ER-Modell für Pliable Objects zum Aufbau von medizinischen Anästhesie-Informationssystemen

Markus Preißner

Arbeitsgruppe Datenbanken
Otto-von-Guericke Universität Magdeburg

Austr.12
73230 Kirchheim / Teck
Markus.Preissner@gmx.de

Abstract: Moderne Informationssysteme sind für die Verarbeitung grosser Datenmengen ausgelegt. Im medizinischen Fachbereich der Anästhesie scheitern sie jedoch, wenn neue Informationen zusätzlich zu verarbeiten oder sie an geänderte Arbeitsabläufe anzupassen sind. Die zu erfassenden Informationen orientieren sich streng an der Erkrankung und dem Grad der Erkrankung des Patienten. Aufgabe eines Anästhesie-Informationssystems ist die Verarbeitung der Gesamtheit aller Informationen, die generell in der Anästhesie zu dokumentieren sind. Dazu zählt auch die Anforderung, neue Informationen ohne Softwareaktualisierung verarbeiten zu können. Im vorliegenden Beitrag wird ein Entity-Relationship-Modell für Pliable[1] Objects vorgestellt. Pliable Objects bieten einen generischen Ansatz zur Lösung dieser Probleme an.

1 Anforderungsdilemma bzgl. Anästhesie-Informationssystemen

Vielfältige Aufgaben werden im medizinischen Fachbereich der Anästhesie während der Patientenversorgung erledigt. Alle Informationen dieser Versorgung sind zu dokumentieren, um die Qualität der Anästhesie zu sichern und um eventuell durch eine Informationsauswertung neue Erkenntnisse zu erlangen. Zur Reduzierung des Aufwandes bei der komplexen Dokumentation sollen Informationssysteme eingesetzt werden. Anästhesie-Informations-Management-Systeme (AIMS) verfolgen dabei das Ziel, die zur Zeit verwendeten papierbasierten Anästhesieprotokolle in ein einheitliches, lesbares und vollständig elektronisches Dokument zu überführen. Im Anästhesieumfeld führen permanente Anpassungen (vgl. [B07], [Pr05], [Pr11]) zu geänderten Protokollen. Aus Sicht der Informatik ist ein Informationssystem zu konzipieren und zu entwickeln, das ebenfalls anpassbar ist.

[1] Pliable: engl. biegsam, nachgiebig, formbar

Informationssysteme werden in der Regel gemäß einem Vorgehensmodell entwickelt. Dieses ist keine einheitliche Definition, sondern dient der Entwicklung als Richtlinie. In der Informatik wird häufig das Wasserfallmodell verwendet, welches relativ linear die Softwareentwicklung mittels einer Kaskade der Phasen Analyse, Design, Entwicklung, Test und Nutzung beschreibt. Während der Phasen Analyse und Design werden die Anforderungen an das Informationssystem gesammelt und in ein Modell überführt, aus welchem der Informatiker seine Entwicklungsaufgaben ableitet und gegen welches er letztlich seine Entwicklung testet (vgl. [V00], [H92]).

Bei der Erhebung der Anforderungen werden jedoch meist unbewusst Fehler gemacht. Ein Anästhesist formuliert die Anforderungen an das zukünftige System mit einem Kontextwissen, welches der Informatiker nicht besitzt. Den Anforderungen mangelt es dadurch an Exaktheit. Der Informatiker erarbeitet auf Grundlage der Anforderungen ein Modell. Dieses Modell ist ganz allgemein ein Muster zu Veranschaulichungszwecken. Der Informatiker verwendet dazu häufig eine Modellierungssprache wie die Strukturierte Analyse (SA), das Entity-Relationship-Modell (ERM) oder die Unified Modelling Language (UML) (vgl. [CYA94], [CYD94], [S94]). Vor Start der Entwicklung wird das erarbeitete Modell mit den Anästhesisten besprochen, um sicherzustellen, dass die Anforderungen korrekt verstanden wurden. Leider sind den Anästhesisten die Prinzipien von SA, ERM, UML, etc. nicht bekannt, so dass die Verifikation des Modells durch den Anästhesisten nicht möglich ist.

Es wird daher die Entwicklung eines Informationssystems gefordert, welches mit überschaubarem Aufwand an die jeweiligen anästhesiespezifischen Bedürfnisse angepasst werden kann. Ein Informationssystem muss dazu mit einer Generik versehen werden, die es ermöglicht, die gewünschten, bislang nicht absehbaren Anpassungen des Informationssystems vornehmen zu können. Einen Ansatz bildet das Modell der Pliable Objects. Sie stellen ein neues Konzept zur Abbildung realer Entitäten dar und beinhalten ferner einen generischen Ansatz, um bestehenden Objekten eine neue Form zu geben bzw. sie derart zu „verbiegen“, dass sie neuen Ansprüchen gerecht werden.

