestimmter Zustandswechsel erreicht werden. Der Zustandswechsel soll vorgenommen werden, wenn eine Aktion aufgerufen wird. Um dies zu gewährleisten, darf (vorerst) keine direkte Bedingung in der Aktion die Aktionsabarbeitung verhindern. Deshalb wird von dem Vorsehen einer Bedingungsanweisung in Aktionen Abstand genommen.

4.5.6 Spezielle Zuweisungsanweisungen

Report und Assign realisieren alle Formen von Wertermittlungen und -zuweisungen. Andere Zugriffsarten auf Attribute existieren nicht. Die Wertermittlung und die Wertzuweisung sind elementare Aktionen, die in Abhängigkeit von Attributen implementiert werden. Sie besitzen einen generischen Charakter, da der Wert vom vorliegenden Wertebereich abhängig ist. Genau genommen wird die Abhängigkeit vom Wertebereich in Informationssystemen von der technischen Realisierung vererbt. Während der Mensch mit alphanumerischen Zeichenketten alle Werte erfassen kann, sind Maschinen auf die Vorgabe der binären Darstellung angewiesen. Im Rahmen dieser Arbeit wird von der technischen Speicherung Abstand genommen. Die technische Speicherung von Werten soll automatisch vom zugrundeliegenden Managementsystem bestimmt werden. Der Anwender soll dadurch von den technischen Restriktionen sowie von Kenntnissen zu Datentypen wie Integer, Long, String etc. befreit werden.

Weil alle Report-Aktionen den aktuellen Wert eines Attributes liefern und alle Assign-Aktionen ein Attribut auf einen übergebenen Wert setzen, seien diese Aktionen allgemein als

GET für das Abfragen des Wertes und als
SET für das Zuweisen eines Wertes

definiert. Die GET-Aktion realisiert eine Funktion auf der Attribut-Definition. Sie liefert den Wert ω_α des Attributes an eine nicht näher spezifizierte Ausgabe.

GET (α): $\alpha \rightarrow \omega_\alpha$

Die SET-Aktion bekommt von einer nicht näher spezifizierten Eingabequelle einen Wert, den sie einem Attribut zuweist:

$$\text{SET}(\alpha, \varepsilon_\omega): \alpha, \varepsilon_\omega \rightarrow \alpha', \text{ mit } \omega_{\alpha'} = \varepsilon_\omega$$

Es wird in dieser Arbeit davon ausgegangen, dass die Existenz eines Attributes die Existenz der Get- und Set-Aktion automatisch nach sich zieht.

Allgemein kann die SET-Aktion folgendermaßen in der Grammatik ausgedrückt werden:

$$\begin{aligned} \text{SET} & ::= \langle \text{attributname} \rangle = \langle \text{attributvalue} \rangle \\ \langle \text{attributname} \rangle & ::= N_\alpha \\ \langle \text{attributvalue} \rangle & ::= \mathbf{ALL}(\Omega_\alpha)^* \end{aligned}$$

Für das Attribut (Geschlecht, Geschlechtsarten, ε) hätte die konkrete „SET-Geschlechtgrammatik“ das Aussehen

$$\begin{aligned} \text{SET} & ::= \langle \text{attributname} \rangle = \langle \text{attributvalue} \rangle \\ \langle \text{attributname} \rangle & ::= \text{Geschlecht} \\ \langle \text{attributvalue} \rangle & ::= \text{männlich} \mid \text{weiblich} \mid \text{intersexuell} \mid \varepsilon \end{aligned}$$

Es sei angenommen, dass eine allgemeine Vorschrift existiert, die in der Lage ist, die anhand der hinterlegten kontextfreien Grammatik die Ausdrücke dahingehend auszuwerten, dass das in den Ausdrücken hinterlegte Verhalten „demonstriert“ bzw. durchgeführt wird.

4.6 Verhaltenskontrolle von Pliable Objects

Die Attribute eines Pliable Objects dienen zur Beschreibung seines Zustands, während die Aktionen eines Pliable Objects dessen Zustand verändern. Ungeklärt ist die Frage, wann bzw. wodurch eine Zustandsänderung ausgelöst wird. Zur Steuerung von Zuständen bzw. Zustandswechseln sollen Regeln dienen. Eine Regel repräsentiert dadurch einen speziellen Teil des (eventuell möglichen) Verhaltens des Pliable Objects.

Allgemein besteht eine Regel aus einer Prämisse bzw. Bedingung, die mit einer Konklusion bzw. Schlussfolgerung verknüpft ist. Mit Hilfe von Regeln werden bestimmte Sachverhalte ausgedrückt, die einen steuernden Charakter besitzen. In Bezug auf Pliable Objects sollen derartige Sachverhalte Abhängigkeiten widerspiegeln, die entweder als Plausibilitäten oder als Strategien und Restriktionen oder Objektbeziehungen hinterlegt werden.

- **Plausibilitäten:** Jedes Pliable Object besitzt bestimmte Regeln, die die Plausibilitäten und Abhängigkeiten der Attribute voneinander und/oder innerhalb des Pliable Objects beschreiben. In den meisten Regeln wird bezüglich der Attribute eine Bedingung abgefragt, die auf das „Vorliegen“ eines bestimmten Wertes des Attributes aus dessen Domäne bezogen sein dürfte. Regeln arbeiten folglich auch auf den Domänen von Attributen. Die Plausibilitäten werden zur Bestimmung der Korrektheit eines Pliable Objects (in der Regel einer Instanz) definiert. Plausibilitäten dienen zur Unterteilung sämtlicher Zustände eines Pliable Objects in gültige und ungültige Zustände.

- **Strategien und Restriktionen:** Strategien und Restriktionen sind Regeln, die das interne Verhalten des Pliable Objects beschreiben. Diese Regeln beobachten das Pliable Object und reagieren auf festgelegte Zustände, indem sie bestimmte Zustandsveränderungen triggern.
Beispielsweise könnte einem Pliable Object *Patient* eine Regel zugeordnet sein, die nach der Eingabe des Geburtsdatums des Patienten die Berechnung des Alters in Bezug zur aktuellen Systemzeit anstößt.
- **Objektbeziehungen:** Objektbeziehungen sind Regeln, die das Verhalten des Pliable Objects im Zusammenspiel mit weiteren Pliable Objects beschreiben. Diese Beziehungen werden über Verknüpfungen bzw. Assoziationen hergestellt. Eine Beziehung selbst wird durch spezielle Attribute realisiert. Dabei steuert eine Regel das Verknüpfen existenter Pliable Objects.
Beispielsweise kann ein Pliable Object *Patient* automatisch mit einem Pliable Object *Narkosevorbereitung* verknüpft werden, wenn ein Operationsdatum eingegeben worden ist.

Definition 4.7 Rule

Eine Rule ρ dient zur Steuerung von Zustandswechseln und sei gegeben als das Tripel

$$\rho = (v_\rho, B_\rho, K_\rho) \text{ mit } B_\rho \neq \emptyset \text{ und } K_\rho \neq \emptyset,$$

wobei v_ρ eine Benennung, B_ρ eine Menge von Restriktionen und K_ρ eine Menge von Aktionen bezeichnet. Die Aktionen wären als Konklusionen auszuführen, wenn die Restriktionen erfüllt sind.

- Die *Benennung* v_ρ ist – analog zur Attributdefinition – eine Bezeichnung, die eine eindeutige Kennzeichnung darstellt. Bildet man die Menge aller Regeln durch Vereinigung aller Regeln sämtlicher Pliable Objects, so kennzeichnet diese Bezeichnung die Regel innerhalb der Menge eindeutig und macht sie für Fachleute identifizierbar.
- Die *Menge der Restriktionen* B_ρ stellt allgemein eine Art Wertebereich der Regel dar, der üblicherweise nicht veränderlich und auch nicht leer ist. In dem Wertebereich sind elementare Bedingungen in Form von Ausdrücken hinterlegt, die einzeln auf die Wahrheitswerte True und False abgebildet werden. Durch Konjunktion wird ein Gesamtwahrheitswert ermittelt, dessen Zustand im Rahmen der Ausführung der Regel die Anwendung der Schlussfolgerung bedingt oder nicht.
- Die *Menge der Konklusionen* K_ρ entspricht einer möglichen Aktion. Sie sind in einer Vorschrift als Anweisungen zusammengestellt und werden bei Erfüllung der Prämisse nacheinander ausgeführt.

Anmerkung: Es ist anzunehmen, dass in der Vorschrift der Schlussfolgerungen nur eine Aktion mit einer einzigen Anweisung hinterlegt wird, die wiederum einen möglichen Zustandswechsel beschreibt.

Notationsvorschriften für Regeln

Die Regel an sich orientiert sich an dem naturgemäßen Aufbau, dass eine Regel aus einem Bedingungs- und einem Konklusionsteil besteht. Es ist an dieser Stelle zu berücksichtigen, dass die Nutzung des Begriffs „Bedingung“ allein schon wieder einen Sachverhalt ausdrückt. Mit diesem Sachverhalt geht der Begriff der Erfüllbarkeit einher. Eine Bedingung drückt quasi „etwas“ aus, das „sein könnte“. Dieses „sein könnte“ muss formulierbar sein. In Bezug auf Pliable Objects muss eine Bedingung derart formalisierbar sein, dass das „Sein“ als ein Ausdruck erfasst werden kann.

Analog zu den Aktionsbeschreibungen sollen Regeln ebenfalls mittels einer Grammatik notationell angegeben werden können. Grundsätzlich soll ein Ausdruck genutzt werden, der sich an der typischen If-Bedingung von Programmiersprachen orientiert.

Eine Regel soll in einem Ausdruck angegeben werden, der mit dem Begriff „IF“ „umschlossen“ wird. Das Umschließen stellt dabei die Verknüpfung des Ausdrucks mit der Frage der Erfüllbarkeit bzw. des „Erfülltseins“ her.

Ein zweiter Teil der Regel folgt dem Begriff „THEN“. Dieser zweite Teil bezeichnet eine Aktion, die auszuführen ist, wenn der Bedingungsteil der Regel erfüllt ist.

Formal genüge eine Regel folgender Grammatik:

```

<Rule>           ::= IF <condition> THEN <expression>
<condition>      ::= <term> {*,<term>}
<term>           ::= <equalterm> | <notequalterm>
<equalterm>      ::= <Attributname> = <Attributwert>
<notequalterm>   ::= <Attributname> ≠ <Attributwert>
<Attributwert>   ::= <Attributdomäne>
<expression>     ::= ALL (Xπ)
  
```

Mit obiger Grammatik können bestimmte Sachverhalte als Regeln formuliert werden. Unklar ist dabei, ob diese Sachverhalte eingehalten werden müssten, eingehalten werden sollten oder eingehalten werden könnten. Bei einer Regelspezifikation soll diese Angabe als separate Charakteristik der Regel angegeben werden. Dadurch erfolgt gewissermaßen eine Klassifikation von Regeln. Für die Charakteristik wird die MoSCoW-Priorisierung in Anlehnung an [St97] herangezogen. Die Charakteristik vereinbart, ob eine Prämisse erfüllt werden muss, soll oder kann.

Priorität	Abkürzung	Bezeichnung	Bedeutung
I	Mo	Must (muss)	Regel ist eine Constraint Bedingung
II	S	Should (soll)	Regel ist eine Plausibilität
III	Co	Could (kann)	Regel ist eine einfache Bedingung
IV	W	Won't (wird nicht)	Regel ist optional ¹⁹

Tabelle 8: MoSCoW – Priorisierung für Regeln

¹⁹ Ob die Erfüllung einer Regel versucht werden soll, ist eine Frage, die zu einem späteren Zeitpunkt geklärt wird.

Die MoSCoW–Klassifikation (Tabelle 8) von Regeln ist in der Definition noch nicht hinterlegt. Um die MoSCoW–Klassifikation einer Regel zu ermöglichen, wird ein weiteres Regelattribut definiert. Vorab werde eine Domäne *Regelarten* vereinbart, die die Klassifikationsarten in einer Auflistung enthält:

$$\delta_{\text{Regelarten}} = (\text{Regelarten}, \{ \text{Constraint, Plausibilität, Bedingung, [optional]} \})$$

Über diese Domäne wird ein Attribut α_{Regelart} definiert.

$$\alpha_{\text{Regelart}} = (\text{Regelart}, \text{Regelarten}, \varepsilon)$$

Der Defaultwert sei für das Attribut α_{Regelart} als $\varepsilon = \text{Bedingung}$ definiert.

4.7 Selbstbeschreibungsmerkmale von Pliable Objects

Die Betrachtung einer Rule schloss mit der Definition eines separaten Attributes, welches Aufschluss über die Erfüllbarkeitsansprüche einer Regel gibt. Es ergibt sich die Frage, ob dieses Attribut der Rule hinzugefügt werden muss. In diesem Fall wäre die Definition der Rule zu überarbeiten, da das neue Attribut noch nicht Teil der Definition ist.

Bevor diese Frage zu beantworten wäre, soll die Aufmerksamkeit auf die Tatsache gelenkt werden, dass im Rahmen der Regelbetrachtung ein neues Attribut erzeugt wurde, welches den Ansprüchen der Attributdefinition genügt. Daraus folgt, dass ein Zusammenhang zwischen Attributen und Regeln existiert. Diesen gilt es näher zu erörtern. Die Erkenntnisse der Erörterung werden zeigen, dass das Modell der Pliable Objects eine Dynamik bereitstellt, die derart nützlich ist, dass eine dynamische Anpassbarkeit von Informationssystemen ermöglicht wird.

4.7.1 Sondereigenschaften

Ein Attribut ist gemäß Definition 4.5 durch Name bzw. Benennung, Domäne und Wert gegeben, eine Aktion gemäß Definition 4.6 durch Benennung, Aktionsattribute und Aktionsliste, und eine Regel gemäß Definition 4.7 durch Benennung, Restriktionen und Konklusionen. In gewisser Weise sind diese sieben Eigenschaften – in einem speziellen Sinn – Attribute. Sie sind quasi Instanzen eines (Pliable) Objektes Attribut. Ferner bilden sie eine Attributmenge, die Objekten zugeordnet sind. Diese Objekte sind, bei abstrakter Interpretation, die Objekte Attribut, Aktion und Regel. Im Folgenden wird geprüft, inwieweit diese Überlegung Bestand hat.

Benennung $Name v_\alpha$

Die Benennung $Name v_\alpha$ kann als Attribut α_{NAME} betrachtet werden. Die Attributbezeichnung dieses speziellen Attributs ist die Bezeichnung $NAME$ selbst. Bei der Bezeichnung handelt es sich um eine Zeichenkette, die einer Domäne

$$\delta_{\text{Zeichenkette}} = (\text{Zeichenketten}, \{ \ll\text{alle Zeichenketten}\gg \})$$

zugeordnet werden kann. Bei dem Wert, also der Attributbezeichnung, die dem Attribut α_{NAME} zugewiesen wird, handelt es sich um eine Zeichenkette $\omega_{\text{Zeichenkette}}$, die ebenfalls ein Element aus der Menge aller möglichen Zeichenketten, der Domäne $\delta_{\text{Zeichenkette}}$, ist.

Die Betrachtung des *Namens* sollte abstrakt und wertfrei sein, d.h. ein Name ist nicht per Default festgelegt. Andernfalls wären keine neuen Namen/Bezeichnungen generierbar. Um die Wertfreiheit zu gewährleisten, wird die *leere Zeichenkette* als Defaultwert vereinbart. Es wird vorausgesetzt, dass die *leere Zeichenkette* $\in \delta_{\text{Zeichenkette}}$ ist und dass das Attribut α_{NAME} grundsätzlich als Voreinstellung die Zuweisung der leeren Zeichenkette besitzt.

Die Benennung $Name v_\alpha$ kann als ein Attribut ausgedrückt werden gemäß:

$$v_\alpha = \alpha_{NAME} = (NAME, \text{Zeichenketten}, \varepsilon)$$

Beispiele für Instanzen von Namen/Bezeichnungen als Attribute wären:

$$\begin{aligned} &(NAME, \text{Zeichenketten}, \text{„Geschlecht“}) \\ &(NAME, \text{Zeichenketten}, \text{„Herzinfarkt“}) \end{aligned}$$

Domäne δ_τ

Die *Domäne* δ_τ kann als Attribut $\alpha_{DOMÄNE}$ aufgefasst werden, wobei die Attributbezeichnung dieses speziellen Attributs selbst die Bezeichnung $DOMÄNE$ trägt.

Gemäß der Erläuterung von Definition 4.5 ist eine Domäne ein Tupel (τ, Ω) mit $\tau \in \mathbf{T}$, der Typbezeichnungsmenge und $\Omega \in \mathbf{W}$, der Wertemengenmenge. Aufgrund der Bijektivität der Relation $\Delta \subseteq \mathbf{T} \times \mathbf{W}$ sind die Typbezeichnungen und Wertemengen eindeutig miteinander verknüpft, so dass für die Domäne des Attributs $\alpha_{DOMÄNE}$ entweder die Typbezeichnungsmenge oder die Wertemengenmenge der Relation Δ verwendet werden kann.

Der Wert des Attributes $\alpha_{DOMÄNE}$ kennzeichnet allgemein einen Wertebereich. Dieser kann durch eine konkrete Wertemenge oder dessen Typbezeichnung angegeben werden. Da diese ebenfalls nicht im Vorfeld feststehen darf, sei auch hier die Angabe der leeren Menge bzw. der leeren Typbezeichnung ε erlaubt.

Die *Domäne* δ_τ kann als Attribut $\alpha_{DOMÄNE}$ ausgedrückt werden gemäß:

$$\begin{aligned} \delta_\tau &= \alpha_{DOMÄNE} = (DOMÄNE, \text{Typbezeichnungsmenge}, \varepsilon) \\ &\quad \text{oder} \\ \delta_\tau &= \alpha_{DOMÄNE} = (DOMÄNE, \text{Wertemengenmenge}, \varepsilon) \end{aligned}$$

Beispiele für Instanzen von Domänen als Attribute wären:

(**DOMÄNE**, Typbezeichnungsmenge, „Geschlechtsarten“)
(**DOMÄNE**, Typbezeichnungsmenge, „Herzinfarktsarten“)

oder alternativ

(**DOMÄNE**, Wertemengenmenge, {männlich, weiblich, intersexuell, ε })
(**DOMÄNE**, Wertemengenmenge, {n.bekannt, <3 Monate, <6 Monate, >6 Monate, ε })

Wert ω_α

Der Wert ω_α kann als Attribut α_{WERT} betrachtet werden, wobei dieses spezielle Attribut die Bezeichnung **WERT** trägt. Das Attribut α_{WERT} soll selbst wiederum einen Wert aufnehmen. Dieser aufzunehmende Wert ist Element einer Wertmenge. Folglich stellt diese Wertemenge die Domäne des Attributes α_{WERT} .

Der Wert des Attributes α_{WERT} ist ein tatsächlicher Wert aus der Wertemenge, die die Domäne von α_{WERT} kennzeichnet. Da der Wert im Vorfeld nicht feststehen darf, sei auch hier die Angabe des leeren Wertes ε erlaubt.

Das spezielle Attribut *Wert* kann ausgedrückt werden durch

$$\omega_\alpha = \alpha_{\text{WERT}} = (\mathbf{WERT}, \Omega_\tau, \varepsilon)$$

Beispiele für Instanzen von Wert als Attribute wären:

(**WERT**, {männlich, weiblich, intersexuell, ε }, ε)
(**WERT**, {n.bekannt, < 3 Monate, < 6 Monate, > 6 Monate, ε }, ε)

Aktionsattribute A_χ

Die *Aktionsattribute* A_χ werden als Attribut $\alpha_{\text{AKTIONSATTRIBUTE}}$ betrachtet. Dieses spezielle Attribut trägt selbst die Bezeichnung *Aktionsattribute*. Das Attribut soll selbst eine Menge von Attributen aufnehmen. Diese Attributsmenge ist dabei eine Teilmenge der Menge aller Attribute. Um alle möglichen Attributteilungen als Domäne zur Verfügung zu stellen, muss die Domäne des Attributes $\alpha_{\text{AKTIONSATTRIBUTE}}$ die Potenzmenge $P(A)$ sein, wobei A die Menge aller Attribute bezeichnet. Der Wert des Attributes sei per Default die leere Menge. Damit kann das spezielle Attribut *Aktionsattribute* ausgedrückt werden durch

$$A_\chi = \alpha_{\text{AKTIONSATTRIBUTE}} = (\mathbf{Aktionsattribute}, P(A), \emptyset)$$

Aktionsliste Σ_χ

Die *Aktionsliste* Σ_χ kann als Attribut $\alpha_{\text{AKTIONSLISTE}}$ realisiert werden. Dieses spezielle Attribut trägt selbst die Bezeichnung ***Aktionsliste***. Das Attribut soll selbst eine Menge von Aktionen bzw. Anweisungen enthalten. Diese Aktionsmenge ist dabei eine Teilmenge der Menge aller Aktionen. Um alle möglichen Aktionsteilmengen als Domäne zur Verfügung zu stellen, muss die Domäne des Attributes $\alpha_{\text{AKTIONSLISTE}}$ die Potenzmenge $P(X)$ sein, wobei X die Menge aller Aktionen bezeichnet. Der Wert des Attributes ist per Default die leere Menge. Damit kann das spezielle Attribut *Aktionsliste* ausgedrückt werden durch

$$\Sigma_\chi = \alpha_{\text{AKTIONSLISTE}} = (\mathbf{Aktionsliste}, P(X), \emptyset)$$

Menge der Restriktionen B_ρ

Die *Menge der Restriktionen* B_ρ wird als Attribut $\alpha_{\text{RESTRIKTIONEN}}$ betrachtet. Dieses spezielle Attribut trägt selbst die Bezeichnung ***Restriktionen***. Das Attribut enthält selbst eine Menge von Bedingungen. Diese Bedingungsmenge ist dabei eine Teilmenge der Bedingungen, die auf den Attributen definiert werden können. Um alle möglichen Bedingungs Mengen als Domäne zur Verfügung zu stellen, muss die Domäne des Attributes $\alpha_{\text{RESTRIKTIONEN}}$ die Potenzmenge $P(B)$ sein, wobei B die Menge aller Bedingungen bezeichnet. Der Wert des Attributes ist per Default die leere Menge. Das spezielle Attribut *Restriktionen* kann ausgedrückt werden durch

$$B_\rho = \alpha_{\text{RESTRIKTIONEN}} = (\mathbf{Restriktionen}, P(B), \emptyset)$$

Menge der Konklusionen K_ρ

Die *Menge der Konklusionen* K_ρ wird als Attribut $\alpha_{\text{KONKLUSIONEN}}$ betrachtet. Dieses spezielle Attribut trägt selbst die Bezeichnung ***Konklusionen***. Das Attribut enthält eine Menge von Aktionen. Diese Aktionsmenge ist analog wie beim Attribut $\alpha_{\text{AKTIONSLISTE}}$ eine Teilmenge der Aktionen, die auf den Attributen arbeiten. Um alle möglichen Aktionsmengen erneut als Domäne zur Verfügung zu stellen, muss die Domäne des Attributes $\alpha_{\text{KONKLUSIONEN}}$ die Potenzmenge $P(X)$ sein, wobei X die Menge aller Aktionen bezeichnet. Der Wert des Attributes ist per Default die leere Menge. Das spezielle Attribut *Konklusionen* kann somit ausgedrückt werden durch

$$K_\rho = \alpha_{\text{KONKLUSIONEN}} = (\mathbf{Konklusionen}, P(X), \emptyset)$$

4.7.2 Methoden

Die obigen Betrachtungen definieren eine Menge von Attributen. Alle Attribute verfügen sowohl über feste als auch über variable Angaben. Um die Variabilität von Angaben erlauben zu können, werden im Folgenden Methoden definiert, die auf diesen Attributen arbeiten und zu Veränderungen von deren Werten führen können.

Gegeben sei ein beliebiges Attribut $\alpha = (v_\alpha, \delta_\tau, \omega_\alpha)$. Um die Informationen zu diesem Attribut, d.h. seinen Namen, seine Domäne und seinen Wert, auslesen zu können, werden folgende Funktionen vereinbart.

Belegungsmethode

Der Wert oder die Belegung²⁰ des Attributs α wird bestimmt durch die Funktion

$$\mathbf{Wert\ W}(\alpha) = \omega_\alpha$$

An dieser Stelle soll mit ω_α der tatsächlich und aktuell hinterlegte Wert des Attributes ausgegeben werden. Nach der Vereinbarung der Sondereigenschaften kann ω_α auch als Attribut aufgefasst werden gemäß

$$\omega_\alpha = \alpha_{\text{WERT}} = (\mathbf{WERT}, \Omega_\tau, \omega_\alpha).$$

Bei der Betrachtung des Modells der Pliable Objects werden temporale Konzepte außer Acht gelassen, um eine Konzentration auf das Modell an sich zu ermöglichen. Die Erweiterung um temporale Konzepte ist möglich, muss jedoch im Rahmen einer weiterführenden Forschung vorgenommen werden. Im Folgenden wird auf eine Berücksichtigung zeitlicher Abhängigkeiten verzichtet.

Domänenmethode

Die Domäne des Attributs α wird durch die Domänenfunktion bestimmt:

$$\mathbf{Domäne\ D}(\alpha) = \delta_\tau$$

Auch für die Domänenfunktion sei mit δ_τ der tatsächlich hinterlegte Typ des Attributes bezeichnet. Nach obigen Überlegungen kann δ_τ ebenfalls als Attribut aufgefasst werden gemäß

$$\delta_\tau = \alpha_{\text{DOMÄNE}} = (\mathbf{DOMÄNE}, \text{Typbezeichnungsmenge}, \delta_\tau).$$

Benennungsmethode

Der Bezeichner des Attributs α wird durch die Bezeichnungsfunktion ausgelesen:

$$\mathbf{Name\ N}(\alpha) = v_\alpha$$

²⁰ Eine **Belegung** (engl. "valuation") für eine Variablenmenge V ist eine Funktion ($V \rightarrow D$), die allen System-Variablen ($v \in V$) entsprechende Werte aus dem zugehörigen Wertebereich D (engl. "domain") zuordnet. Der Wertebereich D beschreibt dabei die möglichen Werteausprägungen der Variable(n) aus V .

Auch für die Benennungsfunktion sei mit v_α der tatsächlich hinterlegte Typ des Attributes bezeichnet. Erneut kann auch v_α als Attribut aufgefasst werden gemäß

$$v_\alpha = \alpha_{\text{NAME}} = (\mathbf{NAME}, \text{Zeichenketten}, v_\alpha).$$

Das Ziel der Belegungsmethode ist die Wertausgabe, während dieses bei der Domänenmethode die Domänenausgabe und bei der Benennungsmethode die Namensausgabe eines Attributes ist. In allen drei Fällen kann die jeweilige Ausgabe selbst wieder als Attribut aufgefasst werden. Die jeweils beabsichtigte Ausgabe entspricht der Wertbestimmung des jeweils gelieferten Attributes. Damit ist nach der jeweilig ausgeführten Methode erneut die Belegungsmethode auf die erhaltenen Attribute aufzurufen, um die gewünschten Informationen zu erhalten:

$$\begin{aligned} \mathbf{W}(\alpha_{\text{WERT}}) &= \alpha_{\text{WERT}} = (\mathbf{WERT}, \Omega_\tau, \omega_\alpha) \\ \mathbf{W}(\alpha_{\text{DOMÄNE}}) &= \alpha_{\text{WERT}} = (\mathbf{WERT}, \Omega_\tau, \delta_\tau). \\ \mathbf{W}(\alpha_{\text{NAME}}) &= \alpha_{\text{WERT}} = (\mathbf{WERT}, \Omega_\tau, v_\alpha). \end{aligned}$$

Im Endeffekt führt die Wertbestimmung auf einen rekursiven Aufruf auf sich selbst, was wiederum als Endlosschleife zu werten ist. Diese gilt es zu vermeiden, weshalb für die Belegungsmethode vereinbart wird, dass sie den Charakter der Endlosschleife erkennt und vermeidet, und einfach anstelle des Attributs α_{WERT} den für das beliebige Attribut α eingetragenen Wert liefert. Der Umstand der Endlosschleife kann eventuell als ein Indiz für die Implementierung genutzt werden (siehe Kapitel 5).

Aktionsattributmethode

Die Menge der Attribute A_χ , auf denen eine Aktion χ arbeitet, liefert die Aktionsattributmethode:

$$\mathbf{Aktionsattribut} \mathbf{AA}(\chi) = A_\chi$$

Auch wenn die Menge der Attribute A_χ als eigenständiges Attribut $A_\chi = \alpha_{\text{AKTIONSATTRIBUT}} = (\mathbf{Aktionsattribut}, P(A), \emptyset)$ aufgefasst werden kann, soll die Aktionsattributmethode die Menge der Attribute liefern, auf denen die Aktion arbeitet.

Aktionslistemethode

Die Liste der Anweisungen Σ_χ der Aktion liefert die Aktionslistemethode:

$$\mathbf{Aktionsliste} \mathbf{AL}(\chi) = \Sigma_\chi$$

Die Liste der Anweisungen kann gemäß $\alpha_{\text{AKTIONSLISTE}} = (\mathbf{Aktionsliste}, P(X), \emptyset)$ ebenfalls als eigenständiges Attribut aufgefasst werden. Im Rahmen der Aktionslistemethode wird vereinbart, dass die Methode die Aktionsliste liefert und nicht das Aktionsliste-Attribut.

Restriktionsmethode

Die Restriktionsmethode liefert die Menge der Restriktionen B_ρ einer Regel.

$$\mathbf{Restriktionen} \mathbf{R}(\rho) = B_\rho$$

Auch für die Restriktionsmethode sei vereinbart, dass die Methode die Menge der Restriktionen liefert, anstelle des Attributs $\alpha_{\text{RESTRIKTIONEN}} = (\mathbf{Restriktionen}, P(B), \emptyset)$ der Regel.

Konklusionsmethode

Die Konklusionsmethode liefert die Menge der Konklusionen K_ρ einer Regel.

$$\mathbf{Konklusionen} \mathbf{K}(\rho) = K_\rho$$

Analog zu den vorherigen Methoden sei auch für diese Methode vereinbart, dass sie die Konklusionen der Regel liefert und nicht das Attribut $K_\rho = \alpha_{\text{KONKLUSIONEN}} = (\mathbf{Konklusionen}, P(X), \emptyset)$.

4.7.3 Bedingungen

Bei einer zusammenhängenden Betrachtung von oben vorgestellten Attributen und auf ihnen arbeitenden Methoden sind inhärente Bedingungen einzuhalten, die sowohl die Attribute als auch die Methoden betreffen:

- Eine *Namensregel* sichert die Existenz einer Bezeichnung einer Eigenschaft. Andernfalls wäre eine Diskussion über konkrete Eigenschaften sinnlos, d.h. es gilt $\mathbf{N}(\alpha) \neq \text{,,}$.
- Ein *Wertereg* schränkt die möglichen Werte einer Eigenschaft auf den Wertebereich ein, den die Domäne des Attributs stellt, d.h. es gilt $\mathbf{W}(\alpha) \in \mathbf{D}(\alpha) = \delta_\tau$.

Vergleicht man beide Bedingungen, können unter beiden Regeln Zusicherungen (Constraints) verstanden werden. Bei der Namensregel ist dies der Fall, bei der Wertereg hingegen ist zu berücksichtigen, dass diese Regel auch verletzt werden darf. Im Hinblick auf die Dokumentation von Informationen in Attributen, die noch nicht für diese Information ausgelegt sind, muss diese Regel verletzt werden dürfen. Somit spiegelt die Wertereg eine Plausibilität wider.

4.7.4 Spezifikation von Sonder-Pliable-Objects

Aus den obigen Attributen, Methoden und Bedingungen können separate Mengen gebildet werden. Bestimmte Mengen können miteinander in einer Art und Weise verknüpft werden, dass diese Verknüpfungen spezielle Pliable Objects repräsentieren. Folgende Mengen können gebildet werden:

- eine Attributmengung A, gegeben durch $A_{\text{ATTRIBUT}} = \{\alpha_{\text{NAME}}, \alpha_{\text{DOMÄNE}}, \alpha_{\text{WERT}}\}$,
- eine Attributmengung A, gegeben durch $A_{\text{ACTION}} = \{\alpha_{\text{NAME}}, \alpha_{\text{AKTIONSATTRIBUTE}}, \alpha_{\text{AKTIONSLISTE}}\}$,
- eine Attributmengung A, gegeben durch $A_{\text{RULE}} = \{\alpha_{\text{NAME}}, \alpha_{\text{RESTRIKTIONEN}}, \alpha_{\text{KONKLUSIONEN}}\}$,
- eine Aktionenmenge X, gegeben durch $X_{\text{ATTRIBUT}} = \{\text{Name, Domäne, Wert}\}$,
- eine Aktionenmenge X, gegeben durch $X_{\text{ACTION}} = \{\text{Name, Aktionsattribute, Aktionsliste}\}$,
- eine Aktionenmenge X, gegeben durch $X_{\text{RULE}} = \{\text{Name, Restriktionen, Konklusionen}\}$,
- eine Regelmengung P, gegeben durch $P_{\text{ATTRIBUT}} = \{\text{Namensregel, Wertregel}\}$ und
- zwei Regelmengungen, gegeben als $P_{\text{ACTION}} = \{\}$ und $P_{\text{RULE}} = \{\}$.

Durch die Verknüpfung je einer Attributmengung mit einer Aktions- und einer Regelmengung sind die drei Sonder-Pliable-Objects π_{ATTRIBUT} , π_{ACTION} und π_{RULE} spezifizierbar.

Sonder-Pliable-Object Attribut

Das Sonder-Pliable-Object Attribut π_{ATTRIBUT} kann wie folgt spezifiziert werden:

$$\pi_{\text{ATTRIBUT}} = (\gamma_{\text{ATTRIBUT}}, A_{\text{ATTRIBUT}}, X_{\text{ATTRIBUT}}, P_{\text{ATTRIBUT}})$$

Die Beschreibungsdomäne kann folgendermaßen angegeben werden:

$$\gamma_{\text{ATTRIBUT}} = (\text{Attribut}, \text{„Ein Attribut dient der Angabe einer Zustandsinformation.“})$$

Dieses Sonder-Pliable-Object beschreibt die Menge aller Attribute, d.h. ein konkretes Attribut ist eine Instanz dieses Pliable Objects π_{ATTRIBUT} .

Sonder-Pliable-Object Action

Analog kann ein Sonder-Pliable-Object Action π_{ACTION} spezifiziert werden:

$$\pi_{\text{ACTION}} = (\gamma_{\text{ACTION}}, A_{\text{ACTION}}, X_{\text{ACTION}}, P_{\text{ACTION}})$$

Die Beschreibungsdomäne wäre wie folgt definiert:

$$\gamma_{\text{ACTION}} = (\text{Action}, \text{„Eine Action spezifiziert einen Zustandswechsel.“})$$

Für eine Action muss ein spezielles Verhalten vereinbart werden. Die Methoden des Sonder-Pliable-Objects stellen den Zugriff auf die Menge von Attributen sicher, die im Attribut $\alpha_{\text{AKTIONSATTRIBUTE}}$ der Aktion aufgelistet sind. Diese Attributmenge stellt eine spezielle Domäne für die Aktion selbst dar. Die Aktion darf nur auf Attribute aus dieser Menge zugreifen und deren Zustand verändern. Eine Aktion ist einem Pliable Object zugewiesen und soll gemäß Definition auf den Attributen des Pliable Objects arbeiten. Daraus folgt, dass die Aktionsattribute der Aktion eine Teilmenge der Attribute des der Aktion zugeordneten Pliable Objects sind.

Soll nun die Aktion auf einem weiteren Pliable Object arbeiten, so muss sichergestellt werden, dass die Aktionsattribute der Aktion ebenfalls eine Teilmenge der Attribute dieses weiteren Pliable Objects sind.

Eine Aktion stellt einen Bestandteil des Verhaltens eines Pliable Objects π dar, dem die Aktion zugeordnet ist, d.h. die Aktion soll auf den Attributen von π arbeiten. Um dies zu gewährleisten, muss die Attributmenge eine Teilmenge der Attributmenge von π sein. Durch einen Vergleich, ob die Aktionsattribute auch dem Pliable Object zugeordnet sind, kann geprüft werden, ob die Aktion für das Verhalten des Pliable Objects π genutzt werden kann.

Die Prüfung kann derart vorgenommen werden, dass die Attribute von $\pi = (\Gamma, A, X, P)$ als eine Domäne $\delta_{\text{Attr}(\pi)}$ angesehen werden.

$$\delta_{\text{Attr}(\pi)} = A$$

Über diese Domäne wird die Potenzmenge gebildet und anschließend geprüft, ob die Aktionsattribute Teilmenge der Potenzmenge sind.

$$\text{IsPartSet} = \text{True} \Leftrightarrow W(\alpha_{\text{AKTIONSATTRIBUTE}}) \in \text{POWERSET}(\delta_{\text{Attr}(\pi)})$$

$$\text{IsPartSet} = \text{False} \Leftrightarrow W(\alpha_{\text{AKTIONSATTRIBUTE}}) \notin \text{POWERSET}(\delta_{\text{Attr}(\pi)})$$

Sonder-Pliable-Object Rule

Abschließend wird ein Sonder-Pliable-Object Rule π_{RULE} spezifiziert:

$$\pi_{\text{RULE}} = (\gamma_{\text{RULE}}, A_{\text{RULE}}, X_{\text{RULE}}, P_{\text{RULE}})$$

Die Beschreibungsdomäne wäre wie folgt definiert:

$$\gamma_{\text{RULE}} = (\text{Rule}, \text{„Eine Rule dient zur Steuerung von Zustandswechslern.“})$$

Beschreibungsdomänen der Sonder-Pliable-Objects

Die Sonder-Pliable-Objects Attribut, Action und Rule stellen abstrakte Sichten auf deren Instanzen dar, d.h. auf konkrete Attribute, Aktionen und Regeln. Durch das Verknüpfen von konkreten Attributen, Aktionen und Regeln werden weitere Pliable Objects erzeugt. Um diese weiteren Pliable Objects genauer zu spezifizieren, ist die Beschreibungsdomäne vorgesehen.

Eine Beschreibungsdomäne ist nach 4.2.4 eine Domäne, die aus einer Typbezeichnung als Fachausdruck und einer umgangssprachlichen Definition besteht. Beide Informationen werden auf folgende Attribute abgebildet:

- Der Fachausdruck als Typbezeichnung einer Beschreibungsdomäne entspricht dem Attribut *Name*:

$$\alpha_{\text{NAME}} = (\text{NAME}, \text{Zeichenketten}, \varepsilon)$$

- Die Definition kann ebenfalls als Attribut dargestellt werden:

$$\alpha_{\text{DEFINITION}} = (\text{DEFINITION}, \text{Zeichenketten}, \varepsilon)$$

Eine Beschreibungsdomäne γ ist somit gegeben als ein Tupel aus einem Fachausdruck und einer Definition:

$$\gamma_{\text{NAME}} = (\alpha_{\text{NAME}}, \alpha_{\text{DEFINITION}})$$

Wird die Beschreibungsdomäne mittels einer Menge von *Attributs* A , einer Menge von *Actions* X und einer Menge von *Rules* P zu einem Pliable Object $\pi = (\Gamma, A, X, P)$ erweitert, so sollen die Attribute der Beschreibungsdomäne mit der Menge der Attribute vereinigt werden, so dass ein Pliable Object angegeben werden kann als

$$\pi_{\text{NAME}} = (A_{\text{NAME}}, X, P) \text{ mit } A_{\text{NAME}} = A \cup \{\alpha_{\text{NAME}}, \alpha_{\text{DEFINITION}}\}$$

Ein spezielles Pliable Object wird formal durch die Tiefstellung des Namens der Beschreibungsdomäne an das Symbol π dargestellt.

4.7.5 Spezielle Pliable-Object-Klassifikation

Ein Pliable Object wird durch seine Attribute, Aktionen und Regeln beschrieben. Um konkrete Pliable Objects miteinander vergleichen zu können, soll zwischen Schema-Pliable-Objects, schematisierten Pliable Objects und instanziierten Pliable Objects differenziert werden. Die Begriffe stellen einen semantischen Bezug zwischen Pliable Objects her und sind wie folgt definiert:

- Schema-Pliable-Object:

Ein Pliable Object heißt *Schema-Pliable-Object* π^S , wenn gilt, dass jedem Attribut aus der Attributmenge des Pliable Objects der Defaultwert der jeweiligen Attributdomäne zugewiesen ist. Die Attribute der Beschreibungsdomäne werden dabei nicht betrachtet:

$$\pi^S = (A, X, P) \Rightarrow \forall \alpha \in A \setminus \Gamma \mid \mathbf{D}(\alpha) = \delta_x \wedge \mathbf{W}(\alpha) = \varepsilon_x \wedge \varepsilon_x \in \delta_x$$

- Instanziiertes Pliable Object

Ein Pliable Object heißt *instanziiertes Pliable Object* π^I oder *Instanz* π^I , wenn jedem Attribut aus der Attributmengende nicht der Defaultwert zugewiesen ist:

$$\pi^I = (A, X, P) \Rightarrow \forall \alpha \in A \mid \mathbf{W}(\alpha) \neq \varepsilon$$

- Schematisiertes Pliable Object:

Ein Pliable Object heißt *schematisiertes Pliable Object* $\pi^{S>I}$, wenn gilt, dass mindestens ein Attribut aus der Attributmengende existiert, welchem der Defaultwert zugewiesen ist, und dass mindestens ein Attribut aus der Attributmengende existiert, welchem der Defaultwert nicht zugewiesen ist:

$$\pi^{S>I} = (A, X, P) \Rightarrow \exists \alpha_1 \in A \setminus \Gamma \mid \mathbf{W}(\alpha_1) = \varepsilon \wedge \exists \alpha_2 \in A \setminus \Gamma \mid \mathbf{W}(\alpha_2) \neq \varepsilon$$

Beispiel:

Gegeben sei die Klasse *Person* (vgl. Abbildung 23).

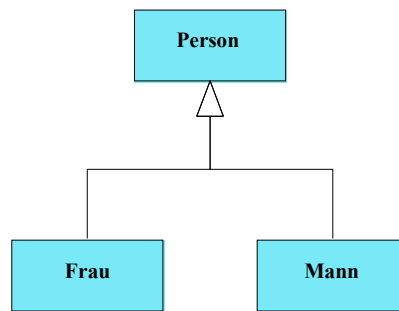


Abbildung 23: Simple Beispiel für Schema-, schematisierte und instanziierte Pliable Objects

Die Klasse besitze als Spezialisierungen die zwei Klassen *Frau* und *Mann*. Die Spezialisierungsklassen basieren auf einer vorgegebenen Wertbelegung eines Attributes *Geschlecht*. In einer möglichen Modellierung als Pliable Objects wäre die Klasse *Person* ein Schema-Pliable-Object, während die Klassen *Frau* und *Mann* schematisierte Pliable Objects wären:

$$\pi_{\text{PERSON}}^S = (\{ \alpha_{\text{NACHNAME}} = (\mathbf{NACHNAME}, \text{Zeichenketten}, \varepsilon), \alpha_{\text{VORNAME}} = (\mathbf{VORNAME}, \text{Zeichenketten}, \varepsilon), \alpha_{\text{GESCHLECHT}} = (\mathbf{GESCHLECHT}, \{\text{männlich}, \text{weiblich}, \text{intersexuell}\}, \varepsilon) \}, \emptyset, \emptyset)$$

$$\pi_{\text{FRAU}}^{S \rightarrow I} = (\{ \alpha_{\text{NACHNAME}} = (\text{NACHNAME}, \text{Zeichenketten}, \varepsilon), \\ \alpha_{\text{VORNAME}} = (\text{VORNAME}, \text{Zeichenketten}, \varepsilon), \\ \alpha_{\text{GESCHLECHT}} = (\text{GESCHLECHT}, \{\text{männlich}, \text{weiblich}, \text{intersexuell}\}, \text{weiblich}) \\ \}, \emptyset, \emptyset)$$

$$\pi_{\text{MANN}}^{S \rightarrow I} = (\{ \alpha_{\text{NACHNAME}} = (\text{NACHNAME}, \text{Zeichenketten}, \varepsilon), \\ \alpha_{\text{VORNAME}} = (\text{VORNAME}, \text{Zeichenketten}, \varepsilon), \\ \alpha_{\text{GESCHLECHT}} = (\text{GESCHLECHT}, \{\text{männlich}, \text{weiblich}, \text{intersexuell}\}, \text{männlich}) \\ \}, \emptyset, \emptyset)$$

Im Gegensatz zur Person würde beim konkreten Anlegen einer Instanz von Frau oder Mann das Geschlechtsattribut mit einem Wert vorbelegt.

Klassifikationszusammenhänge

Aus einem Schema-Pliable-Object wird eine Instanz gewonnen, indem sukzessive die Attribute des Schema-Pliable-Objects mit Werten belegt werden. Eine einzelne Belegung überführt das Object in ein schematisiertes Pliable Object, und sämtliche Belegungen führen zur Instanz.

Aus einem Schema-Pliable-Object können verschiedene Instanzen gewonnen werden; alle möglichen Instanzen werden dem Schemaobjekt als *zugehörig* gekennzeichnet. Grundsätzlich wird angenommen, dass ein Duplikat des Schemaobjektes angelegt wird, das in ein schematisiertes Pliable Object oder eine Instanz überführt wird.

Aus der bisherigen Betrachtung der Sonder-Pliable-Objects ergibt sich, dass die Sonder-Pliable-Objects Attribut, Action und Rule als Schema-Pliable-Objects betrachtet werden können. Die bislang abstrakte Sicht, in der sie gesehen wurden, ist in eine konkrete Betrachtung überführt worden. Instanzen der konkreten Sonder-Pliable-Objects, d.h. konkrete Attribute, Aktionen und Regeln, können aus ihnen, gemäß obiger Überführung, erstellt werden. Diese gewonnenen Instanzen können in neuen Pliable Objects verwendet werden, um zunächst Schema-Pliable-Objects zu spezifizieren. Diesen können erneut in Instanzen überführt werden, die als relevant bewertete Daten aufnehmen.

Die Generierung und Verknüpfung von und mit Pliable Objects geschieht dabei unter Verwendung der vorgestellten Methoden. Grundlegend neue Methoden werden erst benötigt, wenn spezielle Operationen umgesetzt werden sollen. Dabei werden die neuen, bislang unspezifizierten Operationen unter Verwendung der vorgestellten Methoden erstellt.

Damit ergibt sich eine theoretische Vielfalt an Möglichkeiten, die in einer Notation dokumentiert sein sollte. Daher wird zunächst eine Notation für Pliable Objects vorgestellt, bevor weitere semantische Begriffsdefinitionen spezifiziert werden.

4.8 Notationselement von Pliable Objects

Zur graphischen Darstellung von Pliable Objects wird das aus der UML bekannte Notationselement für eine Klasse verwendet. Um auch die Regeldefinitionen mit hinterlegen zu können, wird das Notationselement einer Klasse um ein weiteres Segment erweitert. Ein Pliable Object wird, wie in Abbildung 24 gezeigt, durch ein hochkant stehendes Rechteck notiert, welches vier horizontale Segmente enthält.

