Modellierung temporaler Aspekte betrieblicher Informationssysteme mit UML

Temporale Aspekte und das objektrelationale Modell

Univ. Doz. Dr. Alexander Kaiser

ao. Univ. Prof an der Abteilung für Informationswirtschaft Wirtschaftsuniversität Wien
Augasse 2-6
A-1090 Wien, Österreich
alexander.kaiser@wu-wien.ac.at

Mag. Rinaldo Walter Wurglitsch

Principal Consultant
Oracle GmbH Österreich
Brigittenauer-Lände 50-54
A-1203 Wien, Österreich
rinaldo.wurglitsch@oracle.com

Abstract: Der vorliegende Beitrag beschreibt die Einsatzmöglichkeiten der UML (Unified Modeling Language) für die Modellierung temporaler Aspekte in betrieblichen Informationssystemen unter der Berücksichtigung des Modells RETTE. Dabei wird auf eine durchgängige systematische Anwendung der UML für die Entwurfsschritte der Analyse bis zur Implementierung Wert gelegt. Bei der Umsetzung des zeitbezogenen konzeptionellen Modells in ein objektrelationales Modell wird der Versuch unternommen, unter Einbeziehung kommerziell verfügbarer Möglichkeiten eine praxisrelevante Alternative zur Verfügung zu stellen.

Diese Arbeit stellt ein Modell vor, das alle Ebenen der temporalen Informationssystementwicklung im Bereich des objektrelationalen Modells abdeckt und das mit kommerziell verfügbaren Werkzeugen umsetzbar ist. Dabei wird ein grober Leitfaden zur Umsetzung des temporalen Aspekts in objektrelationalen Systemumgebungen, im speziellen in einer Oracle Umgebung gegeben.

Die Zeit kommt in die Welt durch das kollektive Verhalten vieler Teilchen. - Sie bildet eine reale Illusion.[1]

1 Einführung und Problemstellung

Informationssysteme stellen ein Abbild einer modellierten Realität dar und bilden so Zusammenhänge und Prozesse aus der Realität auf die Ebene von Rechnern ab. Unbestrittener Weise hat die Zeitdimension in unserer Welt eine entscheidende Funktion. Wenn in der modellierten Realität die Zeit eine wichtige Rolle spielt, muss diese auch in Informationssystemen angemessen berücksichtigt werden. Wird die Zeit überhaupt nicht berücksichtigt, so fehlt der Abbildung der modellierten Realität eine ganz wesentliche Dimension und das Ziel einer isomorphen Abbildung bzw. eines möglichst geringen Informationsverlustes kann nicht erreicht werden. Aus diesem Grund ist es wichtig, die Zeit in Informationssystemen im allgemeinen und in Datenbanksystemen im besonderen adäquat einzubinden. Viele meinen, dass eigentlich kaum ein Anwendungsgebiet für Informationssysteme existiert, in dem zeitbezogene Daten zu vernachlässigen wären. Diese Einsicht hat dazu geführt, dass temporale Datenbanken, das sind Datenbanken, welche die Zeitdimension explizit berücksichtigen, zu einem wichtigen Forschungsbereich der Wirtschaftsinformatik geworden sind.

Für die adäquate Berücksichtigung der Zeitdimension bei der Entwicklung relationaler Informationssysteme wurden in den letzten Jahren einige Vorschläge gemacht (vgl. dazu etwa das Modell TimeER ([GJ98] und [GMJ98]), das Modell RETTE [Kai00], das Modell Chrono [BS98], für einen Vergleich von temporal erweiterten Modellen auf der konzeptionellen Ebene vgl. [GJM97]). Im Bereich der Entwicklung objektrelationaler Informationssysteme unter Berücksichtigung der Zeitdimension existieren bis jetzt kaum durchgängige Vorschläge.

Das Modell RETTE bietet Unterstützung für zeitbezogene Informationen von der Analyse bis hin zur Implementierung. Erfährt der temporale Aspekt bei der Informationssystemgestaltung so früh wie möglich Berücksichtigung, so kann in späteren Entwicklungsphasen entsprechend dem Funktionsumfang des DBMS (dieser wird von Version zu Version der kommerziellen und frei verfügbaren DBMS umfangreicher) darauf eingegangen werden. Das Modell RETTE zeigt sich flexibel genug, um modular auf solche Änderungen eingehen zu können, ohne dabei das Gesamtkonzept zu ändern.