2 Entity-Relationship-Modell für Pliable Objects

Nach [Pr11] ist ein *Pliable Object* eine Beschreibungsdomäne, d.h. ein Fachausdruck zur Benennung und eine (umgangssprachliche) Erklärung der Bedeutung und des Zweckes, die dem Pliable Object durch die Experten zugewiesen werden. Das *Pliable Object* wird um eine Menge von *Attributes*, um eine Menge von *Actions* und um eine Menge von *Rules* erweitert. Ein *Attribute* charakterisiert eine Eigenschaft, die für das Pliable Object relevant ist und als Angabe einer Zustandsinformation dient; eine *Action* definiert einen bestimmten Zustandswechsel, der ein spezielles Verhalten eines Pliable Objects beschreibt, und eine *Rule* beschreibt eine Verhältnismässigkeit, die das Verhalten, die Beziehungen und die Attributes eines Pliable Objects verwaltet. Dies schliesst die Erkennung von Ereignissen in seiner Umgebung, die Änderung seiner Attributes und die Interaktion mit anderen Pliable Objects mit ein.

2.1 Semantisches Modell

Mit dem Modell der Pliable Objects wird nicht das Ziel verfolgt, ein neues Modell für die Softwareentwicklung zu spezifizieren, sondern vielmehr soll es ein Konzept realisieren, welches mittels des objektorientierten Modells ausgedrückt werden kann. Dieses Konzept soll zur Definition einer Architektur verwendet werden, welche als Grundlage einer Informationssystemspezifikation dienen kann.

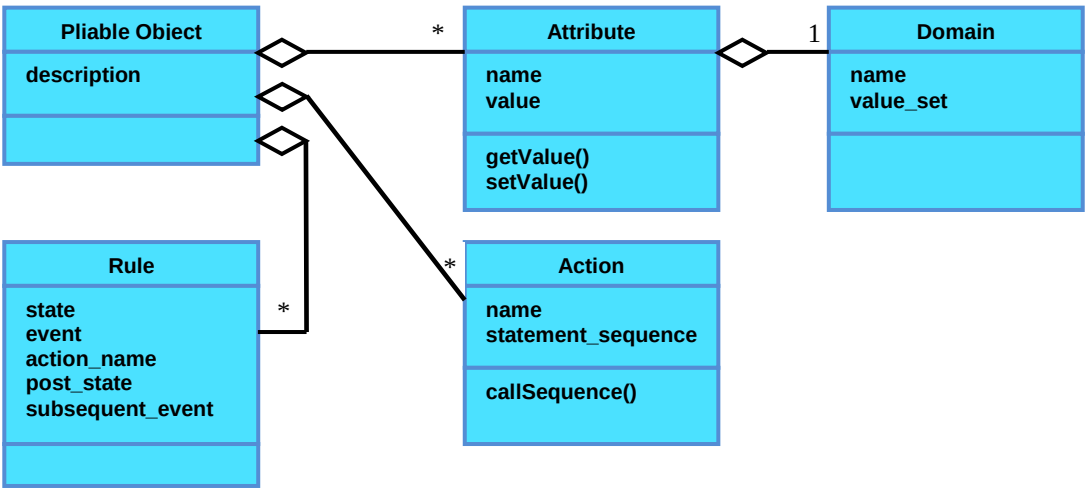

Abbildung 1: Semantisches Pliable-Object-Modell nach [Pr11]

Abbildung 1 zeigt das Semantische Modell von Pliable Objects. 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, welche Bearbeitungs- oder Verfahrensanweisungen darstellen, wenn der Zustand aktiv ist und das Ereignis, auf das die Regel reagieren soll, eintrifft.

2.2 ER-Modell für Pliable Objects

Aus dem semantischen Modell der Pliable Objects kann ein Entity-Relationship-Modell (ER-Modell) abgeleitet werden, auf dessen Grundlage ein Datenbankschema eines Informationssystems erarbeitet werden kann. Abbildung 2 zeigt das ER-Modell für Pliable Objects. Das zentrale Entity *Pliable Object* wird durch eine eindeutige Pliable-Object-ID und eine Bezeichnung beschrieben.