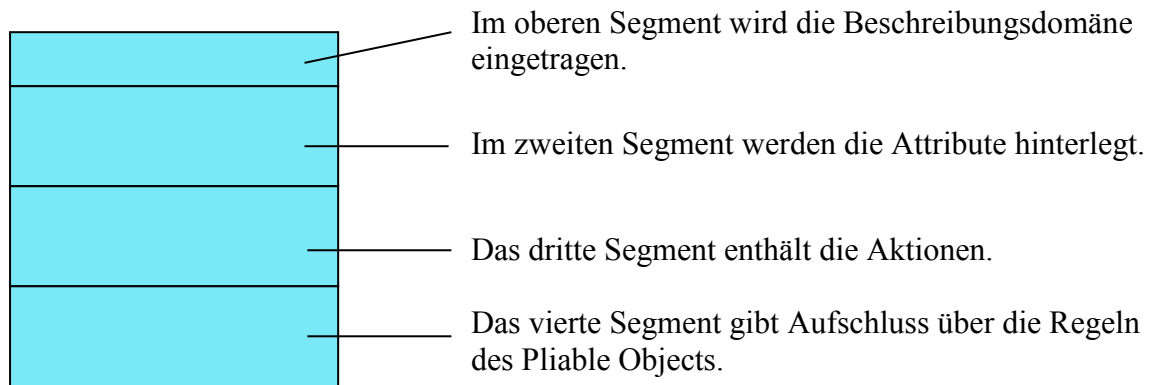


Abbildung 24: Graphisches Notationselement für Pliable Objects

4.8.1 Visualisierung der Sonder-Pliable-Objects

Gemäß der Spezifikation von Sonder-Pliable-Objects (vgl. 4.7.4) können Attribut, Action und Rule in einer Form notiert werden, wie sie Abbildung 25 zeigt.

Pliable-Object-Name: Attribut Pliable-Object-Definition: Ein Attribut dient zur Angabe einer Zustandsinformation.	Pliable-Object-Name: Action Pliable-Object-Definition: Eine Action ist die Definition eines Zustandswechsels.	Pliable-Object-Name: Rule Pliable-Object-Definition: Eine Rule dient der Steuerung von Zustandswechseln.
$\alpha_{\text{NAME}}: \varepsilon$ $\alpha_{\text{DOMÄNE}}: \varepsilon$ $\alpha_{\text{WERT}}: \varepsilon$	$\alpha_{\text{NAME}}: \varepsilon$ $\alpha_{\text{AKTIONSATTRIBUTE}}: \varepsilon$ $\alpha_{\text{AKTIONSLISTE}}: \varepsilon$	$\alpha_{\text{NAME}}: \varepsilon$ $\alpha_{\text{RESTRIKTIONEN}}: \varepsilon$ $\alpha_{\text{KONKLUSIONEN}}: \varepsilon$
Name() Domäne() Wert()	Name() Aktionsattribute() Aktionsliste()	Name() Restriktionen() Konklusionen()
Namensregel Werteregeln	Namensregel	Namensregel

Abbildung 25: Visualisierung der Sonder-Pliable-Objects

Das Pliable Object Attribut besitzt allgemein eine dreielementige Attributmenge, eine dreielementige Aktionsmenge mit den zugehörigen Zugriffsmethoden auf die Attributwerte und eine zweielementige Regelmenge. Das Pliable Object Action verfügt ebenfalls über eine dreielementige Attributmenge sowie der zugehörigen Zugriffsmethoden auf diese Attribute und eine einzelne Regel. Das Pliable Object Rule besitzt einen ähnlichen Aufbau wie die Action, da es ebenfalls über eine dreielementige Attributmenge sowie der zugehörigen Zugriffsmethoden auf diese Attribute und eine einzelne Regel verfügt.

4.8.2 Visualisierung der Pliable-Object-Klassifikationen

Die Visualisierungsform für Pliable Objects kann zusätzlich dazu verwendet werden, die einzelnen Klassifikationsformen (vgl. 4.7.5) von Pliable Objects zu visualisieren.

Pliable-Object-Name: Patient Pliable-Object-Definition: Patient dient zur Aufnahme patientenspezifischer Daten, die für die medizinische Betreuung relevant sind.	Pliable-Object-Name: Patientin Pliable-Object-Definition: Patientin dient zur Aufnahme patientenspezifischer Daten, die für die medizinische Betreuung eines weiblichen Patienten relevant sind.
Patientennachname: ε Patientenvorname: ε Geschlecht: ε	Patientennachname: ε Patientenvorname: ε Geschlecht: weiblich
	Constraint: Geschlecht = weiblich

Abbildung 26: Visualisierungsbeispiel eines Schema- und eines schematisierten Pliable Objects

Abbildung 26 enthält ein Beispiel für das Schema-Pliable-Object *Patient* sowie eines für das schematisierte Pliable Object *Patientin*. Beide verfügen über die gleiche Anzahl an Attributen. Während im Schema-Pliable-Object *Patient* alle Attribute auf den Defaultwert ε gesetzt sind, ist im schematisierten Pliable Object *Patientin* das Attribut *Geschlecht* mit dem Wert *weiblich* vorbelegt. Ferner existiert im schematisierten Pliable Object die Regel, dass das Attribut *Geschlecht* ausschließlich den Wert *weiblich* annehmen darf. Beide Objektvisualisierungen erfüllen somit die Bedingungen für ein Schema-Pliable-Object bzw. ein schematisiertes Pliable Object.

Pliable-Object-Name: Patient Pliable-Object-Definition: Patient dient zur Aufnahme patientenspezifischer Daten, die für die medizinische Betreuung relevant sind.
Patientennachname: Mustermann Patientenvorname: Erwin Geschlecht: männlich

Abbildung 27: Visualisierungsbeispiel eines instanziierten Pliable Objects

Abbildung 27 zeigt die Visualisierung einer Instanz bzw. eines instanziierten Pliable Objects. Alle Attribute des gezeigten Objects sind mit einem Wert belegt.

4.9 Semantische Begriffsdefinitionen

Die Differenzierung zwischen Schema-Pliable-Objects, schematisierten Pliable Objects und instanziierten Pliable Objects ermöglicht den Vergleich von konkreten Pliable Objects miteinander. Um die Vergleiche zu stützen und deren Aussagekraft zu erhöhen, werden im Folgenden semantische Begriffe definiert, die das Vorliegen bestimmter Sachverhalte bei Pliable Objects beschreiben.

4.9.1 Charakterisierende Definition

Ein Pliable Object wird im Wesentlichen durch seine Attribute charakterisiert. Im Detail wird es häufig derart formuliert, dass ein Attribut α ein Attribut eines Pliable Objects π ist. Bezogen auf Pliable Objects soll dieser Sachverhalt begrifflich durch eine spezielle Formulierung zum Ausdruck gebracht werden:

- Aus Attribut-Sicht soll die Formulierung lauten, dass das Attribut α dem Pliable Object π *zu eigen* ist.
- Aus Pliable-Object-Sicht lautet die Formulierung, dass das Pliable Object π das Attribut α *besitzt*.
- Formal sei solch ein Sachverhalt durch $\alpha \in \pi$ ausgedrückt.

So ist im Beispiel aus Abbildung 26 das Attribut *Patientennachname* dem Pliable Object *Patient* zu eigen bzw. das Pliable Object *Patient* besitzt das Attribut *Patientennachname*.

Die Begriffe *zu eigen* und *besitzen* gelten analog für das Ausdrücken der Sachverhalte hinsichtlich einer Action χ und einer Rule ρ .

4.9.2 Äquivalenzrelationale Definitionen

Aus einer Instanz kann das zugehörige Schemaobjekt gewonnen werden, indem das Instanzobjekt dupliziert und für alle Attribute des Duplikats deren Werte auf den Defaultwert gesetzt werden. Nun können Schemaobjekte und Instanzen miteinander oder auch untereinander verglichen werden. Zur Präzision bestimmter Sachverhalte, in denen zwei Objekte in gewisser Hinsicht als gleich anzusehen sind, werden äquivalenzrelationale Begriffe definiert.

Attribut-spezifische Äquivalenzen

Zwei Attribute α_1 , α_2 heißen

- *wertgleich*, wenn gilt: $\mathbf{W}(\alpha_1) = \mathbf{W}(\alpha_2)$
- *äquivalent*, wenn gilt: $\mathbf{Dom}(\alpha_1) = \mathbf{Dom}(\alpha_2)$
- *namensgleich*, wenn gilt: $\mathbf{N}(\alpha_1) = \mathbf{N}(\alpha_2)$

Aufbauend auf diesen Basisäquivalenzen werden ergänzend spezielle Sachverhalte präzisiert:

Zwei Attribute α_1 , α_2 heißen

- *gleich*, wenn die Attribute *namensgleich*, *äquivalent* und *wertgleich* sind,
- *synonym*, wenn die Attribute *äquivalent* und *wertgleich* sind,
- *schematisch gleich*, wenn die Attribute *äquivalent* und *namensgleich* sind.

Häufig werden Pliable Objects hinsichtlich ihrer Attributmengen miteinander verglichen. Zur Vereinfachung solcher Vergleiche werden die attribut-spezifischen Definitionen wie folgt auf Attributmengen erweitert:

Zwei Attributmengen A_1 , A_2 , die die gleiche Anzahl an Attributen besitzen, heißen

- *gleich*, wenn $\forall \alpha_1 \in A_1 \mid \exists \alpha_2 \in A_2$ mit α_1 , α_2 gleich sind,
- *synonym*, wenn $\forall \alpha_1 \in A_1 \mid \exists \alpha_2 \in A_2$ mit α_1 , α_2 synonym sind,
- *schematisch gleich*, wenn $\forall \alpha_1 \in A_1 \mid \exists \alpha_2 \in A_2$ mit α_1 , α_2 schematisch gleich sind.

Aktions-spezifische Definitionen

Für Aktionen ist vereinbart, dass sie einerseits einem Pliable Object zu eigen sind, andererseits auf einer Teilmenge der Attribute dieses Pliable Objects arbeiten. Bezogen auf die jeweiligen Attributmengen zweier Aktionen werden äquivalente Begriffe für Aktionen selbst definiert.

Zwei Aktionen χ_1 , χ_2 heißen

- *gleich*, wenn die Attributmengen der Aktionen χ_1 , χ_2 gleich sind,
- *synonym*, wenn die Attributmengen der Aktionen χ_1 , χ_2 synonym sind,
- *schematisch gleich*, wenn die Attributmengen der Aktionen χ_1 , χ_2 schematisch gleich sind.

Zwei Aktionsmengen X_1 , X_2 , die die gleiche Anzahl an Aktionen besitzen, heißen

- *gleich*, wenn $\forall \chi_1 \in X_1 \mid \exists \chi_2 \in X_2$ mit χ_1 , χ_2 gleich sind,
- *synonym*, wenn $\forall \chi_1 \in X_1 \mid \exists \chi_2 \in X_2$ mit χ_1 , χ_2 synonym sind,
- *schematisch gleich*, wenn $\forall \chi_1 \in X_1 \mid \exists \chi_2 \in X_2$ mit χ_1 , χ_2 schematisch gleich sind.

Regel-spezifische Definitionen

Analog zur Erweiterung äquivalenter Begriffe auf Aktionen wird im Folgenden eine Erweiterung auf Regeln vorgenommen.

Zwei Regeln ρ_1, ρ_2 heißen

- *gleich*, wenn die Attributmengen der Regeln ρ_1, ρ_2 gleich sind,
- *synonym*, wenn die Attributmengen der Regeln ρ_1, ρ_2 synonym sind,
- *schematisch gleich*, wenn die Attributmengen der Regeln ρ_1, ρ_2 schematisch gleich sind.

Zwei Regelmengen P_1, P_2 , die die gleiche Anzahl an Regeln besitzen, heißen

- *gleich*, wenn $\forall \rho_1 \in P_1 \mid \exists \rho_2 \in P_2$ mit ρ_1, ρ_2 gleich sind,
- *synonym*, wenn $\forall \rho_1 \in P_1 \mid \exists \rho_2 \in P_2$ mit ρ_1, ρ_2 synonym sind,
- *schematisch gleich*, wenn $\forall \rho_1 \in P_1 \mid \exists \rho_2 \in P_2$ mit ρ_1, ρ_2 schematisch gleich sind.

Pliable-Object-spezifische Definitionen

Abschließend werden die auf den Begriffsdefinitionen für Attribute, Aktionen und Regeln äquivalente Begriffe auf Pliable Objects selbst definiert:

Zwei Pliable Objects π_1, π_2 heißen

- *gleich*, wenn die jeweiligen Attribut-, Aktions- und Regelmengen gleich sind,
- *synonym*, wenn die jeweiligen Attribut-, Aktions- und Regelmengen synonym sind,
- *schematisch gleich*, wenn die jeweiligen Attribut-, Aktions- und Regelmengen schematisch gleich sind.

Formale Repräsentation

Zur formalen Repräsentation und zur besseren Notation wird für die Gleichheit, die Synonymität und schematische Gleichheit das Gleichheitssymbol verwendet. Eine spezielle tiefgestellte Kennzeichnung dient zur Unterscheidung:

- Die Gleichheit wird durch das Symbol $=$ ausgedrückt,
- die Synonymität durch das Symbol $=_a$ und
- die schematische Gleichheit durch das Symbol $=_s$.

Falls die Gleichheit, die Synonymität oder die Schematische Gleichheit nicht vorliegen, wird das Ungleichheitszeichen \neq mit der tiefgestellten Kennzeichnung verwendet.

4.9.3 Verallgemeinerung und Spezialisierung

Bislang wurde bei Pliable Objects nur der Zusammenhang zwischen Schema-Pliable-Object und Instanz-Pliable-Object betrachtet. Es existieren weitere Zusammenhänge zwischen zwei Pliable Objects, die mengentheoretisch diskutiert werden. Die weitere Betrachtung geschieht auf zwei Pliable Objects π_1, π_2 , die nicht schematisch gleich sind, d.h. $\pi_1 \neq_s \pi_2$.

Attributebene

Die Menge $A_1 \in \pi_1$ sei die Menge der Attribute des Pliable Objects π_1 und die Menge $A_2 \in \pi_2$ die Menge der Attribute von π_2 . Über beide Mengen kann eine Schnittmenge gebildet werden, die drei Ausprägungen besitzt:

1. die Schnittmenge $\Delta_=$ der gleichen Attribute:

$$\Delta_{\alpha=} = A_1 \cap A_2 \text{ mit } \Delta_{\alpha=} = \{ \alpha_- \mid \exists \alpha_1 \in A_1 \wedge \exists \alpha_2 \in A_2 \text{ mit } \alpha_1 =_ = \alpha_2 \}$$

2. die Schnittmenge $\Delta_{\bar{a}}$ der synonymen Attribute:

$$\Delta_{\alpha\bar{a}} = A_1 \cap A_2 \text{ mit } \Delta_{\alpha\bar{a}} = \{ \alpha_{\bar{a}} \mid \exists \alpha_1 \in A_1 \wedge \exists \alpha_2 \in A_2 \text{ mit } \alpha_1 =_{\bar{a}} \alpha_2 \}$$

3. die Schnittmenge Δ_s der schematisch gleichen Attribute:

$$\Delta_{\alpha s} = A_1 \cap A_2 \text{ mit } \Delta_{\alpha s} = \{ \alpha_s \mid \exists \alpha_1 \in A_1 \wedge \exists \alpha_2 \in A_2 \text{ mit } \alpha_1 =_s \alpha_2 \}$$

Allgemein kann angenommen werden, dass die Attributmengen in keinem Zusammenhang stehen, wenn die Mengen A_1 und A_2 disjunkt zueinander sind, d.h. die drei Schnittmengen ergeben sich aus der leeren Menge.

Wenn hingegen die beiden Mengen nicht disjunkt sind, besteht ein Zusammenhang zwischen ihnen, der durch folgende Definitionen genauer spezifiziert wird:

- π_1 ist eine schematische α -Generalisierung von π_2 , wenn die schematische Gleichheit $\Delta_{\alpha s} =_s A_1$ vorliegt.
- π_1 ist eine schematische α -Spezialisierung von π_2 , wenn die schematische Gleichheit $\Delta_{\alpha s} =_s A_2$ vorliegt.
- π_1 und π_2 sind schematische α -Geschwister mit dem schematisch gleichen α -Kern $\Delta_{\alpha s}$, wenn zwischen π_1 und π_2 weder eine schematische α -Generalisierung noch eine schematische α -Spezialisierung vorliegt.
- π_1 ist eine synonyme α -Generalisierung von π_2 , wenn die synonyme Gleichheit $\Delta_{\alpha\bar{a}} =_{\bar{a}} A_1$ vorliegt.
- π_1 ist eine synonyme α -Spezialisierung von π_2 , wenn die synonyme Gleichheit $\Delta_{\alpha\bar{a}} =_{\bar{a}} A_2$ vorliegt.

- π_1 und π_2 sind synonyme α -Geschwister mit dem synonym-gleichen α -Kern $\Delta_{\alpha\bar{a}}$, wenn zwischen π_1 und π_2 weder eine synonyme α -Generalisierung noch eine synonyme α -Spezialisierung vorliegt.
- π_2 ist eine α -Erweiterung von π_1 , wenn die Gleichheit $\Delta_{\alpha=} = A_1$ vorliegt.
- π_1 ist eine α -Erweiterung von π_2 , wenn die Gleichheit $\Delta_{\alpha=} = A_2$ vorliegt.
- π_1 und π_2 sind α -Geschwister mit dem gleichen α -Kern $\Delta_{\alpha=}$, wenn zwischen π_1 und π_2 keine α -Erweiterungen vorliegen.

Aktionsebene

Ein Vergleich der Schnittmengen, gegeben durch die Aktionen, führt zu folgenden Definitionen:

Für die Betrachtungen werden die Menge der Aktionen $X_1 \in \pi_1$ des Pliable Objects π_1 und die Menge der Aktionen $X_2 \in \pi_2$ des Pliable Objects π_2 herangezogen. Die Bildung einer Schnittmenge dieser Mengen führt zu folgenden drei Ausprägungen:

1. die Schnittmenge $\Delta_{=}$ der gleichen Aktionen:

$$\Delta_{\chi=} = X_1 \cap X_2 \text{ mit } \Delta_{\chi=} = \{ \chi_{=} \mid \exists \chi_1 \in X_1 \wedge \exists \chi_2 \in X_2 \text{ mit } \chi_1 = \chi_2 \}$$

2. die Schnittmenge $\Delta_{\bar{a}}$ der synonymen Aktionen:

$$\Delta_{\chi\bar{a}} = X_1 \cap X_2 \text{ mit } \Delta_{\chi\bar{a}} = \{ \chi_{\bar{a}} \mid \exists \chi_1 \in X_1 \wedge \exists \chi_2 \in X_2 \text{ mit } \chi_1 =_{\bar{a}} \chi_2 \}$$

3. die Schnittmenge Δ_s der schematisch gleichen Aktionen:

$$\Delta_{\chi_s} = X_1 \cap X_2 \text{ mit } \Delta_{\chi_s} = \{ \chi_s \mid \exists \chi_1 \in X_1 \wedge \exists \chi_2 \in X_2 \text{ mit } \chi_1 =_s \chi_2 \}$$

Auch für die Aktionen wird angenommen, dass die Aktionsmengen in keinem Zusammenhang stehen, wenn die Mengen X_1 und X_2 disjunkt zueinander sind. Auch hier würden dann die drei Schnittmengen jeweils durch die leere Menge repräsentiert werden.

Wenn hingegen die beiden Mengen nicht disjunkt sind, besteht ein Zusammenhang zwischen ihnen, der durch folgende Definitionen genauer spezifiziert wird:

- π_1 ist eine schematische χ -Generalisierung von π_2 , wenn die schematische Gleichheit $\Delta_{\chi_s} =_s X_1$ vorliegt.
- π_1 ist eine schematische χ -Spezialisierung von π_2 , wenn die schematische Gleichheit $\Delta_{\chi_s} =_s X_2$ vorliegt.

- π_1 und π_2 sind schematische χ -Geschwister mit dem schematisch gleichen χ -Kern $\Delta_{\chi s}$, wenn zwischen π_1 und π_2 weder eine schematische χ -Generalisierung noch eine schematische χ -Spezialisierung vorliegt.
- π_1 ist eine synonyme χ -Generalisierung von π_2 , wenn die synonyme Gleichheit $\Delta_{\chi \bar{a}} =_{\bar{a}} X_1$ vorliegt.
- π_1 ist eine synonyme χ -Spezialisierung von π_2 , wenn die synonyme Gleichheit $\Delta_{\chi \bar{a}} =_{\bar{a}} X_2$ vorliegt.
- π_1 und π_2 sind synonyme χ -Geschwister mit dem synonym-gleichen χ -Kern $\Delta_{\chi \bar{a}}$, wenn zwischen π_1 und π_2 weder eine synonyme χ -Generalisierung noch eine synonyme χ -Spezialisierung vorliegt.
- π_2 ist eine χ -Erweiterung von π_1 , wenn die Gleichheit $\Delta_{\chi =} =_{=} X_1$ vorliegt.
- π_1 ist eine χ -Erweiterung von π_2 , wenn die Gleichheit $\Delta_{\chi =} =_{=} X_2$ vorliegt.
- π_1 und π_2 sind χ -Geschwister mit dem gleichen χ -Kern $\Delta_{\chi =}$, wenn zwischen π_1 und π_2 keine χ -Erweiterungen vorliegen.

Regelebene

Die Schnittmengen, die über die Regeln der Pliable Objects gebildet werden können, lassen entsprechend zu Attributen und Aktionen folgende Definitionen zu. Die Betrachtungen basieren auf der Menge der Regeln $P_1 \in \pi_1$ des Pliable Objects π_1 und auf der Menge der Regeln $P_2 \in \pi_2$ des Pliable Objects π_2 . Die drei Ausprägungen der Schnittmengen lauten wie folgt:

1. die Schnittmenge $\Delta_{=}$ der gleichen Regeln:

$$\Delta_{\rho=} = P_1 \cap P_2 \text{ mit } \Delta_{\rho=} = \{ \rho = \mid \exists \rho_1 \in P_1 \wedge \exists \rho_2 \in P_2 \text{ mit } \rho_1 =_{=} \rho_2 \}$$

2. die Schnittmenge $\Delta_{\bar{a}}$ der synonymen Regeln:

$$\Delta_{\rho \bar{a}} = P_1 \cap P_2 \text{ mit } \Delta_{\rho \bar{a}} = \{ \rho \bar{a} \mid \exists \rho_1 \in P_1 \wedge \exists \rho_2 \in P_2 \text{ mit } \rho_1 =_{\bar{a}} \rho_2 \}$$

3. die Schnittmenge Δ_s der schematisch gleichen Regeln:

$$\Delta_{\rho s} = P_1 \cap P_2 \text{ mit } \Delta_{\rho s} = \{ \rho_s \mid \exists \rho_1 \in P_1 \wedge \exists \rho_2 \in P_2 \text{ mit } \rho_1 =_s \rho_2 \}$$

Auch für die Aktionen wird angenommen, dass die Aktionsmengen in keinem Zusammenhang stehen, wenn die Mengen P_1 und P_2 disjunkt zueinander sind. Auch hier würden dann die drei Schnittmengen jeweils durch die leere Menge repräsentiert werden.

Wenn hingegen die beiden Mengen nicht disjunkt sind, besteht ein Zusammenhang zwischen ihnen, der durch folgende Definitionen genauer spezifiziert wird:

- π_1 ist eine schematische ρ -Generalisierung von π_2 ,
wenn die schematische Gleichheit $\Delta_{\rho s} =_s P_1$ vorliegt.
- π_1 ist eine schematische ρ -Spezialisierung von π_2 ,
wenn die schematische Gleichheit $\Delta_{\rho s} =_s P_2$ vorliegt.
- π_1 und π_2 sind schematische ρ -Geschwister mit dem schematisch-gleichen ρ -Kern $\Delta_{\rho s}$,
wenn zwischen π_1 und π_2 weder eine schematische ρ -Generalisierung noch eine
schematische ρ -Spezialisierung vorliegt.
- π_1 ist eine synonyme ρ -Generalisierung von π_2 ,
wenn die synonyme Gleichheit $\Delta_{\rho \bar{a}} =_{\bar{a}} P_1$ vorliegt.
- π_1 ist eine synonyme ρ -Spezialisierung von π_2 ,
wenn die synonyme Gleichheit $\Delta_{\rho \bar{a}} =_{\bar{a}} P_2$ vorliegt.
- π_1 und π_2 sind synonyme ρ -Geschwister mit dem synonym-gleichen ρ -Kern $\Delta_{\rho \bar{a}}$,
wenn zwischen π_1 und π_2 weder eine synonyme ρ -Generalisierung noch eine
synonyme ρ -Spezialisierung vorliegt.
- π_2 ist eine ρ -Erweiterung von π_1 ,
wenn die Gleichheit $\Delta_{\rho =} =_{=} P_1$ vorliegt.
- π_1 ist eine ρ -Erweiterung von π_2 ,
wenn die Gleichheit $\Delta_{\rho =} =_{=} P_2$ vorliegt.
- π_1 und π_2 sind ρ -Geschwister mit dem gleichen ρ -Kern $\Delta_{\rho =}$,
wenn zwischen π_1 und π_2 keine ρ -Erweiterungen vorliegen.

Pliable-Object-Ebene

Auf Basis der Generalisierungs- und Spezialisierungsdefinitionen der Attribute, Aktionen und Regeln können Definitionen zur Kennzeichnung von Pliable Objects selbst getroffen werden. Erneut werden dazu die zwei Pliable Objects π_1, π_2 betrachtet:

- π_1 ist eine schematische Generalisierung von π_2 , wenn π_1 sowohl eine schematische α -Generalisierung als auch eine schematische χ -Generalisierung als auch eine schematische ρ -Generalisierung von π_2 ist.
- π_2 ist eine schematische Spezialisierung von π_1 , wenn π_2 sowohl eine schematische α -Spezialisierung als auch eine schematische χ -Spezialisierung als auch eine schematische ρ -Spezialisierung von π_1 ist.
- π_1 ist eine synonyme Generalisierung von π_2 , wenn π_1 sowohl eine synonyme α -Generalisierung als auch eine synonyme χ -Generalisierung als auch eine synonyme ρ -Generalisierung von π_2 ist.

- π_2 ist eine synonyme Spezialisierung von π_1 , wenn π_2 sowohl eine synonyme α -Spezialisierung als auch eine synonyme χ -Spezialisierung als auch eine synonyme ρ -Spezialisierung von π_1 ist.
- π_1 ist eine Erweiterung von π_2 , wenn π_1 sowohl eine α -Erweiterung als auch eine χ -Erweiterung als auch eine ρ -Erweiterung von π_2 ist.
- π_1 und π_2 sind Geschwister, wenn π_1 und π_2 sowohl α -Geschwister als auch χ -Geschwister als auch ρ -Geschwister von π_2 sind.

4.10 Verknüpfung von Pliable-Object-spezifischen Informationen

Aus mehreren Instanzen können die jeweils zugehörigen Schema-Pliable-Objects gebildet werden, indem auf den Kopien der Instanzen deren Werte auf den jeweiligen Defaultwert gesetzt werden. Gilt für die auf diese Art und Weise erhaltenen Schema-Pliable-Objects, dass sie als genau ein Schema-Pliable-Objekt π^S ausgedrückt werden können, weil sie paarweise gleich sind, dann können die Instanzen zu einer Wertemenge zusammengefasst werden. Aus dem Schema-Pliable-Objekt π^S wird der Pliable-Name der Beschreibungsdomäne als Typbezeichnung genutzt, welcher zusammen mit der Wertemenge eine neue Domäne δ_π ergibt.

Die möglichen Domänen besitzen einen dynamischen Charakter, da deren Wertemengen aufgrund ihrer Anzahl an Werten schwanken können. Jede neue Instanz erweitert eine zugehörige Domäne umgehend nach der Instanzerzeugung. Wird eine Instanz gelöscht, verringert sich die Werteanzahl der Wertemenge. Die Dynamik durch die Instanzerzeugung hat Auswirkungen auf die Attribute, deren Domäne durch das Schema-Pliable-Object der Instanzen gegeben ist. Besonders die Löschung einer Instanz kann Auswirkungen haben, da die Instanz sowohl eine wertvolle Information eines Attributes sein kann als auch eine wichtige Rolle in einer Aktion oder einer Regel spielen kann.

Ähnliche Auswirkungen ergeben sich eventuell, falls eine Instanz eine Änderung erfährt. Es erfolgen nun weitere Betrachtungen, auf deren Basis die Auswirkungen später abschätzbar sein sollen.

Über die Instanzerzeugung lassen sich die Domänen

$$\begin{aligned}\delta_{\text{Attribute}} &= (\text{Attribut}, \{\langle\langle\text{alle Attribute}\rangle\rangle\}) \\ \delta_{\text{Aktionen}} &= (\text{Aktion}, \{\langle\langle\text{alle Aktionen}\rangle\rangle\}) \\ \delta_{\text{Regeln}} &= (\text{Regel}, \{\langle\langle\text{alle Regeln}\rangle\rangle\})\end{aligned}$$

erzeugen.

Mit Hilfe der Domänen sind folgende Attribute definierbar:

$$\begin{aligned}\alpha_{\text{Attribut}} &= (\text{Attribute}, \text{POWERSET}(\delta_{\text{Attribute}}), \varepsilon) \\ \alpha_{\text{Aktion}} &= (\text{Aktionen}, \text{POWERSET}(\delta_{\text{Aktionen}}), \varepsilon) \\ \alpha_{\text{Regel}} &= (\text{Regeln}, \text{POWERSET}(\delta_{\text{Regeln}}), \varepsilon)\end{aligned}$$

Die drei Attribute sind in der Wertemenge der Domäne δ_{Attribut} enthalten. Ein Pliable Object kann als eine Attributmenge angegeben werden:

$$\text{Pliable Object} = \{ \alpha_{\text{Attribut}}, \alpha_{\text{Aktion}}, \alpha_{\text{Regel}} \}$$

Besondere Bedeutung besitzt das Attribut α_{Attribut} . Da $\alpha_{\text{Attribut}} \in \delta_{\text{Attribut}}$ gilt, kann der Wert von α_{Attribut} die Menge $\{ \alpha_{\text{Attribut}} \}$ sein. Das Attribut enthält sich in dem Fall selbst als Wertemenge. Es existiert ein Selbstbezug durch eine Art der „Selbstenthaltung“, die sich bei rekursiver Auflösung der Wertemenge unendlich fortsetzt.

Aus rein logischer Überlegung heraus mag die fortgesetzte Selbstenthaltung in Frage gestellt sein. Aber daraus kann nicht die Sinnlosigkeit abgeleitet werden. Folglich wird die Selbstenthaltung zugelassen. Es wird angenommen, dass eine rekursive Auflösung nach endlich vielen Schritten von einem Algorithmus erkannt und abgebrochen wird.

Die Selbstenthaltung erlaubt die Angabe von Attributen durch

$$\alpha_{\text{Attribut}} = \{ \alpha_{\text{Attribut}} \},$$

wobei die Domäne minimaler Bestandteil des Wertes, also der Wertemenge von α_{Attribut} , ist.

4.10.1 Synthese von Pliable Objects

Über die Verwendung von dynamischen Domänen in geeigneten Attributen können Beziehungen zwischen Pliable Objects hergestellt werden, die gewisse Abhängigkeiten darstellen, die sich aus der Zusammensetzung bzw. Verknüpfungen von und zwischen Pliable Objects ergeben. Derartige Zusammensetzungen werden durch den Kontext der Anwendung von außen vorgegeben oder auch quasi synthetisiert. Die Arten einer möglichen Informationsverknüpfung sollen im Folgenden allgemeingültig betrachtet und definiert werden.

Attribut-spezifische Gebundenheiten

Es seien zwei Pliable Objects π_1 , π_2 und zwei Attribute α_1 , α_2 mit $\alpha_1 \in \pi_1$ und $\alpha_2 \in \pi_2$ gegeben.

- Wenn durch das Pliable Object π_2 eine Domäne gebildet wird, die wiederum die Attributdomäne von α_1 bildet, dann heißt π_2 an π_1 *gebunden*.
Formal sei dies durch den Ausdruck $\pi_2 \circ \pi_1$ wiedergegeben.
- Bildet zusätzlich das Pliable Object π_1 eine Domäne, die gleichzeitig die Attributdomäne von α_2 ist, dann heißen π_1 und π_2 miteinander *verknüpft*.
Dieser Sachverhalt sei formal über den Ausdruck $\pi_2 \diamond \pi_1$ wiedergegeben.

Abbildung 28 zeigt ein Beispiel, wie eine Verknüpfung zu sehen ist. Es existiert ein Pliable Object *Operation* und ein Pliable Object *Anästhesie*. Der *Operation* ist ein Attribut *Anästhesie* zu eigen, dessen Domäne durch die Instanzen von *Anästhesie* gegeben ist. Das Pliable Object *Anästhesie* ist so an die *Operation* *gebunden*; formal $\pi_A \circ \pi_O$.

Angenommen, das Pliable Object *Anästhesie* besäße zusätzlich ein Attribut *Operation*, dessen Domäne durch die Instanzen von *Operation* gegeben wäre, dann wären *Anästhesie* und *Operation* miteinander verknüpft; formal $\pi_A \diamond \pi_O$.

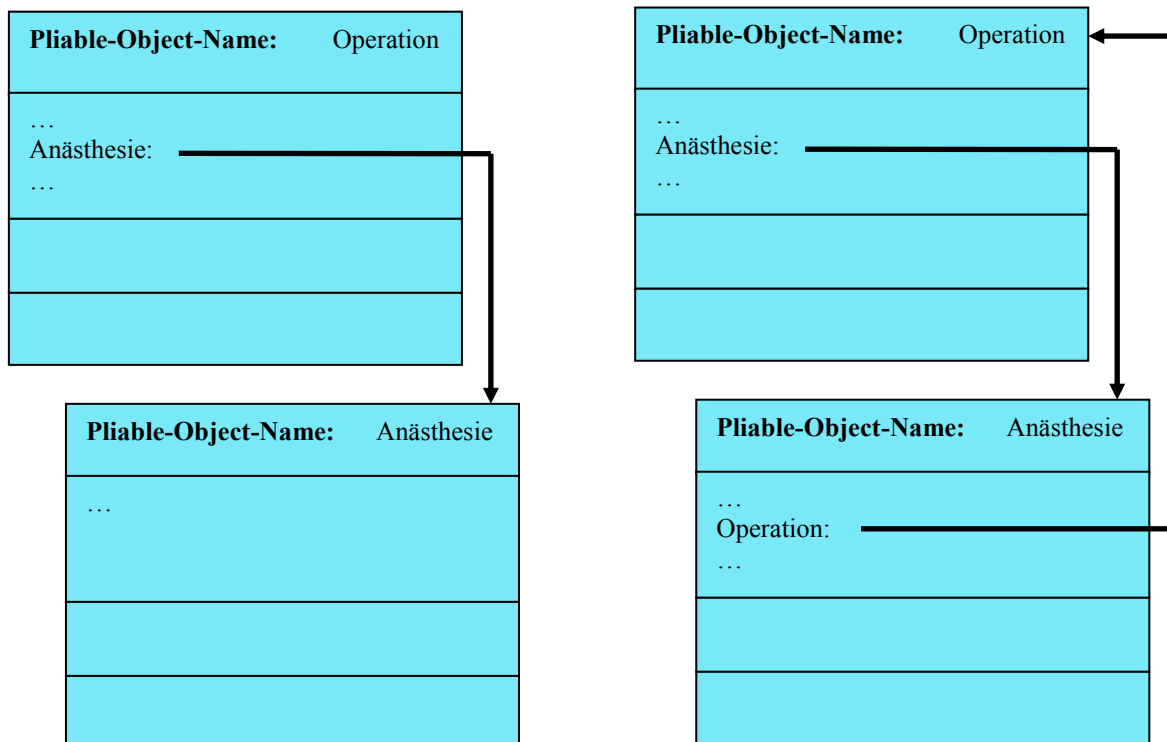


Abbildung 28: Visualisierung von Gebundenheit und Verknüpfung bei Pliable Objects

Spezielle Gebundenheiten bei Sonder-Pliable-Objects

Die vorgestellten Gebundenheiten arbeiten auf durch Pliable Objects gegebenen Domänen. Die Sicht auf die Pliable Objects hatte vorrangig den Charakter einer Datenverknüpfung. Allerdings kann unter einem Pliable Object auch ein Attribut selbst, eine Aktion oder eine Regel verstanden werden. Hinsichtlich dieser Objekte soll die Gebundenheit eine verfeinernde Definition bekommen.

Wenn π_2 an π_1 *gebunden* ist, wobei π_2 ein Pliable Object und π_1 ein Sonder-Pliable-Object ist, dann heißt

- π_2 *aktionsgebunden* an π_1 , wenn π_1 eine Aktion ist,
- π_2 *regelgebunden* an π_1 , wenn π_1 eine Regel ist und
- π_2 *attributgebunden* an π_1 , wenn π_1 ein Attribut ist.

Zur Veranschaulichung sei die Aktion *GET_Anästhesie()* eines nicht näher bestimmten Pliable Objects betrachtet. Diese Aktion soll den aktuellen Wert eines Attributes *Anästhesie* bestimmen, d.h. das Pliable Object, welches in dem Attribut referenziert wird. Das Attribut *Anästhesie* ist *aktionsgebunden* an die Aktion *GET_Anästhesie()*.

Differenzierung von Gebundenheitsklassen

Die bisherigen Betrachtungen der Gebundenheit basierten auf prinzipiellen 1:1-Beziehungen, d.h. nur ein Pliable Object π_2 ist an eine Aktion aktions- oder an eine Regel regelgebunden. Diese Form der Gebundenheit wird als *einfache Gebundenheit* bezeichnet. Sie beinhaltet also den Fall, dass ein Pliable Object nur ein einziges anderes Pliable Object an sich bindet.

Werden dagegen mehrere Pliable Objects an eine Aktion aktionsgebunden, so wird die Aktion als *komplexe Aktion* bezeichnet. Analog definiert sich eine *komplexe Regel*, wenn mehrere Pliable Objects an eine Regel regelgebunden werden. Ein *komplexes Pliable Object* zeichnet sich dadurch aus, dass mehrere Pliable Objects an dieses Pliable Object gebunden sind.

Die Definition eines *komplexen Attributes* erübrigt sich, da ein Attribut generell drei Attribute besitzt und somit per se komplex ist. Ein Pliable Object *Attribut* mit nur einem einzigen Attribut ist daher nicht möglich.

Ein Pliable Object kann in mehreren Pliable Objects gebunden werden. In so einem Fall wird das Pliable Object als *mehrfach gebunden* bezeichnet. Bezieht sich die mehrfache Gebundenheit des Pliable Objects ausschließlich auf mehrere Aktionen, wird es als *mehrfach aktionsgebunden* bezeichnet. Entsprechend bezeichnet *mehrfach regelgebunden* den Umstand, dass ein Pliable Object ausschließlich in mehreren Regeln gebunden wird.

Ein Pliable Object wird als *komplex gebunden* bezeichnet, wenn es sowohl mehrfach aktions- als auch mehrfach regelgebunden ist.

Die Definitionen der Gebundenheit können noch detaillierter gefasst werden, indem zusätzlich die Unterscheidung zwischen interner und externer Gebundenheit getroffen wird. Ein Pliable Object, welches mehrfach oder komplex gebunden ist, wird als *intern mehrfach* oder *intern komplex gebunden*, wenn sich die Gebundenheit ausschließlich auf Pliable Objects bezieht, die alle einem einzelnen Pliable Object *zu eigen* sind. Ist diese Bedingung nicht erfüllt, so gilt es als *extern mehrfach* oder *extern komplex gebunden*.

In Abbildung 28 ist beispielsweise das Pliable Object *Anästhesie* an das Pliable Object *Operation* gebunden. Wenn sichergestellt ist, dass alle Aktionen und Regeln, die auf der *Anästhesie* arbeiten, ausschließlich dem Pliable Object *Operation* zu eigen sind, ist die *Anästhesie intern mehrfach* oder *komplex* an die *Operation gebunden*. Würde eine einzige Aktion oder Regel existieren, die auf der *Anästhesie* arbeitet, aber nicht der *Operation* zu eigen ist, wäre die *Anästhesie extern mehrfach* oder *komplex* an die *Operation gebunden*.

Gebundenheitsbezogene Transitivität

Die Gebundenheit kann auch, wie in Abbildung 29 dargestellt, transitiv gesehen werden. Das Pliable Object *Anästhesie* π_A ist derart aufgebaut, dass der Patient π_P an die Anästhesie und die Anästhesie an die Operation π_O gebunden sind. Formal gilt $\pi_P \circ \pi_A$ und $\pi_A \circ \pi_O$.

Durch eine Verbindung $\pi_P \circ \pi_A \circ \pi_O$ gilt auch $\pi_P \circ \pi_O$, d.h. der Patient ist auch an die Operation gebunden. Da die Bindung durch die Transitivität gegeben ist, soll eine transitive Gebundenheit als indirekte Gebundenheit definiert sein. Formal kenntlich sei dies durch ein tiefgestelltes T am Gebundenheitsoperator \circ_T . Im Beispiel gilt somit $\pi_P \circ_T \pi_O$.

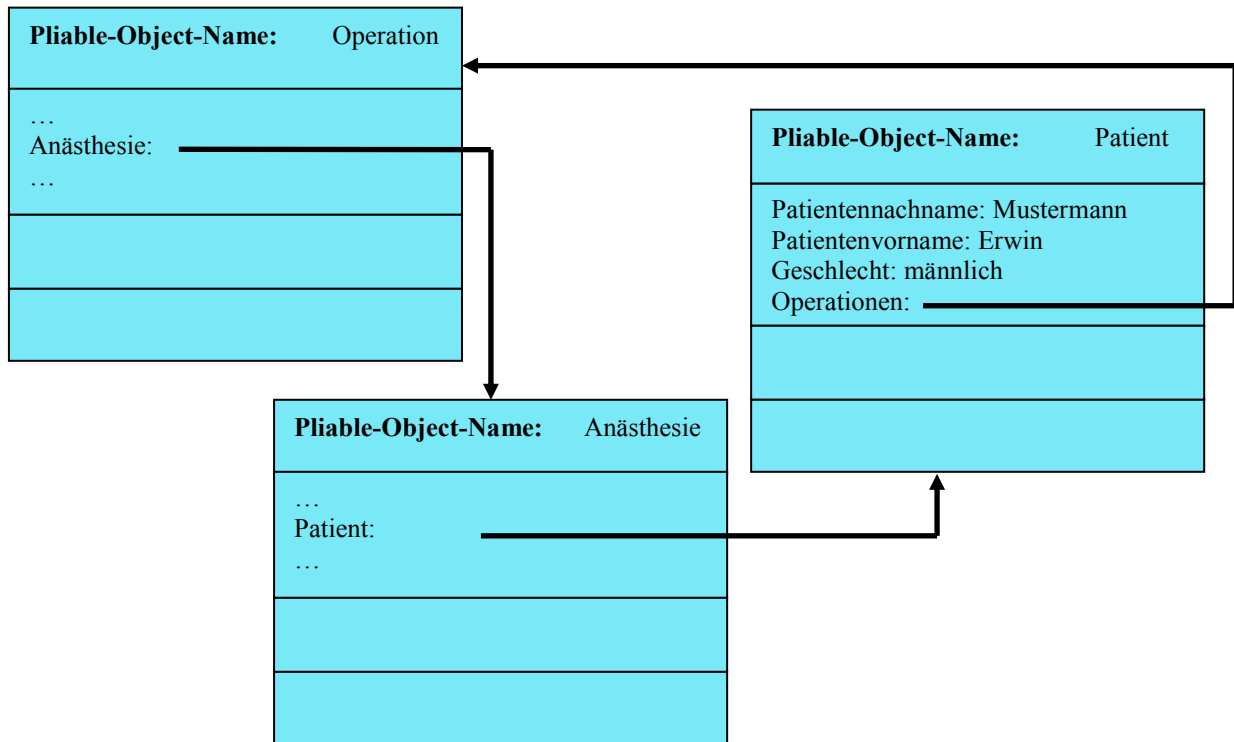


Abbildung 29: Visualisierung einer transitiven Gebundenheit

Analog sei die transitive Verknüpfung von Pliable Objects definiert, d.h. aus $\pi_2 \diamond \pi_1$ und $\pi_1 \diamond \pi_3$ folgt $\pi_2 \diamond_T \pi_3$.

Über die Transitivität stellt sich ein Zusammenhang zwischen Gebundenheit und Verknüpfung her. Aus $\pi_2 \circ \pi_1$ und $\pi_1 \circ \pi_2$ folgt $\pi_2 \circ_T \pi_2$. Wenn ein Pliable Object transitiv mit sich selbst verbunden ist, folgt, dass das Pliable Object mit dem Pliable Object verknüpft ist, welches die transitive Vereinigung ermöglicht. Aus $\pi_2 \circ \pi_1$ und $\pi_1 \circ \pi_2$ mit $\pi_2 \circ_T \pi_2$ folgt $\pi_2 \diamond \pi_1$. Die Verknüpfung ist symmetrisch, da $\pi_2 \diamond \pi_1$ gleich $\pi_1 \diamond \pi_2$ gilt.

Problematisch ist die Transitivität, wenn sich ein Zyklus der Gebundenheit ergibt, der keine direkte Verknüpfung darstellt. Abbildung 29 zeigt beispielsweise die Bindung der Anästhesie an eine Operation. Im Rahmen der Anästhesie wird ein Patient eingebunden. Der Patient hingegen bindet die Operation ein. Es bildet sich ein Zyklus, der eine Verknüpfung zwischen Operation und Patient realisiert. Ein derartiger Zyklus wird als Verknüpfungszyklus definiert.

Aus dem dargestellten Verknüpfungszyklus lässt sich die Verknüpfung durch den Durchlauf der Gebundenheit erstellen. Prinzipiell ist durch den Zyklus die Operation transitiv mit sich selbst verbunden, allerdings repräsentiert dies nicht die Verknüpfung mit dem transitiven Pliable Object, da es davon zwei gibt.

Metrische Betrachtung von Pliable Objects

Um die Gebundenheit und die Verknüpfung konkreter zu definieren, sei eine Abstandsfunktion d gegeben, welche die Anzahl der Gebundenheiten zwischen zwei Pliable Objects angibt.

Gegeben sei die Abbildung $d: P \times P \rightarrow \mathbb{R}$ mit der Menge der Pliable Objects P .

Grundsätzlich gelte, dass der Abstand eines Pliable Objects zu sich selbst gleich Null ist:

$$d(\pi_x, \pi_x) = 0$$

Der Abstand sei ebenfalls 0 für Pliable Objects, die nicht miteinander verknüpft oder aneinander gebunden sind.

Für $\pi_2 \circ \pi_1$ und $\pi_2 \diamond \pi_1$ ist der Abstand d mit 1 definiert:

$$d(\pi_1, \pi_2) = 1$$

Für $\pi_2 \diamond \pi_1$ ist die Verknüpfung symmetrisch, da

$$d(\pi_1, \pi_2) = d(\pi_2, \pi_1).$$

Die Abstandsfunktion soll unabhängig von der Richtung einer Gebundenheit definiert sein. Für $\pi_2 \circ \pi_1$ wird der Abstand ebenfalls symmetrisch definiert, d.h.

$$d(\pi_2, \pi_1) = 1,$$

auch wenn $\pi_1 \circ \pi_2$ nicht existiert.

Für die transitive Gebundenheit $\pi_X \circ_T \pi_Y$ bzw. transitive Verknüpfung $\pi_X \diamond \pi_Y$ gilt, dass der Abstand mindestens 2 ist:

$$d(\pi_X, \pi_Y) \geq 2,$$

da mindestens ein Pliable Object π_Z existieren muss, über welches die Transitivität hergestellt wird. Es folgt, dass $d(\pi_X, \pi_Z) \geq 1$ und $d(\pi_Z, \pi_Y) \geq 1$ ist. Der Transitivitätsabstand ergibt sich aus der Summe der Abstände der Pliable Objects π_X und π_Y zu π_Z .

Durch die Möglichkeit von Verknüpfungszyklen gilt, dass der Abstand eines Pliable Objects zu sich selbst auch größer als 0 sein kann, d.h.

$$d(\pi_x, \pi_x) \geq 0.$$

Über Pliable Objects lässt sich keine Metrik bilden, da die Bedingung

$$d(\pi_x, \pi_y) = 0 \Rightarrow \pi_x = \pi_y$$

nicht erfüllt ist.

4.10.2 Bewertungsoptionen von Pliable Objects

Auf der Basis der Generalisierungen und Spezialisierungen kann eine Hierarchie auf Pliable Objects aufgebaut werden. Entlang der Hierarchie sind dabei Vererbungsmechanismen realisierbar, wie sie aus dem objektorientierten Paradigma bekannt sind. Allerdings bietet die Generalisierung/Spezialisierung von Pliable Objects einen Vorteil, der auf der Gebundenheit von Attributen, Aktionen und Regeln basiert. Unterschiedliche Pliable Objects werden aufgrund der Mengenbetrachtung von Attributen, Aktionen und Regeln vergleichbar. Über eine mengentheoretische Auswertung können Analysen vorgenommen werden, die wiederum zu einer Bewertung der betrachteten Pliable Objects führen. Die Bewertung kann dabei auf Werten, d.h. auf Instanzen von Pliable Objects oder den Attributen, Aktionen und Regeln von Pliable Objects selbst, d.h. auf deren Schema, vorgenommen werden. Die Schlüsse, die aus dem Bewertungsergebnis gezogen werden können, müssen sich an der Anwendung orientieren, die mittels Pliable Objects konzipiert und implementiert werden soll.

Durch die Generalisierung wird eine konkrete Sicht auf Pliable Objects möglich. Es ist anzunehmen, dass viele Pliable Objects in einer Geschwister-Beziehung zueinander stehen. Dank der Generalisierung wird sichtbar, wie viel bzw. wie wenig sie gemeinsam haben. Ein wichtiger Aspekt, den es bei der Generalisierung zu berücksichtigen gilt, ist, dass das generalisierte Pliable Object durch eine Art „Ausblenden“ von Attributen, Aktionen und Regeln gewonnen wird. Dabei besteht die Möglichkeit, dass sich bei in einer Geschwister-Beziehung stehenden Pliable Objects ein Attribut im α -Kern wiederfinden lässt. Dieses Attribut ist eventuell an keine gemeinsame Aktion gebunden, obgleich ein Attribut in jedem Pliable Object an eine bestimmte Aktion gebunden sein sollte. Werden die beiden Aktionen im Zusammenhang verglichen, zeigen sie vermutlich ein ganz unterschiedliches Verhalten.

Allerdings muss eine Generalisierung von mehreren Pliable Objects durch Bildung einer Schnittmenge mit Bedacht angegangen werden. Pliable Objects mögen gleiche bzw. ähnliche Attribute besitzen, das von ihnen abhängige Verhalten ist es sehr wahrscheinlich jedoch nicht. So dürften Pliable Objects mit gleichen Attributen existieren, aber ungleiche Regeln und Aktionen besitzen, die auf diesen Attributen arbeiten. Hier erlauben Pliable Objects die Möglichkeit, über die Schnittmengenbildung betroffene Regeln und Aktionen zu bestimmen und eventuell hinsichtlich ihres Verhaltens anzupassen, um unerwünschte Verhaltenseffekte zu neutralisieren.

Hinsichtlich der Gebundenheit von Pliable Objects über die Attributausbildung ist zu beachten, dass dies häufig einem speziellen Zweck dient. Meist existieren innerhalb des einbindenden Objektes spezielle interne Aktionen und Regeln, die das einzubindende Objekt spezifisch ansprechen. Dabei kann das einbindende Objekt selbst in einem dritten Pliable Object eingebunden sein.

Neben der transitiven Gebundenheit besteht auch das Problem, dass bestimmte Objekte mehrfach referenziert werden. Abbildung 30 zeigt das Beispiel, dass die Pliable Objects *Operation* und *Anästhesie* beide das Attribut *Patient* besitzen. Es sollte sichergestellt sein, dass beide Objekte den gleichen Patienten referenzieren, andernfalls wären die Informationen datentechnisch zueinander inkonsistent.

Problematisch ist der Fall, wenn die Pliable Objects über Aktionen verfügen, die extern gebunden sind. Es besteht die Gefahr, dass sie sich gegenseitig derart aufrufen, dass der Verknüpfungszyklus ebenfalls über die Aktionen nachvollziehbar ist. Solch ein Aktionsverknüpfungszyklus soll von vornherein ausgeschlossen sein, d.h. ein System,

welches die Erstellung von Pliable Objects unterstützt, prüft automatisch die Verknüpfung von Aktionen und unterbindet den fortgesetzten Zyklus. Schwieriger gestaltet sich die Möglichkeit, wenn nicht nur der Verknüpfungszyklus selbst, sondern ebenfalls eine Verknüpfung vorhanden ist.

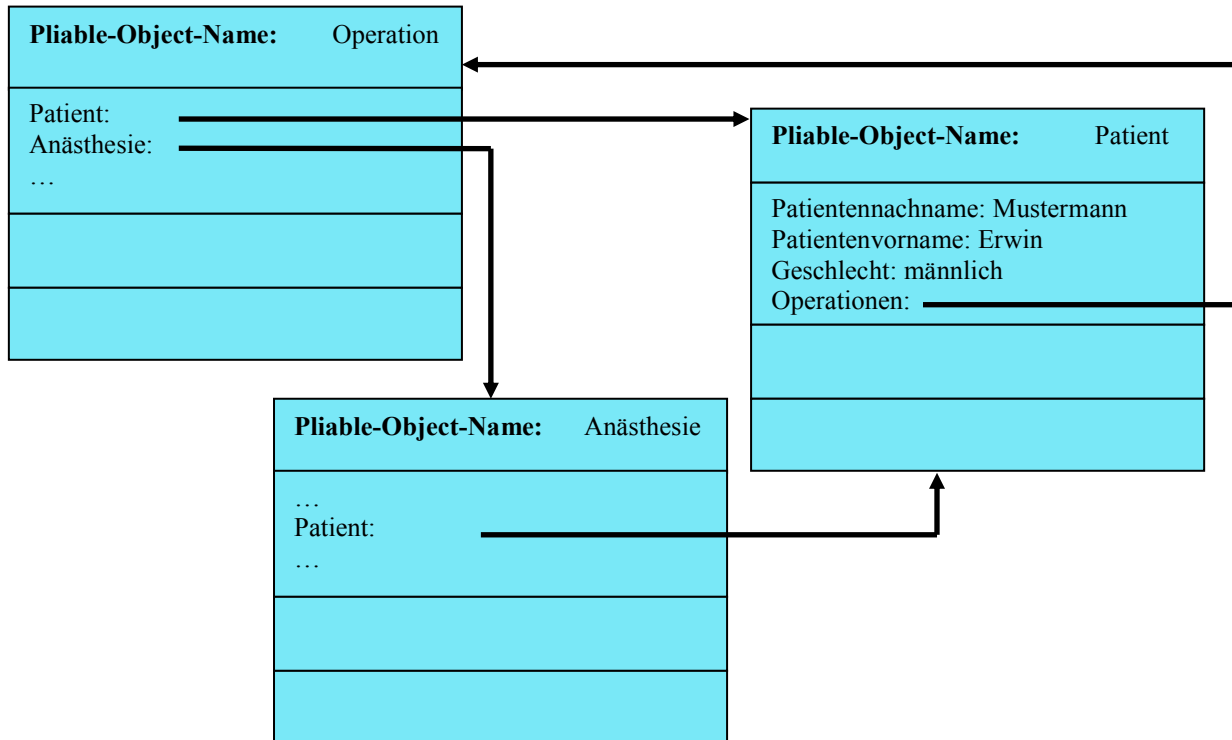


Abbildung 30: Mehrfache Referenzierung eines Pliable Objects in einem Verknüpfungszyklus

Wenn wir annehmen, dass der Patient über eine Aktion verfüge, so wird diese im Rahmen der in der *Operation* existierenden Aktionsausführung aufgerufen. Ferner werde die Aktion des Patienten von einer Aktion des Pliable Objects *Anästhesie* aufgerufen. Der Sachverhalt kann hergestellt werden, wenn die Aktion des Patienten beide Aktionen mit der gleichen Funktionalität bedienen kann. Verlangen die Aktionen aus *Operation* und *Anästhesie* hingegen unterschiedliche Ausprägungen der Patientenaktion, so muss der Patient eventuell zwei relativ ähnliche Aktionen zur Verfügung stellen. Ist dies nicht möglich, muss die gewünschte Funktionalität in die Aktionen der jeweiligen Pliable Objects übertragen werden. Alternativ wäre zur Lösung die dynamische Überladung der Aktion des Patienten denkbar, die beide Ausprägungen erlaubt.

Unabhängig davon, welcher Lösung gefolgt wird, entsteht eine Komplexität im Rahmen des Einsatzes der Aktionen. Diese Komplexität sollte visualisiert werden, um die Konsequenzen, die sich bei Änderungen von Pliable Objects ergeben können, abschätzbar zu machen.

Bei der Betrachtung von Attributen, Aktionen und Regeln als Pliable Objects gelten in deren Zusammenspiel bestimmte Arten der „Zusammenarbeit“. Allgemein gilt, dass Attribute einen Zustand abbilden, der ausschließlich durch Aktionen manipuliert werden kann. Regeln hingegen dienen der Steuerung und benutzen Attribute und Aktionen in einer vorher definierten Weise. Bei einer Regel gilt grundsätzlich für die Prämisse, dass sie auf Attributen

arbeitet, was auch auf Aktionen und Regeln bezogen sein kann. Die Konklusion einer Regel hingegen bedingt nur die Ausführung einer Aktion, welche eventuell zur Überprüfung einer weiteren Regel oder zu einem gewünschten Zustandswechsel führt.

Die Überprüfung einer Regel dient der Präzisierung der Steuerung. Im Rahmen der Präzisierung gilt, dass die Prämisse der Regel dabei immer auf eine *wertmäßige* Betrachtung eines Attributes abzielt. Hinsichtlich Aktionen und Regeln werden spezielle Attribute der Aktion oder Regel wertmäßig abgefragt, d.h. in der Prämisse einer Regel wird grundsätzlich eine Wertprüfung vorgenommen.

5 Informationssystem für Pliable Objects

Im vorherigen Kapitel sind Pliable Objects als ein Datenmodell erarbeitet und formalisiert betrachtet worden. Dieses formalisierte Modell könnte gemäß dem klassischen Ansatz zur Entwicklung eines Informationssystems verwendet werden, um in dem Modell das Datenmaterial der abzubildenden Anwendung aufzubereiten. Nach der Datenaufbereitung wird ein Entwicklungskonzept für die Implementierung erarbeitet, d.h. für die Datenverwaltung wird eine Datenbank verwendet, und die Datenverarbeitung wird von einer in einer höheren Programmiersprache zu implementierenden Applikation übernommen, die wiederum auf den Daten in der Datenbank arbeitet. Der klassische Entwicklungsprozess ist in Kapitel 3.5 kritisch betrachtet worden. Im Folgenden wird der Entwicklungsprozess mittels Pliable Objects vorgestellt. Teilweise sind einige Ergebnisse, die das Kapitel enthält, in [Pr11] und [Pr12] veröffentlicht worden.

5.1 Klassische Realisierung von Informationssystemen

Nach dem klassischen Ansatz wären modellierte Pliable Objects innerhalb des Entwicklungsprozesses abzubilden. Allgemein setzt sich ein Pliable Object aus Attributen, Aktionen und Regeln zusammen. Naheliegender wäre eine Abbildung der Attribute in ein Datenbankschema, damit die Datenbank als Basiseinheit des klassischen Informationssystems die Zustandsabbildung übernimmt. Die Aktionen und Regeln wären anschließend in einer Applikation mittels der Techniken einer höheren Programmiersprache zu implementieren, wobei darauf zu achten wäre, dass wohldefinierte Schnittstellen die Daten und die auf ihnen arbeitenden Methoden derart aufeinander abstimmen, dass das gewünschte Zusammenspiel von Attributen, Aktionen und Regeln der Pliable Objects gewährleistet wird.

In Bezug auf höhere Programmiersprachen bedeutet dies, dass eine geeignete Form der Datenabstraktion gefunden werden muss. Allgemein wird im Rahmen der funktionalen Abstraktion eine Trennung von Datenraum und Programm vorgenommen, während eine Datenabstraktion nach objektorientierten Gesichtspunkten spezifisch zusammengehörende Daten und Prozeduren kapselt (vgl. Kapitel 2). Fundamentale und strukturierte Abstraktionen werden letztlich während der Implementierung in Form von Variablen und Datenstrukturen im Programm ausgedrückt, welche die Applikation realisiert. Hinsichtlich Pliable Objects offenbaren sich darin Schwächen, die einer direkten Umsetzung von Pliable Objects in Objekte einer Programmiersprache widersprechen.

5.1.1 Bedeutung von Variablen

In höheren Programmiersprachen zählt das Variablenkonzept zu den grundlegendsten Konzepten. Variablen sind die Abstraktion von Speicherplätzen im Hauptspeicher. Sie besitzen einen Namen und enthalten einen Wert (vgl. [Lo94]). Variablen ermöglichen Entwicklern die Loslösung ihrer Implementierung von den adressengebundenen Speicherbereichen. Programmiersprachen bieten ein generalisiertes Standardverfahren zur Speichernutzung und -bereinigung, welches automatisch bei Variablen verwendet wird. Entwickler sind mit Variablen freier und flexibler in der Umsetzung ihrer Ideen und werden dadurch in die Lage versetzt, sich auf die Erarbeitung problemorientierter Algorithmen und deren Implementierung in Programme zu konzentrieren. Durch die gedankliche Loslösung von den Speicherbereichen wird der Programmcode nachvollziehbarer. Nachteilig ist, dass die

Programmierung recht fixiert vorgenommen wird, da sie sich weiterhin an den adressierbaren Speicherbereichen und damit an den Hardwarespezifika orientiert.

Eine Variable wird durch ihre Bezeichnung und ihren Wert definiert. Dieser Wert verfügt über die automatisch verwaltete Komponente „Ort“. Es handelt sich um die adressierbare Speicherzelle. Mit der Deklaration einer Variablen wird eine Speicherzelle – für das Programm – fest mit der Variablen gekoppelt. Deklarieren bedeutet, dass der Speicher intern allokiert wird, welcher über den Variablennamen Zugriffen zur Verfügung steht. Die Bitfolge, die darin gespeichert wird, repräsentiert den Wert der Variablen. Für die korrekte Interpretation der Bitfolge ist die Angabe eines Datentyps für die Variable notwendig. Es handelt sich dabei in der Regel um eine eindeutige Bezeichnung (vgl. [BH03]).

Beispielsweise wird durch folgende (pseudocode-ähnliche) Anweisung die Variable *x* vom Datentyp *Integer* deklariert:

```
Int x;
```

Mit der Variablen ist eine Semantik verbunden, die aus solch einer Variablendeklaration nicht ersichtlich ist. Erst durch Hinzuziehen einer eventuell existierenden Entwicklungsdokumentation wird die Semantik deutlich. Falls keine Entwicklungsdokumentation vorliegt, muss anhand des Algorithmus, in welchem die Variable verwendet wird, dessen Bedeutung erschlossen werden.

Zur Erhöhung der Wartbarkeit von Quellcodes gelten häufig Entwicklungsrichtlinien, die dem Verlust der Bedeutung entgegenwirken sollen. Ziel der Richtlinien ist die Bemühung, die Bedeutung einer Variablen zu sichern, indem die Variable „sprechend“ gestaltet wird.

Folgende Deklaration verbindet z.B. den Speicherbereich mit der Bezeichnung *alter*, um für jeden Programmierer die Nachvollziehbarkeit einfacher zu machen und anzudeuten, dass die Variable zur Aufnahme einer Altersinformation und/oder einer Altersberechnung dient.

```
Int alter;
```

An genau dieser Stelle tritt jedoch ein Informationsverlust auf. Nur für den Menschen ist eine Semantik durch dessen natürliches Verständnis der verwendeten Variablenbezeichnung gegeben. Mit der Durchsicht und Bearbeitung des Quellcodes kann die konkrete Semantik der Variablen im Programm erarbeitet werden, falls die Bezeichnung nicht aufgrund unterschiedlicher Interpretationen missverständlich ist. Für ein kompiliertes und laufendes Programm dient die Variablenbezeichnung aus Rechnersicht nur als symbolischer Bezeichner einer Adresse. Während der Kompilierung des Quellcodes geht die Semantik einer Variablen damit verloren.

Der Informationsverlust wird allerdings hingenommen. Schließlich werden Funktionen (Methoden) geschrieben, die gezielt die Verarbeitung der Information in dieser Variablen vornehmen, d.h. sie greifen lesend und schreibend auf die wertmäßige Information in ausschließlich dieser Variablen zu. Die Funktionen sind fixiert und starr mit der Variablen verknüpft. Es existiert kein direkter Informationsbezug zwischen der Variablen und der Methode, außer der starren Implementation, die der Methode den Wert der Variablen zurückgibt. Die Frage, auf welcher Variablen eine Methode arbeitet, kann nur geklärt werden, wenn man auch hier den Zusammenhang direkt in der Entwicklungsdokumentation oder im implementierten Quellcode nachliest.

5.1.2 Verwendung von Datenstrukturen

Die Verarbeitung von zusammenhängenden Daten geschieht anstelle von mehreren Einzelvariablen in der Regel in Datenstrukturen (vgl. [Re00]). Diese setzen sich aus verschiedenen Datentypen zusammen. Die komplexe (Typ-)Struktur entspricht dabei der komplexen Struktur der Entität des Realweltausschnitts, welches beim Modellieren abgebildet wird. Datenstrukturen sind letztlich auf Systemebene die Repräsentation von abstrakten Datentypen (vgl. [Re91], [Re00]). Der Nachteil derartiger Datenstrukturen ist, dass sie nach ihrer Definition festgelegt und schwer änderbar sind. Änderungen können eigentlich nur durch eine Aufdatierung der Applikation, d.h. des Programmes, erreicht werden. Die Starrheit von Datenstrukturen verhindert eine Dynamik, die eine Änderung von Datenstrukturen benötigt.

Klassische Datenstrukturen bilden auf Systemebene ebenfalls die Repräsentation des Klassenkonzeptes des objektorientierten Paradigmas. Bei dessen Implementierung werden durch die Abstraktion von Eigenschaften diese in *Variablen* überführt. Der Datentyp dieser Variablen ist als abstrakter Datentyp gegeben, welcher als Datenstruktur realisiert wird (vgl. [ZW06]). Prinzipiell wird damit die Sichtweise zum Ausdruck gebracht, dass die Datenstruktur als ein Objekt mit einem Zustand interpretiert werden kann. Ferner werden sie als aktiv verstanden, da sie den Zugriff auf den eigenen Speicherbereich und den eigenen Zustand selbst steuern.

Der Zugriff funktioniert über Routinen, die dem Objekt zu eigen und als Methoden definiert sind. Methoden versuchen den Informationsverlust zu kompensieren, der durch den strukturellen Aufbau gegeben ist. Implementierte Methoden setzen oder lesen den Wert einer oder mehrerer Daten in der Struktur. Der Zugriff auf den Speicherbereich wird dadurch gekapselt und typsicher gestaltet; der Informationsverlust hingegen lässt sich nicht vermeiden, da die Methoden selbst wieder fixiert und starr sind.

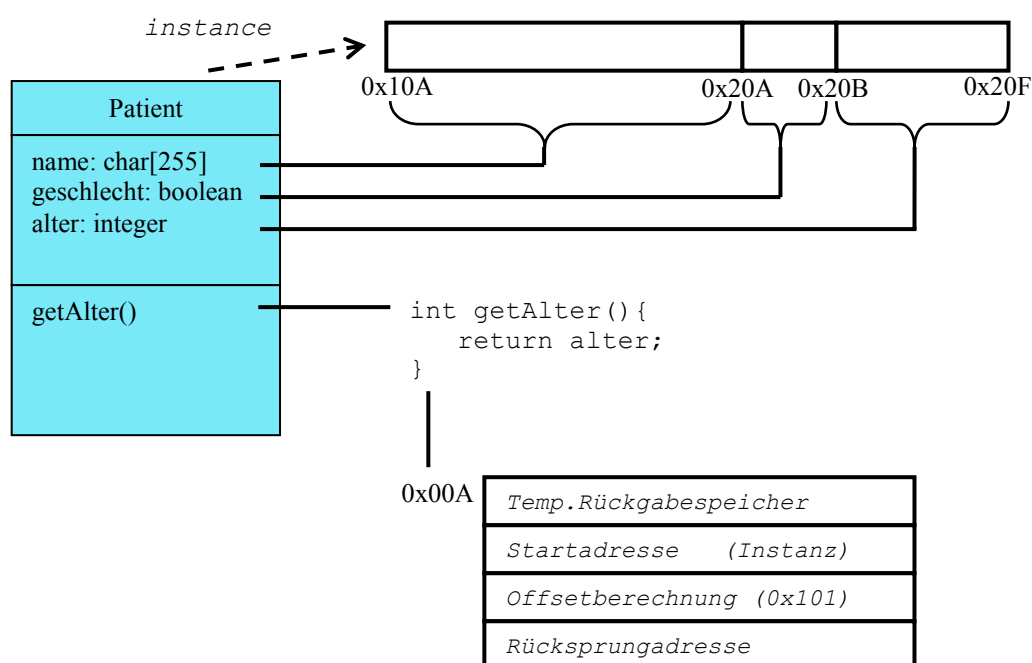


Abbildung 31: Prinzip der Abbildung einer Klasse in Speicherbereiche

Abbildung 31 skizziert das Prinzip der Abbildung einer objektorientierten Klasse in den Speicherbereich eines Systems. In der Klasse *Patient* werden der Patientennamen, das Geschlecht und das Alter des Patienten gespeichert. Eine Methode *getAlter()* liefert das Alter des Patienten an ein aufrufendes und nicht weiter spezifiziertes Objekt. Die Eigenschaften der Klassen verwenden Standarddatentypen (vgl. [HMP04]). Der Name soll als Zeichenkette in einem Characterarray von maximal 256 Zeichen gespeichert werden. Das Geschlecht wird auf einen booleschen Ausdruck abgebildet; letzteres basiert auf der Vorstellung, dass Patienten nur männlichen oder weiblichen Geschlechts sein können. Das Alter wird als 32-Bit Zahl gespeichert.

Die Abbildung der Klasse in einen Speicherbereich geschieht prinzipiell in der Form, da bei Erzeugung einer neuen Instanz der Klasse ein Speicherbereich reserviert wird, dessen Größe sich aus der Summe der Speicherbedarfe der einzelnen Eigenschaften ergibt. Im Abbildung 31 ergeben sich 261 Bytes, 256 Bytes für den Namen, 1 Byte für die Geschlechtsangabe und 4 Byte für das Alter des Patienten, dessen Daten erfasst werden.

Die Methode *getAlter()* wird in Maschinensprache übersetzt, wobei die Kenntnis der Speicherstruktur genutzt wird. Prinzipiell wird die Methode in eine Routine übersetzt, deren Funktionscode im Speicher abgelegt wird. Nach Aufruf des zugehörigen Funktionspointers wird zunächst (etwas vereinfachend ausgedrückt) ein Rückgabespeicher alloziert und die Startadresse der Instanz bestimmt, auf die die Methode angewandt wird. Ausgehend von der Startadresse wird mittels einer Offsetberechnung die Adresse bestimmt, an der das Alter abgelegt ist. Im Beispiel entspricht dieses Offset den 257 Bytes, die sich durch den Patientennamen und das Patientengeschlecht ergeben. Der Wert an der berechneten Adresse wird in den Rückgabespeicher kopiert und die Funktion endet, indem an die Stelle zurück gesprungen wird, an welcher die Methode aufgerufen worden ist.

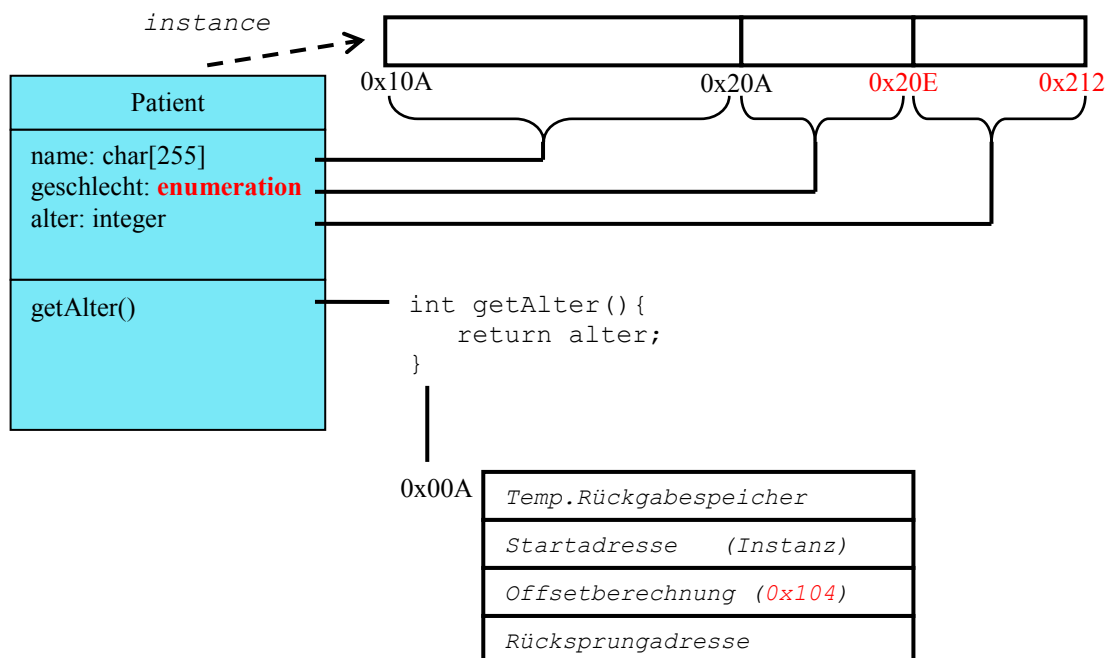


Abbildung 32: Auswirkungen einer Datentypenänderung

Die in dem Beispiel von Abbildung 31 gezeigte starre Festlegung auf Speicherbereiche fester Größe führt zu folgendem Problem:

Allgemein ist die Geschlechtsangabe eines Patienten durch *männlich* bzw. *weiblich* korrekt und wird häufig auch so unterstützt. Allerdings hat die DGAI einen Kerndatensatz für Anästhesieprotokolle verfasst (vgl. [DGAI]), der für die Geschlechtsangabe die Werte *männlich*, *weiblich* und *intersexuell* vorgibt. Diese Richtlinie muss nicht so unterstützt, sollte aber ermöglicht werden. Der im Beispiel gewählt boolesche Datentyp für die Abbildung der Geschlechtsangabe unterstützt nur zwei Werte. Um die drei Werte der Richtlinie zu unterstützen, muss der Datentyp geändert werden. Dies führt zu den Auswirkungen, die in Abbildung 32 skizziert werden.

Zur besseren Veranschaulichung wird für die Geschlechtsangabe eine 32-Bit große Aufzählungsangabe gewählt. Aufgrund der Änderung des Datentyps der Geschlechtsangabe von *boolean* auf die 32-Bit große Aufzählungsangabe vergrößert sich der Speicherbedarf der Klasse. Dies hat Auswirkungen auf die Methode der Klasse. In der Methode *getAlter()* muss auch die Offsetberechnung angepasst werden, da ein Zugriff über die ursprüngliche Offsetgröße ein falsches Ergebnis liefern würde.

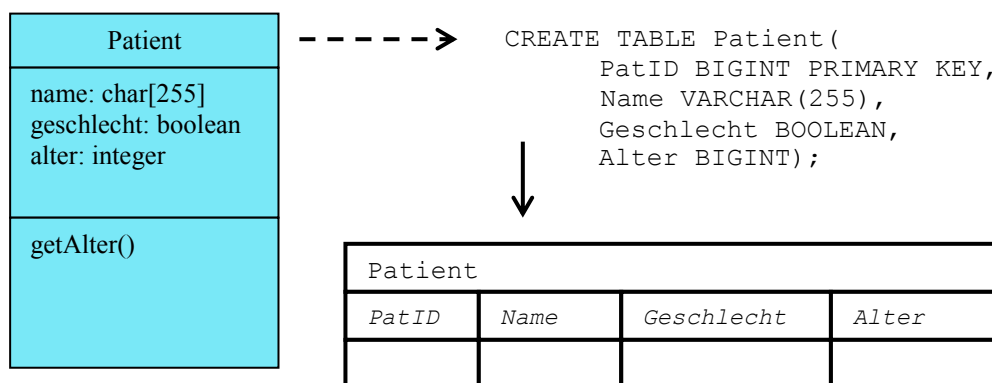


Abbildung 33: Beispiel einer relationalen Datenbankabbildung

Analog lässt sich dieses Problem auch auf Datenbanken abbilden. Die Klasse *Patient* kann auf einer Datenbank abgebildet werden. Abbildung 33 skizziert die Abbildung der Klasse *Patient* mittels SQL auf eine relationale Datenbank. Die erstellte Relation bildet die Eigenschaften der Klasse 1:1 ab, wobei zur Unterscheidung mit der PatID ergänzend eine Identifikation angegeben wurde (vgl. [HMP04]).

Die Problematik hinsichtlich der Typänderung der Eigenschaft *Geschlecht* trifft auch das Datenbanksystem. Allerdings bieten Datenbanken über ihre Anfragesprachen die Möglichkeit, derartige Typänderungen vorzunehmen (vgl. [HMP04]). Im Rahmen einer Schemaevolution wird das Datenbankschema angepasst (vgl. [SST97]). Abbildung 34 zeigt ein mögliches Vorgehen bei der Umsetzung der Änderungen. Im Beispiel wird die Aufzählung, d.h. die erlaubten Werte für die Geschlechtsangabe eines Patienten, in einer separaten Relation angelegt. Anschließend wird der Datentyp der Eigenschaft *Geschlecht* in der Relation *Patient* dahingehend geändert, dass als zulässige Werte ausschließlich die Identifikation GID der Aufzählungsrelation erlaubt sind. Erreicht wird dies durch die

Verwendung einer Fremdschlüsselbeziehung, die von Datenbanken angeboten werden. Das Problem der Datenanpassung des existierenden Datenbestandes im Rahmen der Schemaevolution wird nicht betrachtet.

Während die Schemaanpassung der Datenbank bereits innerhalb der Anfragesprache der Datenbank selbst unterstützt wird, sind weiterhin die Methoden der Klassen sowie unterstützende Funktionen und Routinen an die Änderungen anzupassen.

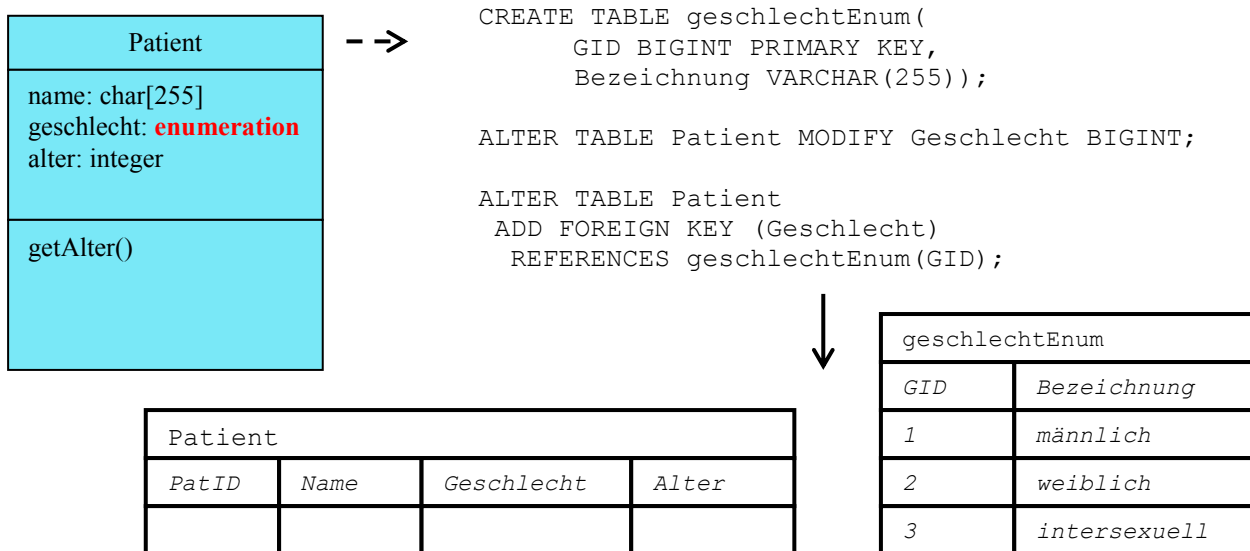


Abbildung 34: Prinzip der Fremdschlüsselbeziehung

Während der (datenrelevante) Zustand einer Instanz der Klasse *Patient* in der Datenbank verwaltet wird, müssen Funktionen zur Verfügung stehen, die den Zustand aus der Datenbank in den Hauptspeicher übertragen, damit mit den Methoden der Klasse der Zustand der Instanz bearbeitet werden kann. Nach der Bearbeitung sind die Änderungen in die Datenbank zu übertragen. Die vorgestellte Anpassung der Typänderung muss auch in derartigen Routinen vorgenommen werden, um die korrekte Datenverarbeitung sicherzustellen.

Diese Anpassungen sind je nach Umfang der Änderungen entsprechend zeitaufwändig. Unabhängig von diesem Zeitaufwand ist festzuhalten, dass die allgemeine Entwicklungsarbeit bei der Implementierung darauf beruht, das spezifische Modell abzubilden. Bei der Abbildung verwendet der Entwickler sein Wissen über die Zusammenhänge, um das gewünschte Verhalten der Applikation zu erzielen. Er weiß, welche Eigenschaften ein Objekt ausmachen und welche Methoden wie auf den Objekten arbeiten sollen. Mit diesem Wissen werden interne Datenstrukturen entworfen und implementiert, die einem Informationsverlust unterworfen sind. Es gehen die Informationen verloren, welche Eigenschaften und Methoden ein Objekt hat. Das nach Abschluss der Implementierung vorliegende Objekt funktioniert, indem die vorgezeichneten Abläufe abgearbeitet werden. Auswertungen, ob z.B. ein Objekt über eine bestimmte Eigenschaft verfügt, können nicht vorgenommen werden.

Das Problem des Informationsverlustes zieht sich durch die gesamte Implementierung. Das objektorientierte Paradigma besitzt eine starre Struktur, auf die, einmal etabliert, eine Standardreihenfolge folgt. Die starre Struktur eines Konzeptes wird in eine starre Implementierung umgesetzt. Dabei werden starre Methoden implementiert, die zwar flexibel

innerhalb der Anwendung genutzt werden, aber außerhalb der Anwendung nicht einsetzbar sind.

Der größte Nachteil der Implementierung offenbart sich beim Design der Schnittstelle. Auf Basis der starren Struktur ergeben sich starre Oberflächen, d.h. fest fixierte (Eingabe-) Masken. Einer Änderung der Struktur folgt zwangsläufig immer eine Änderung der zugehörigen Masken.

Pliable Objects stellen ein neues Konzept zur Abbildung realer Entitäten dar. Im Gegensatz zum objektorientierten Paradigma verfügen Pliable Objects hingegen über Selbstbeschreibungsmerkmale. Wie in Kapitel 4 gezeigt, haben Pliable Objects Kenntnis über ihre Attribute, Aktionen und Regeln. Diese Selbstbeschreibungsmerkmale sind ein wesentlicher Bestandteil des Modells und müssen bei der Abbildung auf Systemebene erhalten bleiben.

Versteht man Pliable Objects als Erweiterung des objektorientierten Paradigmas, so läge es nahe, das Pliable-Objects-Konzept in Programmiersprachen-Konzepte zu integrieren. Es existieren viele objektorientierte Programmiersprachen, die das objektorientierte Paradigma integrieren. Allerdings scheidet die Programmierung von Pliable Objects in einer Programmiersprache aus, da Programmiersprachen den oben vorgestellten Informationsverlust enthalten; diese Schwäche dürfte schwer auszumerzen sein. Für die Entwicklung von Pliable Objects scheiden Programmiersprachen daher aus. Stattdessen wird im Folgenden ein Konzept für ein Informationssystem vorgestellt, welches die Entwicklung und Bearbeitung von Pliable Objects und die Arbeit mit Pliable Objects unterstützt.

5.2 Informationssystem-Architektur für Pliable Objects

Das Ziel ist die Entwicklung eines Informationssystems zur Verarbeitung von Pliable Objects. Unter Verarbeitung wird dabei sowohl das Bearbeiten und Löschen existierender Pliable Objects als auch das Anlegen neuer Pliable Objects verstanden. Da unter Pliable Objects sowohl Schemaobjekte als auch Instanzen verstanden werden, ist zu berücksichtigen, dass das zu entwickelnde Informationssystem - aus objektorientierter Sicht - sowohl Änderungen an bestehenden Klassendefinitionen als auch am Verhalten von Teilaspekten der Instanzen der Klassen zulassen soll.

Das Informationssystem wird intern auf einer Datenbank arbeiten und selbst über die Methodik zur Bearbeitung der Daten der Datenbank verfügen. Bei diesen Daten wird es sich um Pliable Objects handeln. Demzufolge ist für die Datenbank ein Datenbankschema zu erarbeiten, welches Pliable Objects abbildet.

Um zu dem Informationssystem selbst zu gelangen, werden zunächst einige Überlegungen angestellt. Darin wird das Informationssystem selbst als universales Pliable Object interpretiert, welches Entitäten verwaltet. Jede Entität in der realen Welt soll durch ein Pliable Object abgebildet werden. Die Bildung einer Entity-Menge, die alle Entitäten enthält, kann ebenfalls durch die Bildung einer Menge gewonnen werden, die alle Pliable Objects enthält. Das universale Pliable Object wird mittels dieser Entity-Menge gewonnen.

5.2.1 Entity–Menge E

Die Menge **Entity E** sei die Menge aller Pliable Objects.

$$E = \{\pi \mid \pi \text{ ist ein Pliable Object}\}$$

In der Potenzmenge von E finden sich die drei Mengen M_A , M_X und M_P . Die Menge M_A ist die Menge aller Attribute, die Menge M_X die Menge aller Aktionen und die Menge M_P die Menge aller Regeln. Die Mengen M_A , M_X , $M_P \in P(E)$ sind paarweise disjunkt. Diese Überlegung wird dahingehend interpretiert, dass Attribute, Aktionen und Regeln als separate Teilaspekte zu betrachten und ins Informationssystem einzubringen sind.

Ein Schema-Pliable-Object π^S lässt sich durch die symmetrische Differenz der drei Mengen A, X, P ausdrücken:

$$\pi^S = A \diamond X \diamond P \text{ mit } A \in P(M_A), X \in P(M_X), P \in P(M_P)$$

Die Relation $R_\pi \subseteq A \times X \times P$ ist die Menge aller Instanzen von π . Diese Schlussfolgerung deutet an, dass die Instanzen aller Pliable Objects einen weiteren Teilaspekt des Informationssystems ausmachen sollten.

5.2.2 Entity-Domäne

Durch die Zuordnung der Bezeichnung *Entity* zu der Menge E kann die Entity-Domäne δ_{Entity} gebildet werden:

$$\delta_{\text{Entity}} = (\text{Entity}, E)$$

Die Entity-Domäne ermöglicht deren Nutzung als Domäne eines Attributes. Nach der Bildung der Potenzmenge über der Entity-Domäne gilt:

$$E \in P(\delta_{\text{Entity}})$$

Damit kann die ganze Entity-Menge einem Attribut als Wert zugeordnet werden. Dieses Attribut sei bezeichnet als *Menge aller Pliable Objects*.

$$\alpha_{\text{Menge aller POs}} = (\text{Menge aller Pliable Objects}, \text{Powerset}(\delta_{\text{Entity}}), E)$$

Bei dem Attribut $\alpha_{\text{Menge aller BOs}}$ handelt es sich um ein unwiderrufliches Attribut, d.h. sein Wert – die Zuweisung der Menge E – ist fix und unveränderlich. Dieser Umstand lässt die Interpretation zu, dass alle Informationen (die Pliable Objects betreffend) durch ein Attribut ausgedrückt werden können. Ein Attribut sichert einen Zustand, wobei zur Speicherung von persistenten Zuständen Datenbanken die geeignete Wahl sind. Die Datenbank eines Informationssystems, welche Pliable Objects verwaltet, sollte alle relevanten Pliable-Objects-Eigenschaften in der Datenbank abbilden.

5.2.3 Idee der Betrachtung eines Informationssystems als universales Pliable Object

Ein Attribut erfüllt seinen Zweck, wenn es einem Pliable Object zu eigen ist. Es wird vereinbart, dass das Attribut α_{Menge} aller BOs nur einem einzigen Pliable Object zu eigen sein kann. Dieses Pliable Object sei das universale Pliable Object π_U .

Die Aktionen, die π_U besitzt, realisieren die Mengenaxiome (vgl. 4.2.3) und versetzen das universale Pliable Object in die Lage, Pliable Objects zu erzeugen und mit diesen zu arbeiten. Die neuen Pliable Objects sind als Instanzen zu sehen und können die Fähigkeiten von π_U erweitern. Die neuen Pliable Objects erweitern die Entity-Menge und können ferner über die Aktionen des universalen Pliable Objects bearbeitet werden.

Ein Informationssystem besteht im Wesentlichen aus einer Datenbank (den gespeicherten Informationen) und Anwendungsprogrammen (vgl. [Sa93]). Im Vergleich mit dieser Formulierung wird das universale Pliable Object als Informationssystem verstanden, wobei das Attribut *Menge aller Pliable Objects* den gesamten Datenbestand darstellt und somit durch die Datenbank repräsentiert wird. Die Aktionen, die die Mengenaxiome abbilden, werden als das Anwendungsprogramm verstanden, das auf dem Datenbestand arbeitet.

Es wird mit dem Modell der Pliable Objects nicht das Ziel verfolgt, ein neues Modell für die Softwareentwicklung zu spezifizieren, sondern vielmehr soll ein Konzept realisiert werden, welches mittels des objektorientierten Modells ausgedrückt werden kann. Dieses Konzept soll zur Definition einer Architektur verwendet werden, welche als Grundlage einer Informationssystemspezifikation dienen kann.

Das formale Modell der Pliable Objects ist in Kapitel 4 vorgestellt worden. Die Erkenntnisse daraus können in einem semantischen Modell zusammengefasst werden. Abbildung 35 zeigt das semantische Modell von Pliable Objects als UML-Diagramm.

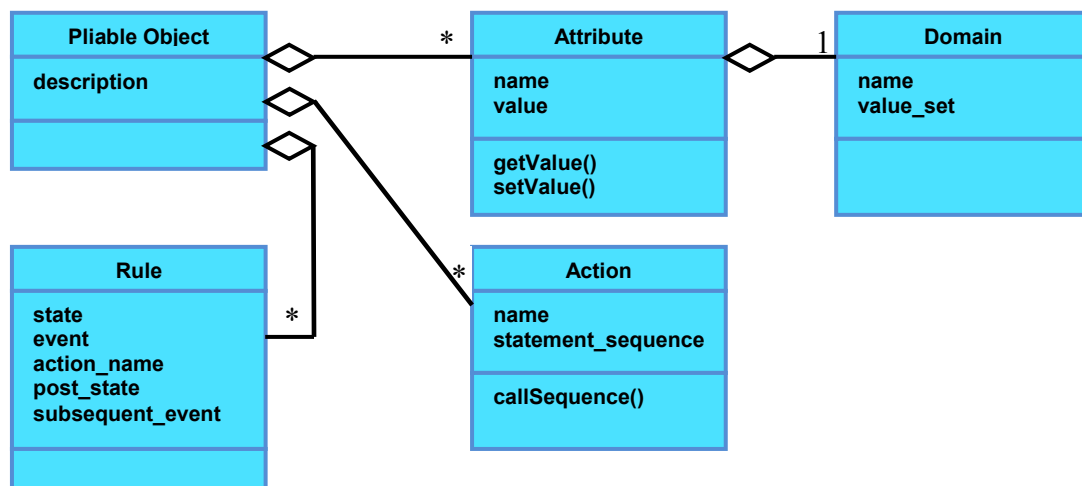


Abbildung 35: Semantisches Pliable-Object-Modell nach [Pr11]

Ein *Pliable Object* wird als Klasse modelliert, der als Komponenten Instanzen von Attributen, Aktionen und Regeln zugeordnet sind. Im Gegensatz zum Objekt des objektorientierten Modells besteht das Pliable Object nicht aus Attributen und Methoden, sondern besitzt jeweils eine Menge von *Attributes* und *Actions*. Die Mengenbetrachtung von *Attributes* und *Actions*

erlaubt einen flexibleren Umgang mit Pliable Objects. Über einfache Mengenoperationen wird es möglich, die Gestalt eines Pliable Objects zu ändern - nicht nur hinsichtlich seiner Eigenschaften, sondern auch in Bezug auf sein Verhalten. Besondere Dynamik erhält ein Pliable Object durch die Zuweisung von *Rules*. Eine Regel erweitert ein Pliable Object um einen Zustand inklusive Transitionen, die Bearbeitungs- oder Verfahrensanweisungen darstellen. Prinzipiell gilt, wenn der Zustand aktiv ist, und das Ereignis eintritt, auf das die Regel reagieren soll, werden die Verarbeitung gemäß den Bearbeitungs- oder Verfahrensanweisungen fortgesetzt. Der Klasse *Attribut* wiederum ist eine Domäne als Komponente zugeordnet. Die Klasse *Domäne* dient zur Verwaltung von beliebigen, d.h. dynamischen Wertebereichen. Mit Hilfe dieser drei Klassen können beliebige Pliable Objects zumindest strukturell angelegt werden. Dies bedeutet, dass mit Hilfe der drei Klassen Schema-Pliable-Objects verwaltet werden können. Die einzelnen Klassen werden im Rahmen dieses Kapitels noch detaillierter vorgestellt werden.

5.3 Datenbankschema für Pliable Objects

Die Grundlage eines Informationssystems bildet eine Datenbank. Diese benötigt ein Datenbankschema, um die anfallenden Informationen verarbeiten zu können. Im Folgenden wird das semantische Modell aus Abbildung 35 in ein Datenbankschema für Pliable Objects überführt werden.

5.3.1 Relationale Abbildung von Pliable Objects

Wesentlicher Vorteil von Pliable Objects ist die Selbstbeschreibung des Pliable-Object-Verständnisses durch sich selbst. Um diesen Sachverhalt konkret ausdrücken zu können, wird das Entity-Relationship-Modell verwendet, da sich die Darstellung von Beziehungen zwischen Entities hierfür eignet.

Aus der formalen Aufbereitung aus Kapitel 4 und aus dem semantischen Modell ergibt sich, dass jedes Pliable Object eindeutig gekennzeichnet und sein Name sowie seine Beschreibungsdomäne hinterlegt werden muss. Alle weiteren Attribute, die dem Pliable Object zu eigen sind, sollten referenziert werden, wofür sich eine Beziehung eignet. Dadurch wäre auch gewährleistet, dass Attribute nicht zwingend einem Pliable Object zu eigen sein müssen oder auch mehreren Attributen zu eigen sein können. Attribute selbst wären als Entities abzubilden, inklusive eines Bezugs zu einer Domäne. Attribute realisieren den Zustand einer Eigenschaft eines Pliable Objects. Die separate Betrachtung als eigenständige Entities erlaubt eine von Pliable Objects losgekoppelte Bearbeitung. Über einen analogen Mechanismus wären Pliable Objects auch Aktionen und Regeln zuzuordnen, die einem oder mehreren Pliable Objects zu eigen sind.

Konkrete Zustände von speziellen Pliable Objects sind durch Instanzen gegeben. Diese werden als eigenständige Entities betrachtet und verwaltet. Generell wäre denkbar, dass Instanzen immer einem Pliable Object zu eigen sind.