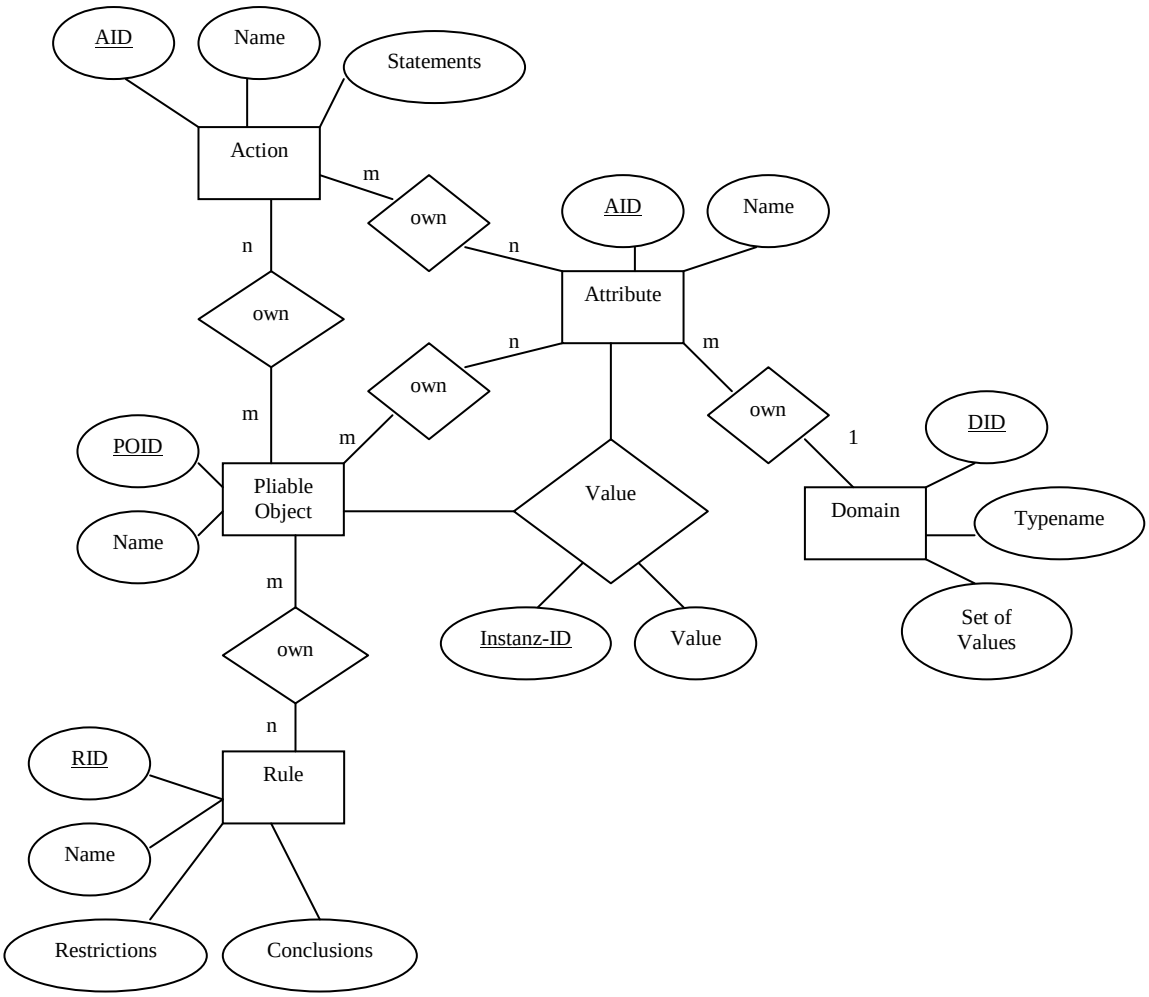

Abbildung 2 ER-Modell für Pliable Objects

Die Eigenschaften und die Verhaltensbeschreibungen eines Pliable Objects sind über 3 *besitzt(own)*-Beziehungen dem Objekt zugeordnet. Einzelne Eigenschaften werden als Attribut-Entitäten und bestimmte Verhaltensbeschreibungen als Action- bzw. Rule-Entitäten abgebildet.

Ein Attribut ist eine Zustandsinformation, die ebenfalls durch eine eindeutige Attribut-ID und eine Bezeichnung beschrieben ist. Die Art der Informationen, die in dem Attribut hinterlegt werden kann, ist durch eine Domäne gegeben.

Jedes Attribut besitzt genau eine Domäne. Die Domäne selbst wird durch eine eindeutige Domänen-ID, eine (Typ-)Bezeichnung sowie durch eine Wertemenge beschrieben.

Für die Wertemenge wird vereinbart, dass sie durch eine Aufzählung der in ihr enthaltenen Werte angegeben wird. Falls dies zu umfangreich werden sollte, kann die Wertangabe - sofern möglich - alternativ durch einen regulären Ausdruck beschrieben werden. Es sei speziell darauf hingewiesen, dass es sich bei Domänen um willkürliche Zusammenstellungen von Werten handelt. Der Umstand, dass eine Wertemenge mit Hilfe der Ausdrucksmöglichkeiten eines regulären Ausdrucks beschrieben werden kann, ist keine Verpflichtung dafür, dass ein regulärer Ausdruck zwingend erforderlich ist.

Der Zustand eines Attributs eines Pliable Objects kann über eine Action beeinflusst werden. Eine Action wird durch eine eindeutige Action-ID, eine Bezeichnung sowie durch eine Liste von Anweisungen (statements) ausgedrückt. Für Actions wird angenommen, dass sie ausschliesslich auf den Attributen des Pliable Objects arbeiten, dem sie selbst zugeordnet sind.

Die möglichen Zustände, die ein Pliable Object annehmen kann, werden mittels Rule-Entitäten verwaltet. Eine Rule selbst ist durch eine eindeutige Rule-ID, eine Bezeichnung, eine Menge von Restriktionen und Konklusionen wiedergegeben. Eine Restriktion beinhaltet einerseits den Zustand, in dem die Rule zu beachten und andererseits, bei welchem Ereignis sie explizit zu prüfen ist. In der Konklusion kann eine Action referenziert werden, die aufzurufen ist, wenn die zugehörige Restriktion erfüllt ist. Ferner kann mit der Konklusion ein Zustandswechsel vorgenommen werden und ein Ereignis zur weiteren Verarbeitung gemeldet werden.

Die spezifischen Werte einer Instanz eines Pliable Objects werden in einer Value-Beziehung hinterlegt. Über die Beziehung wird ein eingegebener oder ermittelter Wert einer eindeutigen Instanz zugeordnet. Die Zuordnung wird ergänzt durch die Angabe eines konkreten Attributs und des speziellen Pliable Objects, dem die Instanz angehört. Für den eigentlichen Wert wird vereinbart, dass er als Zeichenkette (String) abgelegt wird.

2.3 Relationales Beispiel

Im Folgenden soll kurz skizziert werden, wie eine relationale Umsetzung am Beispiel eines rudimentären präoperativen Anästhesieprotokolls aussehen könnte.

Für dieses rudimentäre Anästhesieprotokoll wird angenommen, dass es aus 3 Attributen besteht: einer Patientenummer, einer Dringlichkeit und einer Aufnahmeart. Das Pliable Object *Präoperatives Protokoll* zeigt Tabelle 1.

Für die Erfassung und Speicherung seien zwei Actions definiert, die dem Pliable Object *Präoperatives Protokoll* zugeordnet sind, wie es Tabelle 2 zeigt.

Pliable Object		
POID	**Pliable-Name**	**Pliable-Definition**
111	Präoperatives Protokoll	Anästhesieteilprotokoll zur Erfassung des Patientenzustandes

Own	
POID	**AttID**
111	221
111	222
111	223

Attribut	
AttID	**Name**
221	Patientennummer
222	Dringlichkeit
223	Aufnahmeart

Own	
AttID	**DID**
221	331
222	332
223	333

Domänen		
DID	**Typbezeichnung**	**Wertemenge**
331	Zahl	[0-9]*
332	Dringlichkeiten	elektiv, dringlich, Notfall
333	Aufnahmearten	ambulant, stationaer, teilstationaer

Tabelle 1 Beispiel eines Pliable Object *Präoperatives Protokoll*

Pliable Object		
POID	**Pliable-Name**	**Pliable-Definition**
111	Präoperatives Protokoll	Anästhesieteilprotokoll zur Erfassung des Patientenzustandes

Own	
POID	**ActID**
111	441
111	442

Action		
ActID	**Name**	**statement sequence**

441	Protokoll erfassen	select_protocol_attributes();
		generate_GUI();
		show_GUI();

442	Protokolldaten speichern	get_user_inputs();
		create_instance();
		store_data();

Tabelle 2 Actionbeispiel des Pliable Objects *Präoperatives Protokoll*

Mit dem Start der Action *Protokoll erfassen* werden alle Attribute des Protokolls bestimmt. Auf Basis der Menge der Attribute wird eine Maske der Bedienoberfläche generiert, die die Anwendereingaben entgegennimmt. Eine mögliche Maske zeigt Abbildung 3. Nachdem die Daten eingegeben sind, werden diese gespeichert. Die Speicherung übernimmt die Action *Protokolldaten speichern*.