Aus diesen Überlegungen heraus kann ein Entity-Relationship-Diagramm (vgl. 2.3) gewonnen werden, welches in Abbildung 36 wiedergegeben ist und welches als Basismodell für ein Datenbankschema dient.

Die zentrale Rolle spielt das Entity-Pliable-Object, dem Attribut-, Aktions- und Regelmengen über die Own-Beziehungen zugeordnet sind. Das Entity-Action besitzt dabei die

Besonderheit, dass es ebenfalls mittels einer Own-Beziehung eine Attributmenge zugeordnet bekommt. Die Attributentitäten bekommen mittels der Own-Beziehung jeweils eine Domäne zugewiesen.

Von besonderer Bedeutung ist die zweite Beziehungsrelation zwischen Pliable Object und Attribut. Mittels der Value-Beziehung werden die Instanzen von Pliable Objects verwaltet. Eine Instanz besitzt eine eindeutige Instanz-ID. Allerdings ist die Speicherung von Instanzen wertorientiert, d.h. in der Value-Relation werden die Attributwerte der Instanz separat hinterlegt. Jedem Wert ist ein dreiattributiger Schlüssel zugeordnet, der sich aus der Pliable-Object-ID, der Attribut-ID und der Instanz-ID zusammensetzt. Mittels dieser ternären Assoziation wird die Is-a-Beziehung zum Pliable Object angegeben, um über dessen Regeln und Aktionen darin gespeicherte Plausibilitätskontrollen zu triggern.

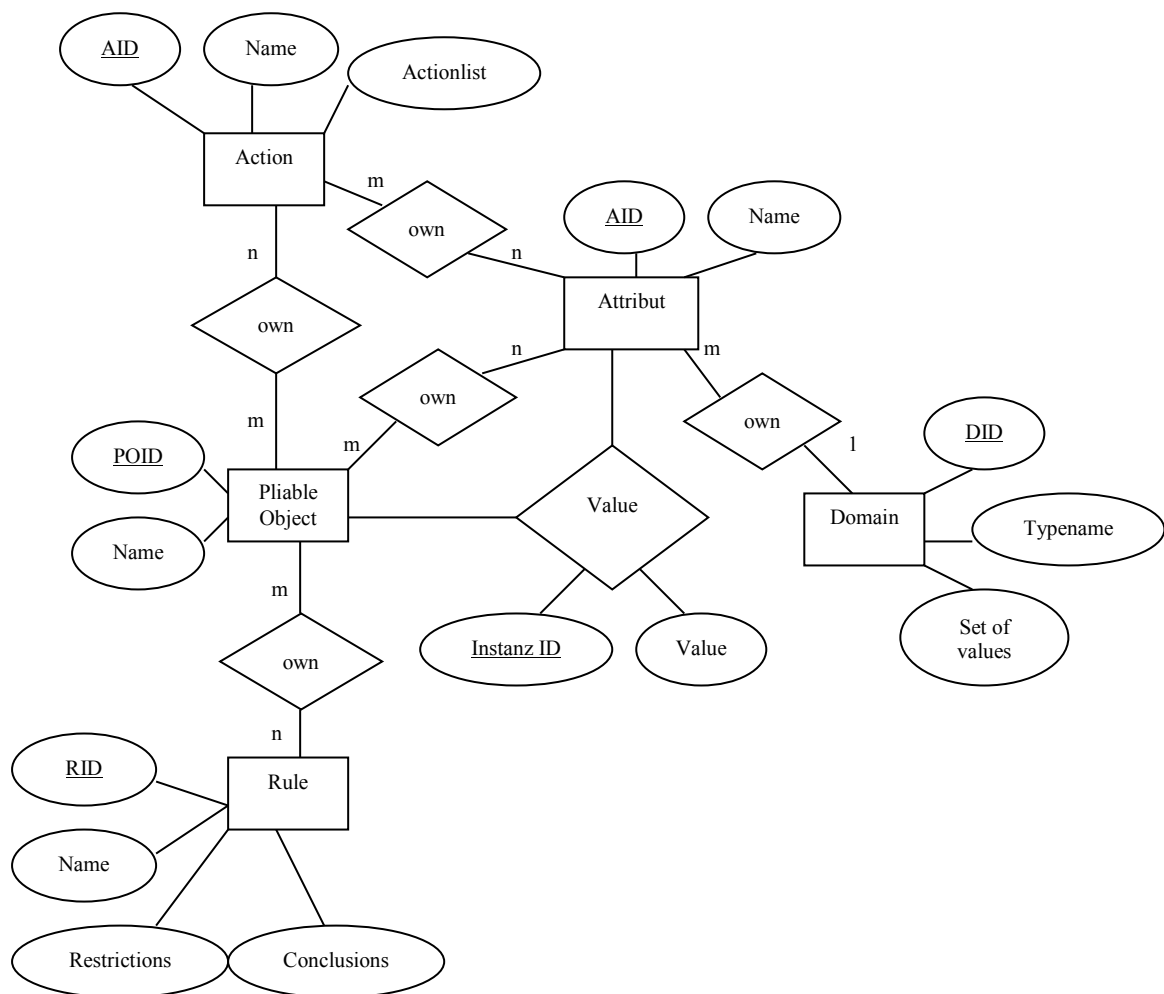


Abbildung 36: ER-Modell für Pliable Objects

Zur Veranschaulichung einer Implementierung von Pliable Objects wird eine relationale Abbildung gewählt (vgl. 2.3). Die tabellenhafte Darstellung erlaubt eine konkretere Sichtweise, die hier einfacher zu präsentieren und auch verständlicher ist.

5.3.2 Relation Domäne

Pliable Objects nehmen Werte auf, über die sich ihr Zustand definiert. Werte werden nach Kapitel 4.2 als gegeben betrachtet. Ihre Repräsentation und Nutzung geschieht durch Bildung von Domänen; diese besitzen einen dynamischen Charakter, so dass eine geeignete Relation realisiert werden muss, die die Verwaltung von Domänen ermöglicht.

Eine Domäne ist ein aus einer Typbezeichnung und einer Wertemenge bestehendes Tupel. Die Typbezeichnung ist eine alphanumerische Bezeichnung, die Wertemenge wird entweder durch eine Aufzählung oder als regulärer Ausdruck gegeben. Um zuzulassen, dass gleichbezeichnete Domänen verwaltet werden können, werden die Domänen generell durch eine eindeutige ID gekennzeichnet. Domänen können damit anwenderspezifisch angelegt werden.

Beispielsweise kennzeichnet das Symbol \mathbb{N} allgemein die Menge der natürlichen Zahlen. Ob die Zahl 0 Element dieser Menge ist, ist Definitionssache des Definierenden. Die Verarbeitung beider Definitionsfälle wäre über die kennzeichnende ID der Domäne möglich.

Eine mögliche Visualisierung der Relation für Domänen zeigt Tabelle 9. Eine SQL-Anweisung zur Erzeugung der Relation ist im Anhang beschrieben.

Domänen		
ID	Typbezeichnung	Wertemenge
1	Domänen	<<alle Domänen>>
2	Zeichenkette	[a-zA-Z0-9]*
3	Zahl	[0-9]*

Tabelle 9: Relation Domäne

Auf den einzelnen Domänen können Operationen definiert werden, die funktionale Zusammenhänge zum Ausdruck bringen. Unter einer Operation kann eine Aktion verstanden werden, weshalb eine Domäne ebenfalls als Pliable Object aufgefasst werden kann.

Die Domäne *Zahl* kann z.B. als Pliable Object betrachtet werden, wobei die vier Grundrechenarten als entsprechende Aktionen hinterlegt sind. In Anlehnung an formale Systeme kann die Addition als folgende symbolersetzen Vorschrift angegeben werden:

Beispiel Addition:

Ersetze zwei mit dem Symbol ‚+‘ verknüpfte Ziffern gemäß

$$\begin{aligned} 0+1 &= 1, \\ 0+2 &= 2, \\ &\dots \\ 9+9 &= 8 \text{ mit Übertrag}, \\ 9+0 &= 9. \end{aligned}$$

Die Realisierung von Berechnungen, wie die Grundrechenarten, in Form solcher Aktionen ist müßig. Da ein Rechner solche Berechnungen vornehmen kann, wird grundsätzlich davon ausgegangen, dass diese Berechnungen anhand der üblichen Symbolik automatisch vom auf den Pliable Objects arbeitenden System erkannt und durchgeführt werden.

5.3.3 Relation Attribut

Zum Kern von Pliable Objects zählen Attribute, da sie Auskunft über den Zustand eines konkreten Pliable Objects, dem sie zugeordnet sind, geben. Attribute werden in einer eigenen Relation hinterlegt. Ein Attribut ergibt sich als Tupel, bestehend aus Name, Domäne und Wert; diese wiederum sind jeweils Attribute der Attributdefinition, wobei alle drei Attribute wortwörtlich geschrieben sind. Folglich werden die Attribute in die Relation aufgenommen. Ihnen wird vorerst die Domäne „Zeichenkette“ zugewiesen.

Attribut			
ID	Name	Domäne (ID)	Wert
1	Name	2	
2	Domäne	2	
3	Wert	2	
4	Aktionsattribute	2	
5	Aktionsdefinition	2	
6	Prämisse	2	
7	Schlussfolgerung	2	
8	Pliabledefinition	2	

Tabelle 10: Relation Attribut

Tabelle 10 visualisiert die Relation der Attribute. In der Relation sind bereits die Attribute der Pliable Objects *Attribut*, *Aktion* und *Regel* mit angegeben, um sie im Folgenden verwenden zu können. Eine SQL-Anweisung zur Erzeugung der Relation ist im Anhang beschrieben. Der Wert (eines Attributs) bleibt bislang unbesetzt. In der Relation dient er zur Angabe vordefinierter Zustände, die als konstant zu betrachten sind. Auf diese Weise ist die Generierung schematisierter Pliable Objects möglich.

5.3.4 Relation Pliable Object

Ein Pliable Object ist das zentrale Entity, welches realisiert werden muss. Alle Pliable Objects werden in der Relation „Pliable Objects“ abgebildet. Als zentrale Relation gibt sie Auskunft über die Existenz von Pliable Objects. Um Pliable Objects voneinander unterscheiden zu können, werden sie durch eine eindeutige ID gekennzeichnet. Zusätzliche Angaben sind der Name und die Beschreibungsdomäne der Pliable Objects. Tabelle 11 visualisiert die Relation. Im Anhang ist eine SQL-Anweisung zur Erzeugung der Relation beschrieben.

Pliable Object		
ID	Pliable-Name	Pliable-Definition
1	Pliable Object	Allgemeines Pliable Object
2	Attribut	Zustand
3	Aktion	Zustandswechsel
4	Regel	Zustandssteuerung

Tabelle 11: Relation Pliable Objects

Gemäß Kapitel 4.7.4 wird von der Existenz der Sonder-Pliable-Objects *Attribut*, *Aktion* und *Rule* ausgegangen. Neben dem allgemeingültigen Pliable Object werden diese Sonder-Pliable-Objects in der Relation hinterlegt. Damit stehen Pliable Objects abstrakt und inklusive Zustands- und Verhaltensbeschreibung im Zugriff.

5.3.5 Relationen der Pliable Objects zu eigenen Attributen

Attribute sind Pliable Objects zu eigen, genauso Aktionen und Regeln. Diese Zuordnungen werden in entsprechenden „Own“-Relationen verwaltet. Tabelle 12 visualisiert diese Relationen. Allgemein werden einem über seinen POID referenzierten Pliable Object in den jeweiligen Relationen die Attribute, Aktionen und Regeln über deren eindeutige ID zugewiesen, die dem Pliable Object zu eigen sind.

Own-Attribut	
POID	Attribut-ID
1	1
1	8
2	1
2	2
2	3
3	1
3	4
3	5
4	1
4	6
4	7
4	9

Own-Action	
POID	Action-ID
1	1

Own-Rule	
POID	Rule-ID
1	1

Tabelle 12: Beziehungsrelationen: links Attribute, rechts Aktionen und Regeln

In der Tabelle für die Attribut-Zuordnungen sind ferner die Verknüpfungen wiedergegeben, die die Sonder-Pliable-Objects rein über die Attribute beschreiben. Mit den bereits vorgestellten Relationen sind alle Pliable Objects damit schon schematisch erfasst.

5.3.6 Relation Value

Neben der schematischen Erfassung erlauben die bereits vorgestellten Relationen die Erzeugung von neuen Schema Pliable Objects; einfach durch Anlegen neuer Pliable Objects in der zugehörigen Relation, sowie das Anlegen der Attribute, Aktionen und Regeln der neuen Pliable Objekts inklusive der Erstellung der entsprechenden Verknüpfungen über die own-Relationen.

Um für die Schema-Pliable Objects auch Instanzen hinterlegen zu können, wird die Relation Value genutzt. In dieser Relation werden alle Instanzen hinterlegt, die im Informationssystem zu speichern sind. Eine SQL-Anweisung zur Erzeugung der Relation ist im Anhang beschrieben.

Value			
ID	POID	AttributID	Wert
1	1	1	Protokoll

Tabelle 13: Relation Value

Tabelle 13 skizziert den Aufbau der Relation *Value*. Darin werden die Instanzen von Pliable Objects verwaltet. Die Relation stellt in gewisser Weise eine Art Werteliste dar. Über die Werte wird der Zustand eines Pliable Objects zum Ausdruck gebracht. Eine Instanz selbst ist einzigartig und von anderen Instanzen unterscheidbar, selbst wenn die Instanzen den gleichen

Zustand besitzen. Die Einzigartigkeit einer Instanz wird in der Relation durch eine eindeutige ID gewährleistet. Jede Instanz kann einem Pliable Object schematisch zugeordnet werden. Die Instanz ist mit dem Pliable Object assoziiert, das durch die angegebene POID referenziert wird. Abschließend wird die Instanz rein werttechnisch erfasst, d.h. jeder aktuell angenommene Einzelwert eines jeden Attributs wird gespeichert. Die eindeutige Attribut-ID referenziert das Attribut, während unter *Wert* der zu speichernde Wert abgelegt wird.

Das Modell der Pliable Objects dient zunächst dazu, um Informationssysteme zu beschreiben, die vorrangig protokollarische Unterstützung geben. Die zu speichernden Informationen sollten von Personen gelesen werden können. Um dies zu gewährleisten, wird für die Werte der Datentyp der Zeichenkette verwendet. Dies stellt sicher, dass alle eingegebenen Werte gespeichert werden können.

Hinsichtlich des Wertes gilt die Richtlinie, dass der Wert aus der Domäne des referenzierten Attributes stammen sollte. Allerdings kann diese Richtlinie auch gebrochen werden. Falls die Person, die die Daten eingibt, es wünscht, einen anderen Wert für ein Attribut einzugeben, besteht durch die Relation Value dazu die Möglichkeit.

Die Attribut-ID hat in der Relation einen starken Bezug zur Instanz. Die Instanz referenziert zwar ein Schema Pliable Object, allerdings muss das Attribut nicht dem Schema Pliable Object zu eigen sein. Das Instanz Pliable Object bekommt über die Relation Value die Möglichkeit, zusätzliche Attribute zugewiesen zu bekommen. Von einem anderen Standpunkt aus gesehen, kann das Schema Pliable Object bearbeitet werden, dass z.B. ein Attribut entfernt wird. Eine derartige Änderung hat auf die Instanz bzw. auf alle Instanzen des Schema Pliable Objects keine Auswirkungen. Die Instanzen bleiben davon unberührt.

Die Referenz zu einem Pliable Object in der Relation erlaubt das Ändern der damit realisierten Is-a-Beziehung. Eine Instanz hat keinen starren Bezug mehr zu einem konkreten Pliable Object. Die Beziehung kann dahingehend geändert werden, dass die Instanz einem anderem (Schema) Pliable Object zugeordnet wird.

5.3.7 Relation Action

Die bisher vorgestellten Relationen dienen einerseits dazu, Pliable Objects selbst zu beschreiben und andererseits, um Zustände bestimmter Pliable Objects zu sichern. Das Ändern von Zuständen geschieht durch Aktionen. Normalerweise werden Zustandsänderungen durch implementierten Code innerhalb der Applikation vorgenommen, der auf der Datenbank arbeitet (vgl. [Be96]). Im Rahmen von Pliable Objects werden anstelle von implementiertem Code Aktionen spezifiziert, die ebenfalls in der Datenbank zu speichern sind. Gemäß Kapitel 4.5 sind Aktionen durch eine Liste von Statements gegeben, die inhaltlich die einer kontextfreien Grammatik genügen.

Wichtig ist dabei, dass die Aktionen nur in der Datenbank gespeichert werden. Sie selbst nehmen dadurch keinerlei Zustandsänderungen in der Datenbank vor. Dies obliegt nach wie vor der auf der Datenbank arbeitenden Applikation. Die Aktionen beschreiben ausschließlich die Zustandsänderung in notationeller Form. Er wird vorausgesetzt, dass das Datenbankmanagementsystem die Applikation dabei unterstützt, die Aktionen auszulesen, damit die Applikation die in den Aktionen beschriebenen Zustandswechsel vornehmen kann, d.h. die Applikation verfügt über die Funktionalität, die in der kontextfreien Grammatik gegebenen Anweisungen auszuführen.

Dies ermöglicht Pliable Objects, dass keine spezielle kontextfreie Grammatik definiert werden muss. Je nach Applikation kann applikationsspezifisch eine kontextfreie Grammatik gewählt werden, die im Rahmen der Aktionen von Pliable Objects verwendet werden soll.

Für die Speicherung von Aktionen in einer Datenbank wird eine Relation verwendet, wie sie Tabelle 14 zeigt.

Action			Own	
ActID	Name	Statement sequence	Action-ID	Attribut-ID
1	PO-Schematabestimmung	SELECT POID, Name FROM Pliable Object	1	1

Tabelle 14: Relation Action

Eine Aktion ergibt sich als Tupel, bestehend aus Name, Aktionsattributen und Aktionsliste. Der Name und die Anweisungsliste werden zusammen mit einer eindeutigen ID in der Relation Action gespeichert. Die Aktionsattribute werden in der separaten Own-Relation referenziert, indem in dieser Relation für jedes verwendete Attribut dessen Attribut-ID zusammen mit der Aktions-ID eingetragen wird.

5.3.8 Relation Rule

Eine Regel dient gemäß Kapitel 4.6 zur Steuerung von Zuständen bzw. Zustandswechseln. Durch die Zuweisung von Regeln zu Pliable Objects erhalten diese eine besondere Dynamik. Regeln repräsentieren einen speziellen Teil des (eventuell möglichen) Verhaltens von Pliable Objects.

Allgemein besteht eine Regel aus einer Prämisse bzw. Bedingung, die mit einer Konklusion bzw. Schlussfolgerung verknüpft ist. Dies bedeutet, dass mit Hilfe von Regeln bestimmte Sachverhalte ausgedrückt werden, die einen steuernden Charakter besitzen. Praktisch betrachtet sollen Regeln ein Pliable Object in die Lage versetzen, auf bestimmte äußere Ereignisse in situationsspezifischer Weise zu reagieren. In der UML werden derartige Verhaltensweisen mittels Zustandsdiagrammen modelliert (vgl. Kapitel 2.4.2). Bei Pliable Objects werden die Verhaltensweisen allerdings nicht nur modelliert, sondern sollen direkt beim Pliable Object hinterlegt werden (vgl. Kapitel 4.5). In der praktischen Umsetzung bedeutet dies, dass die Regeln ein Pliable Object um eine Zustandsmaschine erweitern.

Zustandsmaschinen

Allgemein wird unter einer Zustandsmaschine eine Menge von Zuständen verstanden, denen jeweils eine Menge von Transitionen zugeordnet ist. Eine Transition definiert einen Zustandswechsel derart, dass in einem aktiven Zustand ein Ereignis auftritt, welches zu einer Verarbeitung führt. Die Verarbeitung wiederum führt eventuell dazu, dass ein anderer Zustand zum aktiven Zustand wird.

Das interne Verhalten von Pliable Objects soll über eine Zustandsmaschine zum Ausdruck gebracht werden. Diese Zustandsmaschine spezifiziert die möglichen Auswirkungen, die ein einzelnes Pliable Object besitzt, wenn es in einen bestimmten Zustand tritt. Die Zustandsmaschine selbst ist durch alle Zustände gegeben, die das Pliable Object von seiner

Erzeugung bis zu seiner Löschung und/oder während der Ausführung einer Aktion der Reihe nach einnehmen kann. Diesen Sachverhalt skizziert Abbildung 37. Ein einzelner Zustand der Zustandsmaschine ist als aktiver Zustand gekennzeichnet.

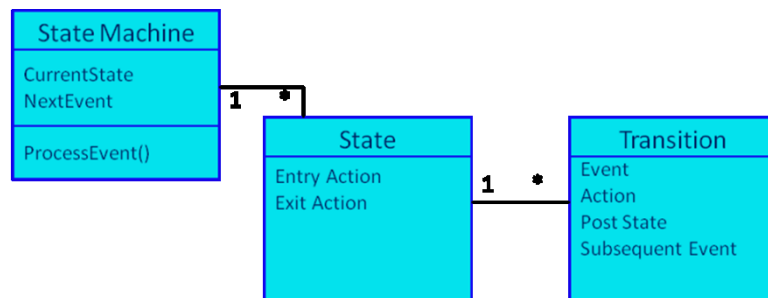


Abbildung 37: UML-Diagramm einer Zustandsmaschine

Ein Zustand charakterisiert eine bestimmte Situation. Er wird durch ein eigenschaftsloses Objekt abgebildet. Der Zustand verfügt ausschließlich über zwei Aktionen. Die *Entry*-Aktion ist diejenige Aktion, die aufzurufen ist, wenn der Zustand zum aktiven Zustand der Zustandsmaschine wird, während die *Exit*-Aktion aufzurufen ist, wenn der Zustand seinen Status als aktiven Zustand verliert.

Jedem Zustand ist eine Menge an Transitionen zugeordnet, die die Reaktion auf bestimmte äußere Ereignisse beschreiben und einen möglichen Zustandsübergang auslösen. Eine Transition ist gegeben aus dem Ereignis, auf welches die Transition reagieren soll, einer Aktion, die aufzurufen ist, wenn das Ereignis eintritt, und einem Folgezustand, den die Zustandsmaschine aktivieren soll, wenn die Aktion der Transition abgearbeitet worden ist. Optional kann ein Folgeereignis angegeben werden, welches zu verarbeiten ist, wenn der Folgezustand aktiviert worden ist.

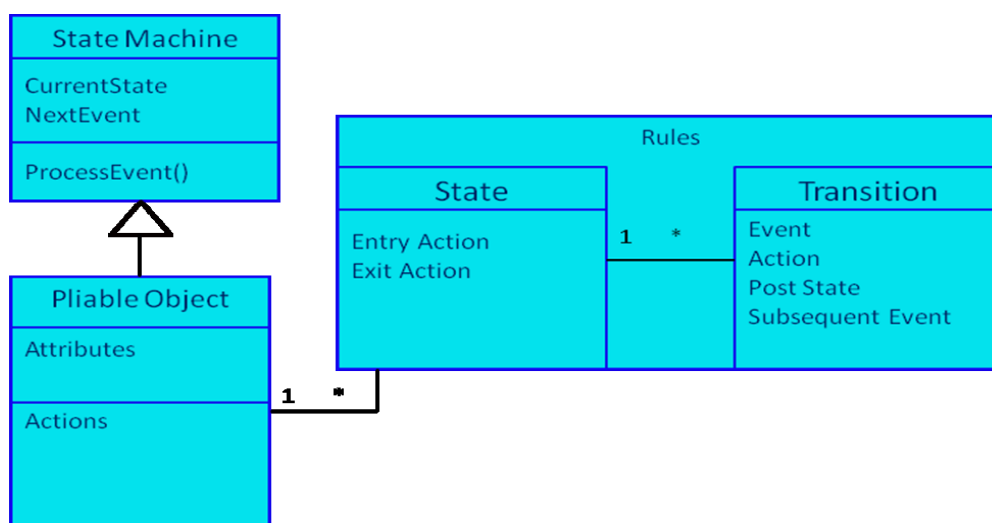


Abbildung 38: Integration der Zustandsmaschine ins Pliable Object Klassenmodell

Pliable Objects zeichnen sich durch Attribute, Aktionen und Regeln aus. Während die Abbildung von Attributen und Aktionen bereits vorgestellt worden ist, sollen für die Umsetzung der Regeln Zustandsmaschinen verwendet werden. Diese Zustandsmaschinen sollen dabei einen ähnlichen dynamischen Charakter besitzen wie Attribute und Aktionen, d.h. sie sollen erweiterbar sein.

In Analogie zu obiger Klassenbeschreibung einer Zustandsmaschine wird zunächst die Integration der Zustandsmaschine in Pliable Objects vorgenommen. Abbildung 38 zeigt die Annahme, dass sich ein Pliable Object von der Klasse der Zustandsmaschine ableitet. Für die Klasse der Zustandsmaschine sind keine Zustände und Transitionen vorgegeben. Die Regeln des Pliable Objects werden durch Zustände und Transitionen repräsentiert, so dass das Pliable Object als Zustandsmaschine durch seine Regeln die Zustandsmaschine ausprägt.

Jede Regel erweitert ein Pliable Object um einen Zustand inklusive Transitionen, die Bearbeitungs- oder Verfahrensanweisungen darstellen, wenn der Zustand aktiv ist, und das Ereignis, auf das die Regel reagieren soll, eintritt. Die relationale Abbildung von Regeln zeigt Tabelle 15. Im Anhang sind SQL-Anweisungen zur Erzeugung der Relationen beschrieben.

Event	Rule					
Event-ID	POID	State-ID	Event-ID	Action-ID	PostState-ID	SubsequentEvent-ID
E_NoPO	1	S_Create	E_NoPO	1	S_Create	E_Selected
E_Selected						

Tabelle 15: Regelspezifische Relationen

Zunächst sind in der Relation Event alle Ereignisse zu definieren, die allgemein spezifiziert werden. Die Relation Rule bildet die Grundlage der Regelabbildung. In ihr werden alle Zustände des zugeordneten Pliable Objects über eine State-ID definiert. Für jede Transition wird das Eintrittsereignis durch die Event-ID, die aufzurufende Action durch die Action-ID, der Folgezustand durch die PostState-ID und ein eventuelles Folgeereignis durch die SubsequentEvent-ID referenziert. Das Tupel POID, State-ID und Event-ID dient der Relation als Primärschlüssel.

Nach Kapitel 4.6 ist eine Regel aus Restriktion und Konklusion definiert. Mit obiger Relation bildet der Primärschlüssel die Restriktion ab, während die Konklusion durch die aufzurufende Action, den anzunehmenden Folgezustand und das eventuell angegebene Folgeereignis gegeben ist. Auf diese Weise werden alle Regeln eines Pliable Objects in Angaben der Zustandsmaschine überführt.

5.4 Generische Verhaltensabbildung von Pliable Objects

Das oben vorgestellte Datenbankschema stellt eine Verwaltungsstruktur dar, welche von einem Datenbank-Managementsystem verwendet werden kann, um darin das Datenmaterial eines Realitätsausschnitts zu erfassen (vgl. Kapitel 2.1). Eine die Datenverarbeitungsebene bildende Softwareapplikation kann die Daten verwenden, um sie zu verarbeiten und auszuwerten. Für die Interaktion mit einem Bediener stellt die Softwareapplikation ein Interface bereit.

Die Softwareapplikation muss über das Zusammenspiel mit dem Datenbank-Managementsystem die Pliable Objects verarbeiten, d.h. die Zustände der Pliable Objects erfassen und unter Verwendung von deren Aktionen die Verarbeitung unter Berücksichtigung deren Regeln vornehmen. Zur Unterstützung der Verarbeitung können generische Aktionen erarbeitet werden, die der Softwareapplikation zu Verfügung stehen.

5.4.1 Allgemeingültige Kontrollflussschleife mit generischer Ausprägung

Zunächst muss die Softwareapplikation in die Lage versetzt werden, auf die Zustände eines Pliable Objects zu reagieren, wobei die Reaktion durch die Regeln des Pliable Objects gegeben ist. Die Berücksichtigung der Zustände und die Kontrolle der Reaktion kann durch einen einzelnen Algorithmus zum Ausdruck gebracht werden. Dieser Algorithmus realisiert das Verhalten einer allgemeingültigen Zustandsmaschine. Das Verhalten ist in Abbildung 39 wiedergegeben.

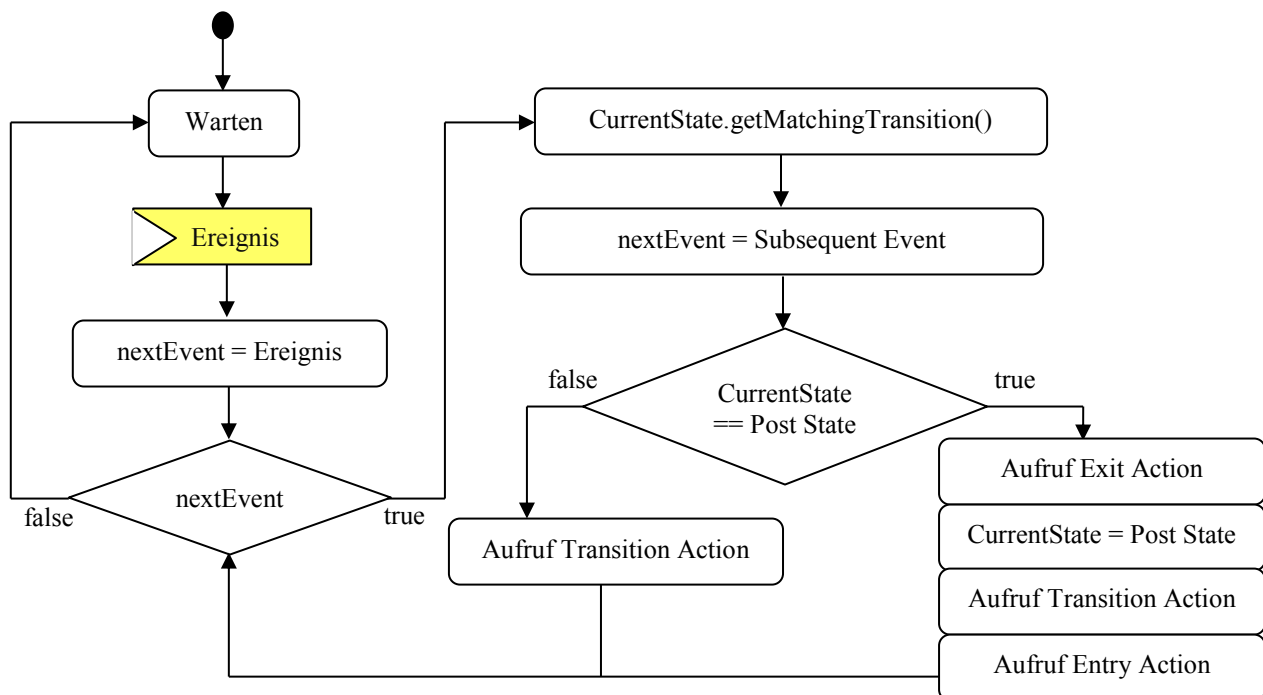


Abbildung 39: Aktivitätsdiagramm zur allgemeingültigen Verarbeitung der Zustandsmaschinen, die durch Pliable Objects ausgeprägt werden

Der Algorithmus selbst realisiert prinzipiell wieder eine abstrakte Zustandsmaschine, die nur zwei Zustände kennt, entweder der Algorithmus wartet auf ein Ereignis oder er verarbeitet ein Ereignis. Der Ausgangs- oder Startzustand ist der Wartezustand, in welchem zusätzlich ein aktiver Zustand eines Pliable Objects referenziert wird. Das Referenzieren bzw. die Kennzeichnung eines Zustands als aktiven Zustand bedeutet, dass alle auftretenden Events nur mit den Transitionen dieses Zustandes verglichen werden.

Wenn ein Ereignis eintrifft, beispielsweise getriggert über das Bedieninterface der Softwareapplikation, wird das Ereignis verarbeitet. Dazu wechselt der Algorithmus in den Verarbeitungszustand. Die Verarbeitung beginnt mit der Bestimmung der Transition-Action. Diese Action ist im aktiven Zustand des referenzierten Pliable Objects mit dem Ereignis verknüpft. Anschließend wird das Folgeereignis aus der Transition bestimmt, also das Ereignis, das nach Beendigung des Aufrufs der Action zu senden bzw. zu verarbeiten wäre.

Die Hauptverarbeitung unter Verwendung von Aktionen zur Durchführung von Zustandswechseln startet mit der Überprüfung, welcher Zustand nach dem Action-Aufruf als aktiv zu markieren ist. Falls es sich um den Zustand handelt, der aktuell aktiv markiert ist, wird nur die Action aufgerufen und anschließend das Folgeereignis verarbeitet. Es sei angemerkt, dass die Action die Option besitzt, das Folgeereignis zu überschreiben, um damit eine Abweichung vom Standardablauf zu initiieren.

Falls ein anderer Zustand als aktiv markiert werden soll, wird zunächst die eventuell angegebene Exit-Action des derzeit aktiven Zustands aufgerufen. Es folgen die Aktiv-Markierung des neuen Zustands und der Aufruf der Transition-Action. Abschließend wird die Entry-Action des neuen Zustands aufgerufen und ein eventuell vorliegendes Folgeereignis verarbeitet.

Die Verarbeitung stoppt, wenn kein Folgeereignis mehr vorliegt. In diesem Fall kehrt der Algorithmus in den Ausgangszustand zurück

5.4.2 Generische Prozesse

Basierend auf den vorgestellten Relationen und der allgemeingültigen Kontrollflussschleife können Aktionsfolgen spezifiziert werden. Eine einzelne, spezifizierte Aktionsfolge definiert einen Prozess. Eine einzelne Aktion dient der Beschreibung eines Zustandswechsels. Zustandswechsel werden durch die Verwendung der Ausführungsfunktionalität (vgl. 4.5.2) in Verbindung mit jeweils einer Aktion vorgenommen. Ein als Aktionsfolge gegebener Prozess realisiert folglich eine Kette von Zustandswechseln.

Ein Prozess wird zielführend definiert, d.h. die Kette von Zustandswechseln, die den Prozess spezifizieren, soll bei der Durchführung der Zustandswechsel zu einem beabsichtigten Ziel führen. Im Normalfall besitzt der Prozess einen anwendungsspezifischen Zweck, d.h. er bewirkt etwas Konkretes und dieses Konkrete ist das beabsichtigte Ziel. Die Beabsichtigung des Prozesses bzw. dessen gewünschte, konkrete Wirkung, ist in einer präzisen Beschreibung angegeben. Sie wird wiederum durch die Verkettung von Aktionen realisiert.

Eine Anwendung umfasst in der Regel mehr als einen einzelnen Prozess. Generell kann angenommen werden, dass sich mehrere präzise Beschreibungen verschiedener Prozesse paarweise unterscheiden, so dass keine Beschreibung redundant vorliegt. Auch wenn die Beschreibungen unterschiedlichen und konkreten Charakter besitzen, so können sie doch nach ihrem Zweck klassifiziert werden. Der Zweck ist allgemein charakterisierbar, da er ein allgemeines Verhalten besitzt, welches durch die Beschreibung speziell geprägt wird.

Allen Prozessen ist gemein, dass sie ausgeführt werden können. Ein Prozess existiert nicht ohne die Existenz eines Ausführungscharakters. Bei einer Ausführung kann unterschieden werden, ob es sich um eine

- Erzeugung,
- Löschung,
- Veränderung,
- Ausführung,
- Verarbeitung,
- Erfassung oder
- Überwachung

von Pliable Objects handelt. Die Klassifikation von Prozessen nach einer derartigen Zweckbestimmung führt zu der Annahme, dass für jeden einzelnen bestimmbar Zweck, gegeben als eine Klassifikation, ein einzelner Prozess ausprägbar ist. Dieser ausprägbare Prozess besitzt einen generischen Charakter, da sich der Prozess an dem Pliable Object orientiert, welches er gerade bearbeitet. Dies bedeutet, dass je nach genannter Klassifikation ein einzelner Prozess beschrieben werden kann, welcher in Abhängigkeit von einem gewählten Pliable Object seine Ausprägung findet.

Erzeugungsprozess

Der Erzeugungsprozess dient dem Anlegen neuer Pliable Objects. Dabei kann es sich um eine neue Instanz eines konkreten schematisch bekannten Pliable Objects, wie z.B. ein neues Attribut, handeln oder um ein neues Schema Pliable Object. Der Erzeugungsprozess realisiert die Hinzufüge-Operation **INSERT**, die eine gegebene Menge um ein neues Element erweitert. Die Erzeugung von Pliable Objects selbst reduziert sich auf das **Anlegen** der Pliable Objects in den Relationen.

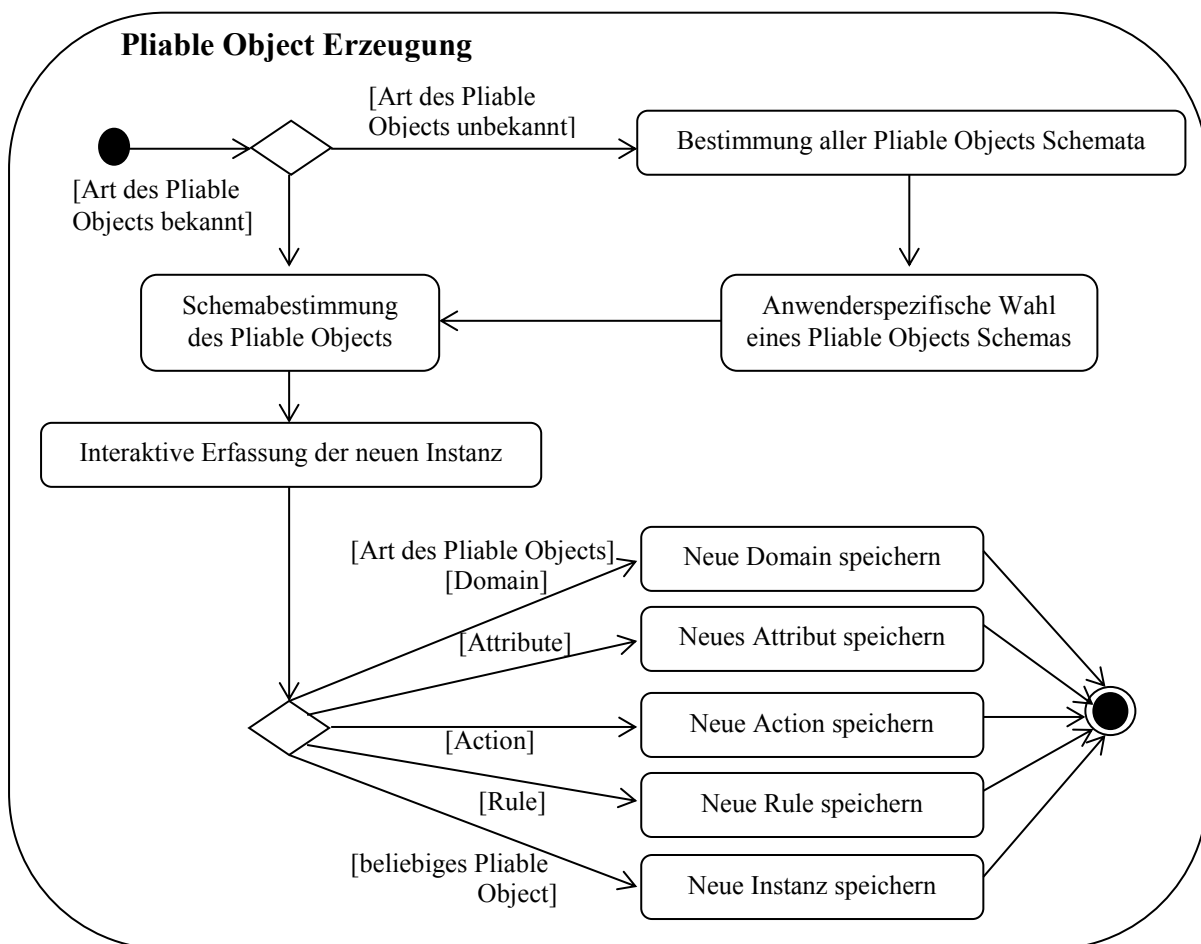


Abbildung 40: Darstellung des Erzeugungsprozesses

Die Funktionalität des Prozesses wird in Abbildung 40 skizziert. Falls das anzulegende Object unbekannt ist, beginnt der Prozess mit der Bestimmung aller Schema Pliable Objects, d.h. mit dem Auslesen der Relation *Pliable Object*. Aus den zur Verfügung stehenden Pliable Objects wählt der Anwender jenes Schemaobject aus, zu welchem er eine Instanz anzulegen wünscht.

Anschließend werden die dem gewählten Schema Pliable Object zu eigenen Attribute bestimmt und der Anwender gibt die gewünschten Werte für die Attribute ein. Nach der Erfassung wird die neue Instanz als Pliable Object gespeichert. Bei der Speicherung wird unterschieden, ob es sich bei der neuen Instanz um eine neue Domäne, ein neues Attribut, eine neue Aktion, eine neue Regel oder eine Instanz eines beliebigen Pliable Objects handelt. Der Prozess trägt beim Speichern der neuen Instanz die Sorge dafür, dass die Informationen in den richtigen Relationen gespeichert werden.

Löschungsprozess

Der Lösungsprozess löscht ein existentes Pliable Object und realisiert entsprechend die Entferne-Operation **DELETE**. Der Prozess selbst ist in Abbildung 41 visualisiert. Ist die zu löschende Instanz noch nicht bekannt, beginnt das Löschen mit der Bestimmung aller Schema Pliable Objects, d.h. mit dem Auslesen der Relation *Pliable Object*. Die Schema Pliable Objects dienen in diesem Fall als Klassifikation auf sämtliche Instanzen in der Datenbank. Die Auswahl eines Schemas stellt den Zugriff auf dessen Instanzen zur Verfügung. Aus den Instanzen des Schema Pliable Objects wählt der Anwender jene Instanz aus, welche er zu löschen wünscht. Anschließend wird die Instanz gelöscht. Dabei wird erneut berücksichtigt, ob es sich bei der Instanz um eine Domäne, ein Attribut, eine Aktion, eine Regel oder eine Instanz eines anderweitig definierten Pliable Objects handelt.

Aus Sicht des Informationssystems vernichtet der Lösungsprozess Informationen. Der entstehende Informationsverlust kann zu Inkonsistenzen führen. In Abbildung 41 ist auf die Prüfung möglicher Inkonsistenzen verzichtet worden. Allgemein wird gefordert, dass das Informationssystem trotz existierenden Inkonsistenzen im Datenbestand weiter zur Verfügung steht. Hinsichtlich der Löschung rückt die Verantwortung des Anwenders in den Vordergrund. Wenn ein Anwender, ausgestattet mit allen notwendigen Berechtigungen, die Löschung einer Information will, dann ist diese Löschung nicht zu verweigern, auch wenn dadurch das Informationssystem zu Schaden kommt.

Mit dem Lösungsprozess ist beispielsweise auch die Löschung von Basisinformationen, wie z.B. dem Schema Pliable Object *Attribut* möglich. Anschließend könnte das Informationssystem keine Eigenschaften mehr verarbeiten. Dies käme quasi der Streichung des Eigenschaftsgedankens aus dem menschlichen Bewusstsein selbst gleich. Die Frage, ob ein Mensch ohne Merkmale und ohne Verständnis von Merkmalen existieren kann, kann nur die Philosophie beantworten und wird an dieser Stelle nicht weiter betrachtet.

Bei dem Lösungsprozess handelt es sich um eine Art Basisaktion, die je nach Anwendungsfall angepasst werden kann. Durch die Definition entsprechender Regeln, die der Prozess vor der Löschung prüft, kann die Löschung eingeschränkt oder nicht erlaubt werden. Die Löschung z.B. des Attributes *Name* sollte verhindert werden, da andernfalls keine Bezeichnung mehr ausgewertet werden kann. Eine Vorgabe für solche einschränkende Regeln könnte an dieser Stelle getroffen werden, allerdings muss einem Anwender die Freiheit zugestanden werden, solche Vorgaben zu ändern oder sogar zu löschen. Um dem Anwender die größtmögliche Freiheit zu lassen, wird von einer Vorgabe abgesehen und der Prozess nur abstrakt betrachtet

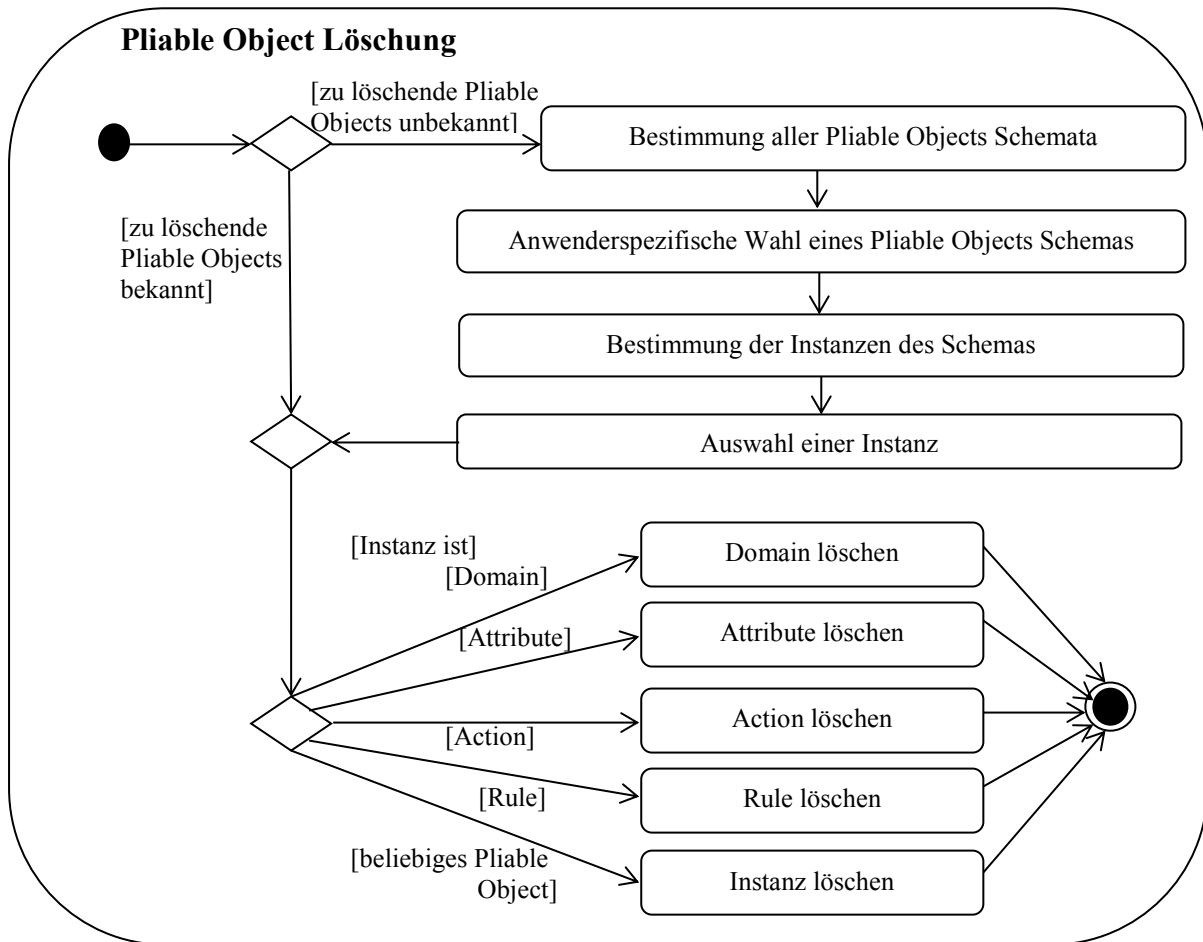


Abbildung 41: Darstellung des Lösungsprozesses

Veränderungsprozess

Der Veränderungsprozess realisiert die Bearbeitung von Pliable Objects. Die Bearbeitung besitzt zwei Ausprägungen. Zum einen kann der Zustand eines Pliable Object rein auf die Wertigkeit seiner Attribute bezogen verändert werden, zum anderen kann der Zustand der Attribut-, Aktions- und Regelmengen des Pliable Objects bearbeitet werden. Der Veränderungsprozess dient auch der Manipulation von Pliable Objects und dem Herstellen von Verknüpfungen dergestalt, dass durch diesen Prozess Pliable Objects um Attribute, Aktionen und Regeln erweitert oder um diese vermindert werden. Über den Veränderungsprozess werden Gebundenheiten und Verknüpfungen realisiert. Er kann folglich auch auf den Relationen, die die Zuordnung realisieren, arbeiten.