Abbildung 3 Beispiel einer generierten Bedienmaske

Die Actions zur Datenerfassung und –speicherung sind beim Pliable Object hinterlegt. Der Aufruf der Actions wird über die Rules des *Präoperativen Protokolls* gesteuert. Bezogen auf das Beispiel ist eine minimale Realisierung in Tabelle 3 gegeben. Mit der Vereinbarung, dass eine Applikation existiert, die auf dem Pliable-Object-Modell arbeitet, wird für die Ausgangssituation angenommen, dass über die Applikation das neue Anlegen eines präoperativen Protokolls gestartet wird. Die Applikation sendet daraufhin das Ereignis *E_new_instance* an das Pliable Object, woraufhin gemäss der Rules die Action *Protokoll erfassen* aufgerufen wird. Während der Interaktion mit dem Bediener ist das Object im Zustand *S_new*. Nach der Eingabe der Daten wird der Bediener die Erfassung entweder mit *OK* oder mit *Cancel* beenden. Die Applikation wird daraufhin die entsprechenden Events an das Object senden.

Pliable Object		
POID	**Pliable-Name**	**Pliable-Definition**
111	Präoperatives Protokoll	Anästhesieteilprotokoll zur Erfassung des Patientenzustandes

Own	
POID	**RID**
111	551

Rule					
RID	**State**	**Event**	**Action**	**Post state**	**Subsequent Event**
551	S_IDLE	E_new_instance	Protokoll erfassen	S_new	-
551	S_new	E_cancel	Clear Screen	S_IDLE	-
551	S_new	E_OK	Protokolldaten speichern	S_IDLE	-

Tabelle 3 Minimales Beispiel für Rules

Unabhängig vom gesendeten Event wechselt das Pliable Object in den Ausgangszustand, wobei die eingegebenen Daten im Falle des Ereignisses *E_OK* gespeichert werden. Für den Fall, dass die Eingabe abgebrochen wird, soll eine von der Applikation bereitgestellte Routine aufgerufen werden, die eventuell eingegebene Daten verwirft und sonstige Aufräumarbeiten im Speicher vornimmt. Diese Routine wird nicht weiter betrachtet.

Im Fall der Eingabe von Daten wird eine neue Instanz eines Präoperativen Protokolls angelegt, wie es Tabelle 4 zeigt.

Value			
InstanzID	**POID**	**AttID**	**Value**
1	111	221	4711
1	111	222	elektiv
1	111	223	stationaer

Tabelle 4 Beispiel einer neuen Instanz

Hinsichtlich der Rules in Tabelle 3 sei angemerkt, dass sich eine Restriktion aus dem Tupel (RID, State, Event) ergibt, während die Konklusion durch das Tupel (Action, Post State, Subsequent Event) realisiert wird.

2.4 Schematische Erweiterung und Verhaltensanpassung

Die Aufbereitung der Informationen eines Pliable Objects in Attributes, Actions und Rules erlaubt die nachträgliche Erweiterung der Objekte sowie auch eine Verhaltensanpassung. Hierzu werden die normalen Möglichkeiten eines Datenbankmanagementsystems verwendet; beispielsweise soll die Zustandsbeurteilung eines Patienten im Präoperativen Protokoll um die Eigenschaft *Bewusstsein* mit der Auswahl *normal, somnolent, komatös, sediert* erweitert werden. Mit bekannten Standardmitteln, wie z.B. entsprechenden SQL-Statements, kann das Pliable Object erweitert werden, wie es Tabelle 5 zeigt.

Pliable Object		
POID	**Pliable-Name**	**Pliable-Definition**
111	Präoperatives Protokoll	Anästhesieteilprotokoll zur Erfassung des Patientenzustandes

Own	
POID	**AttID**
111	221
111	222
111	223
111	224

Attribut	
AttID	**Name**
221	Patientennummer
222	Dringlichkeit
223	Aufnahmeart
224	Patientenbewusstsein

Own	
AttID	**DID**
221	331
222	332
223	333
224	334

Domänen		
DID	**Typbezeichnung**	**Wertemenge**
331	Zahl	[0-9]*
332	Dringlichkeiten	elektiv, dringlich, Notfall
333	Aufnahmearten	ambulant, stationaer, teilstationaer
334	Bewusstsein	normal, somnolent, komatoes, sediert

Tabelle 5 Erweitertes Pliable Object

Es wird ein neues Attribut mitsamt einer neuen Domäne angelegt. Dieses neue Attribut wird dem Pliable Object zugeordnet. Aufgrund der generischen Actions ist diese Erweiterung bereits ausreichend um die Ergänzung der neuen Information vorzunehmen.

Ebenso einfach kann das Verhalten des Pliable Objects ergänzt werden. So soll z.B. eine Action die Werteingabe auf Korrektheit prüfen. Diese könnte gemäss Tabelle 6 ergänzt werden.

Diese neue Action soll die spezielle Bewusstseinseingabe dahingehend überprüfen, ob der eingegebene Wert in der Domäne enthalten ist. Ist dies nicht der Fall, so wird ein Fehlerereignis gesendet.

Pliable Object		
POID	**Pliable-Name**	**Pliable-Definition**
111	Präoperatives Protokoll	Anästhesieteilprotokoll zur Erfassung des Patientenzustandes

Own	
POID	**ActID**
111	441
111	442
111	443

Action		
ActID	**Name**	**statement sequence**

441	Protokoll erfassen	select_protocol_attributes();
		generate_GUI();
		show_GUI();

442	Protokolldaten speichern	get_user_inputs();
		create_instance();
		store_data();

443	Bewusstsein prüfen	get_bewusstein_input();
		if input not in domain bewusstein
		send_event(E_error);

Tabelle 6 Ergänzung einer Action

Der Aufruf der zusätzlichen Action kann z.B. über die Rules in das System eingebracht werden. Alternativ wäre auch eine Anpassung einer im Ablauf befindlichen Action denkbar. Tabelle 7 zeigt Anpassungen mittels der Aktualisierung der Rules-Beziehung.

Pliable Object		
POID	**Pliable-Name**	**Pliable-Definition**
111	Präoperatives Protokoll	Anästhesieteilprotokoll zur Erfassung des Patientenzustandes

Own	
POID	**RID**
111	551

Rule					
RID	**State**	**Event**	**Action**	**Post state**	**Subsequent Event**
551	S_IDLE	E_new_instance	Protokoll erfassen	S_new	-
551	S_new	E_cancel	Clear Screen	S_IDLE	-
551	S_new	E_OK	Bewusstsein prüfen	S_check	E_OK
551	S_check	E_OK	Protokolldaten speichern	S_IDLE	-
551	S_check	E_error	Show Error	S_IDLE	-

Tabelle 7 Erweiterung der Rules

Der Ablauf nach der Dateneingabe wird dahingehend geändert, dass dann zunächst die neue Action aufgerufen wird. Das Objekt wechselt in den neuen Zustand *S_check*. Per Default wird vorgesehen, dass nach der Prüfung das Folgeereignis *E_OK* verarbeitet wird. Wenn die Prüfung korrekt verläuft, wird der bereits bekannte Ablauf fortgesetzt. Falls die Prüfung allerdings fehlschlägt, wird das Default-Folgeereignis von der Action mit *E_error* überschrieben. In den Rules wird für diesen Fall eine Fehlerverarbeitung vorgesehen, in deren Rahmen das Objekt in den Ausgangszustand zurückversetzt wird.

Angenommen, nach der Erweiterung wird ein neues Präoperatives Protokoll fehlerfrei erfasst, so würde die neue Instanz in einer Form gespeichert werden, wie sie Tabelle 8 zeigt.

Value			
InstanzID	**POID**	**AttID**	**Value**
1	111	221	4711
1	111	222	elektiv
1	111	223	stationaer
2	111	221	4712
2	111	222	dringlich
2	111	223	ambulant
2	111	224	normal

Tabelle 8 Erweiterte Instanz-Relation

Bei der Speicherung der neuen Instanz ist festzuhalten, dass diese neue Instanz die neue Eigenschaft mit erfasst. Der bislang erfasste Datensatz bleibt von der Ergänzung unberührt. Dies ist ein Vorteil der Pliable Objects, da u.a. auch die Anforderung im Raum steht, dass nach einer Aktualisierung des Informationssystems Altinformationen unberührt von der Aktualisierung vorliegen sollen.

2.5 Implementierungskonzept

Pliable Objects beinhalten eine spezielle Form der Selbstbeschreibung, die dahingehend genutzt werden kann, eine allgemeingültige Applikation für die Bearbeitung von Pliable Objekts konzipieren zu können.