Abbildung 42 zeigt den Veränderungsprozess. Falls das zu verändernde Pliable Object noch nicht bekannt ist, beginnt der Prozess mit der Bestimmung aller Schema Pliable Objects, d.h. mit dem Auslesen der Relation *Pliable Object*. Die Schema Pliable Objects dienen erneut als Klassifikation auf sämtliche Instanzen in der Datenbank. Die Auswahl eines Schemas stellt den Zugriff auf dessen Instanzen zur Verfügung. Aus den Instanzen des Schema Pliable Objects wählt der Anwender jene Instanz aus, welche er zu bearbeiten wünscht. Auf der zu bearbeitenden Instanz können sowohl die Attributwerte verändert werden, als auch die Anzahl der Attribute, als auch die Anzahl der Aktionen, als auch die Anzahl der Regeln. Die

Änderung der Anzahl auf den Attribut-, Aktions- und Regelmengen geschieht auf Basis der Gebundenheit / Verknüpfung. Mit der Durchführung der Veränderung wird vorausgesetzt, dass der neue Zustand der Pliable Object Instanz im Informationssystem gespeichert wird.

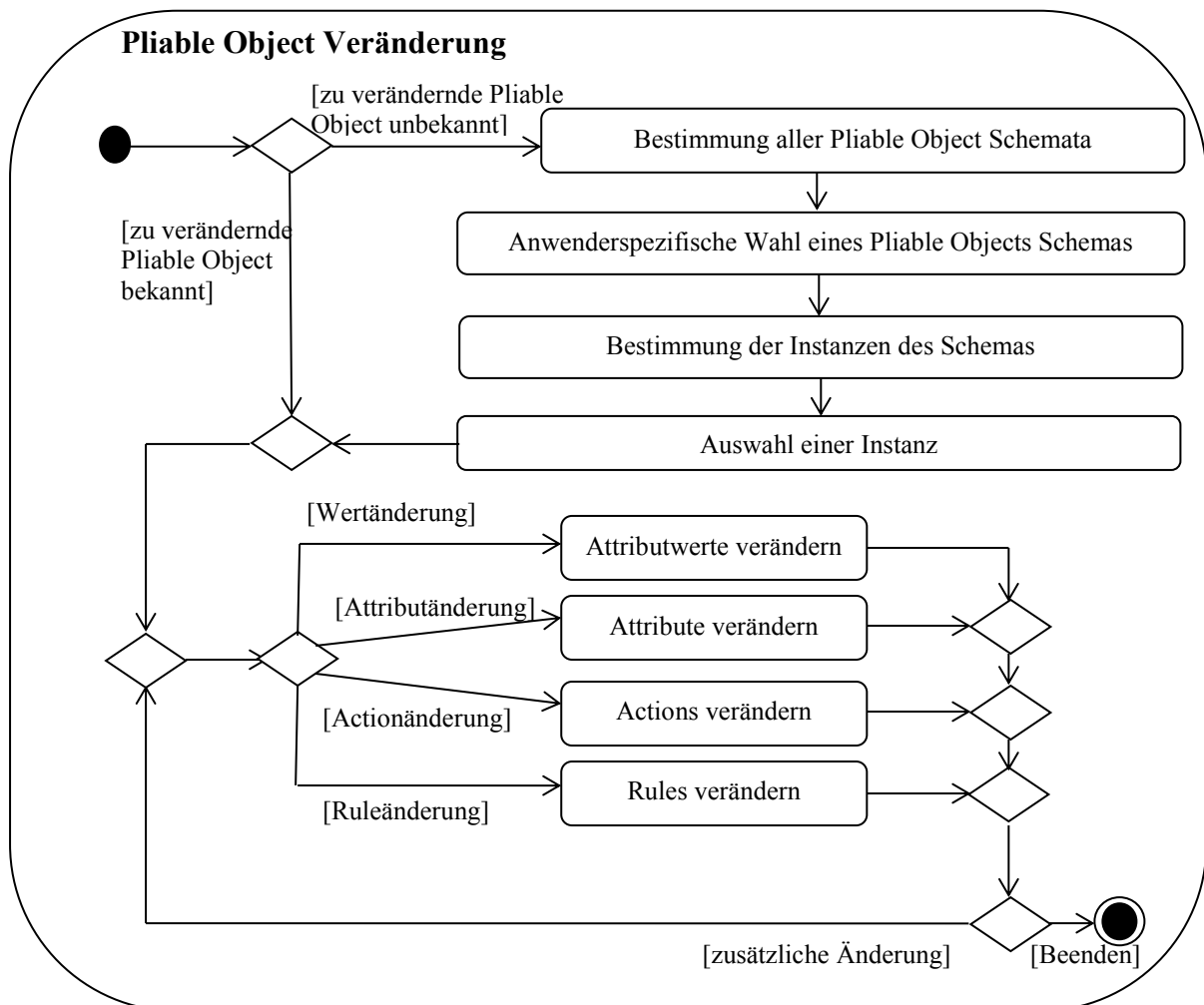


Abbildung 42: Darstellung des Veränderungsprozesses

Erfassungsprozess

Die bisherigen Prozesse setzen auf eine überwiegende Interaktion mit dem Anwender. Ohne Dateneingaben seitens der Anwender können auch keine weiteren Informationen in das Informationssystem aufgenommen werden. Um eine Interaktion konkreter zu unterstützen, soll die Interaktion durch einen speziellen Erfassungsprozess automatisiert werden. Der Erfassungsprozess ist für die Interaktion mit den Anwendern verantwortlich und kann von den weiteren Prozessen eingebunden werden. Die weiteren Prozesse werden dadurch von der Verwaltung der Interaktion befreit.

Der Erfassungsprozess startet eine Bedienoberfläche, über die die Anwenderinteraktion vorgenommen wird. Die Daten, die ein Anwender eingibt, werden gespeichert. Der Erfassungsprozess dient vorrangig der Generierung eines Pliable-Object-Zustandes. Nach der Auswahl einer bekannten Instanz eines Pliable Objects werden die Attributwerte dieser Instanz erfasst. Der Erfassungsprozess dient dem Anlegen gültiger Zustände von Pliable

Objects. Im Rahmen der Interaktion werden eventuell die weiteren Prozesse getriggert, vorausgesetzt, die Triggerung ist im Rahmen der Interaktion zugelassen. Nach Beendigung der Interaktion wird auch die Bedienoberfläche (GUI) beendet.

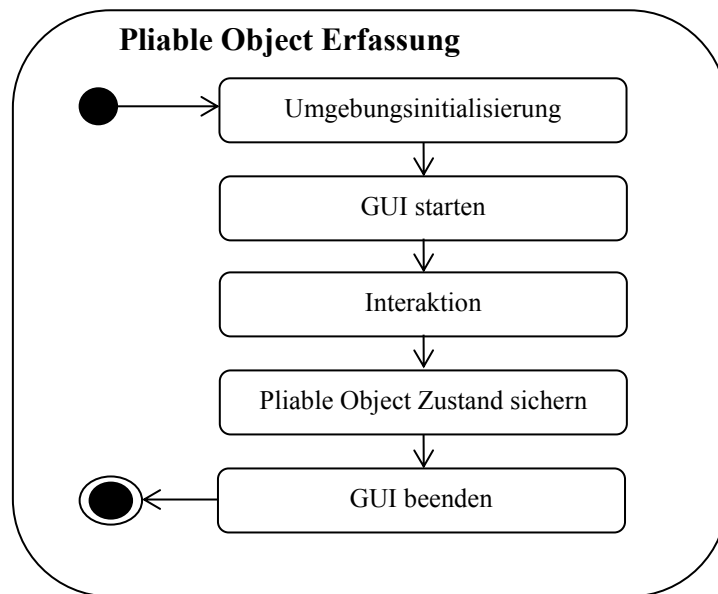


Abbildung 43: Darstellung des Erfassungsprozesses

Abbildung 43 zeigt den Erfassungsprozess. Dieser initialisiert die Umgebung für die graphische Präsentation und startet die graphische Bedienoberfläche (GUI). In der GUI werden die Daten durch den Anwender erfasst. Nach Beendigung der Datenerfassung werden die Informationen in einem Pliable Object gespeichert. Bei der Speicherung ist es unerheblich, ob der Zustand des Pliable Objects gültig ist. Es wird vorausgesetzt, dass der Anwender die Daten so zu speichern wünscht, wie er sie eingegeben hat. Diese Anforderung muss zugesichert werden, für den Fall, dass ein Anästhesist beispielsweise seine Dateneingabe unterbrechen muss. In solch einem Fall sind noch nicht alle Daten im System hinterlegt, so dass ein Pliable Object als inkonsistent gelten kann. Um dem Anwender die Option zu bieten, seine Datenerfassung zu einem späteren Zeitpunkt fortzusetzen, muss auch der inkonsistente Zustand des Pliable Objects gespeichert werden können.

Das Starten einer Bedienoberfläche bedeutet nicht unbedingt, dass generell nur eine einzige GUI zur Verfügung steht. Die Möglichkeit, dass eine Bedienoberfläche speziell für die gewünschte Erfassung vorliegt oder gar generiert wird, soll Bestand haben. Eine Bedienoberfläche kann ebenfalls als ein Pliable Object verstanden werden. Im Falle einer Interaktion reduziert sich der Erfassungsprozess auf einen Ausführungsprozess mittels eines GUI-Pliable-Objects.

Ausführungsprozess

Der Ausführungsprozess dient zur Initiierung einer automatischen Verarbeitung (vgl. 4.5.2). Er ist eingebettet in die Verarbeitung, die mittels der allgemeingültigen Kontrollflussschleife kontrolliert wird (vgl. 5.4.1). Der Prozess ist dafür zuständig, die Anweisungsliste einer

Aktion zu bestimmen und jede Anweisung einzeln nacheinander aufzurufen. Die Abarbeitung der Anweisung kann zu einem beabsichtigten Zustandswechsel führen.

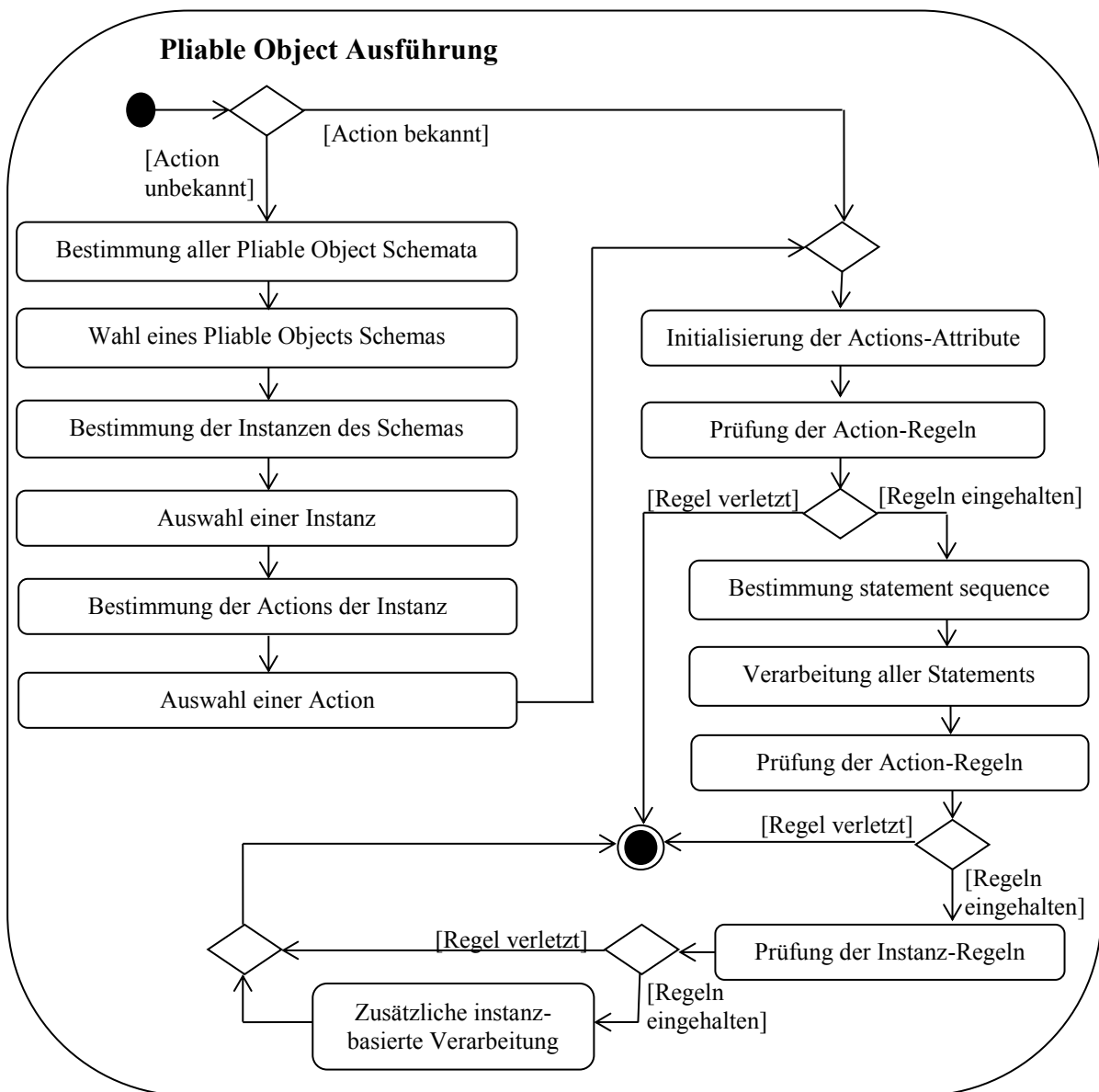


Abbildung 44: Darstellung des Ausführungsprozesses

Abbildung 44 zeigt den Ausführungsprozess. Der Prozess selbst beginnt mit der Bestimmung der Aktion eines Pliable Objects, die ausgeführt werden soll, falls diese nicht bereits bekannt ist. Von der auszuführenden Aktion werden deren Aktionsattribute initialisiert, d.h. ihr aktueller Zustand aus dem Informationssystem bestimmt. Anschließend folgt eine Prüfung der Regeln, die die Aktion selbst besitzt, um eventuelle Vorbedingungen zu berücksichtigen. Sofern eine Regel verletzt ist, wird die Verarbeitung abgebrochen. Im Rahmen eines Abbruchs wird vorausgesetzt, dass eine Fehlerverarbeitung vorgenommen wird. Eine konkrete Fehlerverarbeitung wird nicht spezifiziert; vielmehr wird angenommen, dass im Fall von Fehlern zusätzliche Ereignisse definiert werden, die die Ausführung spezieller Aktionen triggern, die die eigentliche Fehlerverarbeitung realisieren.

Ist die Erfüllung der Aktionsregeln gegeben, werden die Anweisungen der Aktion bestimmt und einzeln nacheinander verarbeitet. Zur Verarbeitung wird der Verarbeitungsprozess herangezogen (s.u.). Nach der Verarbeitung werden die Aktionsregeln auf eventuelle Nachbedingungen geprüft. Die Ausführung selbst endet mit der Überprüfung der Regeln des Pliable Objects, dem die Aktion zu eigen ist, um auf Zustandsänderungen reagieren zu können. Sind bestimmte Bedingungen erfüllt, ist es über entsprechende Regeln möglich, Instanz-spezifische, weitere Verarbeitungsaktionen ausführen zu lassen.

Verarbeitungsprozess

Der Verarbeitungsprozess arbeitet generell mit dem Ausführungsprozess zusammen, da er von diesem getriggert wird. Er bietet die Unterstützung allgemeiner Berechnungsfunktionalitäten und ist anweisungskonform konstruiert. Der Prozess führt die einzelnen Anweisungen durch, die in Aktionen eingebunden sind. Der Verarbeitungsprozess ist in der Lage existente Funktionalitäten einzubinden, wie z.B. Addition von Zahlen, und liefert so beispielsweise den Aktionen ein Berechnungsergebnis, falls eine Anweisung dies beinhaltet.

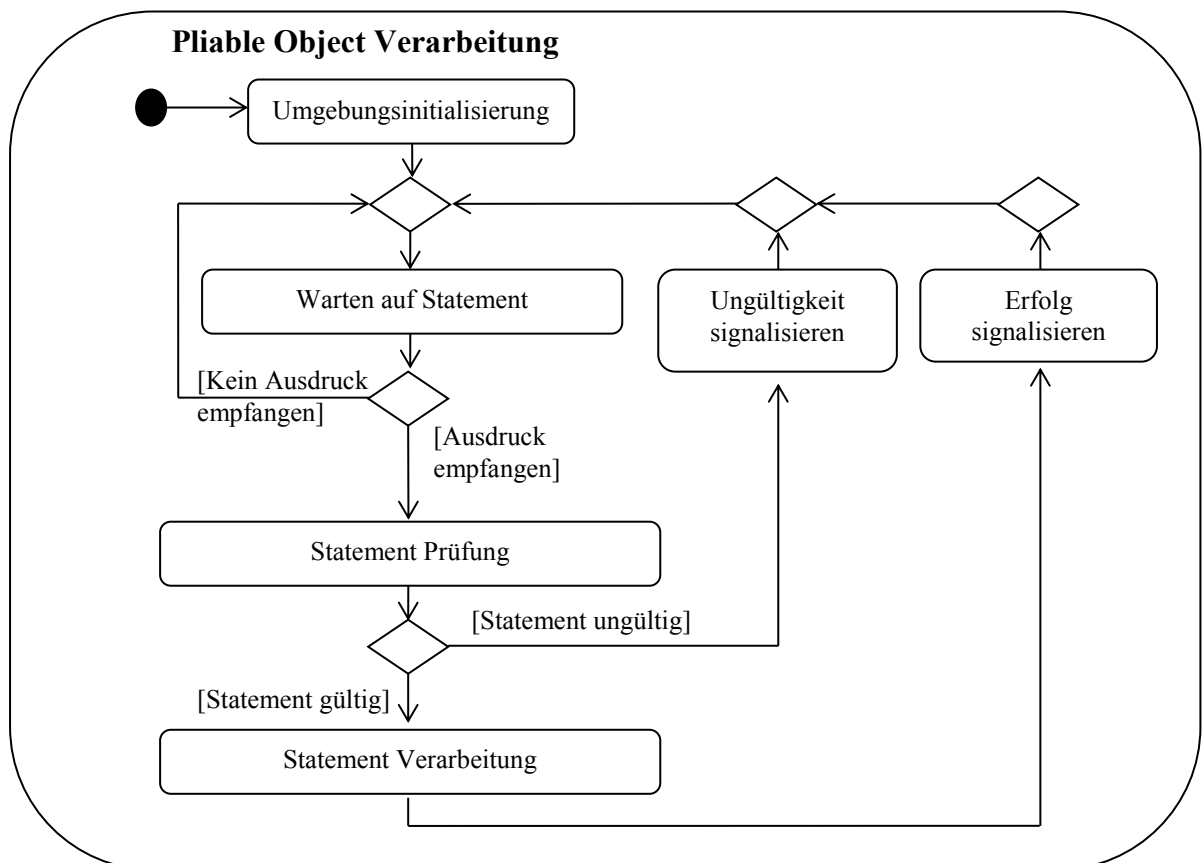


Abbildung 45: Darstellung des Verarbeitungsprozesses

Abbildung 45 zeigt den Verarbeitungsprozess. Der Prozess selbst realisiert eine Endlosschleife. Er beginnt bei seinem Start mit der Initialisierung seiner Umgebung, d.h. er initialisiert die Funktionalität, um Anweisungen verarbeiten zu können, die der kontextfreien Grammatik genügen, die zur Formulierung der Anweisung verwendet wurde bzw. verwendet werden kann. Es bleibt anzumerken, dass nicht ausschließlich genau eine kontextfreie

Grammatik initialisiert werden muss. Der Verarbeitungsprozess ist prinzipiell derart ausgelegt, dass er beliebig viele kontextfreie Grammatiken unterstützen kann. Die genaue Spezifikation des Verarbeitungsprozesses wird im Rahmen dieser Arbeit allerdings nicht vorgenommen.

Nach der Initialisierung der notwendigen Umgebung wartet der Prozess auf den Empfang einer Anweisung. Nach dem Empfang einer Anweisung überprüft der Prozess, ob die Anweisung syntaktisch korrekt angegeben ist. Ist diese nicht gegeben, wird die Ungültigkeit der Anweisung signalisiert und der Verarbeitungsprozess wechselt in den Wartezustand. Liegt die Anweisung in einer korrekten Syntax vor, wird die Anweisung ausgeführt. Diese Ausführung orientiert sich an der Art der Anweisung, die bereits in Kapitel 4.5.3 vorgestellt wurde. Im Anschluss an die Ausführung der Anweisung wird die erfolgreiche Ausführung signalisiert und der Prozess wechselt in den Wartezustand.

Vom Ausführungsprozess, der dem Verarbeitungsprozess die einzelnen Anweisungen übergibt, wird erwartet, dass das Signal zur erfolgreichen Verarbeitung dazu verwendet wird, die nächste Anweisung der Aktion an den Verarbeitungsprozess zu übergeben. Das Ungültigkeitssignal hingegen soll innerhalb der Regeln der Aktion, über deren Ausführung die Anweisungsverarbeitung beauftragt worden ist bzw. innerhalb der Regeln des Pliable Objects, dem die Aktion zu eigen ist, derart abgefangen werden, um eine spezifische Fehlerverarbeitung zu starten.

Überwachungsprozess

Der Ausführungs- und der Verarbeitungsprozess arbeiten überwiegend automatisiert. Sie werden allerdings durch eine von einem Anwender ausgehende Aktivierung angestoßen. Im Rahmen des Verarbeitungsprozesses können bei der Regelüberprüfung von Pliable Objects weitere Ausführungen bzw. Verarbeitungen getriggert bzw. initiiert werden. Dies setzt die Existenz einer entsprechenden Regel voraus. Regeln selbst sind bislang mit konkreten Regelausprägungen verknüpft.

Im Rahmen der Verarbeitung kann es auch möglich sein, dass zu einer Prämisse verschiedene Konklusionen existieren. Dabei gilt, dass zu einem bestimmten Zeitpunkt nur eine Konklusion mit der Regel verknüpft ist. Eine Regel, die solch ein Verhalten zeigt und über einen Zeitraum hinweg mindestens zwei unterschiedliche Konklusionen besitzt, wird als Ereignis definiert.

Um Ereignisse verarbeiten zu können, ist ein Überwachungsprozess nötig, der sich durch eine ständige Ereigniserwartung auszeichnet und bei Eintreffen eines Ereignisses dessen Verarbeitung vornimmt. Abbildung 46 zeigt den Überwachungsprozess.

Dieser Überwachungsprozess ist der elementarste Prozess. Er wird durch ein spezielles Aktivierungsereignis gestartet. Einmal gestartet, besitzt der Prozess zwei Zustände, zwischen denen er wechselt. Der Prozess erwartet grundsätzlich Ereignisse, von denen angenommen wird, dass sie dem Prozess bekannt sind. Tritt ein bekanntes Ereignis in Erscheinung, reagiert der Prozess darauf, indem er die mit dem Ereignis verbundene Verarbeitung vornimmt, d.h. der Überwachungsprozess triggert einen Verarbeitungsprozess mit der aktuellen Konklusion der Regel, die auf die eingetretenen Ereignisse zugeschnitten ist. Nach der Abarbeitung der getriggerten Prozesse kehrt der Überwachungsprozess in den Erwartungszustand zurück. Diese Ablaufbeschreibung entspricht der Verarbeitungsbeschreibung der allgemeingültigen

Kontrollflussschleife (vgl. 5.4.1). Dies bedeutet, dass ein allgemeingültiger Prozess gegeben sein muss, der durch die Kontrollflussschleife repräsentiert wird.

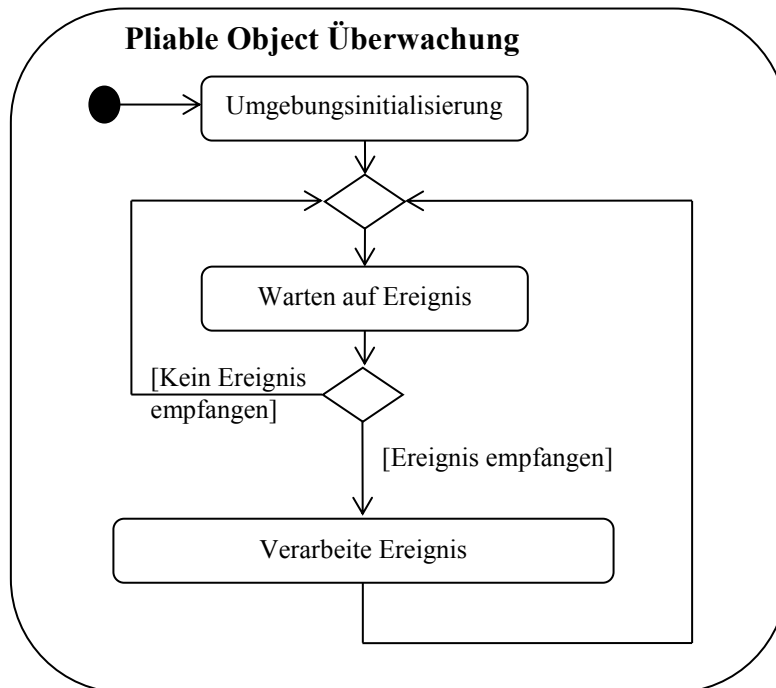


Abbildung 46: Darstellung des Überwachungsprozesses

Es bleibt anzumerken, dass das Terminierungsereignis für den Überwachungsprozess ein Ereignis ist, welches durch den Prozess selbst verarbeitet wird. Falls solch ein spezielles Terminierungsereignis vereinbart ist, terminiert der Prozess bei dessen Auftreten quasi durch die Terminierungsverarbeitung. Allerdings verbleibt der Prozess danach in dem Zustand, in dem er nur auf ein einziges Ereignis wieder reagiert. Dies ist das Aktivierungsereignis. Dessen Verarbeitung ist daher auch Teil der eigenen Prozessverarbeitung. In der obigen Abbildung 46 ist daher keine Terminierung vorgesehen.

5.4.3 Verwaltung von Fehlerzuständen

Die vorgestellten sieben Prozessklassen beschreiben die Realisierung der Basisfunktionalitäten, die notwendig sind, um die gesamte Be- und Verarbeitung von Pliable Objects zu gewährleisten. Jedes Pliable Object, inklusive spezifischer Attribute, spezifischer Aktionen und spezifischer Regeln, ist durch die einzelnen Prozessklassen erzeugbar und kann mit anderen Pliable Objects verknüpft werden. Die Prozessklassen verfügen über einen generischen Charakter. Dieser äußert sich dadurch, dass die Prozessklassen selbst als Pliable Objects betrachtet werden können. Damit können einzelne Prozesse der Prozessklassen erstellt, individuell angepasst und speziell ausgeprägt werden.

Es wird vorausgesetzt, dass Aktionen auch Prozesse realisieren. Wird eine Prozessaktion aufgerufen, wird deren Aktionsbeschreibung abgearbeitet. Inwieweit davon Abhängigkeiten, wie z.B. Konsistenzprüfungen, einbezogen werden, hängt vom Anwender ab. Je sicherer man ein System machen möchte, desto mehr Attribute müssten definiert werden. Über Aktionen

und Regeln müssten sie aufeinander abgeglichen werden. Welche Regelung dafür jedoch notwendig ist, hängt von der persönlichen Bewertung des Anwenders ab. Dies ist vor allem bei einer Fehlerbehandlung von Bedeutung

Ein Prozess sollte in der Regel mit einem vorher bestimmbar Ergebnis terminieren. Damit ist die durch die Aktionsbeschreibung festgelegte Berechnung bzw. der Algorithmus gemeint. Ein nicht vorher festgelegtes Ergebnis, d.h. ein ungültiger Zustand, wird als Fehler definiert und muss „behandelt“ werden. Ein Fehlermechanismus kann auf verschiedene Weise realisiert werden; so kann z.B. in der allgemeinen Aktionsdefinition ein Attribut integriert werden, welches über einen Fehlerzustand der Aktion Aufschluss gibt. Eine aufrufende Aktion muss den Fehlerzustand abfragen und eine eigene Fehlerbehandlung für die aufzurufende Aktion integrieren.

Eventuell ist zu berücksichtigen, dass ein Prozess von außen abgebrochen und zur Terminierung gezwungen werden kann. Die sich daraus ergebenden Fehlerzustände müssen behandelt werden.

Eine Aktion kann die Prozesstypen selbst involvieren, um gewünschte Ziele zu erreichen. Ein denkbares Szenario wäre die Möglichkeit des operationellen Ableitens.

5.4.4 Operationelle Attribute

Eine Möglichkeit, die durch die Verkettung von Prozessen ermöglicht wird, ist das operationelle Ableiten von Attributwerten. Darunter wird der Vorgang verstanden, dass nicht ein spezifisches Attribut wertmäßig direkt durch Anwender angegeben, sondern der Wert des Attributes im Rahmen einer Aktionsausführung zugewiesen wird, wobei die Aktion durch eine Regelprüfung gestartet wird. Die spezifischen Attribute, die auf diese Weise ihre Werte erhalten, werden als **operationelle Attribute** definiert, d.h. ein operationelles Attribut sei definiert als ein Attribut, dessen Wert sich ausschließlich aus der Anwendung einer Regel ableitet.

Beispielsweise ist das *Alter* eines Patienten ein operationelles Attribut. Das Patientenalter definiert eine Zeitspanne, die gemäß einer Berechnungsvorschrift bestimmt wird. Allgemein gibt das Alter die Zeitspanne an, die (beginnend mit dem Geburtsdatum des Patienten) bis zu einem Bezugsdatum reicht. Häufig wird das aktuelle Tagesdatum als Bezugsdatum herangezogen, wobei die Einheit des Alters in Jahren gegeben ist.

Ein Wert für das Alter kann berechnet werden. Ist dieser Wert einmal berechnet, hat er solange Bestand, bis die Berechnung bzw. die Regel neu angestoßen wird. Generell sind operationelle Attribute von natürlichen Gegebenheiten abhängig. Ändern die natürlichen Gegebenheiten ihren Zustand, so müssen die operationellen Attribute angepasst werden. Das Alter ist generell abhängig von der Zeit und ändert sich in Abhängigkeit derselben. Das in einem Anästhesieprotokoll als aktuelles hinterlegte Patientenalter bezieht sich in der Regel auf das ebenfalls eingetragene Operationsdatum.

Verfügt ein Pliable Object über operationelle Attribute, ist der Pliable-Object-Zustand grundsätzlich Änderungen unterworfen.

5.4.5 Operationelle Aktionen

Neben der Möglichkeit, Attributwerte operationell abzuleiten, kann man auch Attribute selbst durch Aktionen ableiten. Die Verkettung von Prozessen ermöglicht spezielle Verknüpfungen, die in Aktionsanweisungen hinterlegt sind. Unter konkreten Entitäten, die mit Pliable Objects abgebildet werden, können auch konkrete Vorgänge verstanden werden. Diese Vorgänge seien dadurch gekennzeichnet, dass zwei Pliable Objects ausschließlich durch den Vorgang miteinander verknüpft werden. Z.B. wird eine Anästhesie durch einen Anästhesisten vorgenommen. Abgebildet werden diese beiden Entitäten durch die Pliable Objects *Anästhesie* und *Anästhesist*. Eine Aktion *Anästhesie ausführen* beschreibt die Verknüpfung zweier Instanzen der Pliable Objects. Wird die Aktion gestartet, so wird automatisch erkannt, dass die beiden Pliable Objects *Anästhesie* und *Anästhesist* miteinander verknüpft werden. In der Instanz des Pliable Objects *Anästhesie* ist im Attribut *Anästhesist* der verantwortliche Anästhesist angegeben. Dieser wird durch eine Instanz des Pliable Objects *Anästhesist* repräsentiert, in welchem die *Anästhesie* wiederum zu hinterlegen ist. Daraus ist die Attributdefinition

$$\alpha_{\text{Anästhesist}} = (\text{Anästhesist}, \delta_{\text{Anästhesist}}^*, \varepsilon)$$

ableitbar. Das Attribut sollte existieren und der Attributmenge des Pliable Objects *Anästhesie* angehören. Falls dem nicht so wäre, so wäre die automatische Generierung des Attributes und die Zuweisung zur Attributmenge des Pliable Objects unter Verwendung der vorgestellten Prozesse denkbar.

Analog ergibt sich, dass das Pliable Object *Anästhesist* folgendes Attribut besitzen sollte:

$$\alpha_{\text{Anästhesie}} = (\text{Anästhesie}, \delta_{\text{Anästhesie}}^*, \varepsilon)$$

Dieses Attribut könnte ebenfalls angelegt und dem Pliable Object *Anästhesist* zugewiesen werden. Es würde alle Anästhesien aufnehmen, die der durch das Pliable Object abgebildete Anästhesist betreut hätte.

Die Generierungen wären vom Verarbeitungsprozess durchführbar, gegebenenfalls unterstützt durch eine interaktive Auswahl der jeweiligen Instanzen. Aktionen, die das Anlegen von Attributen aus einer Anweisung ableiten, werden als **operationelle Aktionen** definiert.

5.4.6 Unwiderrufliche Zustände

Einer Instanz soll es möglich sein, unveränderliche Zustände zu besitzen. Solche Zustände sind beispielsweise realisiert durch Konstanten wie die Zahl π oder die Lichtgeschwindigkeit.

Ein unveränderlicher Zustand ist ebenfalls erreicht, wenn ein Attribut einen Wert zugewiesen bekommen hat und anschließend seinen Zustand nicht mehr ändern darf. Solche unwiderruflichen Zustände treten in Zusammenhang mit (meist) zeitkritischen Entscheidungen auf. Ist beispielsweise die Narkose eines Patienten eingeleitet und der Arzt hat sich entschieden zu operieren, d.h. er hat demzufolge die eigentliche Operation bereits mit einem Schnitt begonnen, so ist diese Entscheidung nicht mehr widerrufbar. Im Dokumentationssystem ist der Zeitpunkt des Schnittes fest einzutragen und kann nicht mehr verändert werden.

Pliable Objects sind flexibel mit der Wertzuweisung und den erlaubten Zustandswechseln. Sie verhalten sich eher nachsichtig und befürworten Änderungen bzw. zügige Zustandswechsel. Sie verfügen über einen optimistischen Charakter, der beliebige Änderungen zulässt. Eine Unwiderrufbarkeit muss auf das flexible System aufgesetzt werden. Hierzu sind spezielle Aktionen und Regeln sowie Attribute zu entwickeln, die die Unwiderrufbarkeit sicherstellen. Dabei besteht die Gefahr, die Unwiderrufbarkeit nicht immer einhalten zu können, da nicht alle sich ergebenden Einschränkungen abgefangen werden.

Alternativ können Pliable Objects eher pessimistisch gesehen werden, d.h. die Unwiderrufbarkeit wird dergestalt berücksichtigt, dass Pliable Objects prinzipiell von einem Start- in einen Folgezustand wechseln, der nicht veränderbar ist. Durch spezielle Aktionen und Regeln muss eine gewünschte Änderungsmöglichkeit realisiert werden. Es besteht jedoch die Gefahr, dass die Änderungsmöglichkeiten das gewünschte Ziel nicht erreichen und nicht alle benötigten Einschränkungen aufheben. Zudem besteht auch noch die Gefahr, dass die Ermöglichung der Änderung „über das Ziel hinausschießt“, d.h. es werden mehr Änderungen zulässig als ursprünglich angenommen.

Um eine Änderbarkeit oder Unwiderrufbarkeit zu erreichen, wird ein weiteres Attribut definiert. Das Attribut *Änderbar* besitze die Domäne

$$\delta_{\text{Änderbarkeit}} = (\text{Änderbarkeit}, \{\text{Veränderbar}, \text{Unwiderrufbar}\})$$

mit dem Defaultwert $\varepsilon = \text{Veränderbar}$.

Um das Attribut nutzen zu können, muss es generell allen Pliable Objects zugewiesen sein, der Verarbeitungsprozess muss es berücksichtigen und entsprechend reagieren. Für Pliable Objects, denen mehrmalig ein Wert zugewiesen wird, wie z.B. Nachname, Alter, Adresse eines Patienten, ist der Änderbarkeitszustand *Veränderbar*. Für Pliable Objects, die einen einmal erreichten Zustand (wie z.B. Aktionen) behalten, ist der Änderbarkeitszustand *Unwiderrufbar*.

Mit der Aufnahme der Änderbarkeit in die Definition eines Pliable Objects wird die Änderbarkeit zu einer generellen Eigenschaft von Pliable Objects. Für konkrete Entitäten, die unter Pliable Objects eingebunden sind, gibt der Änderbarkeitszustand Sicherheit in Bezug auf Änderungsauswirkungen. Konkrete Entitäten sind selbst Pliable Objects. Wenn sie nicht mehr geändert werden, können sie bei einer Gebundenheit an Pliable Objects als konstante Werte betrachtet werden. Sie besitzen einen über ihre Ausprägungen definierten fixen Zustand. Zustandsänderungen sind nicht zu erwarten und geben der Gebundenheit oder sogar Verknüpfung Sicherheit, da keine Zustände eintreten können, die eine inkonsistente Sicht auf eine Gebundenheit verursachen können.

Die Änderbarkeit als Definitionsbestandteil ist unerlässlich, da sie eine Betrachtung von Pliable Objects als Werte nicht nur ermöglicht, sondern sogar die Behandlung von Pliable Objects als Werte ermöglicht. Die Bildung einer aus Pliable Objects bestehenden Wertemenge ist somit erlaubt. Andernfalls wäre eine Diskussion von Pliable-Object-Mengen erschwert.

5.4.7 Abbildung von Werten

Über den Mechanismus der Verknüpfung des Erzeugungsprozesses mit dem Veränderungsprozess kann eine Pliable-Object-Domäne mit den Attributen *Typbezeichnung* und *Wertemenge* erstellt werden. Ein Anlegen sowie eine Integration neuer Domänen ist mit Hilfe des Mechanismus möglich.

Eine Domäne bezeichnet eine Wertemenge. Ein Wert, der Element der Menge ist, ist bislang nicht genau spezifiziert worden. Von einem System kann ein tatsächlicher Wert nicht verarbeitet werden; vielmehr wird der Wert in einer alphanumerischen Repräsentation codiert. Aus Gründen der Lesbarkeit wird angenommen, dass ein Wert als Zeichenkette gespeichert wird. Bei solch einem Wert kann es sich auch um die Angabe einer Wertemenge handeln. Um diesen Umstand berücksichtigen zu können, wird ein Wert grundsätzlich als ein regulärer Ausdruck verstanden.

Ein regulärer Ausdruck ist eine Zeichenkette, die sich aus Metazeichen und Literalen zusammensetzt (vgl. [Fr01]). Zum Verständnis muss die Menge aller bekannten Zeichen in zwei disjunkte Mengen unterteilt werden. Die Menge der Metazeichen enthält die Symbole oder Spezialzeichen, die eine grammatikalische Rolle spielen. Über der Menge, die alle übrigen Zeichen enthält, werden die Literale gebildet, die durch beliebige Kombinationen der Zeichen entstehen. Literale nehmen so quasi eine Rolle von Wörtern ein. Ein regulärer Ausdruck entsteht, wenn die Wörter durch die grammatikalischen Regeln, d.h. die Metazeichen, kombiniert werden. Auf diese Weise kann ein regulärer Ausdruck als Zeichenkette dargestellt werden, die folgenden Inhalt besitzen können:

- ein Literal
- eine Aufzählung von Literalen
- ein regulärer Ausdruck
- eine Aufzählung von regulären Ausdrücken
- eine Aufzählung von Literalen und regulären Ausdrücken.

Da die beiden letzten Aufzählungen selbst wieder einen regulären Ausdruck darstellen, kann der Inhalt eines regulären Ausdrucks durch die ersten drei Angaben beschrieben werden.

Ein Informationssystem, welches die Pliable Objects in einer Datenbank verwaltet, muss über eine Parser-Komponente verfügen, die einen Wert als einen regulären Ausdruck auswerten kann. Bei der Erstellung neuer Domänen kann die Wertemenge als regulärer Ausdruck angegeben werden. Das System wird selbstständig die weitere Verarbeitung und Speicherung der Wertemenge in der Datenbank vornehmen. Die Angabe eines einzigen Literals in der Wertemenge führt zu der Speicherung des Wertes in der Domänen-Relation selbst. Wird die Wertemenge durch eine Aufzählung angegeben, kann diese in eine separate Relation übernommen werden. Tabelle 16 zeigt die Ableitung einer Relation am Beispiel der Regelarten.

Regelart-Aufzählung		
Domänen-ID	Wert-ID	Wert
4	1	Constraint
4	2	Plausibilität
4	3	Bedingung

Tabelle 16: Beispiel zur Abbildung von Domänen auf Einzelwerte

Die Zuordnung eines konkreten Regelartwertes zu einer Regel muss das System selbstständig auflösen. Die automatische Auflösung von Zuweisungsverhältnissen muss ebenfalls bei mehrwertigen Zuordnungen einer Aufzählungsdomäne realisiert werden. Die Ableitung der mehrwertigen Zuordnung kann in der Instanzen-Relation geschehen. Angenommen, das Attribut *Regelart* einer Regel wäre mehrwertig und die Regel sollte sowohl als Constraint als auch als Bedingung gekennzeichnet sein, so müsste diese Mehrwertigkeit in der Instanz in einem separaten Tupel gespeichert werden. Tabelle 17 zeigt die Referenzierung von Einzelwerten anhand der Instanz X des Pliable Objects Rule.

Instanzen			
ID	POID	Attribut-ID	Wert
..
X	4	9	1
X	4	9	3

Tabelle 17: Beispiel für die Referenzierung von Einzelwerten abgebildeter Domänen

Wird die Wertemenge einer Domäne durch einen regulären Ausdruck angegeben, so muss für den Wert eines Attributes, dem die Domäne zugeordnet ist, gelten, dass bei der Instanz der konkrete Wert gespeichert wird, wobei der Parser sicherstellt, dass der Wert auch der Domäne angehört. Dies geschieht durch die Überprüfung, ob der Wert von dem regulären Ausdruck abgeleitet werden kann.

Von der Mächtigkeit des Parsers, der den regulären Ausdruck auswertet, hängt die Mächtigkeit der Funktionalität ab, die dadurch realisiert werden kann. Für den Parser muss eine genaue Sprache gefunden werden, die die Anwendungsspezifikation erfüllt.

5.5 Basisarchitektur eines Informationssystems zur Verarbeitung von Pliable Objects

Pliable Objects beinhalten eine spezielle Form der Selbstbeschreibung, die dahingehend genutzt werden kann, eine allgemeingültige Applikation für die Bearbeitung von Pliable Objects konzipieren zu können.

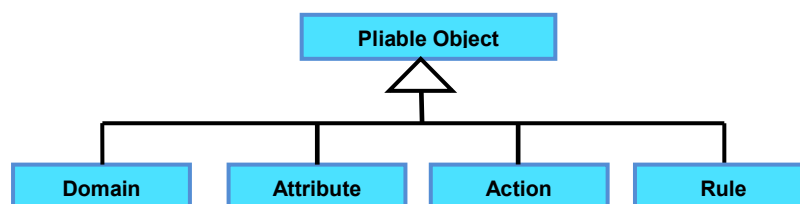


Abbildung 47: Spezielle Klassenhierarchie von Pliable Objects

Das semantische Modell in Abbildung 35 beinhaltet die Objekte *Pliable Object*, *Attribute*, *Domain*, *Action* und *Rule*. In dieser Abbildung stehen *Attribute*, *Action* und *Rule* in einer Komponentenbeziehung zum *Pliable Object*. Aufgrund der Ausprägung der Selbstbeschreibung kann die Schlussfolgerung gezogen werden, dass die Objekte *Attribute*, *Domain*, *Action* und *Rule* selbst wieder als Pliable Objects zu betrachten sind (vgl. Abbildung 47). Deren Instanzen zeichnen sich dadurch aus, dass sie andere Pliable Objects auf einer

schematischen Ebene beschreiben und zur Gestaltung der Instanzen dieser anderen Pliable Objects dienen. Oder anders formuliert, ein Schema-Pliable Object setzt sich aus Instanzen weiterer Pliable Objects zusammen. Die Art und Anzahl der genutzten Instanzen prägen den Aufbau und das Verhalten der Instanzen des Schema-Pliable Objects. Das Verhalten der verwendeten Instanzen wird dagegen dazu verwendet, um Instanzen des Schema-Pliable Objects anzulegen, zu bearbeiten oder zu löschen. Bei letzteren Instanzen kann es sich wieder um Schema Pliable Objects handeln, die dazu dienen, eine Anwendung zu beschreiben. Die Gestaltung und die Ausprägung verschiedener weiterer (Schema-)Pliable Objects sind dabei anwendungsbezogen und müssen gemeinsam mit den jeweiligen Anwendern fachlich spezifiziert werden.

Unabhängig von einer fachbezogenen Anwendungsspezifikation kann eine allgemeingültige Basis einer komponentenorientierten Software-Architektur (vgl. [Gr98], [Wh02], [Zö09]) konzipiert werden, die bereits eine unabhängige Minimal-Applikation beschreibt. Abbildung 48 zeigt die elementaren Komponenten, die im Rahmen einer solchen Applikation zu entwickeln wären, um Pliable Objects zur Datenverarbeitung konzipieren und gleichzeitig verwenden zu können. Die einzelnen Komponenten können nach standardisierten Entwicklungsprinzipien realisiert werden (vgl. [CCG00], [De05], [Ga95]), weshalb deren Einzelrealisierungen in dieser Arbeit nicht diskutiert werden.

Die zentrale Komponente ist eine generische Verarbeitungseinheit mit der Basisfunktionalität zur Be- und Verarbeitung von Pliable Objects. Die Verarbeitungseinheit arbeitet auf den Pliable Objects, die in einer Datenbank gespeichert sind. Die Datenbank wiederum wird über ein Datenbankmanagementsystem verwaltet.