Das semantische Modell in Abbildung 1 beinhaltet die Objekte *Pliable Object, Attribute, Domain, Action* und *Rule*. Diese Objekte verfügen nach objektorientiertem Verständnis über die Attribute *description, name, value, value_set, statement_sequence, state, event, action_name, post_state* und *subsequent_event* sowie über die Methoden *getValue, setValue* und *callSequence*. Über die Interpretation der 5 Objekte als Pliable Objects folgt, dass die Objekte über das ER-Modell zum Ausdruck gebracht werden können; d.h. die 5 Pliable Objects beschreiben einerseits das ER-Modell, andererseits bilden sie Entities im ER-Modell. Hierzu werden die 5 Objekte mitsamt der 9 Attribute und 3 Methoden (als Aktionen) analog zum Beispiel unter 2.3 und 2.4 angelegt. Tabelle 9 zeigt exemplarisch die Ausprägung der Selbstbeschreibung auf Basis des semantischen Modells. Auf die Darstellung der jeweiligen Own-Relationen ist in Tabelle 9 aus Übersichtsgründen verzichtet worden.

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 4). 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. Die Gestaltung und die Ausprägung verschiedener weiterer Pliable Objects sind dabei anwendungsbezogen und müssen gemeinsam mit den jeweiligen Anwendern fachlich spezifiziert werden.

Pliable Object		
POID	**Pliable-Name**	**Pliable-Definition**
1	Pliable Object	Basisobjekt zur Definition und Bearbeitung von zweckgebundenen Objekten
2	Attribute	Basisobjekt zur Definition und Bearbeitung von Zustandsinformationen
3	Domain	Basisobjekt zur Definition und Bearbeitung von Wertebereichen
4	Action	Basisobjekt zur Definition und Bearbeitung von Zustandswechseln
5	Rule	Basisobjekt zur Definition und Bearbeitung von Verhaltenskontrollen

Attribut	
AttID	**Name**
1	description
2	name
3	value
4	value_set
5	statement_sequence
6	state
7	event
8	action_name
9	post_state

Action		
ActID	**Name**	**statement sequence**
1	getValue	
2	setValue	
3	callSequence	

Tabelle 9 Ausprägung der Selbstbeschreibung von Pliable Objects

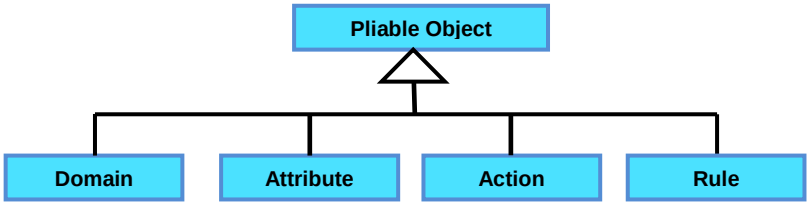

Abbildung 4 Spezielle Klassenhierarchie von Pliable Objects

Unabhängig von einer fachbezogenen Anwendungsspezifikation kann eine allgemeingültige Basis einer Software-Architektur konzipiert werden, die bereits eine unabhängige Minimal-Applikation beschreibt. Abbildung 5 zeigt die elementaren Komponenten, die im Rahmen einer solchen Applikation zu entwickeln wären, um Pliable Objects zur Datenverarbeitung konzipieren und gleichzeitig verwenden zu können.