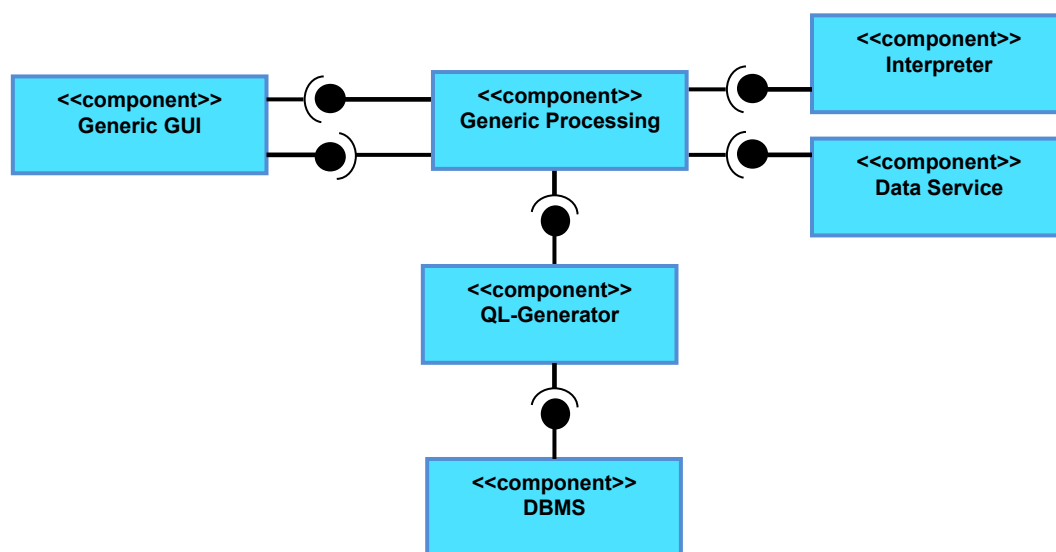


Abbildung 48: Elementare Software-Architektur zur Verarbeitung von Pliable Objects

Bei Pliable Objects handelt es sich um dynamische Objekte, d.h. ihre momentane Gestalt und Ausprägung muss zur Laufzeit ermittelt werden. Hierzu verwendet die Verarbeitungseinheit einen Query-Language-(QL)-Generator. Wird beispielsweise eine relationale Datenbank mit SQL als Anfragesprache verwendet, können aufgrund der Tatsache, dass die Attribute der Pliable Objects *Pliable Object*, *Attribute*, *Domain*, *Action* und *Rule* selbst als Daten in der Datenbank eingetragen sind (vgl. Tabelle 11), die notwendigen Select-, Insert- und Update-Anfragen generisch bestimmt werden.

Im Prinzip initialisiert sich die Verarbeitungseinheit am Anfang mit der Funktionalität zum Anlegen, Löschen und Verarbeiten von Pliable Objects. Über eine Bedienoberfläche kann ein Anwender interaktiv diese Funktionalität nutzen. So können dem Anwender beispielsweise zunächst alle Pliable Objects zur Verfügung gestellt werden. Aus dieser Menge wählt er eines aus, welches er bearbeiten möchte. Dies schließt auch das Anlegen neuer Instanzen als auch neuer Pliable Objects selbst mit ein.

Nach einer getroffenen Auswahl initialisiert die Verarbeitungseinheit sich und die Bedienoberfläche mit der Funktionalität, die mit dem ausgewählten Pliable Object in der Datenbank verknüpft ist, d.h. dessen Attributes, Actions und Rules werden aus der Datenbank bestimmt.

Die Verarbeitungseinheit verwendet die Rules, um die darin definierte Verhaltenskontrolle im Zusammenspiel mit der Bedienoberfläche zu realisieren. Startet der Anwender interaktiv eine spezielle Funktionalität, wird ein zugehöriger Event ausgelöst, der zum Aufruf der mit dem Event verknüpften Action führt. Die Verarbeitungseinheit bestimmt die Verarbeitungssequenz der Action aus der Datenbank und nutzt einen Interpreter, um die Verarbeitungssequenz abzuarbeiten.

Im Rahmen dieser Abarbeitung soll eventuell eine Datenbestimmung von oder eine Datenübermittlung auf externe Schnittstellen vorgenommen werden. Zur Datenbestimmung und -übermittlung wird die Datenservicekomponente verwendet, die die notwendige Funktionalität dazu bereitstellt. Wichtig ist dabei, dass die Komponente nur die Funktionalität zur Verfügung stellt. Die Protokolle, die für den Datenaustausch notwendig sind, sollten über Actions generisch gehandhabt werden können, wobei diese Actions wieder in der Datenbank gespeichert sind. Da die Änderung von Actions über die Änderungsfunktionalität von Pliable Objects realisiert wird, sind selbst Protokolle für einen gezielten Datenaustausch variabel und anpassbar.

Die Grundlage der Minimal-Applikation bildet eine Datenbank. Da Datenbanken die Grundlage eines Informationssystems sind (vgl. [Sa93]), realisiert die Minimal-Applikation ein Informationssystem. Dieses Informationssystem ist in der Lage, Aktionen und Regeln von Pliable Objects zu verarbeiten. Aktionen und Regeln sind Pliable Objects zu eigen. Da sie eigene Zustände besitzen, erweitern sie die Zustandsmöglichkeiten der Pliable Objects. Die möglichen Zustände werden in der Datenbank gespeichert.

5.5.1 Zustandsabbildung

Die möglichen Zustände eines Pliable Objects werden über dessen Attribute abgebildet. Die Abbildung eines Zustandes wird über die Zuweisung von Werten zu diesen Attributen vorgenommen.

Eine Wertbelegung eines einzelnen Attributs kennzeichnet einen Zustand ζ des Attributes. Die Menge aller Zustände, die das Attribut α annehmen kann, sei mit Z_α bezeichnet. Zur Unterscheidung von verschiedenen Zuständen werden die Zustände nummeriert; die Nummerierung wird als Indizierung angegeben, z.B. $\zeta_1, \zeta_2, \text{etc.}$

Die Kardinalität der Wertemenge der zugeordneten Domäne entspricht der Anzahl der Zustände, die das Attribut annehmen kann, d.h.

$$\text{card}(Z_\alpha) = \text{count}(\delta_\tau^*) \text{ mit } \delta_\tau^* \text{ ist die Domäne des Attributes } \alpha.$$

Durch die Existenz eines Defaultwertes gilt, dass ein Attribut grundsätzlich einen Zustand besitzt, der eben durch die Wertbelegung mit dem Defaultwert gegeben ist. Dieser Zustand wird mit ζ_ε bezeichnet.

Beispielsweise sind für das Attribut Geschlecht folgende drei Zustände denkbar:

- („Geschlecht“, {m,w, ε}, m),
- („Geschlecht“, {m,w, ε}, w) und
- („Geschlecht“, {m,w, ε}, ε)

Einem Pliable Object sind mehrere Attribute zugeordnet; jedes einzelne Attribut besitzt einen Zustand. Aus der Gesamtheit der Zustände der Attribute resultiert der Zustand des Pliable Objects. Die Anzahl der möglichen Zustände eines Pliable Objects ergibt sich durch

$$\text{card}(Z_\pi) = \prod \text{card}(Z_\alpha), \forall \alpha \in \pi.$$

Jedes Pliable Object hat immer einen Ausgangs- oder Startzustand. Dieser Zustand wird mit ζ_ε bezeichnet. Er kennzeichnet, dass sich jedes Attribut des Pliable Objects im Zustand ζ_ε befindet.

Attribute bilden relevante Zustände ab. Nun können nicht nur Attribute, sondern auch Regeln und Aktionen relevante Zustände besitzen, die im Rahmen der Bearbeitung wichtig sind. Um dies gewährleisten zu können, müssen die Zustände von Aktionen und Regeln erfasst werden.

Dies geschieht durch eine Erweiterung der Aktionen und Regeln um Zustandsattribute. In einer Zustandsdomäne werden diejenigen Zustandsarten zusammengefasst, die als Domäne der Zustandsattribute dienen.

Beispielsweise wäre folgende Zustandsdomäne denkbar:

$$\delta_{\text{Zustand}} = (\text{Zustände}, \{\text{bereit, in Ausführung, terminiert}\})$$

Über die Domäne wären folgende Attribute anlegbar, die die jeweilige Attributmenge der Aktion und der Regel erweitern:

$$\begin{aligned} \alpha_{\text{Aktionszustand}} &= (\text{Aktionszustand}, \delta_{\text{Zustand}}, \varepsilon) \\ \alpha_{\text{Regelzustand}} &= (\text{Regelzustand}, \delta_{\text{Zustand}}, \varepsilon), \\ &\text{mit jeweils dem Defaultwert } \varepsilon = \text{bereit}. \end{aligned}$$

Für die Nutzung der Attribute muss die zugehörige Aktionsdefinition jeweils Anweisungen enthalten, die den Zustand der Attribute setzen, um die Zustände von außen ablesbar zu gestalten. Dies muss das Informationssystem gewährleisten.

5.5.2 Transitionsbetrachtung

Eine Transition definiert einen Zustandswechsel. Unter einer Transition versteht man im Allgemeinen den Übergang eines Objektes von einem bestimmten Ausgangs- in einen Folgezustand. Dieser Übergang kann von bestimmten Ereignissen ausgelöst und selbst auch wiederum mit bestimmten Folgeereignissen verbunden sein.

Bezogen auf Pliable Objects überführt eine Transition ein Pliable Object π von Zustand ζ_i in Zustand ζ_j , mit $\zeta_i, \zeta_j \in Z_\pi$, formal wiedergegeben durch $\zeta_i \rightarrow \zeta_j$.

Bei Attributen ist damit eine Änderung der Wertausprägung gemeint. Bei Pliable Objects handelt es sich dagegen in der Regel um die Durchführung einer Belegungsänderung von Attributen.

Laut Pliable-Object-Vereinbarung sind Aktionen für Zustandswechsel verantwortlich, d.h. Transitionen werden durch Aktionen repräsentiert. Diese Aktionen eines Pliable Objects π definieren eine Relation auf die Zustandsmenge Z_π . Die Relation verknüpft zwei Zustände dergestalt miteinander, dass ein Zustand als Ausgangszustand deklariert ist und der zweite Zustand den Folgezustand bildet.

Ein Pliable Object heißt änderbar, wenn eine Zustandsrelation existiert, die das Pliable Object aus dem Default- in einen Folgezustand überführt. Hierfür muss eine Aktion ausgeführt werden. Die Ausführung einer Aktion ist im Rahmen einer Steuerung erlaubt, die durch eine Regel abgedeckt wird. Damit eine Regel „feuert“, muss ein Ereignis eintreten, auf welches sie „gewartet“ hat, d.h. ein Ereignis wird von einer Regel „erkannt“. Als Folge löst die Regel eine Aktion aus, welche wiederum eine neue Attributbelegung erwirkt.

Das Informationssystem muss gewährleisten, dass durch eine Aktion Folgeereignisse angestoßen werden können, durch die Regeln dazu veranlasst werden, eventuell eine neue Aktion anzustoßen. Um dieses Pliable-Object-Verhalten zu realisieren, muss das Informationssystem selbst ein bestimmtes Verhalten realisieren. Eine durch ein Ereignis ausgeführte Transition skizziert Abbildung 49.

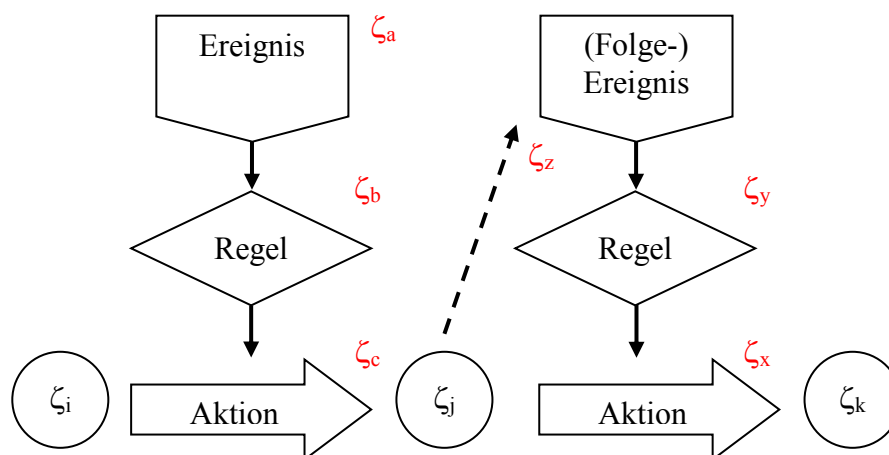


Abbildung 49: Skizzierung von Zustandsübergängen

Das Verhalten des Informationssystems, das die Aktionen und Regeln eines Pliable Objects auswertet und die Wertänderungen in den Attributen des Pliable Objects kontrolliert, gestaltet

sich in einer Weise, die durch eine Zustandsabfolge skizziert wird, die wie folgt aufgelistet werden kann:

$$\zeta_i \rightarrow \zeta_a \rightarrow \zeta_b \rightarrow \zeta_c \rightarrow \zeta_j \rightarrow \zeta_z \rightarrow \zeta_y \rightarrow \zeta_x \rightarrow \zeta_k$$

Anfangs befindet sich das Pliable Object im Zustand ζ_i . Ein Ereignis tritt ein, d.h. das Informationssystem wechselt in den Zustand ζ_a , in welchem das Ereignis berücksichtigt wird. Dies führt dazu, dass nach einer Regel gesucht wird, die die Behandlung des Ereignisses beschreibt. Nach erfolgreicher Lokalisierung der Regel wechselt das Informationssystem in den Zustand ζ_b , in welchem die Regel ausgeführt, d.h. das Ereignis verarbeitet wird. Diese Verarbeitung entspricht wiederum dem Aufruf einer Aktion und wird durch den Wechsel in den Zustand ζ_c signalisiert. In diesem Zustand wird die Aktion ausgeführt. Anschließend ist das Pliable Object im Zustand ζ_j . Ein Folgeereignis tritt ein, d.h. das Informationssystem wechselt in den Zustand ζ_z . Nach der Bestimmung der Regel, die dieses Folgeereignis behandelt, wird der Zustand ζ_y angenommen, um die Regelausführung zu behandeln. Es folgt der Wechsel in den Zustand ζ_x , der die Aktionsausführung kennzeichnet. Abschließend befindet sich das Pliable Object im gewünschten Zielzustand ζ_k , während das Informationssystem wieder in einen Zustand wechselt, in dem auf Ereignisse gewartet wird.

Aus dem Beispiel wird ersichtlich, dass durch Aktionen und Regeln verschiedene Zustände miteinander zu einer Zustandssequenz verkettet werden. Durch die Existenz mehrerer Zustandssequenzen werden mögliche Zustände isoliert, die nie erreicht werden können und sollen. In der Regel sind solche nie erreichbaren Zustände die so genannten inkonsistenten Zustände. Einem Pliable Object *Patient* sind beispielsweise die Attribute für das Geburtsdatum und das Alter des Patienten zu eigen. Ohne Angaben zu weiteren Einschränkungen sind inkonsistente Angaben möglich, wie z.B., dass es sich um ein negatives Alter handelt oder (bezogen auf das aktuelle Datum) dieses nicht der Differenz zum Geburtsdatum entspricht. Zur Vermeidung solch inkonsistenter Zustände dienen Regeln und Aktionen, die die Zustandsmenge eines Pliable Objects einschränken. Aktionen dienen dem bestimmten Zweck, nur zu konsistenten Zuständen zu führen. Regeln greifen steuernd über die Auswahl der Aktionen ein. Zusätzlich ermöglichen Regeln den Zustandswechsel aus einem Startzustand. Die konkreten Aktionen und Regeln werden vom Anwender vorgegeben. Das Informationssystem soll den Anwender bei der Eingabe von Aktionen und Regeln unterstützen. Somit liegt es in der Verantwortung des Anwenders, die Zustände, die er zu verarbeiten wünscht, zu definieren und im Informationssystem zu realisieren.

Die konsistenten Zustände beziehen sich an dieser Stelle auf die Zustände, die das Informationssystem besitzt. Mit den vorgestellten Optionen der Pliable Objects, besteht die Möglichkeit, dass Pliable Objects einen inkonsistenten Zustand besitzen, der nach subjektiven Maßstäben nicht erlaubt sein sollte, aber im Informationssystem gespeichert werden sollte.

Beispielsweise kann im Rahmen einer Interaktion ein Wert für ein Attribut von einem Anwender eingegeben werden. Bei der Werteingabe besteht die Möglichkeit, den eingegebenen Wert über eine Regel mit der entsprechenden Domäne des Attributes abzugleichen. Angenommen, der eingegebene Wert kann nicht der Domäne zugeordnet werden, dann kann in Abhängigkeit von der Regel entschieden werden, wie in diesem Fall weiter zu verfahren ist. Bei einer Constraint-Zusicherung kann der Wert umgehend zurückgewiesen und der Anwender um eine Korrektur gebeten werden. Falls die Regel allerdings eine Plausibilität realisiert, kann der Wert übernommen und gespeichert werden, da Werte bislang als Zeichenkette bzw. Literal angesehen werden. Das Pliable Object würde

dadurch nur als unplausibel und, bezogen auf sich selbst, als inkonsistent bewertet werden, aber das Informationssystem kann mit diesem Zustand umgehen. Pliable Objects unterstützen ungültige Zustände der Objekte, die das Informationssystem verarbeiten können muss.

5.5.3 Workflowbasierte Betrachtung von Pliable Objects

Zustandswechsel werden durch die Ausführung von Aktionen hervorgerufen. Bezieht sich ein hervorgerufener Zustandswechsel auf ein einzelnes Pliable Object und handelt es sich bei der Aktion um eine interne Aktion des Pliable Objects, so wird die Aktion als *Task* bezeichnet.

Beispielsweise wäre eine Berechnungsaktion für das Alter eines Patienten zum gegebenen Geburtsdatum ein Task. In einem Task werden dabei nur Stützfunktionen eingebunden; bei diesen Stützfunktionen handelt es sich um Routinen, die als existent betrachtete Basisdaten liefern, wie z.B. das Bestimmen des aktuellen Tagesdatums.

Wenn andere interne Aktionen des Pliable Objects, die als Task deklariert sind, in einem Task aufgerufen werden, wird dieser Task als *Aktivität* bezeichnet. Falls ein Task nach seiner Beendigung einen weiteren Task induziert, weil durch den Zustandswechsel die Startvoraussetzung des zweiten Task erfüllt ist, wird die Verknüpfung der beiden Tasks als Aktivität betrachtet. Eine Aktivität zeichnet sich dabei dadurch aus, dass sie komplett automatisiert abläuft; in einer Aktivität können auch mehrere Tasks induziert werden.

Ebenfalls sei es in einer Aktivität erlaubt, andere Aktivitäten zu starten. Ferner dienen Aktivitäten dem Datenaustausch zwischen Pliable Objects. Sie lesen und schreiben Daten in Pliable Objects, die an das Pliable Object gebunden sind. In einer Aktivität ist es ebenfalls erlaubt, Tasks bzw. Aktivitäten von eingebundenen Pliable Objects zu starten; sie starten allerdings keine Tasks oder Aktivitäten in Pliable Objects, in die ihr zugeordnetes Pliable Object eingebunden ist. Es sei denn, über einen Verknüpfungszyklus besteht dazu die Möglichkeit.

Aktivitäten, die nicht automatisiert ablaufen, werden als *Prozesse* bezeichnet und sind abhängig von externen Dateneingaben, wie z.B. Anwendereingaben. Prozessen ist es möglich, Daten zu lesen und schreiben - sowohl in gebundenen Pliable Objects als auch in Pliable Objects, die von ihrem Pliable Object eingebunden werden.

Um den Datenbestand der Datenbank zu bearbeiten, dient die Ausführen-Aktion. Diese Aktion soll automatisch die Prozessstypen Task, Aktivität und Prozess unterstützen und entsprechend handhaben. Die verschiedenen Aktionen der Pliable Objects liegen in der Datenbank als Prozessbeschreibung vor und sind somit auch Teil des Datenbestandes. Es handelt sich dabei um vom Menschen und von der Maschine lesbaren Code. Die Ausführen-Aktion liest diese Prozessbeschreibung ein und startet einen Prozess, der das Verhalten zeigt, welches mit der Prozessbeschreibung gegeben ist. Im Rahmen des Prozesses kann es sein, dass weitere Prozesse initiiert werden.

Für diese Prozesse wird grundsätzlich angenommen, dass sie nur ausgeführt werden, wenn eine Beabsichtigung vorliegt. Auch wenn Prozesse automatisiert gestartet bzw. auch durch andere Prozesse induziert werden können, wird davon ausgegangen, dass ein Anwender gezielt einen initialen Prozess startet. Der Anwender muss Kenntnis von Startereignissen besitzen, die er initiieren kann und die zum Start von Prozessen führen. Hierfür ist die Existenz einer Regel notwendig, die auf das Startereignis reagiert und den Prozess initiiert.

Zur Übersicht ist es dienlich, Prozesse in Pliable Objects gezielt zur Verfügung zu stellen. Um dies zu gewährleisten, müssen allerdings konkrete Prozesse existieren, die die Verarbeitung von Prozessen ermöglichen. Diese Prozesse entsprechen den vorgestellten Prozessen, die das Verhalten der Pliable Objects realisieren. Das Informationssystem zur Be- und Verarbeitung von Pliable Objects stellt die Prozesse zur Verfügung. Ein Anwender hat die Möglichkeit, mittels der Interaktion mit dem Informationssystem, die Pliable Objects nach seinen Gestaltungsvorstellungen auszuprägen und dadurch auch die Mächtigkeit des Informationssystems festzulegen.

5.6 Zusammenfassung

In den Abschnitten dieses Kapitels ist das in Kapitel 4 vorgestellte Modell der Pliable Objects in ein Basisinformationssystem überführt worden. Dies gelang mittels der Erarbeitung eines Datenmodells im Entity-Relationship-Modell für Pliable Objects und der anschließenden Visualisierung in Form einer relationalen Ansicht.

Ermöglicht wurde dies durch die in Kapitel 4.7.4 spezifizierten Sonder-Pliable-Objects. Aus den spezifizierten Methoden, die auf den Sonder-Pliable-Objects arbeiten, konnten allgemeingültige Prozesse abgeleitet werden, die eine generische Verarbeitung von und auf Pliable Objects zulassen.

Die Grundlage der generischen Prozesse bildet eine speziell ausgeprägte Ereignisverarbeitung, die durch eine Zustandsmaschine realisiert wird, deren Ausprägung durch die Regeln der zu verarbeitenden Pliable Objects in einer dynamischen Art und Weise erreicht wird.

Diese Dynamik macht den besonderen Charakter von Pliable Objects aus. Das erarbeitete Basisinformationssystem ermöglicht dadurch einerseits die dynamisch ausprägbare Verarbeitung von Pliable Objects und andererseits die dynamische Definition neuer Pliable Objects, die anschließend in die laufende Verarbeitung einfließen. Damit eröffnet das Modell der Pliable Objects einen neuen Ansatz für die Entwicklung zukünftiger Informationssysteme.

Gemäß dem klassischen Ansatz zur Entwicklung eines Informationssystems wird zunächst ein formalisiertes Modell verwendet (vgl. [Ba00], [He92], [RP02], [Sa93], [Sp05], [Vo00]). Das Modell bereitet das Datenmaterial der abzubildenden Anwendung auf. Anschließend wird ein Entwicklungskonzept für die Implementierung erarbeitet. Dabei wird für die Datenverwaltung eine Datenbank verwendet. In einer höheren Programmiersprache wird die Datenverarbeitung der zu implementierenden Applikation programmiert. Die darin abgebildete Algorithmik arbeitet auf den Daten in der Datenbank (vgl. Kapitel 3.5).

Das Basisinformationssystem für Pliable Objects erlaubt eine andere Art, um das Ziel einer laufenden Applikation zu erreichen. Ebenso wie im klassischen Ansatz wäre das Modell der Anwendung zu erstellen. Im Gegensatz zur Erarbeitung eines Entwicklungskonzeptes kann das Anwendungsmodell, umgesetzt mit Pliable Objects, direkt im Basisinformationssystem gespeichert werden. Das Anwendungsmodell dient dabei der Ausprägung des Basisinformationssystems in jenes Informationsmodell, welches den Zweck erfüllt, der durch die Anwendung gewünscht ist.

Die Dynamik von Pliable Objects, welche auch das Informationssystem gewährleistet, bietet den Vorteil, dass sich das Informationssystem durch eine Anpassung des Modells dynamisch

an die geänderten Bedürfnisse ausrichten lässt, wenn sich der Zweck des ursprünglich angedachten Anwendungssystems ändert. Dies geschieht dabei unter Beibehaltung des ursprünglich eingegebenen Datenbestandes.

Im folgenden Kapitel wird die Ausprägung des Basisinformationssystems in eine Realisierung eines Anästhesie-Dokumentationssystems beschrieben.

6 Anästhesie-Management mit Pliable Objects

Im vorherigen Kapitel ist ein Informationssystem vorgestellt worden, welches Pliable Objects be- und verarbeitet. Die Pliable Objects dienen dabei der Ausprägung des Informationssystems. In diesem Kapitel werden mögliche Pliable Objects vorgestellt, die bereits eine Anästhesie-Dokumentation realisieren. In Verbindung mit dem vorher vorgestellten Informationssystem würden diese Pliable Objects ein Anästhesie-Informationssystem zur Dokumentation von Anästhesien darstellen. Diese Anästhesie-Dokumentation ist dabei einfach gehalten, um die prinzipielle Vorgehensweise zur Erstellung eines Informationssystems zur Anästhesie-Dokumentation mittels Pliable Objects zu präsentieren. Beachtenswert ist dabei, dass das Informationssystem die Funktionalität zur Verfügung stellt, die folgenden Anästhesie-Pliable-Objects zu verändern. Dies bedeutet, dass der vorgestellte Vorschlag zur Anästhesie-Dokumentation gänzlich nach bislang unbekanntem anwendungs- und anwenderspezifischen Kriterien komplett verändert werden kann, ohne dabei den eigentlichen Anspruch an eine Anästhesie-Dokumentation aufzugeben. Das Anästhesie-Dokumentationssystem ist somit anpassbar und speziell ausprägbar. Einige Ergebnisse, die das Kapitel enthält, sind teilweise in [Pr12] veröffentlicht worden.

6.1 Basisklassenmodell der Anästhesie

Um einen individuellen Dokumentationsanspruch mit einem Informationssystem zu gewährleisten, muss ein Modell abgebildet werden, welches generische Aspekte enthält, die eine individuelle Gestaltung der Datenverarbeitung ermöglichen.

Nach Kapitel 3.1 unterteilt sich die Protokollierung in drei Phasen, die den Arbeitsablauf widerspiegeln, in dem die Daten anfallen, die von einem Anästhesie-Dokumentationssystem zu erfassen und aufzubereiten wären. Aus der in dem Kapitel angeführten idealisierten Betrachtung des Vorgangs der Anästhesie muss zunächst ein Referenzmodell abgeleitet werden; dieses entspricht dem Datenmodell, welches die Basis des zukünftigen Informationssystems bildet. Aufbauend auf dem Modell kann die Modellierung der Dokumentation erfolgen, die der Protokollierung der Anästhesie entspricht. Ein Basisklassenmodell der Anästhesie zeigt Abbildung 50.

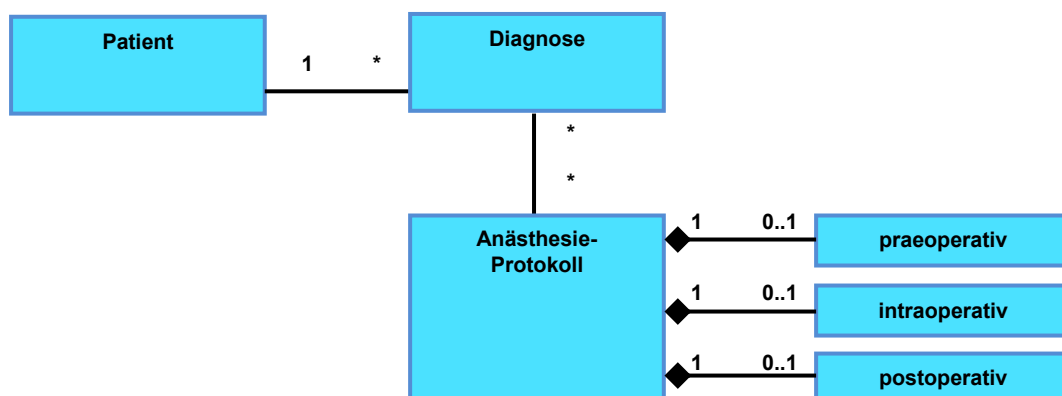


Abbildung 50: Basisklassendiagramm der Anästhesie

Die Grundlage des Modells bilden die drei Pliable Objects *Patient*, *Diagnose* und *Anästhesieprotokoll*:

1. *Patient*: Zu den elementarsten Informationen, die als Ausgangsbasis zu erfassen sind, zählen die Patienteninformationen. Das Object *Patient* ist die Repräsentation eines Patienten und nimmt die Daten auf, welche eine aufgenommene Person beschreiben.
2. *Diagnose*: Der Patient hat ein „Leiden“, welches anhand von Symptomen diagnostiziert wird. Die zugehörigen Diagnose-Informationen, die den Zustand des Patienten beschreiben, werden in dem Object *Diagnose* verwaltet. Grundsätzlich gilt, dass ein Patient mehrere Diagnosen ausgestellt bekommt. Ein Patient kann hinsichtlich mehrerer Krankheiten behandelt werden oder muss sich im Laufe seines Lebens mehreren unterschiedlichen Operationen unterziehen. Die Diagnose spielt im weiteren Verlauf eine zentrale Rolle. Auf Basis der Diagnosedaten wird der gesamte weitere Verlauf der Anästhesie geplant.
3. *Anästhesieprotokoll*: Das Object *Anästhesieprotokoll* enthält die anästhesiespezifischen Daten einer Operation. Diese Daten umfassen die prä-, intra- und postoperativen Anästhesiedaten. Aufgrund des Datenumfanges unterteilt sich das eigentliche Protokoll in anästhesiespezifische Teilprotokolle, die der Datenerhebung während der Prämedikation, der Narkose und des Aufwachraums dienen. Im Informationssystem werden diese Teilprotokolle durch die Objects *Präoperativ*, *Intraoperativ* und *Postoperativ* repräsentiert.

Die Vielfalt der Daten, die nach den Richtlinien der DGAI zu erfassen sind, erfordert die Generierung von Domänen. Über den Domänen können die Attribute gebildet werden, die den Pliable Objects zugeordnet werden, die die Grundlage der Abbildung der Anästhesie-Dokumentation in ein Informationssystem bilden. Die folgende Vereinbarung orientiert sich am Kerndatensatz der DGAI (vgl. [DGAI]).

6.1.1 Anästhesie-Domänen

Im Kerndatensatz werden einige Basisinformationen gespeichert, die einen allgemeingültigen Charakter besitzen und über allgemeingültige Domänen abgebildet werden können. Diese Domänen sollen über einen regulären Ausdruck abgebildet werden. Diese können wie folgt angegeben werden:

1. (Zahlendomäne, [0-9]*)
2. (Zeichenkette, [a-zA-Z]*)
3. (Wahrheitswerte, (ja|nein))
4. (Datum,
 $(([0][1-9]) | ([1|2][0-9]) | (30) | (31) \setminus$
 $([0][1-9]) | (10) | (11) | (12)) \setminus$
 $([19][0-9][0-9] | [0-9][0-9]) | ([20][0-9][0-9] | [0-9][0-9]))$)
5. (Zeit,
 $(([0|1][0-9]) | (20) | (21) | (22) | (23) \setminus (([0|1|2|3|4|5][0-9])))$)

Hinsichtlich der allgemeingültigen Domänen ist zu beachten, dass auch diese Domänen in einem Informationssystem, welches Pliable Objects verarbeitet, mit den Mitteln des Informationssystems geändert und an individuelle Ausprägungen angepasst werden können.

Die weiteren Domänen, die in der Anästhesie-Dokumentation verwendet werden, stellen überwiegend Aufzählungen dar und können wie folgt angegeben werden:

1. (Versorgungsstufendomäne,
{Fachkrankenhaus, Grundversorgung, Regelversorgung, Maximalversorgung,
Medizinisches Versorgungszentrum, Ambulanzzentrum})
2. (Geschlechtdomäne, {männlich, weiblich, intersexuell})
3. (Fallartdomäne, {ambulant, stationär})
4. (Fachabteilungsdomäne,
{AIN, ACH, NCH, MKG, PWC, PCH, MED, NEU, HNO, URO, AUG, DRM,
GYN, PED, RAD, NUC, NRD, PSY, TCH, HCH, GCH, HDC, MSZ, SON})
5. (Dringlichkeitsdomäne, {Elektiveingriff, Dringlich, Not/Soforteingriff})
6. (ASADomäne, {ASA I, ASA II, ASA III, ASA IV, ASA V, ASA VI})
7. (Tracerdomäne,
{kein Tracer, Section Caesarea, Adenotomie, laparoskopische Cholezystektomie,
transuethrale Prostataresektion, arthroskopischer Knieeingriff})
8. (Verlegungsdomäne,
{Intensiv, Normalstation, Sonstige Verlegung, Exitus, unerwartete stationäre
Aufnahme nach ambulanten Eingriffen, ungeplante Aufnahme IMC/ICU})
9. (Domäne der Art einer Anästhesiologischen Verlaufs-Beobachtung(AVB),
{Laryngospasmus, Bronchospasmus, Aspiration, ...²¹})
10. (Domäne des Schweregrads einer Anästhesiologischen Verlaufs-Beobachtung (AVB),
{Verlängerter Aufenthalt im Aufwachraum und/oder besondere Nachbeobachtung
auf Allgemeinstation, Problem kann im AWR nicht zufriedenstellend gelöst
werden und bedingt Verlegung auf Intensiv- oder Wachstation, Tod des
Patienten})
11. (Domäne des Zeitpunkts einer Anästhesiologischen Verlaufs-Beobachtung (AVB),
{intraoperativ, postoperativ})

6.1.2 Unbestimmte Anästhesie-Domänen

Es existieren weitere Domänen, die einen dynamischen Charakter besitzen. Diese müssen separat als Pliable Objects realisiert werden. Deren Ausprägung hängt sehr von den individuellen Bedürfnissen der Anwender des Informationssystems ab, so dass kein konkreter Vorschlag vorgestellt werden kann.

Die Instanzen dieser individuell festzulegenden Pliable Objects bilden selbst wieder eine Domäne, die für Eigenschaften genutzt werden, die wiederum in der individuellen Ausprägung des Anästhesieprotokolls verwendet werden können. Bei diesen individuellen Pliable Objects handelt es sich um die technische Repräsentation von Ärzten, Pflegekräften und Diagnosen sowie Eingriffen.

²¹ Die Liste der möglichen Anästhesiologischen Verlaufs-Beobachtungen ist recht umfangreich und kann dem Anhang des Kerndatensatzes der Anästhesie unter [DGAI] entnommen werden.

Ärzte und Pflegekräfte stellen sich jeweils als ein Pliable Object mit im Minimum zwei Eigenschaften dar. Diese Eigenschaften geben Auskunft über die Personalnummer und den Namen des Arztes bzw. der Pflegekraft. Zusätzlich dürfte in der Regel jede medizinische Versorgungsanstalt ergänzende individuelle Eigenschaften definieren wollen, die für die interne Datenverarbeitung der medizinischen Versorgungsanstalt wichtig sind.

Diagnosen und Eingriffe sind nach den Standards ICD und ICPM vereinbart. Die Vereinbarung besteht für jeden Eingriff und jede Diagnose aus einem eindeutigen Schlüssel sowie aus einer erläuternden Beschreibung. Als Pliable Objects werden Diagnosen und Eingriffen daher mit jeweils zwei entsprechenden Attributen realisiert. Deren Ausprägung ist von individuellen Bedürfnissen geprägt, so dass kein allgemeingültiger Vorschlag erarbeitet werden kann.

6.1.3 Anästhesie-Pliable-Objects

Unter Verwendung der Domänen können Attribute gebildet werden, in denen die während einer Anästhesie anfallenden Informationen gespeichert werden können. Die Attribute sind an die oben vorgestellten Pliable Objects wie folgt gebunden.

Patient	Diagnose	Anästhesie
Patienten-ID: Zahlendomäne Name: Zeichenkette Alter: Zahlendomäne Altereinheit: Zeichenkette Geschlecht: Geschlechtdomäne	ICD: Zahlendomäne ICPM: Zahlendomäne	Einrichtungs-ID: Zahlendomäne Versorgungsstufe: Versorgungstufendomäne Fallart: Fallartdomäne Anästhesiedatum: Datum Fachabteilung: Fachabteilungsdomäne Dringlichkeit: Dringlichkeitsdomäne

Abbildung 51: Basisinformationen des Kerndatensatzes der DGAI

Abbildung 51 zeigt die Pliable Objects *Patient*, *Diagnose* und *Anästhesie*, auf die die Basisinformationen als Attribute verteilt sind, die für den Kerndatensatz der Anästhesie von der DGAI festgelegt worden sind. Bei den Basisinformationen in der Abbildung handelt es sich um die so genannten Header-Informationen und die allgemeinen Daten des Kerndatensatzes. Die Pliable Objects sind in UML-Notation angegeben. Den Attributen sind die entsprechenden Domänen zugeordnet. Einzig das Object *Diagnose* und deren Eigenschaften sind nicht dem Kerndatensatz entnommen. Allerdings ist für jede Operation ein Grund zu definieren und dafür eignen sich die Vereinbarungen des ICD- und des ICPM- Standards.

Abbildung 52 zeigt die Ausprägungen der Pliable Objects *Präoperativ*, *Intraoperativ* und *Postoperativ*, ebenfalls in UML-Notation. Bezogen auf den Kerndatensatz ist im präoperativen Pliable Object die Risikobewertung dokumentiert. Dies entspricht dem Vorgehen bei der papierbasierten Erfassung, wie es Abbildung 12 zeigt. Das intraoperative Pliable Object dokumentiert die Zeiterfassung, das Anästhesieverfahren, Angaben zum Luftweg und zur Atmung/Beatmung, zum erweiterten Monitoring, der Operationsart, zu Anästhesiologischen Verlaufsbeobachtungen und der Entlassung (aus dem Operationssaal). Das postoperative Pliable Object nimmt Angaben zu besonderen Qualitätsmerkmalen entgegen. Auf eine detaillierte Beschreibung wird verzichtet, da diese dem veröffentlichten Kerndatensatz [DGAI] entnommen werden kann²².

²² Die Angaben zu technischen Feldern des Kerndatensatzes sind nicht berücksichtigt worden.

Präoperativ
ASA: ASADomäne Herz: Wahrheitswerte Lunge: Wahrheitswerte Kreislauf: Wahrheitswerte Neurologie/ZNS: Wahrheitswerte Stoffwechsel: Wahrheitswerte Extreme Adipositas: Wahrheitswerte
Intraoperativ
Arztbindungszeit: Zeit Schnitt-Naht-Dauer: Zeit TIVA: Wahrheitswerte Balancierte Anästhesie: Wahrheitswerte RSI: Wahrheitswerte Spinalanästhesie: Wahrheitswerte EDA lumbal: Wahrheitswerte EDA thorakal: Wahrheitswerte Periphere Regionalanästhesie: Wahrheitswerte Regionale mit Katheterverfahren: Wahrheitswerte Analgosedierung: Wahrheitswerte Stand by / Monitored Care: Wahrheitswerte Maske: Wahrheitswerte Supraglottischer LW: Wahrheitswerte Intratrachealer LW: Wahrheitswerte Endobronchialer LW: Wahrheitswerte Spontanatmung: Wahrheitswerte Unterstützte Spontanatmung: Wahrheitswerte Kontrollierte Atmung: Wahrheitswerte Sonstige Beatmung: Wahrheitswerte Invasives Blutdruckmonitoring: Wahrheitswerte Erweitertes Monitoring inkl.HZV: Wahrheitswerte ZNS-Monitoring: Wahrheitswerte Ultraschall Herz / TEE: Wahrheitswerte Ultraschall bei RA: Wahrheitswerte Ultraschall bei ZVK-Anlage: Wahrheitswerte Tracer: Tracerdomäne AVB: Wahrheitswerte AVB 1-Art: Domäne AVB-Art AVB 1-Schweregrad: Domäne AVB-Schweregrad AVB 1-Zeitpunkt: Zeit AVB 2-Art: Domäne AVB-Art AVB 2-Schweregrad: Domäne AVB-Schweregrad AVB 2-Zeitpunkt: Zeit AVB 3-Art: Domäne AVB-Art AVB 3-Schweregrad: Domäne AVB-Schweregrad AVB 3-Zeitpunkt: Zeit Aufwachraum: Wahrheitswerte Verlegung: Verlegungsdomäne
Postoperativ
Aufwachraum-Dauer: Zeit Übelkeit: Wahrheitswerte Erbrechen: Wahrheitswerte Zittern Grad > 1: Wahrheitswerte Hypothermie: Wahrheitswerte Schmerzen VAS > 3: Wahrheitswerte RA: Partieller Effekt: Wahrheitswerte RA: Unzureichender Effekt: Wahrheitswerte Awareness: Wahrheitswerte

Abbildung 52: Hauptinformationen des Kerndatensatzes der DGAI

6.2 Anästhesie-Prozess

Die Anästhesie untergliedert sich in die prä-, intra- und postoperative Phase (vgl. Kapitel 3.1). Die in den einzelnen Phasen zu dokumentierenden Daten werden durch die Attribute der drei Pliable Objects *Präoperativ*, *Intraoperativ* und *Postoperativ* abgebildet. Die Abhängigkeiten, die eventuell zwischen den drei Objekten existieren, werden über das Anästhesieobjekt (vgl. Abbildung 50) repräsentiert.

Das Anästhesieobjekt ist neben der Repräsentation des zusammenfassenden Dokumentationsobjektes auch eine Repräsentation des Vorgangs der Anästhesie, der als Prozess gegeben ist. Im Rahmen des Prozesses werden die Pliable Objects der Anästhesie, welche die Zustandsinformationen, die in den einzelnen Phasen anfallen, aufnehmen, miteinander verknüpft. Die Verknüpfung kann durch direkte oder indirekte Prozessabbildung realisiert werden.

6.2.1 Direkte Prozessabbildung

Die direkte Prozessabbildung verwendet zur Prozessabbildung ein eigenständiges Prozess-Pliable-Object. Dieses Prozessobjekt kann beispielsweise die *Anästhesie* sein. An das Prozessobjekt werden die Pliable Objects *Patient*, *Diagnose*, *Präoperativ*, *Intraoperativ* und *Postoperativ* über entsprechende Attribute gebunden, wie es Abbildung 53 zeigt.



Abbildung 53: Direkte Prozessabbildung am Anästhesieobjekt

Zusätzlich ergänzen verschiedene Aktionen das Anästhesieobjekt. Die Aktionen *getPatient()*, *getDiagnosis()*, *getPraeoperativ()*, *getIntraoperativ()* und *getPostoperativ()* dienen dazu, interaktiv die Attribute der entsprechenden Objekte zu erfassen.

Die Aktion *checkConsistency()* realisiert eine besondere Aktion. Diese Aktion prüft Plausibilitäten, wie sie ebenfalls im Anhang des Kerndatensatzes der DGAI vorgeschlagen werden. Die Plausibilitäten stellen inhaltliche Bedingungen dar, die die Korrektheit von Daten prüfen. Die tatsächliche Prüfung wird in der Aktion vorgenommen. Falls eine Plausibilität verletzt ist, soll die Aktion das Event *E_inconsistent* senden, um auf die Verletzung einer

Plausibilität hinzuweisen. Falls alle Plausibilitäten erfüllt sind, wird stattdessen das Event *E_consistent* gesendet.

Die Aktion *consistenceHandling()* realisiert eine individuell auszugestaltende Aktion, die dazu dient eine Art Fehlerbearbeitung vorzunehmen, wenn eine Plausibilität verletzt worden ist. Die Entscheidung, wie auf eine verletzte Plausibilität zu reagieren ist, obliegt den verantwortlichen Anästhesisten. Für den allgemeingültigen Fall wird an dieser Stelle angenommen, dass das Pliable Object Anästhesie um ein Attribut *consistenceFlag* erweitert wird, dessen Wert von der Aktion auf falsch gesetzt wird, um anzuzeigen, dass die betrachtete Instanz der Anästhesie inkonsistent bzgl. seiner Plausibilitäten ist.

Die Aktion *finish()* beendet die Verarbeitung mit der Instanz, während die Aktion *abort()* eine Abbruchaktion realisiert, die die notwendigen Aufräumarbeiten vornimmt, falls ein Anwender die interaktive Datenerfassung abbricht.

Anästhesieregeln				
State	Event	Action	PostState	SubsequentEvent
S_IDLE	E_Start	getPatient()	S_Patient	E_Diagnosis
S_Patient	E_Diagnosis	getDiagnosis()	S_Diagnose	E_Praeoperativ
S_Patient	E_Cancel	abort()	S_IDLE	-
S_Diagnose	E_Praeoperativ	getPraeoperativ()	S_Praeoperativ	E_Intraoperativ
S_Diagnose	E_Cancel	abort()	S_IDLE	-
S_Praeoperativ	E_Intraoperativ	getIntraoperativ()	S_Intraoperativ	E_Intraoperativ
S_Praeoperativ	E_Cancel	abort()	S_IDLE	-
S_Intraoperativ	E_Intraoperativ	getPostoperativ()	S_Postoperativ	E_Postoperativ
S_Intraoperativ	E_Cancel	abort()	S_IDLE	-
S_Postoperativ	E_Postoperativ	checkConsistency()	S_ConsistentCheck	-
S_Postoperativ	E_Cancel	abort()	S_IDLE	-
S_ConsistentCheck	E_consistent	finish()	S_IDLE	-
S_ConsistentCheck	E_inconsistent	consistenceHandling()	S_IDLE	-

Tabelle 18: Regeln des Anästhesieobjektes

Die Abfolge der Aufrufe dieser Aktionen wird in den Regeln des Anästhesieobjektes festgelegt. Tabelle 18 skizziert eine mögliche Realisierung. Darin befindet sich das System zunächst im Zustand *S_IDLE*. Die Erfassung einer neuen Anästhesie startet mit dem Event *E_Start*. Dies bedeutet, dass eine neue Instanz des Anästhesieobjektes angelegt und über die Aktion *getPatient()* interaktiv ein bekannter Patient ausgewählt oder ein neuer Patient im System aufgenommen wird. Während der Interaktion befindet sich das Anästhesieobjekt im Zustand *S_Patient*. Im Normalfall wird nach Abschluss der Patientenaufnahme die Diagnose gestartet. Im System wird dies über die anschließende automatische Verarbeitung des Events *E_Diagnosis* erreicht. Das Anästhesieobjekt wechselt in den Zustand *S_Diagnose*, in dem die interaktive Erfassung zur Diagnose vorgenommen wird. Nach der Diagnose wird die Prämedikationsphase gestartet. Getriggert durch das Event *E_Praeoperativ* wird in den Zustand *S_Praeoperativ* gewechselt und mittels der Aktion *getPraeoperativ()* die Erfassung der präoperativen Daten vorgenommen. Nach Abschluss dieser Erfassung wird durch das Senden des Event *E_Intraoperativ* in den Zustand *S_Intraoperativ* gewechselt und die interaktive Erfassung der intraoperativen Daten über die Aktion *getIntraoperativ()* aktiviert. Nach erfolgreicher Erfassung wird in den Zustand *S_Postoperativ* gewechselt. Das Senden des Event *E_Postoperativ* veranlasst diesen Zustandswechsel, wobei über den Aufruf der

Aktion *getPostoperativ()* die interaktive Erfassung der postoperativen Daten vorgenommen wird. In allen Zuständen, in denen interaktiv Daten erfasst werden, besteht die Möglichkeit, dass ein Anwender die Interaktion abbricht. In solch einem Fall wird über das Event *E_Cancel* die *abort()*-Aktion aufgerufen, die mögliche „Aufräumarbeiten“ veranlasst und dabei die Instanz in den Zustand *S_IDLE* versetzt.

Hat hingegen der Anwender alle Daten eingeben ohne einen Abbruch zu erzwingen, wird abschließend die Aktion *checkConsistency()* aufgerufen. Das Anästhesieobjekt wechselt in den Zustand *S_ConsistentCheck*, um anzuzeigen, dass die Plausibilitätsprüfung vorgenommen wird. Nach Aufruf der Aktion muss die Aktion ein Folgeevent aus der Überprüfung heraus senden. In den skizzierten Regeln ist deshalb kein automatischer Folgeevent definiert. Für die Aktion wird angenommen, dass die Aktion über die Events *E_consistent* und *E_inconsistent* signalisiert, ob eine Plausibilitätsverletzung erkannt worden ist. Wird keine Plausibilitätsverletzung erkannt, beendet das System die Verarbeitung des Anästhesieobjektes. Wird hingegen eine Plausibilitätsverletzung erkannt, wird durch den Aufruf der Aktion *consistenceHandling()* die Fehlerverarbeitung gestartet, die für diesen Fall vorgesehen ist.

6.2.2 Indirekte Prozessabbildung

Die indirekte Prozessabbildung verwendet die Zustände aller Pliable Objects, um in ihnen die Restriktionen und die Prozesse zu integrieren. Die Prozesse werden in Aktionen abgelegt, die ereignisorientiert getriggert werden. Hierzu ist ebenfalls die Verknüpfung oder Bindung der Pliable Objects untereinander nötig, d.h. den Pliable Objects werden Attribute zu eigen gemacht, deren entsprechende Domänen durch die weiteren Pliable Objects gebildet werden.

Zusätzlich wird das Informationssystem in die Lage versetzt, ein aktives Pliable Object zu verarbeiten. Dies bedeutet, dass immer nur ein Pliable Object aktiv ist und alle Ereignisse, die das System verarbeitet, mit den Regeln des aktiven Pliable Objects verglichen werden. Ferner werden Pliable Objects in die Lage versetzt, über ihre Aktionen die Aktivierung anderer Pliable Objects zu veranlassen und diesen Ereignisse zu schicken.

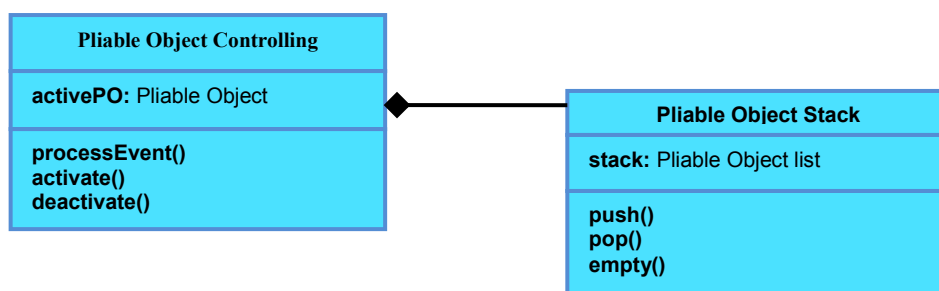


Abbildung 54: Erweiterung der Steuerungsfunktionalität um einen Kellerautomaten

Abbildung 54 skizziert ein Steuerungsobjekt, welches den unter Kapitel 5.4.2 erläuterten Überwachungsprozess realisiert. Das Objekt wird um einen Stack erweitert. Der Aktivierungsprozess ist nun dergestalt ausgeprägt, dass bei der Aktivierung eines anderen Pliable Objects, das aktuell aktive Pliable Object auf den Stack gepackt wird. Jedes aktive Pliable Object hat die Möglichkeit, sich über einen Aktionenaufruf selbst zu deaktivieren. Mit

der Deaktivierung wird das auf dem Stack abgelegte Pliable Object aktiviert, d.h. das vorher aktive Pliable Object wieder reaktiviert.

Auf Basis dieser Vereinbarung können die obigen Pliable-Objects *Patient*, *Diagnose* und *Anästhesie* miteinander verknüpft werden, wie es Abbildung 55 zeigt. Der Patient erhält zwei zusätzlich Attribute, die ein Diagnoseobjekt und ein Anästhesieobjekt referenzieren. Die drei Objekte verfügen ferner über eine *get()*-Aktion, die jeweils die interaktive Erfassung der in der Abbildung ausgeblendeten Attribute der Objekte erfassen.

Das Anästhesieobjekt selbst bindet zusätzlich das präoperative Objekt ein, welches wiederum das intraoperative Objekt einbindet, das letztlich das postoperative Objekt einbindet. Die drei Objekte erhalten die Aktionen, die in Kapitel 6.2.1 vorgestellt wurden. Die einzige Einschränkung ist, dass die Aktionen nur auf dem jeweiligen Objekt arbeiten.

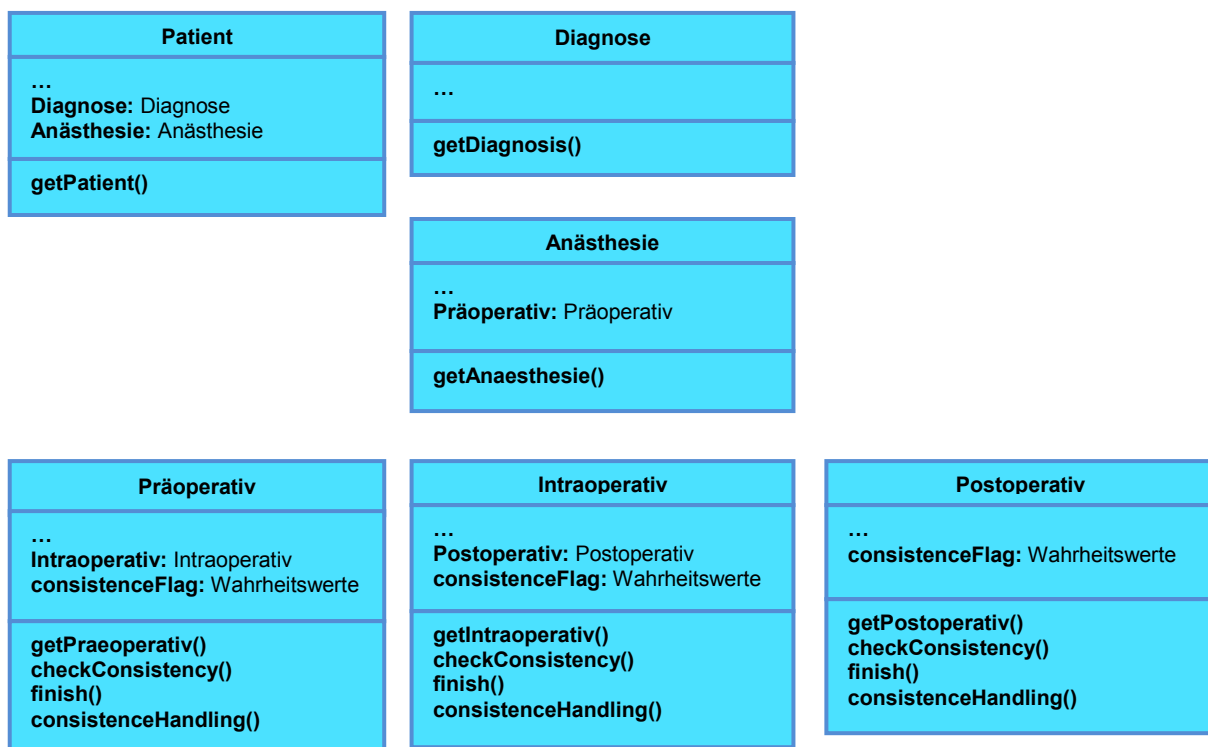


Abbildung 55: Indirekte Prozessabbildung der Anästhesie

Restriktionen bzw. Plausibilitätsprüfungen können dann in die einzelnen Pliable Objects verschoben werden, die die spezifischen Attribute besitzen und die zu prüfen wären. Eine mögliche Realisierung zeigen Tabelle 19 bis Tabelle 24.

Tabelle 19 skizziert die Regeln des Pliable Objects *Patient*. Für den betrachteten Fall wird angenommen, dass ein Patient im Rahmen der Aufnahme in eine medizinische Versorgungsanstalt angelegt wird. Nach dem Anlegen befindet sich das Objekt im Zustand *S_IDLE*. Nach dem Anlegen wird das Ereignis *E_Patient* vom Objekt verarbeitet, welches die interaktive Erfassung der Patientendaten mittels *getPatient()* startet und das Pliable Object in den Zustand *S_Patient* wechseln lässt.

Patientenregeln				
State	Event	Action	PostState	SubsequentEvent
S_IDLE	E_Patient	getPatient()	S_Patient	E_DiagnosisCreate
S_Patient	E_Cancel	-	S_IDLE	-
S_Patient	E_DiagnosisCreate	CREATE Diagnose	S_Diagnose	E_Diagnosis
S_Diagnose	E_Cancel	DELETE Diagnose	S_Patient	-
S_Diagnose	E_Diagnosis	ACTIVATE Diagnose	S_Diagnose	E_Diagnosis
S_Diagnose	E_AnaesthesieCreate	CREATE Anaesthesie	S_Anaesthesie	E_Anaesthesie
S_Anaesthesie	E_Anaesthesie	ACTIVATE Anaesthesie	S_Anaesthesie	E_Anaesthesie
S_Anaesthesie	E_Cancel	DELETE Anaesthesie	S_Diagnose	-
S_Anaesthesie	E_Done	-	S_IDLE	

Tabelle 19: Regeln des Pliable Objects Patient

Bricht der Anwender die Interaktion ab, wird das Ereignis *E_Cancel* verarbeitet, welches das Pliable Object *Patient* in den Zustand *S_IDLE* versetzt. Dies ermöglicht das erneute Starten der interaktiven Erfassung der Patientendaten.

Wird die interaktive Erfassung erfolgreich beendet, folgt automatisch das Senden des Ereignisses *E_DiagnosisCreate*. Dessen Auswertung führt zum Anlegen eines neuen Pliable Objects *Diagnose*, sowie zum Wechsel in den Zustand *S_Diagnose*. Durch die Verarbeitung des Ereignisses *E_Diagnosis* wird dieses Objekt aktiviert und das Event an das Diagnose-Pliable-Object weitergeleitet. Das Pliable Object *Patient* bleibt im Zustand *S_Diagnose*.

Diagnoseregeln				
State	Event	Action	PostState	SubsequentEvent
S_IDLE	E_Diagnosis	getDiagnosis()	S_Diagnose	E_finish
S_Diagnose	E_finish	DEACTIVATE self	S_IDLE	-
S_Diagnose	E_Cancel	DEACTIVATE self	S_IDLE	E_Cancel

Tabelle 20: Regeln des Pliable Objects Diagnose

Tabelle 20 zeigt die Regeln des Pliable Objects *Diagnose*. Das Pliable Object selbst befindet sich nach seiner Erzeugung im Zustand *S_IDLE*. Das über das Pliable Object *Patient* gesendete Ereignis *E_Diagnosis* startet die Interaktion zur Erfassung der Diagnosedaten. Unabhängig davon, ob die Interaktion abgebrochen oder erfolgreich abgeschlossen wird, wird anschließend in den Zustand *S_IDLE* gewechselt und die eigene Deaktivierung veranlasst. Einzig im Fall eines Abbruchs wird nach der Deaktivierung das Ereignis *E_cancel* gesendet. Bezogen auf dieses Beispiel wird aufgrund der Deaktivierung der Diagnose der Patient wieder aktiv. Falls ein Abbruch der Auslöser war, wird durch den Abbruch beim Pliable Object *Patient* die Verarbeitung angestoßen, welche das Pliable Object *Diagnose* löscht und das Object in den Zustand *S_Patient* wechseln lässt. Damit wäre sichergestellt, dass die Erfassung einer Diagnose erneut gestartet werden kann.

Für den Fall, dass die Erfassung der Diagnose erfolgreich abgeschlossen worden ist, verbleibt das Pliable Object *Patient* im Zustand *S_Diagnose*. Dieser Zustand wird verlassen, wenn eine Anästhesie angesetzt wird. Das Ereignis *E_AnaesthesieCreate* signalisiert diesen Fall. Es sorgt für das Anlegen des Pliable Objects *Anästhesie*, dessen Aktivierung, sowie für das Senden des Ereignisses *E_Anaesthesie* an das erzeugte Pliable Object. Die Regeln des Pliable Objects *Anästhesie* zeigt Tabelle 21.

Anästhesieregeln				
State	Event	Action	PostState	SubsequentEvent
S_IDLE	E_Anaesthesie	getAnaesthesie()	S_Anaesthesie	E_Done
S_Anaesthesie	E_Cancel	DEACIVATE self	S_IDLE	E_Cancel
S_Anaesthesie	E_Done	CREATE Praeoperativ	S_Praeoperativ	E_Praeopertiv
S_Praeopertiv	E_Praeopertiv	ACTIVATE Praeoperativ	S_Praeoperativ	E_Praeopertiv
S_Praeoperativ	E_Cancel	DELETE Praeoperativ	S_Anaesthesie	E_Cancel
S_Praeoperativ	E_Done	DEACTIVATE self	S_IDLE	E_Done

Tabelle 21: Regeln des Pliable Objects Anästhesie

Der Startzustand des Pliable Objects *Anästhesie* ist der Zustand *S_IDLE*. Mit Erhalt des Ereignisses *E_Anaesthesie* wechselt es in den Zustand *S_Anaesthesie*, während die interaktive Erfassung der allgemeinen Anästhesiedaten vorgenommen wird. Bei einem Abbruch der interaktiven Erfassung deaktiviert sich das Anästhesieobjekt und sendet das Ereignis *E_Cancel*. Dieses Ereignis wird daraufhin vom Pliable Object *Patient* verarbeitet, welches das Anästhesieobjekt löscht und selbst in den Zustand *S_Diagnosis* wechselt, um anschließend auf das erneute Startereignis einer Anästhesieerfassung zu warten.

Wird hingegen die interaktive Anästhesieerfassung erfolgreich abgeschlossen, wird das Ereignis *E_Done* gesendet und anschließend verarbeitet. Es folgt der Wechsel in den Zustand *S_Praeoperativ*, welcher das Anlegen eines Pliable Objects *Präoperativ* kennzeichnet. Nach dem Anlegen wird das Objekt aktiviert und es wird das Ereignis *E_Praeoperativ* geschickt. Die Regeln des Pliable Object *Präoperativ* zeigt Tabelle 22.

Das Pliable Object *Präoperativ* startet nach seiner Erzeugung ebenfalls im Zustand *S_IDLE*. Die Verarbeitung des Ereignisses *E_Praeoperativ* führt zum Wechsel in den Zustand *S_Praeoperativ* und damit zum Start der interaktiven Erfassung der präoperativen Anästhesiedaten.

Ein Abbruch der interaktiven Erfassung lässt das Objekt zurück in den Zustand *S_IDLE* wechseln. Im Rahmen der weiteren Verarbeitung wird es deaktiviert. Das Anästhesieobjekt erhält das Ereignis *E_Cancel*, woraufhin es in den Zustand *S_Anaesthesie* wechselt, das Pliable Object *Präoperativ* gelöscht wird und eine weitere Verarbeitung des Ereignisses *E_Cancel* stattfindet. Im Zustand *S_Anaesthesie* führt dieses Ereignis zu dem oben beschriebenen Wechsel in den Zustand *S_IDLE* und zum Deaktivieren des Anästhesieobjektes. Abschließend erhält das Pliable Object *Patient* das Ereignis *E_Cancel*.

Ein erfolgreicher Abschluss der interaktiven Erfassung der prämedikativen Anästhesiedaten führt zum Senden des Ereignisses *E_Done*. Durch dieses Ereignis wird die Konsistenzprüfungsaktion *checkConsistency()* aufgerufen, während das Objekt in den Zustand *S_ConsistentCheck* wechselt. Für diese Aktion wird angenommen, dass sie die eingegebenen Daten auf inhaltliche Korrektheit überprüft. Die möglichen Prüfungen und die Art und Weise der jeweiligen Prüfungen sind in der Aktionsbeschreibung zu hinterlegen. In Abhängigkeit vom Prüfungsergebnis wird erwartet, dass die Aktion mit dem Senden eines Ereignisses endet. Daher wird von der automatischen Verarbeitung eines Defaultereignisses abgesehen.

Präoperativregeln				
State	Event	Action	PostState	SubsequentEvent
S_IDLE	E_Praeoperativ	getPraeoperativ()	S_Praeoperativ	E_Done
S_Praeoperativ	E_Cancel	DEACTIVATE self	S_IDLE	E_Cancel
S_Praeoperativ	E_Done	checkConsistency()	S_ConsistentCheck	-
S_Praeoperativ	E_consistent	CREATE Intraoperativ	S_Intraoperativ	E_Intraoperativ
S_Praeoperativ	E_inconsistent	CREATE Intraoperativ	S_Intraoperativ	E_Intraoperativ
S_Intraoperativ	E_Intraoperativ	ACTIVATE Intraoperativ	S_Intraoperativ	E_Intraoperativ
S_Intraoperativ	E_Cancel	DELETE Intraoperativ	S_Praeoperativ	E_Cancel
S_Intraoperativ	E_Done	DEACTIVATE self	S_IDLE	E_Done
S_ConsistentCheck	E_consistent	finish()	S_Praeoperativ	E_consistent
S_ConsistentCheck	E_inconsistent	consistenceHandling()	S_Praeoperativ	E_inconsistent

Tabelle 22: Regeln des Pliable Objects Präoperativ

Im Fall von konsistenten Daten soll von der Aktion das Ereignis *E_consistent* gesendet werden, im Fall von inkonsistenten Daten das Ereignis *E_inconsistent*. Unabhängig vom jeweiligen Ereignis wechselt das Objekt in den Zustand *S_Praeoperativ*, während abhängig vom Ereignis eine Verarbeitungsaktion ausgeführt wird. Für den konsistenten Fall soll die Aktion *finish()* die Daten als korrekt markieren, während im inkonsistenten Fall die Aktion *consistenceHandling()* Einstellungen vornimmt, um inkorrekte Daten anzuzeigen. Dabei soll allerdings die weitere Verarbeitung nicht beeinträchtigt werden. Die Bewertung auf inkonsistente Daten kann ein Informationssystem zwar vornehmen, aber die Festlegung, ob Daten wirklich inkonsistent sind, trifft der Anwender.

Intraoperativregeln				
State	Event	Action	PostState	SubsequentEvent
S_IDLE	E_Intraoperativ	getIntraoperativ()	S_Intraoperativ	E_Done
S_Intraoperativ	E_Cancel	DEACTIVATE self	S_IDLE	E_Cancel
S_Intraoperativ	E_Done	checkConsistency()	S_ConsistentCheck	-
S_Intraoperativ	E_consistent	CREATE Postoperativ	S_Postoperativ	E_Postoperativ
S_Intraoperativ	E_inconsistent	CREATE Postoperativ	S_Postoperativ	E_Postoperativ
S_Postoperativ	E_Postoperativ	ACTIVATE Postoperativ	S_Postoperativ	E_Postoperativ
S_Postoperativ	E_Cancel	DELETE Postoperativ	S_Intraoperativ	E_Cancel
S_Postoperativ	E_Done	DEACTIVATE self	S_IDLE	E_Done
S_ConsistentCheck	E_consistent	finish()	S_Intraoperativ	E_consistent
S_ConsistentCheck	E_inconsistent	consistenceHandling()	S_Intraoperativ	E_inconsistent

Tabelle 23: Regeln des Pliable Objects Intraoperativ

Die weitere Verarbeitung startet mit der Erzeugung und der Aktivierung des Pliable Object *Intraoperativ*. Die Regeln dieses Objektes zeigt Tabelle 23. Sie realisieren ein analoges Verhalten des Objektes im Vergleich zu dem Pliable Object *Präoperativ*.

Das Pliable Object startet im Zustand *S_IDLE*. Die Verarbeitung beginnt mit der interaktiven Erfassung der intraoperativen Anästhesiedaten. Dem erfolgreichen Abschluss der Interaktion folgt eine Konsistenzprüfung der Daten. Unabhängig von der Konsistenzprüfung wird das Pliable Object *Postoperativ* angelegt und aktiviert. Die Regeln dieses Objektes zeigt Tabelle 24. Auch dieses Pliable Object zeigt ein analoges Verhalten, wie es bereits für die Objekte *Präoperativ* und *Intraoperativ* beschrieben worden ist.

Postoperativregeln				
State	Event	Action	PostState	SubsequentEvent
S_IDLE	E_Postoperativ	getPostoperativ()	S_Postoperativ	E_Done
S_Postoperativ	E_Cancel	DEACTIVATE self	S_IDLE	E_Cancel
S_Postoperativ	E_Done	checkConsistency()	S_ConsistentCheck	-
S_Postoperativ	E_consistent	DEACTIVATE self	S_IDLE	E_Done
S_Postoperativ	E_inconsistent	DEACTIVATE self	S_IDLE	E_Done
S_ConsistentCheck	E_consistent	finish()	S_Postoperativ	E_consistent
S_ConsistentCheck	E_inconsistent	consistenceHandling()	S_Postoperativ	E_inconsistent

Tabelle 24: Regeln des Pliable Objects Postoperativ

Mit Abschluss der Konsistenzprüfung der postoperativen Daten wird das Pliable Object deaktiviert und das Ereignis *E_Done* gesendet. Dieses Ereignis führt zum sequentiellen Deaktivieren der Pliable Objects *Intraoperativ*, *Präoperativ* und *Anästhesie*. Letztlich wird im Pliable Object *Patient* durch das Ereignis *E_Done* die Anästhesie als abgeschlossen markiert. Im Pliable Object *Patient* wird danach auf die Erstellung einer neuen Diagnose und die Durchführung einer neuen Anästhesie gewartet.

6.2.3 Nutzen der Erweiterbarkeit von Pliable Objects

Die direkte Prozessabbildung ist transparent und überschaubar, besitzt allerdings eine starre Struktur, die unflexibel wirkt. Die Ursache dieses Eindrucks basiert auf der möglichen Reduzierung der Anzahl an Objekten, die zur Realisierung einer Anwendung verwendet werden. Die Pliable Objects an sich verfügen über eine hohe Anzahl an Attributen und Aktionen. Proportional dazu steigt die Anzahl der Regeln in den jeweiligen Objekten, die unter Umständen eine große Menge an Objektzuständen abzudecken versuchen. Die Komplexität der Anwendung wird in die Regeln verlagert.

Die indirekte Prozessabbildung ist ungemein flexibel, bietet einen dynamischen Charakter und erlaubt eine individuelle Prozessgestaltung. Die Attribute und Aktionen werden über viele Pliable Objects verteilt. Die Regeln dieser Pliable Objects sind ähnlich proportional wie bei der direkten Prozessabbildung, d.h. sie liegen in eher geringer Menge vor, wobei sie die Objektzustände relativ überschaubar halten. Während die Pliable Objects für sich transparent erscheinen, wird die Komplexität der Anwendung über viele Objekte verteilt. Dadurch leidet eventuell die Übersicht über die eigentliche Anwendung.

In der Regel wird sich hinsichtlich einer Anwendung eine Mischform etablieren. Die Entscheidung, welche Attribute, Aktionen und Regeln in welchen Pliable Objects zu hinterlegen sind, entscheidet der Entwickler, der mit der Umsetzung einer Anwendung betraut ist. Die Entscheidung zwischen direkter und indirekter Prozessabbildung hat keine entscheidenden Auswirkungen auf die Anwendung, da im Rahmen einer Evolution der Anwendung, die Entscheidung revidiert und geändert werden kann. Die ist durch die Erweiterbarkeit des Pliable-Objects-Modell möglich, die das Modell bereits enthält.

Ein Entwickler kann frei bestehende Attribute, Aktionen und Regeln anderen Pliable Objects oder auch neu angelegten Pliable Objects zuweisen. Ebenso besteht die Möglichkeit, neue Attribute, Aktionen und Regeln in bestehende Pliable Objects zu integrieren. Im Folgenden wird gezeigt, wie der oben beschriebene Ansatz zur Realisierung des Kerndatensatzes der DGAI um weitere Informationen ergänzt werden kann.

Bei Patienten, die einmal einen Herzinfarkt erlitten haben, ist die Reinfarkttrate innerhalb des ersten halben Jahres nach dem Infarkt besonders groß. Daher muss sich ein Anästhesist vergewissern, ob ein Patient bereits einmal einen Herzinfarkt erlitten hat. Liegt ein Infarkt maximal ein halbes Jahr zurück, wird empfohlen, einen neuen Operationstermin festzulegen, sofern die Operation aufschiebbar ist - andernfalls sollten zusätzliche medizinische Überwachungsmethoden zum Einsatz kommen (vgl. [KS01], [KT06]).

Bei der Erweiterung dieser Überprüfung steht das Prinzip der Einzelinformationsabbildung vorrangig im Fokus. Die Ermittlung des Patientenzustandes wird außen vor gelassen.

Zunächst ist eine Domäne mit den *Herzinfarktzeiten* $\{ < 3 \text{ Monate}, < 6 \text{ Monate}, > 6 \text{ Monate} \}$ anzulegen. Diese Domäne wird dem anzulegenden Attribut *Herzinfarkt* mit dem Defaultwert *nicht bekannt* zugewiesen. Das Attribut wird dem Pliable Object *Präoperativ* zugewiesen, welches die Daten enthält, die der Anästhesist bei der Patientenzustandsbeurteilung erfasst.

Das Pliable Object *Präoperativ* wird zusätzlich um weitere Aktionen ergänzt, die die neue Information erfassen, bewerten und in Abhängigkeit von der Bewertung den weiteren Verlauf beeinflussen:

1. *getHeartData()*: Diese Action erfasst die herzinfarktrelevanten Daten.
2. *checkHeartData()*: Die Action gleicht die erfasste Herzinfarktangabe mit der Domäne ab. Bei einem möglichem Konflikt, der besagt, dass der Patient innerhalb des letzten halben Jahres einen Infarkt erlitten hat, überschreibt die Action das Folgeereignis mit dem Ereignis *E_Attention*, um die Prüfung einer möglichen Verschiebung anzustoßen.
3. *checkMovement()*: Mit dieser Action wird eine Verschiebung der Operation geprüft. Wenn die Verschiebung nicht möglich ist, überschreibt die Action das Folgeereignis mit dem Ereignis *E_NoMove*, um zu signalisieren, dass keine Verschiebung möglich ist.
4. *cancelOperation()*: Diese Action verschiebt die geplante Operation.

Die oben beschriebenen Erweiterungen können dynamisch zur Laufzeit des Informationssystems vorgenommen werden. Abbildung 56 zeigt das erweiterte Pliable Object *Präoperativ*. Abschließend werden die Regeln des Pliable Objects erweitert, um damit den bislang festgelegten Ablauf zur Beurteilung des Patientenzustands während der präoperativen Phase derart abzuändern, dass auch die neuen Informationen erfasst werden. Die Regelerweiterung wird in Tabelle 25 veranschaulicht. Die Änderungen sind in der Abbildung kursiv hervorgehoben.

Präoperativ
ASA: ASADomäne Herz: Wahrheitswerte Lunge: Wahrheitswerte Kreislauf: Wahrheitswerte Neurologie/ZNS: Wahrheitswerte Stoffwechsel: Wahrheitswerte Extreme Adipositas: Wahrheitswerte Herzinfarkt: Herzinfarktzeiten
getPraeoperativ() checkConsistency() finish() consistenceHandling() getHeartData() checkHeartData() checkMovement() cancelOperation()

Abbildung 56: Dynamisch erweitertes Pliable Object Präoperativ

Präoperativregeln				
State	Event	Action	PostState	SubsequentEvent
S_IDLE	E_Praeoperativ	getPraeoperativ()	S_Praeoperativ	E_Done
S_Praeoperativ	E_Cancel	DEACTIVATE self	S_IDLE	E_Cancel
S_Praeoperativ	E_Done	getHeartData()	S_Heart	E_Check
S_Heart	E_Check	checkHeartData()	S_Heart	E_OK
S_Heart	E_Attention	checkMovement()	S_Move	E_Move
S_Heart	E_OK	checkConsistency()	S_ConsistentCheck	-
S_Move	E_Move	cancelOperation	S_Praeoperativ	E_Cancel
S_Move	E_NoMove	checkConsistency()	S_ConsistentCheck	-
S_Praeoperativ	E_consistent	CREATE Intraoperativ	S_Intraoperativ	E_Intraoperativ
S_Praeoperativ	E_inconsistent	CREATE Intraoperativ	S_Intraoperativ	E_Intraoperativ
S_Intraoperativ	E_Intraoperativ	ACTIVATE Intraoperativ	S_Intraoperativ	E_Intraoperativ
S_Intraoperativ	E_Cancel	DELETE Intraoperativ	S_Praeoperativ	E_Cancel
S_Intraoperativ	E_Done	DEACTIVATE self	S_IDLE	E_Done
S_ConsistentCheck	E_consistent	finish()	S_Praeoperativ	E_consistent
S_ConsistentCheck	E_inconsistent	consistenceHandling()	S_Praeoperativ	E_inconsistent

Tabelle 25: Dynamisch erweiterte Regeln des Pliable Objects Präoperativ

Für den zu erweiternden Ablauf wird angenommen, dass im Anschluss an die Ermittlung des bisherigen Patientenzustands die Frage hinsichtlich eines etwaigen Herzinfarktes geklärt wird. Im Normalfall werden keine Probleme erwartet, so dass generell nach dieser Klärung mit der Erfassung der intraoperativen Anästhesiedaten fortgefahren werden kann. In Tabelle 25 ist dies derart hinterlegt, dass nach der Erfassung der Daten zum Patientenzustand anstelle der Check-Aktion die Erfassung der Angabe zum Herzinfarkt durch Aufruf der Aktion *getHeartData()* vorgenommen wird. Das Pliable Object *Präoperativ* wechselt in den neuen

Zustand *S_Heart*, um die zusätzliche Verarbeitung zu kennzeichnen. Es folgt die Prüfung, ob die Angabe operationsrelevante Folgen haben kann. Die allgemeine Erwartungshaltung ist, dass keine operationsrelevanten Folgen festgestellt werden, d.h. es wird das neue Ereignis *E_OK* gesendet, welches dafür sorgt, dass der ursprüngliche Ablauf mit der Fortführung der Konsistenzprüfung wieder aufgenommen wird.

Die Ausnahme tritt ein, wenn bei der Bewertung eines etwaigen Herzinfarktes eine Sonderprüfung vorgenommen werden muss. Dieser Sachverhalt wird von der Prüf-Action *checkHeartData()* über das Ereignis *E_Attention* signalisiert. Dieses Ereignis überschreibt das automatische Senden des Ereignisses *E_OK*. Die Verarbeitung des Ereignisses *E_Attention* unterbricht den Standardablauf und sorgt für die Prüfung einer Operationsverschiebung. Es wird in den neuen Zustand *S_Move* gewechselt und die Prüf-Action *checkMovement()* aufgerufen. Als Vorgabe für diese Ausnahme wird angenommen, dass die Operation verschoben werden kann, d.h. es wird vorgesehen, automatisch das Ereignis *E_Move* zu senden. Wenn die Verschiebung möglich ist, wird die Durchführung der Verschiebung über die Action *cancelOperation()* gestartet. Parallel wechselt das Pliable Object in den Zustand *S_Praeoperativ* und anschließend wird das Senden des Ereignisses *E_cancel* vorgesehen, um den bereits ursprünglich geplanten Ablauf für einen Abbruch starten zu lassen.

Falls sich hingegen herausstellt, dass trotz des Herzinfarkttrisikos die Operation nicht aufschiebbar ist, wird das Standardereignis durch das Senden von *E_NoMove* überschrieben. Dieses Ereignis sorgt für die Fortführung des Ablaufs mit der Konsistenzprüfung.

Auf dem vorgestellten Weg kann prinzipiell ohne Änderung an den Actions die Reihenfolge der Action-Aufrufe durch simple Änderungen in den Regeln erreicht werden. Analog zur Erweiterung am Beispiel des Herzinfarkttrisikos sind weitere Erweiterungen denkbar. Existiert beispielsweise eine Action *getDiabetes()*, die eine eventuelle Zuckererkrankung des Patienten erfasst und wünscht ein Anästhesist, dass die eventuelle Zuckererkrankung nach der Erfassung des Herzinfarktes erfasst werden soll, wären in den Regeln zusätzliche Zustände zu definieren, über die die Action *getDiabetes()* aufzurufen und deren Ergebnisbewertung abzuhandeln wäre.

6.3 Anästhesie-Dokumenten-Management

Die bisher vereinbarten Pliable Objects erlauben die Beschreibung einer konkreten Anästhesie. Die Informationen, die in den Attributen der Pliable Objects gespeichert werden, bilden konkrete Zustände, die unmittelbar mit einer Anästhesie zu tun haben. Im Folgenden seien diese Informationen als primäre Informationen verstanden.

Einzelne Anästhesien werden durch Abarbeitung eines entsprechenden Prozesses erfasst. Sie werden durch das System nachvollziehbar. Um mehr Einblick zu gewinnen, sollen nicht nur die Anästhesien selbst, sondern auch Informationen zum Prozessablauf gespeichert werden. Hierzu dient im Realen bislang das Narkoseprotokoll, welches durch ein Informationssystem abgelöst werden soll. Die Betrachtung der Prozesse führt zu einer Ausgestaltung der Prozesse durch prozessbezogene Zustandsinformationen, d.h. ein Prozess-Pliable-Object erhält neben den anästhesiebezogenen Attributen neue Attribute, die prozessspezifisch sind. Diese Informationen sollen als sekundäre Informationen klassifiziert werden.

Starken Einfluss auf die Wahl und Benennung der Attribute haben die Anwender. Sie nutzen die Definition solch „prozessoraler“ Zustandsinformationen zur abstrakten Betrachtung der Prozesse. Diese Prozesszustandsinformationen dienen so einer Qualitätssicherung. Durch

abstrakte Bewertungen und Gewichtungen der Prozesszustandsinformationen werden Planungen möglich, die eine Optimierung der Prozessabläufe zur Folge haben können.

Die Unterteilung in primäre und sekundäre Informationen ist aus technischer Sicht für Pliable Objects irrelevant. Technisch betrachtet werden alle Informationen, die in Attributen zu speichern sind, gleichrangig behandelt. Die Unterteilung ist somit nur ein Bewertungskriterium für die Anwendung, nicht aber für die technische Realisierung. Allerdings steht es den Entwicklern frei, die Erfassung primärer und sekundärer Informationen entweder zusammen in Pliable Objects zu erfassen oder diese Informationen auf verschiedene Pliable Objects zu verteilen. Aus Systemsicht ist die Verarbeitung der Einzelinformationen davon unabhängig. Eine rein anwenderspezifische, mögliche Bewertung wird im Folgenden kurz vorgeschlagen.

Ein Pliable Object, welches ein um Prozesszustandsattribute erweitertes Prozess-Pliable-Object darstellt, wird als **Dokument Pliable Object** definiert. Die Wahl des Begriffes leitet sich vom Begriff *Dokument* ab, da reale papiergestützte Dokumente im Bereich der Anästhesie als Protokoll verwendet werden. Ein Protokoll dient der Qualitätssicherung. Ein Dokumenten-Management-System soll ebenfalls qualitätssichernde Aufgaben wahrnehmen, weshalb ein zentrales Dokument in Form eines Pliable Objects etabliert wird.

Die Informationen, die ein *Dokument Pliable Object* aufnimmt, haben einen prozessbegleitenden Charakter. Sie werden als sekundäre Informationen vereinbart, die unterstützend einen Prozess begleiten.

Die Art der Informationen, die im Dokument zu erfassen sind, werden vom Anwender bestimmt und vorgegeben. Aus diesem Grund können die Daten nur abstrakt klassifiziert werden. Eine mögliche Klassifikation bleibt wegen der Unkenntnis der Anwenderwünsche spekulativ.

Die sekundären Informationen dienen rein administrativen Zwecken. Allgemein kann angenommen werden, dass sie sich zusammensetzen aus:

- Anwenderinformationen, die restriktiven Charakter besitzen, wie z.B. Verantwortlichkeiten und Berechtigungen,
- temporären Informationen, wie die konkreten Zeitbezüge in Form von (zeitlichen) Erfassungsdaten,
- ordnenden Informationen, die fachliche Zuordnungen repräsentieren.

Das *Dokument Pliable Object* bekommt mit den sekundären Informationen einen Rechtscharakter, der zusätzlich Aufschluss über die Archivierungswürdigkeit gibt. Im Zusammenhang ähnelt das *Dokument Pliable Object* von seiner Bedeutung her dem papierbasierten Dokument. Eine Repräsentation eines *Dokument Pliable Object* kann das Pliable Object Anästhesie aus Abbildung 51 sein.

Die Einrichtung des *Dokument Pliable Objects* erlaubt auch die Definition von Aktionen, die die tatsächlichen Umgänge von realen Dokumenten auf Rechner Ebene widerspiegeln. Auf diese Weise wird das System, welches Pliable Objects verwaltet, auf charmante Weise zu einem Dokumenten-Management-System erweitert. Verwendet man für die Abläufe, die sich an einem Dokument orientieren, ebenfalls Prozess-Pliable-Objects, so werden Vorgänge

realisiert, die den Ansprüchen von einfachen Workflows gerecht werden. In solch einem Fall erscheint das Dokumenten-Management-System mit Pliable Objects im Bereich der Anästhesie workflowbasiert.

6.4 Fazit

Die obigen Kapitelabschnitte zeigen die Ausprägung des in Kapitel 5 vorgestellten Basisinformationssystems in ein Anästhesie-Dokumentationssystem. Hinsichtlich der Anforderungen ist dabei vorrangig nur die online verfügbaren Informationen der DGAI zurückgegriffen worden (vgl. [DGAI]).

Bei diesen Angaben handelt es überwiegend um zu erfassende Daten einer Anästhesie. Diese Daten sind im vorgestellten Fall auf Attribute abgebildet worden, die das Basisinformationssystem ergänzten. Die neuen Attribute wurden mit neuen Pliable Objects verknüpft, die zusammen ein individuelles Anästhesie-Datenmodell beschreiben. Auf eine besondere Spezifikation spezieller Algorithmen konnte verzichtet werden, da die angedachte Datenerfassung und –verarbeitung der neuen Attribute mit den Standardmethoden bewältigt werden kann, die das Basisinformationssystem zur Verfügung stellt. Die Verarbeitungsreihenfolge der Pliable Objects ergab sich automatisch durch die Regeln der vorgestellten Pliable Objects.

Abschließend ist eine individuelle Erweiterung des Kerndatensatzes skizziert worden, wie man sie im praktischen Umfeld des Einsatzgebietes von Anästhesie-Dokumentationssystemen erwarten kann.

Die vorgestellte Informationssystemausprägung zeigt, dass ein auf dem Basisinformationssystem für Pliable Objects aufbauendes Anästhesie-Dokumentationssystem alle Ansprüche erfüllen kann, die von den Anästhesisten gestellt werden (vgl. Kapitel 3.2.7):

Grundlegende Anforderungen

Die Dokumentation mittels Pliable Objects erfolgt in einem einheitlichen, lesbaren und vollständig elektronischen Dokument. Der zu erfassende Datensatz des Dokumentes kann individuell zusammengestellt werden. Der Anwender entscheidet, ob dabei der komplette Kerndatensatz der DGAI abgebildet wird. Ferner ist jedes Datum eines Datensatzes individuell einstellbar und individuelle Datenaggregationen für betriebswirtschaftliche, wissenschaftliche und qualitätssichernde Zwecke werden unterstützt.

Funktionsspezifische Anforderungen

Individuelle Wertebereiche und die Eingabe unzulässiger Daten ist erlaubt. Neue medizinische Erkenntnisse können über neue Pliable Objects ins System gebracht werden. Plausibilitäten können dynamisch zur Laufzeit eingegeben und bearbeitet können. Die spezielle Ausprägung zur Übernahme interoperativer Daten, dem Aufbau einer Kommunikation zu einem übergeordneten Krankenhausinformationssystem (KIS), die revisionssichere Speicherung und das gewünschte Systemverhalten muss individuell vorgenommen werden. Das Modell unterstützt die dynamische Ergänzung und Änderung.

Technische Anforderungen

Der individuelle anästhesiologische Arbeitsablauf kann vollständig abgedeckt werden. Der Grad der Abdeckung richtet sich nach den Anwenderwünschen. Die Unterstützung neuer Technologien kann im laufenden System ergänzt werden.

Bedienungsorientierte Anforderungen

Die Anpassung des Dokumentationssystems an die Arbeitsabläufe der Anästhesisten wird durch Definition entsprechender Pliable Objects erlaubt. Der Umfang des „intelligenten“ Systemverhaltens hängt von den Ansprüchen der Anwender ab. Die Regeln der Pliable Objects erlauben die Verarbeitung nach den Qualitätskriterien, die der Anwender vorgibt.

Betriebswirtschaftliche Anforderungen

Das vorgestellte Modell der Pliable Objects lässt sich in günstigen Umgebungen realisieren, d.h. eine Umsetzung mit Freeware bzw. Open-Source-Software ist möglich.

Optionale Anforderungen

Das Konzept der Pliable Objects beinhaltet die Flexibilität, die optional gefordert wird. Prinzipiell ist jeder Erfassungsprozess individuell ausprägbar. Auch eine Prozessdokumentation kann ergänzt werden, indem entsprechende Pliable Objects spezifiziert werden.

Die Unterstützung der Netzwerktopologie des jeweiligen Krankenhauses hängt von der Ausgestaltung der Pliable Objects ab und liegt somit in der Verantwortung der Anwender.

Aufgrund der freien Festlegung von Domänen, werden per se auch Wertebereiche unterstützt, die im Rahmen einer multimedialen Erfassung dokumentationsbegleitend genutzt werden können. Das Informationssystem kann auch um gänzlich neue Technologien ergänzt werden.

Offene Anforderungen

Eine Personalisierung der Bedienoberflächen ist nicht diskutiert worden, genauso wenig wie Sicherheitsaspekte bei der Dokumentation. Derartige Aspekte sind durch Definition entsprechender Pliable Objects denkbar, allerdings stellen sie so fundamentale Systemanforderungen dar, dass es hier um eine spezielle Erweiterung des Basisinformationssystems handelt. Die Umsetzung dieser Anforderung ist jedoch äußerst umfangreich und würde den Umfang der vorgestellten Arbeit sprengen.

7 Zusammenfassung und Ausblick

Die schnelle und flächendeckende Umsetzung neuester Erkenntnisse des medizinischen Fortschrittes ist eine Aufgabe nationaler Gesundheitssysteme. Darin eingebunden ist die Anpassung medizinischer Informationssysteme und diese sollte die Erfassung neu spezifizierter Daten, die technische Unterstützung neuer Behandlungsmethoden sowie die Auswertung der neuen Daten und Behandlungsmethoden umfassen. Die Herausforderung für die Informationssysteme ist dabei, dass die neuen Erkenntnisse und Behandlungsmethoden zum Zeitpunkt der Inbetriebnahme der Informationssysteme noch nicht bekannt oder absehbar sind.

Im medizinischen Fachbereich der Anästhesie zeichnen sich die zu verarbeitenden Daten einerseits durch ihre Menge aus, andererseits durch eine anwenderbestimmte Interpretation der Daten. Beispielsweise kann in einer medizinischen Versorgungsanstalt die Patientenidentifikation durch eine Nummer definiert sein, während eine andere dies über eine alphanumerische Zeichenkette abdeckt. Derartige Interpretationsspielräume weisen auf eine inhomogene Struktur der zu erfassenden Daten hin, die von den Informationssystemen effizient, sicher und persistent zu speichern und zu verwalten sind, und auf denen die Informationssysteme Analysen ermöglichen müssen.

Das Bereitstellen von Funktionalitäten zur Erfassung und Speicherung dieser Daten sollte die Eingabe/Übernahme interaktiver Daten von externen Datenquellen, wie erreichbaren Datenbanken und von Analysegeräten umfassen. Zugriffe auf die Daten sollen wahlfrei und individuell ausprägbar sein.

Die bestehenden Softwarelösungen können diese Aufgabe nicht erfüllen. Aus dem Blickwinkel der Informatik startet die Entwicklung eines Informationssystems mit der Zusammenstellung der Anforderungen. Mittels einer Anforderungsanalyse wird ein Konzept für ein Softwaresystem zur Datenhaltung und Datenanalyse abgeleitet. Zur Erstellung des Konzeptes werden gängige Modelle verwendet, die die Grundlage für die Implementierung bilden. Modelle verfolgen das Ziel, eine direkte Beziehung zwischen einem modellierten Objekt bzw. einer modellierten Entität und deren Entsprechung in der Datenbank des Informationssystems und in der Programmiersprache, mit der die Funktionalität des Informationssystems implementiert wird, herzustellen. Die Grundlage der Beziehung ist dabei eine starre Definition des Objektes bzw. der Entität. Während der Modellierung kann ein Objekt zwar verschiedentlich verändert werden, aber nach Fertigstellung des Modells und dem damit verbundenen Abschluss des Konzeptes, sind alle Objekte bzw. Entitäten und alle darauf definierten Funktionalitäten des Modells unveränderlich. Dies hat zur Folge, dass nach Abschluss der Implementierung zwar eine Gesamtlösung zur Verfügung steht, die bzgl. der Anforderungen wartbar, effizient und ökonomisch arbeitet, in sich allerdings nicht veränderlich ist. Eine spätere Anpassung ist nicht automatisch gegeben, womit der Einsatz des Informationssystems nur solange gewährleistet ist, solange sich keine Anforderung ändert.

Um die Anforderung zu erfüllen, dass Informationssysteme an Änderungen angepasst werden können, auch solche, die zum Zeitpunkt der Anforderungsermittlung nicht absehbar waren, wurde das Modell der Pliable Objects entwickelt. Pliable Objects sind ein flexibles Konzept, welches die Wesentlichkeiten von Entitäten abstrahiert, um diese in einem Informationssystem erfassen und verarbeiten zu können.

Ein Pliable Object realisiert ein Aggregat bestehend aus je einer Menge von Attributen, Aktionen und Regeln. Dieses Aggregat besitzt keine direkte Umsetzung auf ein spezielles

Konzept, das in einer Programmiersprache zur Verfügung steht, wie z.B. das Klassenkonzept einer objektorientierten Programmiersprache. Das Modell der Pliable Objects basiert vielmehr auf einer Selbstbeschreibung, die von einem Informationssystem verstanden und ausgewertet werden muss. Über die Auswertung der Selbstbeschreibung wird die gewünschte Anwendung auf generische Art und Weise realisiert.

Innerhalb der Selbstbeschreibung werden Zustandsinformationen über die Attribute erfasst. Mögliche Änderungen an den Informationen werden als Zustandswechsel über Aufrufe von Aktionen vorgenommen. Die Steuerung von Zustandswechseln wird über die Regeln der Pliable Objects erreicht.

Attribute, Aktionen und Regeln sind jeweils als Menge ausgerichtet. Mittels Mengenoperationen sind diese veränderbar, womit diese Mengenoperationen die dynamische Änderbarkeit repräsentieren, die eine Auszeichnung von Pliable Objects darstellt.

Im Verlauf der Vorstellung des Pliable-Object-Modells konnte gezeigt werden, dass Attribute, Aktionen und Regeln selbst wieder eine spezielle Art von Pliable Objects darstellen. Diese Pliable Objects zeichnen sich dadurch aus, dass sie gleichberechtigt neben anwenderspezifisierbaren Pliable Objects stehen und keine Pliable-Object-Hierarchie im Sinne von Meta-Hierarchien aufbauen. Dies bedeutet, dass die Mengenoperationen, die als Funktionalitäten realisiert werden können, um Attribute, Aktionen und Regeln zu ermöglichen, automatisch dazu verwendet werden können, um individuelle Pliable Objects anzulegen, zu löschen und zu ändern.

Die Mengenoperationen realisieren eine grundlegende Form von Basisaktionen, die sich durch eine generische Ausprägung auszeichnen. Aufbauend auf den Basisaktionen konnte ein Basisinformationssystem spezifiziert werden, welches in der Lage ist, dynamisch um Pliable Objects erweitert werden zu können und diese zusätzlich zu verarbeiten. Es basiert auf Relationen und Funktionen, die skizziert und veranschaulicht wurden. Die auf den skizzierten Relationen und skizzierten Funktionen arbeitenden Basisaktionen realisieren eigenständige Prozesse, die wiederum der Verarbeitung von Daten dienen. Die einzelnen Prozesse entsprechen nach ihrer Realisierung den vorgestellten Mengenoperationen.