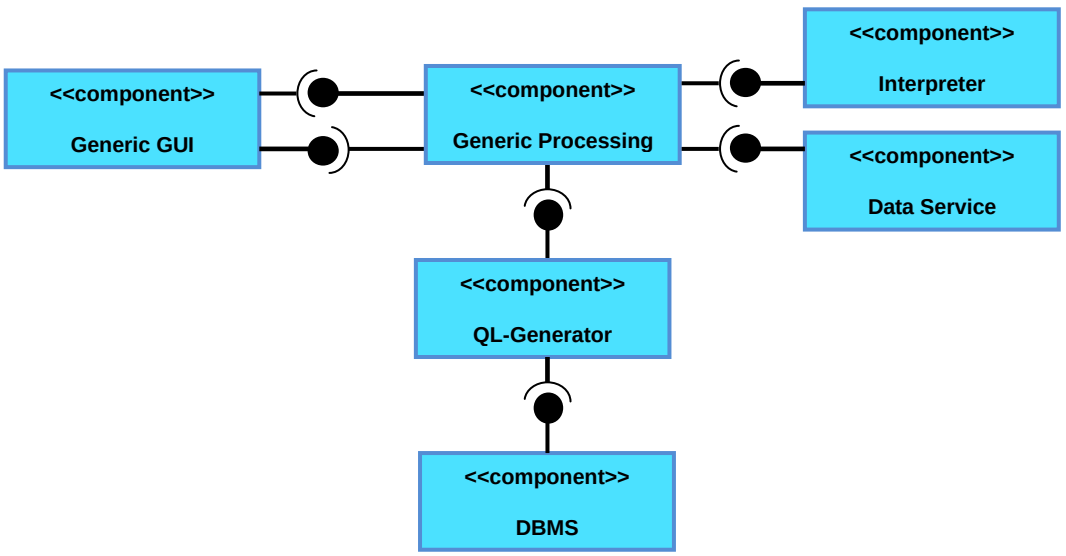

Abbildung 5 Elementare Software-Architektur zur Verarbeitung von Pliable Objects

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.

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 9), 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.

3 Zusammenfassung

Die Speicherung von Tabellen und ihren Inhalten ist eine Kernaufgabe von relationalen Datenbanken. Das gezeigte ER-Modell für Pliable Objects bereitet dieses auf eine Abbildung in ein Datenbankschema vor. Die Abbildung unterstützt dabei die generische Aufarbeitung von Informationen. Mit den Zugriffsmitteln eines Datenbankmanagementsystems ist es erlaubt, dass die Definition von Pliable Objects selbst hinsichtlich ihrer Attributes, ihrer Actions und ihrer Rules verändert werden kann. So können neue Eigenschaften und Verhaltensweisen ergänzt oder bestehende geändert werden. Bereits angelegte Instanzen der Pliable Objects bleiben davon unberührt.

Die Anpassungen, die durch das Datenbankmanagementsystem möglich werden, sind zur Laufzeit des Systems möglich; d.h. das Erweitern neuer Informationen, von denen Anwender sich wünschen, diese zusätzlich ins Informationssystem zu bringen, wird unterstützt. Ebenso wird berücksichtigt, dass eventuelle Fehlinterpretationen von Anforderungen an ein Informationssystem durch Pliable Objects zügig ausgeräumt werden können.

Anästhesie-Informationssysteme, die das Konzept der Pliable Objects integrieren und die Informationen mittels Pliable Objects erfassen und verarbeiten, würden die Optionen anbieten können, die Erfassung und die Verarbeitung an geänderte Umstände anpassen zu können, ohne dass eine Softwareaktualisierung vorzunehmen wäre.

Eine Realisierung von Pliable Objects in Informationssystemen steht noch aus.

Literaturverzeichnis

[BS02] Broy, M., Siedersleben, J.: Objektorientierte Programmierung und Softwareentwicklung – Eine kritische Einschätzung, Informatik Spektrum, 25.2.2002

[B97] Bruder, M., Rehfeld, W., Seeger, T., Strauch, D. (Hrsg.): Grundlagen der praktischen Information und Dokumentation, 4. völlig neu gefasste Ausgabe, Band 1 und 2, K.G.Saur Verlag, 1997

[B07] Branitzki et al.;Spezielle Empfehlungen und Anforderungen zur Implementierung eines Anästhesie-Informations-Management-Systems; Anästh. Intensivmed., 2007;48:282-290

[C96] Casanave, Cory (ed.): OMG Common Facilities RFP-4 Common Business Objects and Business Object Facility, OMG TC Document CF/96-01-04, www.omg.org

[CYA94] Coad, P., Yourdon, E.: OOA – Objektorientierte Analyse, Prentice Hall Verlag, 1994

[CYD94] Coad, P., Yourdon, E.: OOD – Objektorientiertes Design, Prentice Hall Verlag, 1994

[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

[GHJV95] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Entwurfsmuster, Addison-Wesley, 1995

[H92] Heuer, A.: Objektorientierte Datenbanken, Konzepte, Modelle, Systeme, Addison-Wesley, 1992

[KS01] Kretz, F.J., Schäfer, J.: Anästhesie, Intensivmedizin, Notfallmedizin, Schmerztherapie, Springer-Verlag, 2001

[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

[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 zur 41. Jahrestagung der GI, 2011

[RT04] Roewer, T., Thiel, H.: Taschenatlas der Anästhesie, 2. aktualisierte Auflage, Thieme Verlag, 2004

[S94] Sims, O.: Business Objects, Delivering cooperative objects for client-server, McGraw – Hill, 1994

[SPCH97] Sutherland, J., Patel, D., Casanave, C., Hallowell G., Miller, J.(eds): Business Object Design and Implementation, OOPSLA’95 Workshop Proceedings, Springer Verlag, 1997

[V00] Vossen, G.: Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme, Oldenbourg, 2000