Relationen und Funktionen werden in der Regel in tabellarischer Form angegeben. Die Speicherung von Tabellen und ihren Inhalten ist eine Kernaufgabe von relationalen Datenbanken. Da die Ablauflogik der Prozesse auf eine Relation abgebildet wurde, bedeutet dies, dass mittels Pliable Objects die spezielle Ablauflogik, die bislang in Programmdateien fest codiert wird, in die Datenbank verschoben wird. Dort kann sie mittels SQL-Anweisungen verändert werden. Relationale Datenbanken wiederum bilden häufig die Basis für Informationssysteme. Ein Informationssystem, das Pliable Objects abbildet, gewährt geschultem Fachpersonal mittels generalisierten Zugriffen die Möglichkeit, Ablauflogiken in die gewünschte Form zu bringen. Die umfasst eventuell auch die Ausprägung der generischen Prozesse. Auch wenn bislang die relationale Abbildung favorisiert wurde, ist eine Realisierung in einer objektorientierten Datenbank nicht ausgeschlossen.

Unter der Annahme eines realisierten Informationssystems für Pliable Objects ist abschließend eine Anästhesie-Dokumentation vorgestellt worden. Als Grundlage ist dafür der von der DGAI verabschiedete Kerndatensatz verwendet worden.

7.1 Fazit

Pliable Objects bezeichnen eine Methodik, mit der Daten zunächst geschickt aufbereitet werden, um diese anschließend generisch zu bearbeiten und auszuwerten. Die Realisierung über ein Informationssystem ergibt folgende Vorteile.

- **Dynamische Erweiterbarkeit:**
Das Modell der Pliable Objects zeichnet sich durch eine hohe Dynamik aus. Diese Dynamik kann durch Informationssysteme gewährleistet werden und ermöglicht dadurch einerseits die dynamisch ausprägbare Verarbeitung von Pliable Objects und andererseits die dynamische Definition neuer Pliable Objects, die anschließend in die laufende Verarbeitung einfließen. Ein auf Pliable Objects basierendes Informationssystem ist darauf vorbereitet, neue Attribute, Aktionen und Regeln zu erstellen und diese miteinander über neue Pliable Objects zu verknüpfen oder in bestehenden Pliable Objects zu ergänzen. Bestehende Attribute, Aktionen und Regeln können verändert oder gar gelöscht werden. Damit eröffnet das Modell der Pliable Object einen neuen Ansatz für die Entwicklung zukünftiger Informationssysteme, da sich der Zweck des ursprünglich angedachte Anwendungssystems an neue Bedürfnisse anpassen lässt. Dies geschieht dabei unter Beibehaltung des ursprünglich eingegebenen Datenbestandes.
- **Gemeinsame Datenhaltung:**
Das Modell der Pliable Objects und die Daten der Pliable Objects werden in der gleichen Datenbank gehalten und können mit den gleichen Methoden bearbeitet werden. Die Klassen-Instanz-Beziehung, die normalerweise für die Speicherteilung verwendet wird, wird in der Pliable-Object-Verarbeitung als Is-a-Beziehung verstanden, die auch wieder gelöst werden kann. Eine Instanz kann dadurch von einer Klasse in eine andere Klasse verschoben werden oder ohne Verbindung zu einem Schema-Pliable-Object in der Datenbank des Informationssystems gespeichert sein.
- **Datenschnittstelle zwischen den Aktionen:**
Aktionen holen ihre Zustandsinformationen aus der Datenbank und schreiben die eventuell veränderten Zustandsinformationen wieder dahin zurück. Eine direkte Verknüpfung zwischen Aktionen ist nur gegeben, wenn es vom Anwender fest codiert wird. Ohne die Verknüpfung ist die unabhängige Verwendung von Aktionen möglich.
- **Separierte Steuerungslogik:**
Regeln dienen zur Steuerung von Zustandswechseln. Diese Logik ist in Informationssystemen häufig in Programmdateien unveränderbar hinterlegt. Pliable Objects trennen die Steuerungslogik von einer zustandsbehafteten Änderungslogik und erlauben so die Einflussnahme auf die Abläufe. Die Steuerung von Kontrollflüssen wird ohne Beeinflussung auf bestimmte Zustandswechsel möglich.
- **Einheitliche Bedienoberfläche:**
Die grafische Bedienoberfläche wird sich einheitlich gestalten, trotz der Unterschiede, die sich durch die Anzahl möglicher Pliable Objects ergibt und trotz der möglichen unterschiedlichen ablauftechnischen Ausgestaltung der Aktionen. Aus der Bedienoberfläche heraus werden alle Aktionenaufrufe erfolgen.
- **Dokumentation und Reproduzierbarkeit:**
Wenn man davon ausgeht, dass eine Analyse nur über eine Sequenz von mehreren Aktionen mit der entsprechenden Anzahl an Attributen und Zwischenergebnissen möglich

ist, entsteht das Problem der konsistenten Verwaltung dieser Datensätze und der durch den Aktionenaufruf generierten Aktionendaten. Pliable Objects bieten dem Anwender die Option, das Problem nach seinen Wünschen zu lösen und die gewünschten Informationen in entsprechenden Attributen zu speichern. Der Anwender hat die Möglichkeit, diese Daten untereinander und mit Basisdaten zu verlinken, um eine lückenlose Dokumentation nach seinen Wünschen zu gewährleisten. Diese Spezifikation kann einerseits als Protokoll bzw. Dokumentation des Ablaufs angesehen werden, andererseits aber, wenn man einen Interpretier zur Abarbeitung der Aktionenfolge nutzt, können Anästhesiespezifikationen ohne weitere interaktive Nutzereingriffe automatisch ausgeführt werden.

- **Skalierbarkeit:**
Ein Informationssystem für Pliable Objects erlaubt ein integriertes Arbeiten in der Form, dass das System vollständig vom Anwender erweitert werden kann und gemäß seinen Ansprüchen an die Skalierbarkeit ausgeprägt werden kann. Die Mächtigkeit der Einflussnahme basiert auf dem Realisierungsstand des allgemeinen Basiskonzepts von Pliable Objects.

7.2 Ausblick

Das entwickelte Modell und das vorgestellte Basisinformationssystem für Pliable Objects bilden den Ausgangspunkt für zahlreiche mögliche Folgearbeiten. Einige mögliche Erweiterungen werden kurz vorgestellt.

7.2.1 Ableitung intuitiver Bedienoberflächen

Heutige Informationssysteme verfügen generell über eine geeignete visuelle Ausgabe in Form graphischer Bedienoberflächen. Die Bedienoberflächen besitzen verschiedene Elemente, die auf codierte Tastatur- und Mausereignisse reagieren und sie als Anwendereingaben verarbeiten.

Für Pliable Objects müssen geeignete Bedienoberflächen gestaltet werden. Im Gegensatz zur allgemeinen Entwicklungslage, dass Oberflächen von Hand starr implementiert werden, wäre auf Basis von Pliable Objects eine Generierungskomponente denkbar. Die Generierungskomponente bekommt die Pliable-Object-Daten übermittelt und baut eine Bedienoberfläche auf, in der beispielsweise Textfelder für Dateneingaben für die entsprechend zu erfragenden Attribute dienen. Den Textfeldern wären in Form von Labels die Attributnamen beigeordnet. Die unterstützenden Prozesse könnten auf der Oberfläche in Form generierter Buttons zur Verfügung gestellt werden. Die Beschriftung der Buttons wäre durch die Bezeichnungen der Prozesse gegeben und ein separat zu spezifizierendes Pliable Object triggert je nach Anwendereingabe das zugehörige Ereignis.

Auf diese Weise sind so genannte elektronische Formulare (E-Forms) generierbar, die automatisch die Unterstützung für die Eingabe, Anzeige, Ausgabe und Verwaltung variabler Informationen, wie sie in papierbasierten Formularen verwendet werden, bieten. Der Mechanismus zur automatischen Generierung der Bedienoberfläche muss differenzierter untersucht und erforscht werden. Das Ziel sollte sein, eine intuitiv bedienbare Bedienoberfläche aufzubauen. Hierzu sind Erkenntnisse zum Design zu finden, die Beschränkungen definieren, die zur Gestaltung der Bedienoberflächen verwendet werden. Z.B. dürfen automatisch generierte Bedienoberflächen nicht mit Eingabeelementen

überfrachtet sein. Geeignete Mechanismen sollten automatische Seitenumbrüche etc. verwenden, um eine geschickte Datenaufbereitung über mehrere Fensterseiten zu erreichen.

7.2.2 Geeigneter Mechanismus zur Bearbeitung von Fehlern und Fehlzuständen

Das Informationssystem soll im Idealfall zu einem Assistenten werden, der dem Anästhesisten die Tätigkeit der Protokollierung seiner Arbeit abnimmt und seinen Anweisungen Folge leistet. Dabei sollte es „intelligent“ sein und ihn vor Fehlern bewahren sowie Fehleingaben ablehnen, ähnlich wie eine gute Sekretärin den diktierten Brief ihres Chefs entsprechend aufbereitet. Die Ablehnung von Fehleingaben muss allerdings konfigurierbar sein. Letztlich obliegt die Feststellung für die Korrektheit eines Wertes dem Anwender, d.h. seine Eingabe darf erst dann abgelehnt werden, wenn der Anwender das System derart konfiguriert hat, dass die Eingabe abzulehnen ist.

Ein derartiges konfigurierbares Verhalten sollte auch mit etwaigen Fehlzuständen umgehen können. Im Rahmen der Anwendung und in Verbindung mit einer Datenbank muss eine Art Konsistenzprüfer entwickelt werden. Dieser wird bei Bedarf durch die vorgestellten Prozesse eingebunden. Aufgrund der überschaubaren Relationen des Pliable-Object-Modells sind zur Wahrung der Konsistenz wenige Anfragen durchzuführen.

So kann eine Attribut-Wert-Prüfung über eine Suche auf die Instanzrelation abgebildet werden, die all diejenigen Attribute filtert, deren angegebener Wert nicht Element der Domäne des Attributes ist. Über die gefundenen Elemente werden die Instanzen gefunden, die einen attributbedingten inkonsistenten Zustand besitzen. Die gefundenen Pliable-Objekt-Instanzen sind in einer geeigneten Form aufzubereiten, so dass sie nachbearbeitet werden können. Dabei ist zu berücksichtigen, dass vom System fehlerhaft markierte Pliable Objects dennoch vom Anwender als korrekt eingestuft werden können.

Ähnlich zu einer Attribut-Wert-Prüfung wäre eine Prüfung von Aktionen denkbar, die eine Beurteilung über die Qualität einer Aktion macht. So könnte geprüft werden, ob alle in der Aktion bearbeiteten Attribute auch dem Pliable Object zu eigen sind, an welches die Aktion gebunden ist. Zusätzlich sollte untersucht werden, ob eine Prüfung auf semantische Korrektheit von Aktionen möglich sein könnte.

Analog dazu könnte eine Regelprüfung kontrollieren, ob die Regeln eines Pliable Objects in sich konsistent sind und alle Zustandswechsel des Pliable Objects abbilden. Es sollte vermieden werden, dass Pliable Objects in ungewöhnlichen Zuständen verharren und diese nie wieder verlassen können.

Über solch einen Mechanismus wäre es auch denkbar, alle Pliable Objects zu finden, die einen regelbedingten inkonsistenten Zustand besitzen.

7.2.3 Integration eines rollenbasierten Bedienkonzeptes

Das Pliable-Object-Modell ist bislang nicht restriktiv. Weitere Forschungsuntersuchungen sollten angestellt werden, inwieweit die Einhaltung von Datenschutz-Richtlinien in das Pliable-Object-Modell integriert werden kann. Dies bedeutet, dass Restriktionen auf Werte definierbar sein sollten, die sich aus dem Anwendungszusammenhang und der anwenderspezifischen Gewichtung der Werte ergeben. Zugriffsrestriktionen, die allgemein

auf einem Benutzerkonzept basieren, wurden nicht betrachtet. Ein entsprechendes Anwenderkonzept auf Basis von Pliable Objects muss noch erarbeitet werden.

Erreicht werden könnte dies durch die Generierung eines Pliable Objects, welches den Anwender repräsentiert. Ein Attribut des Anwenders wäre für die möglichen Rollen, die ein Anwender annehmen kann, anzugeben. Dessen Attributdomäne spezifiziert die Domänen, durch die alle Rollen gegeben sind. Diese Domäne ist anzulegen.

Die Basisprozesse, die die Verarbeitung der Pliable Objects gewährleisten, sind um entsprechende Prüfroutinen zu erweitern und dahingehend zu ergänzen, dass rollenabhängige Zugriffsrestriktionen erfüllt werden. Wünschenswert wäre die Etablierung eines Rollenkonzeptes in einem Anästhesie-Dokumentationssystem, da derzeit die Zugriffsbeschränkung aktueller Anästhesie-Dokumentationssysteme nur durch die Benutzerverwaltung des zugrundeliegenden Betriebssystems gegeben ist. Hierfür wäre die Spezifikation von Aktionen denkbar, die eventuell die Benutzerverwaltung des Betriebssystems ins Informationssystem integrieren.

7.2.4 Unterstützung von teamorientierten Arbeiten im Sinne von Groupware

Groupware bezeichnet die Unterstützung zur gemeinsamen Verwendung und Organisation von Informationen in einer verteilten Mehrbenutzerumgebung (vgl. [SSU01]). Über ein geeignetes Rollenkonzept kann eine Mehrbenutzerumgebung zur Verfügung gestellt werden. Auf der Mehrbenutzerumgebung können Arbeitsgruppen aufgesetzt werden, die koordinierte Aufgabenwahrnehmung im Team ermöglichen. Werden ferner Pliable Objects generiert, die entsprechend geeignete und aufgabenorientierte Werkzeuge zur Kooperation, Kommunikation und Koordinierung von Arbeitsgruppen darstellen, wäre das System als Groupware-unterstützendes System gekennzeichnet. Die typischen Merkmale von allgemeiner Groupware sind dynamische, datenbankgestützte Ablage, Verwaltung von Informationen aus Bürokommunikationsanwendungen, kooperative Bearbeitung von Dokumenten durch zahlreiche Anwender, gesteuerte Replikation von Informationen zur mehrfachen Nutzung sowie der koordinierte und restriktive Zugriff auf Dokumente mittels gezielter Suchstrategien. Die Merkmale wären durch entsprechende Pliable Objects und Prozessketten einzurichten. Wahrnehmbar wäre die Groupwareunterstützung, wenn die grundlegende, aufbauende Infrastruktur auf Pliable Objects die Kommunikation zwischen allen Anwendern steuert. Dies schließt die Unterstützung von dynamischer Zuteilung, Koordination und Planung von Anwendern, Ressourcen und Terminen mit ein. Diskussionsforen auf Pliable-Object-Basis nehmen die Informationen auf und stellen sie allen Zugriffsberechtigten zur Verfügung.

Im Rahmen der allgemeinen Anästhesie-Dokumentation bzw. des allgemeinen Anästhesie-Managements sind keinerlei Groupware-Mechanismen spezifiziert. Durch die täglichen Vorgänge in der Anästhesieabteilung einer Klinik sind sie allerdings faktisch gegeben. Um eine Abbildung auf Pliable-Object-Basis zu gewähren, müsste ein möglicher Anforderungskatalog erstellt werden. Sukzessive ist dieser durch geeignete Pliable Objects in das System zu integrieren.

Außerdem besteht häufig der Wunsch, zwei Systeme miteinander zu koppeln (vgl. [GS05]). Dies wird im Rahmen einer Systemmigration erreicht. Dabei müssen redundante Daten aufeinander abgebildet werden. Zusammenhänge auf Attributebene werden relativ schnell gefunden; auf einer Verhaltensebene ist dies dahingehend äußerst schwierig. Pliable Objects erlauben das Finden und Vergleichen von Aktionen, die auf denselben Attributen arbeiten.

Damit wird es möglich, den Aufwand abzuschätzen, der benötigt wird, um die Funktionalität, die die Anwender gewohnt sind, im migrierten System zu realisieren. Dies ist bislang ein häufig unterschätzter Kostenfaktor, da die Unverträglichkeit zwischen einzelnen Algorithmen erst in der Testphase neuer Systeme sichtbar wird.

7.2.5 Aufhebung der Beschränkung durch Schema-Pliable-Objects

Das vorgestellte Pliable-Object-Modell ist in der betrachteten Art und Weise datenorientiert. Im Rahmen dieser Orientierung ist ein Datum, repräsentiert als Wert, einem Attribut zugeordnet, welches einem Schema-Pliable-Object zu eigen ist. Daraus resultiert der Fall, dass jedes Instanz-Pliable-Object einem Schema-Pliable-Object zugeordnet ist. Die Zuordnung Schema zu Instanz entspricht in Analogie der Klassen-Instanz-Beziehung des Objektorientierten Modells.

Vor der klaren Verarbeitung von Informationen müssen somit immer Schema-Pliable-Objects angelegt werden. Es besteht allerdings die Möglichkeit einer Anforderung, die das Erfassen von Informationen unabhängig von Schema-Pliable-Objects verlangt. Beispielsweise könnte verlangt werden, ein Informationssystem aufzubauen, das zunächst konkrete Erkenntnisse speichert. Auf den gespeicherten Erkenntnissen werden Analysen durchgeführt, deren Bewertung zur Definition von Schema-Pliable-Objects führt.

Ein derartiges Vorgehen wird aktuell nicht unterstützt. Quasi bedeutet dies, dass schemalose Instanzen zu verarbeiten wären, deren Verarbeitung letztlich zum Anwendungsmodell führt. Die Realisierung eines solchen Vorgehensmodells wäre ein weiterer, interessanter Forschungsschwerpunkt.

8 Anhang

Ergänzend zum vorgestellten Modell der Pliable Objects werden im Folgenden Informationen präsentiert, die im Fließtext der Arbeit nicht untergebracht werden konnten.

8.1 SQL-Anweisungen eines Pliable-Object-Datenbankschemas

In Kapitel 5 wird das relationale Modell einer Datenbank präsentiert, die in der Lage ist, Pliable Objects zu verarbeiten. In Ergänzung zu den dort vorgestellten Relationen könnten die SQL-Anweisungen zum Aufbau einer den Relationen entsprechenden Datenbank wie folgt angegeben werden.

8.1.1 Create-Anweisungen für die Sonder-Pliable-Objects

Die Sonder-Pliable-Objects *Pliable Object*, *Attribut*, *Domaene* und *Action* können durch folgende Create Statements angelegt werden.

```
CREATE TABLE IF NOT EXISTS PliableObject (  
    POID BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    Bezeichnung LONGTEXT NOT NULL);
```

```
CREATE TABLE IF NOT EXISTS Domaene (  
    DomaeneID BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY ,  
    Name VARCHAR(255) Not NULL,  
    Wertemenge LONGTEXT Not NULL);
```

```
CREATE TABLE IF NOT EXISTS Attribut (  
    AttributID BIGINT NOT NULL AUTO_INCREMENT Primary key,  
    Bezeichnung LONGTEXT NOT NULL,  
    Domaene BIGINT,  
    Wert LONGTEXT);
```

```
CREATE TABLE IF NOT EXISTS Action (  
    ActionID BIGINT NOT NULL AUTO_INCREMENT Primary key,  
    Bezeichnung LONGTEXT NOT NULL,  
    Ablauf LONGTEXT);
```

8.1.2 Create-Anweisungen für Zustandsmaschinen

Das Pliable Object Regel wird durch eine Zustandsmaschine realisiert. Zustandsmaschinen können in Relationen verwaltet werden, die sich durch folgende Create Statements erzeugen lassen.

```
CREATE TABLE IF NOT EXISTS Event (  
    ID BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    Name VARCHAR(255) Not NULL);
```

```
CREATE TABLE IF NOT EXISTS State (  
    ID BIGINT NOT NULL AUTO_INCREMENT Primary key,  
    Name VARCHAR(255) Not NULL,  
    EntryCallbackID BIGINT,  
    ExitCallbackID BIGINT);
```

```

CREATE TABLE IF NOT EXISTS Transition (
    StateID BIGINT NOT NULL references State(ID),
    EventID BIGINT NOT NULL references Event(ID),
    PostStateID BIGINT NOT NULL references State(ID),
    CallbackID BIGINT,
    DefaultFollowEvent BIGINT,
    PRIMARY KEY (StateID, EventID));

```

8.1.3 Create-Anweisung für die Relation Value

Die Relation Value nimmt die Instanzen von beliebigen Pliable Objects auf. Die zugehörige Relation wäre durch folgendes Create Statement realisierbar.

```

CREATE TABLE IF NOT EXISTS Instanz (
    InstanzID BIGINT NOT NULL AUTO_INCREMENT ,
    POID BIGINT NOT NULL,
    AttributID BIGINT NOT NULL,
    Wert LONGTEXT NOT NULL,
    PRIMARY KEY (InstanzID, POID, AttributID));

```

8.1.4 Create-Anweisung für die Own-Relationen

Die Verknüpfungen zur Abbildung der Zugehörigkeit von Attributen, Aktionen und Regeln werden in den Own Relationen abgelegt. Diese sind durch folgende Create-Statements erzeugbar.

```

CREATE TABLE IF NOT EXISTS ZuEigen (
    POID BIGINT NOT NULL,
    AttributID BIGINT NOT NULL,
    PRIMARY KEY (POID, AttributID));

```

```

CREATE TABLE IF NOT EXISTS ZuEigeneActions (
    POID BIGINT NOT NULL,
    ActionID BIGINT NOT NULL,
    PRIMARY KEY (POID, ActionID));

```

```

CREATE TABLE IF NOT EXISTS ZuEigeneRules (
    POID BIGINT NOT NULL,
    StateID BIGINT NOT NULL,
    PRIMARY KEY (POID, StateID));

```

```

CREATE TABLE IF NOT EXISTS ZuEigeneInstanzen (
    POID BIGINT NOT NULL,
    InstanzID BIGINT NOT NULL,
    PRIMARY KEY (POID, InstanzID));

```

8.2 Initialisierung der Selbstbeschreibung

Pliable Objects zeichnen sich durch ihre Selbstbeschreibung aus. Um diese Basiseigenschaft in der Datenbank zu hinterlegen und so die Funktionalität zur Verfügung zu stellen, dienen folgende Insert-Statements, die die Basisdaten bereitstellen, um die Selbstbeschreibung zu ermöglichen.

8.2.1 Beispiel für Domänen

Als Beispiel für Domänen dienen die Datentypen Integer und Zeichenkette, deren Wertemengen jeweils über einen regulären Ausdruck beschrieben werden.

```
INSERT INTO Domaene VALUES (1, \"BigInt\", \"[\\d]*\");
INSERT INTO Domaene VALUES (2, \"VarChar\", \"[.]*\");
```

Ferner stellen die Instanzen von Pliable Objects selbst Wertebereiche dar. Dies wird durch folgende Insert-Statements in der Datenbank hinterlegt.

```
INSERT INTO Domaene VALUES (4, \"Attribut\", \"\");
INSERT INTO Domaene VALUES (5, \"Domaene\", \"\");
INSERT INTO Domaene VALUES (6, \"PliableObject\", \"\");
INSERT INTO Domaene VALUES (7, \"Action\", \"\");
INSERT INTO Domaene VALUES (8, \"Rule\", \"\");
```

8.2.2 Basisattribute

Die Basisattribute werden durch folgende Insert-Statements in die Datenbank eingetragen.

```
INSERT INTO Attribut values ( 1, \"AttributID\", 1, \"\");
INSERT INTO Attribut values ( 2, \"Bezeichnung\", 2, \"\");
INSERT INTO Attribut values ( 3, \"Domaene\", 5, \"\");
INSERT INTO Attribut values ( 4, \"Wert\", 2, \"\");
INSERT INTO Attribut values ( 5, \"DomaeneID\", 1, \"\");
INSERT INTO Attribut values ( 6, \"Name\", 2, \"\");
INSERT INTO Attribut values ( 7, \"Wertemenge\", 2, \"\");
INSERT INTO Attribut values ( 8, \"POID\", 1, \"\");
INSERT INTO Attribut values ( 9, \"InstanzID\", 1, \"\");
INSERT INTO Attribut values (10, \"ActionID\", 1, \"\");
INSERT INTO Attribut values (11, \"EventID\", 1, \"\");
INSERT INTO Attribut values (12, \"StateID\", 1, \"\");
INSERT INTO Attribut values (13, \"PostStateID\", 1, \"\");
INSERT INTO Attribut values (14, \"DefaultFollowEvent\", 1, \"\");
INSERT INTO Attribut values (15, \"CallbackID\", 1, \"\");
INSERT INTO Attribut values (16, \"EntryCallbackID\", 1, \"\");
INSERT INTO Attribut values (17, \"ExitCallbackID\", 1, \"\");

INSERT INTO PliableObject values ( 1, \"Attribut\" );
INSERT INTO PliableObject values ( 2, \"Domaene\" );
INSERT INTO PliableObject values ( 3, \"PliableObject\" );
INSERT INTO PliableObject values ( 4, \"Action\" );
INSERT INTO PliableObject values ( 5, \"Rule\" );
```


9 Literaturverzeichnis

- [Am03] Ammenwerth, E.: Die Bewertung von Informationssystemen des Gesundheitswesens, Habilitationsschrift 2003, <http://iig.uit.at/dokumente/r17.pdf> [Stand: 01.02.2011]
- [AMA05] Adelman, S.; Moss, L.; Abai, M.: Data Strategy, Pearson Education Inc., 2005
- [ASU99] Aho, A.V.; Sethi, R.; Ullmann, J.D.: Compilerbau, Oldenbourg, 1999
- [Ba00] Balzert, H.: Lehrbuch der Software-Technik Band I Software-Entwicklung, Spektrum Akademischer Verlag, 2. Auflage, 2000
- [Be96] Becker, S. U.: Constraint-Netze zur Bearbeitung zeitkritischer Koordinationsprobleme in der industriellen Fertigung, Dissertation, Shaker Verlag, 1996
- [BH98] Bauer, B.; Höllerer, R.: Übersetzung objektorientierter Programmiersprachen – Konzepte, abstrakte Maschinen und Praktikum „Java-Compiler“, Berlin; Heidelberg; New York; Barcelona; Budapest; Hongkong; London; Mailand; Paris; Santa Clara; Singapur; Tokio, Springer, 1998
- [BH03] Bryant, R. E.; O’Hallaron, D.: Computer Systems, Prentice Hall, 2003
- [BI68] Brauer, W.; Indermark, K.: Algorithmen, Rekursive Funktionen und Formale Sprachen, BI Hochschultaschenbuch 817, Mannheim, 1968
- [BMS06] Bernus, P.; Mertins, K.; Schmidt, G. (eds.): Handbook on Architectures of Information Systems, second edition, 2006
- [Bo94] Booch, G.: Object-oriented Analysis and Design with applications, 2. Auflage, Benjamin Cummings, Redwood/CA., 1994
- [Bö00] Böhm, M.: Entwicklung von Workflow-Typen, Springer, 2000
- [Br07] Branitzki, P. ; Junger, A.; Bleicher, W.; Pollwein, B.; Prause, A.; Röhrig, R.; Specht, M.: Spezielle Empfehlungen und Anforderungen zur Implementierung eines Anästhesie-Informationen-Management-Systems (AMS). (2007) Anästh Intensivmed 48:Suppl. 282-290, http://www.uniklinik-freiburg.de/anaesthesie/live/medizintechnik/ONline/ONlinePub/Narkose_Artikel_5_07.pdf [Stand: 21.02.2010]
- [Br09] Brugger, R.: IT-Projekte strukturiert realisieren – Situationen analysieren, Lösungen konzipieren – Vorgehen systematisieren, Sachverhalte visualisieren – UML und EPKs nutzen, 2.Auflage, Vieweg + Teubner, Unveränderter Nachdruck, 2009
- [Br97a] Bruder, M.; Rehfeld, W.; Seeger, T.; Strauch, D. (Hrsg.): Grundlagen der praktischen Information und Dokumentation, 4., völlig neu gefasste Ausgabe, Band 1, K. G. Saur Verlag, 1997

- [Br97b] Bruder, M., Rehfeld, W., Seeger, T., Strauch, D. (Hrsg.): Grundlagen der praktischen Information und Dokumentation, 4., völlig neu gefasste Ausgabe, Band 2, K. G. Saur Verlag, 1997
- [BS02] Broy, M.; Siedersleben, J.: Objektorientierte Programmierung und Softwareentwicklung – Eine kritische Einschätzung, Informatik Spektrum, 25.2.2002
- [Ca96] Casanave, C. (ed.): OMG Common Facilities RFP-4 Common Business Objects and Business Object Facility, OMG TC Document CF/96-01-04, <http://www.omg.org> [Stand: 21.02.2010]
- [Ca97] Casanave, C.: Business-Object Architectures and Standards, In [Su97]: OOPSLA'95 Workshop Proceedings, Springer Verlag, 1997; S.7 - 28.
- [CC01] Cuhls, M.; Cuhls, H.: Patienteninformation – Ablauf einer Narkose in Bildern, 15.06.2001, <http://www.meb.uni-bonn.de/institute/kliansint/patient/inhalt.htm> bzw. <http://www.ukb.uni-bonn.de/kai/Patient/narkose.html> [Stand: 21.02.2010]
- [CCG00] Carey, J.; Carlson, B.; Graser, T.: SanFrancisco Design Patterns, Addison-Wesley, 2000
- [Ch76] Chen, P.: The entity-relationship model-toward a unified view of data, ACM Transactions on Database Systems (TODS) Volume 1 Issue 1, March 1976, ACM New York, NY, USA
- [CY94a] Coad, P.; Yourdon, E.: OOA – Objektorientierte Analyse, Prentice Hall Verlag, 1994
- [CY94b] Coad, P.; Yourdon, E.: OOD – Objektorientiertes Design, Prentice Hall Verlag, 1994
- [De05] Deacon, J.: Object-Oriented Analysis and Design – A pragmatic approach, Pearson Education Limited, 2005
- [De78] DeMarco, T.: Structured Analysis and System Specification. NewYork, Yourdon Press, 1978
- [DGAI] http://www.dgai.de/06_1_00tabelle.htm#zu_viii [Stand: 21.02.2010]
- [DHW71] Dreßler, H.; Hammelmann, I.; Wilhelm, H.A.: Entscheidungstabellen – ein Lehrprogramm, a i v – d i d – a k t e a m, Druckerei Ph. Reinheimer, Darmstadt, 1971
- [Ed99] Stefan Edelkamp; Datenstrukturen und Lernverfahren in der Zustandsraumsuche; In „Dissertationen zur Künstlichen Intelligenz“ Band 201, Sankt Augustin, infix Verlag, 1999

- [Er01] Erler, T.: Business Objects als Gestaltungskonzept strategischer Informationssystemplanung, Bochumer Beiträge zur Unternehmensführung und Unternehmensforschung; Band 58, Peter Lang GmbH Europäischer Verlag der Wissenschaften, 2001
- [Er00] Erler, T.: UML – Das Einsteigerseminar, vmi-Buch, 2000
- [Er08] Ernst, H.: Grundkurs Informatik – Grundlagen und Konzepte für die erfolgreiche IT-Praxis – Eine umfassende, praxisorientierte Einführung, 4., vollständig überarbeitete Auflage, 2008
- [ES98] Eeles, P.; Sims, O.: Building Business Objects, Wiley & Sons, 1998
- [FG08] Freund, J.; Götzer, K.: Vom Geschäftsprozess zum Workflow, Carl Hanser Verlag München, 2008
- [Fr01] Friedl, J. E. F. : Reguläre Ausdrücke, O'Reily, 2001
- [Ga95] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: Entwurfsmuster, Addison-Wesley, 1995
- [GBB01] Grässle, P.; Baumann, H.; Baumann, P.: UML projektorientiert – Geschäftsprozessmodellierung, IT-System-Spezifikation und Systemintegration mit der UML, 1. Nachdruck der 1. Auflage, Galileo Computing, 2001
- [Go11] Goll, J.: Methoden und Architekturen der Softwaretechnik, Vieweg + Teubner, 2011
- [Gr98] Griffel, F.: Componentware - Konzepte und Techniken eines Softwareparadigmas, dpunkt.verlag, 1998
- [GS05] Güting, R. H.; Schneider, M.: Moving Objects Databases, Morgan Kaufmann, 2005
- [Gü95] Günther, A. (Hrsg.): Wissensbasiertes Konfigurieren – Ergebnisse aus dem Projekt PROKON, Infix, 1995
- [Ha05] Haas, P.: Medizinische Informationssysteme und Elektronische Krankenakte, Springer-Verlag, 2005
- [He92] Heuer, A.: Objektorientierte Datenbanken, Konzepte, Modelle, Systeme, Addison-Wesley, 1992
- [He96] Heckmann, W.: Grundlagen der Dokumentenverarbeitung, Addison-Wesley, 1996
- [Hi11] Hildebrand, K.; Gebauer, M.; Hinrichs, H.; Mielke, M.(Hrsg): Daten- und Informationsqualität – Auf dem Weg zur Information Excellence, 2., aktualisierte und erweiterte Auflage, Vieweg + Teubner, 2011

- [HK06a] Hänsgen, P.; Kühne, S.: Modellgetriebene Softwareentwicklung zur Lösung von Integrations- und Migrationsproblemen am Beispiel des E-Commerce-Systems Intershop Enfinity Suite, In: Klaus-Peter Fähnrich; Stefan Kühne; Andreas Speck; Julia Wagner (Hrsg.): Integration betrieblicher Informationssysteme: Problemanalysen und Lösungsansätze des Model-Driven Integration Engineering. Eigenverlag Leipziger Informatik-Verbund (LIV), September 2006,
<http://orvia.informatik.uni-leipzig.de/uploads/Projekt.OrviaBuch2006/HaeKue06.pdf> [Stand: 21.02.2010]
- [HK06b] Han, J.; Kamber, M.: Data Mining – Concepts and Techniques, Second Edition, Morgan Kaufmann Publishers, 2006
- [HMP04] Hoffer, J.; McFadden, F.; Prescott, M.B.: Modern Database Management, 7th ed., Prentice Hall, 2004
- [Ho08] Hoffmann, D. W.: Software-Qualität, Springer Verlag Berlin Heidelberg, 2008
- [Ho96] Houy, C.: Datenmanagement für Workflowprozesse, Gabler Verlag, 1996
- [HP96] Haas, P.; Pietrzyk, P.: Generelle Vorgehensweise und Projektphasen bei der Systemauswahl, Uniklinik Göttingen, Auszug aus Tagungsband „Praxis der Informationsverarbeitung im Krankenhaus“, ECOMED 1996,
www.uni-due.de/~tmi030/ak_chirurgie/info/kistg-v2.doc
 [Stand: 21.02.2010]
- [HS00] Herzum, P.; Sims, O.: Business Component Factory, John Wiley & Sons, 2000
- [HS96] Heidl, P.; Stein, K.: Behandlung semantischer Fehler im Workflow Management, GI-Datenbank-Rundbrief, 17.Mai 1996
- [Hü01] Hübscher, H.; Petersen, H. J.; Rathgeber, C.; Richter, K.; Scharf, D.: IT-Kompodium, Westermann Verlag, 2001
- [HV04] Henning, P. A.; Vogelsang, H.: Taschenbuch Programmiersprachen, Fachbuchverlag Leipzig, 2004
- [IK08] Imboden, D. M.; Koch, S.: Systemanalyse – Einführung in die mathematische Modellierung natürlicher Systeme, 3. korrigierter Nachdruck der 1. Auflage, Springer, 2008
- [Ja92] Jacobson, I. et al.: Object-Oriented Software Engineering. A Use Case Driven Approach. Reading/Mass.: Addison Wesley, 1992
- [JBS97] Jablonski, S.; Böhm, M.; Schulze, W. (Hrsg.): Workflow-Management, dpunkt Verlag, 1997

- [Ki04] Kieß, A.: Tom DeMarco - Strukturierte Analyse und System Spezifikation, Seminararbeit, Universität Leipzig, 2004,
<http://ebus.informatik.uni-leipzig.de/www/media/lehre/seminar-pioniere04/kiess-ausarbeitung-demarco.pdf> [Stand: 21.02.2010]
- [KKB08] Kastens, U.; Kleine Büning, H.: Modellierung – Grundlagen und formale Methoden, 2., überarbeitete und erweiterte Auflage, Hanser verlag München, 2008
- [Kl01] Klingelhöller, H.: Dokumentenmanagementsysteme Handbuch zur Einführung, Springer, 2001
- [KM99] Kampffmeyer, U.; Merkel, B.: Dokumenten–Management Grundlagen und Zukunft, Project Consult, 1999
- [Ko01] Korthaus, A.: Komponentenbasierte Entwicklung computergestützter betrieblicher Informationssysteme, Informationstechnologie und Ökonomie Band 20, Peter Lang GmbH Europäischer Verlag der Wissenschaften, 2001
- [Kö03] König, W.; Rommelfanger, H.; Ohse, D.; Wendt, O.; Hofmann, M.; Schwind, M.; Schäfer, K.; Kuhnle, H.; Pfeifer, A.: Taschenbuch der Wirtschaftsinformatik und Wirtschaftsmathematik, 2., überarbeitete und erweiterte Auflage, Verlag Harri Deutsch, 2003
- [Kr00] Krcmar, H.: Informationsmanagement, Springer, 2000
- [Kr05] Kroenke, D.: Database Processing. Fundamentals, Design, and Implementation 10th ed., Prentice Hall, 2005
- [KS01] Kretz, F. J.; Schäfer, J.: Anästhesie, Intensivmedizin, Notfallmedizin, Schmerztherapie, Springer-Verlag, 2001
- [KT06] Kretz, F. J.; Teufel, F.(Hrsg.): Anästhesie und Intensivmedizin, Springer-Verlag, 2006
- [La87] Langenscheidts Großes Schulwörterbuch English-Deutsch, 18. Auflage, Langenscheidt Berlin München Wien Zürich, 1987
- [Le05] Lehmann, T. M.(Hrsg.): Handbuch der Medizinischen Informatik, 2., vollständig neu bearbeitete Auflage, Hanser Verlag München : Wien, 2005
- [LGH97] Leiner, F.; Gaus, W.; Haux, R.: Medizinische Dokumentation: einführendes Lehrbuch, 2. Aufl. – Stuttgart – Schattenauer, 1997
- [Li01] Limper, W.: Dokumenten-Management, Wissen, Informationen und Medien digital verwalten, dtv, 2001
- [Li98] Lioubinski, M.: Business Objects als Hilfe bei der Unternehmensmodellierung, Diplomarbeit, 1998

- [LM98] Lockemann, P. C.; Mayr, H. C.: Rechnergestützte Informationssysteme, Berlin, Heidelberg, New York : Springer, 1978
- [Lo94] Louden, K. C.: Programmiersprachen, Grundlagen, Konzepte, Entwurf, International Thomson Publishing, 1. Auflage, 1994
- [LR09] Lahres, B.; Rayman, G.: Objektorientierte Programmierung – Das umfassende Handbuch, Galileo Press, 2009
- [MBR04] Michel-Backofen, A.; Röhrig, R.: Erfahrungen mit der Anwendung von HL7 bei Intensiv- und Narkosedokumentationssystemen, Online-Artikel
<http://www.egms.de/static/en/meetings/gmds2004/04gmds041.shtml> [Stand: 21.02.2010]
- [Mö03] Möller, D. P. F.: Rechnerstrukturen-Grundlagen der Technischen Informatik, Springer Verlag, 2003
- [Mü97] Müller, O.: Objektorientiert Entwerfen – Teil 3, Linux Magazin 03/1997;
<http://www.linux-magazin.de/Artikel/ausgabe/1997/03/OOP/oop.html> [Stand: 21.02.2010]
- [NDO05] NarkoData-Online Beschreibung:
<http://doku.imeso.de/NarkoData.htm/index.htm>, 01.04.2005
 [Stand: 21.02.2010]
- [Ot09] Ott, H. J.: Spezifikation von Daten mit Datenmodellen;
<http://sla03.ex.ba-heidenheim.de/script/41dmodspec.htm>
 (ehemals <http://www.kecos.de/script/41dmodspec.htm>), 2009
 [Stand: 14.02.2010]
- [OW02] Ottmann, T.; Widmayer, P.: Algorithmen und Datenstrukturen, 4. Auflage Spektrum Akademischer Verlag, 2002
- [OW99] Opderbecke, H. W.; Weissauer, W.(Hrsg.): Entschließungen, Empfehlungen, Vereinbarungen, Ein Beitrag zur Qualitätssicherung in der Anästhesie, Kerndatensatz Anästhesie – Version 2.0/1999,
http://www.dgai.de/06_1_00tabelle.htm#zu_viii (Stand: 21.02.2010)
- [Pe99] Persson, E.: Shibboleth of Many Meanings – An Essay on the Ontology of Business Objects. In C. Atkinson, N. Baker, S. Uehara und M. Schader (Hrsg.): Proceedings of the third International Enterprise Distributed Object Computing Conference (EDOC'99), 27.-30. Sept.1999, University of Mannheim, Germany. IEEE, Piscataway, NJ, USA. S. 1 – 17.
- [Pf01] Pfeifer, T.: Qualitätsmanagement, Hanser Verlag, 2001
- [PR09] Pohl, K.; Rupp, C.: Basiswissen Requirements Engineering, dpunkt.verlag, 2009

- [Pr05] Preißner, M.: Modellierung und Entwicklung von Pliable Objects zur Unterstützung des Aufbaus von Informationssystemen im medizinischen Anwendungsgebiet der Anästhesie, 17. Workshop Grundlagen von Datenbanken, Tagungsband, 2005
- [Pr11] Preißner, M.: Pliable Objects zur Konzeption medizinischer Anästhesie-Informationssysteme, Workshop Datenmanagement und Interoperabilität im Gesundheitswesen (DIG), Tagungsband, 2011
- [Pr12] Preißner, M.: ER-Modell für Pliable Objects zum Aufbau von medizinischen Anästhesie-Informationssystemen, Workshop Sicherheit, Datenschutz, Management und Interoperabilität medizinischer Daten, Tagungsband, 2012
- [Qi03] Quinzio, A. L. F.: Einführung der computergestützten Anästhesie-Dokumentation am Universitätsklinikum Giessen, Inaugural Dissertation 2003, <http://geb.uni-giessen.de/geb/volltexte/2003/1336/pdf/QuinzioLorenzo-2003-12-09.pdf> [Stand: 21.02.2010]
- [Re91] Rembold, U. [Hrsg.]: Einführung in die Informatik: für Naturwissenschaftler und Ingenieure, 2., bearb. Aufl., München; Wien, Hanser Verlag, 1991
- [Re00] Rechenberg, P.: Was ist Informatik? Eine allgemeinverständliche Einführung, 3. überarbeitete und erweiterte Auflage, München; Wien, Hanser Verlag, 2000
- [RP02] Rechenberg, P.; Pomberger, G.: Informatik Handbuch, 3. aktualisierte und erweiterte Auflage, Hanser Verlag, 2002
- [RT04] Roewer, T.; Thiel, H.: Taschenatlas der Anästhesie, 2. aktualisierte Auflage, Thieme Verlag, 2004
- [Ru91] Rumbaugh, J. et al.: Object-oriented Modelling and Design. Englewood Cliffs: Prentice Hall, N.J., 1991
- [Ru05] Rupp, C.; Hahn, J.; Queins, S.; Jeckle, M.; Zengler, B.: UML 2 glasklar – Praxiswissen für die UML-Modellierung und –Zertifizierung, 2. Auflage, Hanser Verlag, 2005
- [Sa93] Saake, G.: Objektorientierte Spezifikation von Informationssystemen, Stuttgart: Leipzig: Teubner, 1993
- [Sc02] Scheer, A. W.: ARIS – Vom Geschäftsprozess zum Anwendungssystem, Springer Verlag, 2002
- [Se05] Sebesta, R. W.: Concepts of Programming Languages, 7. Auflage, Addison Wesley, 2005
- [SH96] Specht, G.; Hofmann, M.: Auswertung der Migration eines Multimedia-Informationssystems von einem relationalen auf ein objektorientiertes Datenbanksystem, GI, 1996

- [Si70] Sinowjew, A. A.: Komplexe Logik. Grundlagen einer logischen Theorie des Wissens, Vieweg, 1970
- [Si94] Sims, O.: Business Objects, Delivering cooperative objects for client-server, McGraw – Hill, 1994
- [SM09] Stepanic, A.; McJones, P.: Elements of programming, Pearson Education Inc., 2009
- [Sp05] Specker, A.: Modellierung von Informationssystemen, 2., überarb. und erw. Auflage, vdf Hochschulverlag AG an der ETH Zürich, 2005
- [SSH08] Saake, G.; Sattler, K.-U.; Heuer, A.: Datenbanken Konzepte und Sprachen, Dritte, aktualisierte und erweiterte Auflage, mitp, 2008
- [SST97] Saake, G.; Schmitt, I.; Türker, C.: Objektdatenbanken, Bonn; Albany; Attenkirchen: Internat. Thomson Publ., 1997
- [SSU01] Schwabe, G.; Streitz, N.; Unland, R. (Hrsg.): CSCW-Kompodium, Springer, 2001
- [St07] Stock, W. G.: Information Retrieval – Informationen suchen und finden, R. Oldenbourg Verlag München Wien, 2007
- [St73] Stachowiak, H.: Allgemeine Modelltheorie, Springer, Wien, 1973
- [St97] Stapleton, J.: DSDM Dynamic Systems Development Method – The Method in Practise, Addison-Wesley, 1997
- [Su97] Sutherland, J.; Patel, D.; Casanave, C.; Hallowell G.; Miller, J. (eds): Business Object Design and Implementation, OOPSLA'95 Workshop Proceedings, Springer Verlag, 1997
- [SW07] Schneider, U.; Werner, D.: Taschenbuch der Informatik, Hanser Verlag München, 2007
- [Ta06] Tabelaing, P.: Softwaresysteme und ihre Modellierung, Springer Verlag Berlin: Heidelberg, 2006
- [Te97a] Teich, J.: Digitale Hardware/Software-Systeme – Synthese und Optimierung, Berlin; Heidelberg; New York; Barcelona; Budapest; Hongkong; London; Mailand; Paris; Santa Clara; Singapur; Tokio, Springer, 1997
- [Te97b] Teille, K.: BRENDA Modellierung und Implementierung eines objektorientierten Informationssystems über Enzyme mit besonderer Berücksichtigung des Metabolismus-Engineering, Shaker Verlag, 1997
- [USZ03] UniversitätsSpital Zürich, Institut für Anästhesiologie: Anästhesie Allgemein <http://www.anaesthesie.usz.ch/PatientenUndBesucher/AnaesthesieAllgemein/Seiten/default.aspx> [Stand: 21.02.2010]

- [Ve91] Vetter, M.: Aufbau betrieblicher Informationssysteme mittels objektorientierter, konzeptioneller Datenmodellierung, 7., neubearb. und erw. Aufl. – Stuttgart: Teubner, 1991
- [Vo00] Vossen, G.: Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme, Oldenbourg, 2000
- [Wh02] Whitehead, K.: Component-based Development. Principles and Planning for Business Systems, Addison-Wesley Longman, Amsterdam, 2002
- [ZGK04] Zuser, W.; Grechening, T.; Köhle, M.: Software Engineering mit UML und dem Unified Process, 2. überarbeitete Auflage, Pearson Studium, 2004
- [Zö09] Zöller-Greer, P.: Software-Architekturen – Grundlagen und Anwendungen, 2. Auflage, Composita Verlag, 2009
- [ZW06] Zeppenfeld, K.; Wolters, R.: Generative Software-Entwicklung mit der MDA, Spektrum Akademischer Verlag, 2006
- [Zw99] Zwally, B.: Pflegestandards für die Anästhesieabteilung im OP 04, 1998/99, <http://www.klinikum.uni-muenchen.de/Campus-fuer-Alten-und-Krankenpflege/download/inhalt/Anaesthesie/Kinderanaesthesie/Kinderanaesthesie.pdf> [Stand: 21.02.2010]