[1] siehe [Gen96] Seite 41

In der vorliegenden Arbeit werden zwei Fragestellungen untersucht:

1.) Auf welche Art und Weise kann die Zeitdimension bei der Entwicklung von objektrelationalen Informationssystemen mittels UML berücksichtigt werden?

2.) Eignet sich das Modell RETTE, das für die Behandlung zeitbezogener Aspekte für relationale Systeme entworfen wurde, auch für das objektorientierte bzw. objektrelationale Modell?

2 Das Modell RETTE

Das Modell RETTE wurde im Rahmen einer Habilitation an der Wirtschaftsuniversität Wien entwickelt und ausführlich in [Kai00] und überblicksmäßig in [Kai99] beschrieben, so dass in der vorliegenden Arbeit lediglich auf die wesentlichen Eckpunkte des Modells eingegangen werden kann. In [KW00] wurden eine konkrete Anwendung und Umsetzung des RETTE-Modells vorgestellt.

Kernstück des RETTE-Modells sind die fünf unterschiedlichen Attributarten. Die Zuordnung dieser Attributarten zu Entitätstypen und Beziehungstypen erlaubt eine differenzierte Abbildung von zeitbezogenen Entitäts- und Beziehungstypen, gleichzeitig können jedoch nicht-zeitbezogene Sachverhalte mit den konventionellen Elementen des ER-Modells so wie bisher dargestellt werden.

Zeitstempelattribute: Die beiden Zeitstempelattribute T_B und T_E sind auf der Domäne Zeit definiert und markieren den Beginn und das Ende einer Gültigkeitsperiode.

Zeitunabhängige Attribute: Die Werte von zeitunabhängigen Attributen bleiben für jedes Objekt im Zeitverlauf eines zeitbezogenen Entitätstyps konstant. Ein typisches Beispiel für ein zeitunabhängiges Attribut wäre etwa das Geburtsdatum eines Mitarbeiters.

Zeitabhängige Attribute im weiteren Sinn: Die Werte von zeitabhängigen Attributen im weiteren Sinn können sich für jedes Objekt im Zeitverlauf eines zeitbezogenen Entitätstyps ändern. Die früheren Werte dieser Attribute sind jedoch im Kontext der modellierten Realität nicht relevant. Das bedeutet, dass die Historie solcher Attribute nicht nachvollziehbar sein muss und dass immer nur der momentan gültige und aktuelle Wert dieses Attributs relevant ist. Ein Beispiel für ein zeitabhängiges Attribut im weiteren Sinn wäre etwa die Adresse eines Mitarbeiters, wenn man davon ausgehen kann, dass jeweils nur die aktuelle Adresse eines Mitarbeiters bedeutend ist.

Zeitabhängige Attribute im engeren Sinn: Die Werte von zeitabhängigen Attributen im engeren Sinn können sich für jedes Objekt im Zeitablauf eines zeitbezogenen Entitätstyps ändern. Die früheren Werte dieser Attribute sind relevant, d.h. die Historie dieser Attribute muss nachvollziehbar sein. Wenn sich der Wert eines zeitabhängigen Attributs im engeren Sinn ändert, wird eine neue Version angelegt. Alle früheren Versionen müssen natürlich erhalten bleiben. Ein Beispiel für ein zeitabhängiges Attribut im engeren Sinn wäre die Funktion eines Mitarbeiters.

Zeitabhängige zyklische Attribute: Die Werte von zeitabhängigen zyklischen Attributen können sich für jedes Objekt im Zeitablauf eines zeitbezogenen Entitätstyps ändern. Die früheren Werte dieser Attribute sind relevant, d.h. die Historie dieser Attribute muss nachvollziehbar sein. Wenn sich der Wert eines zeitabhängigen zyklischen Attributs ändert, wird eine neue Version angelegt. Wenn es zu keiner Wertänderung kommt, obwohl prinzipiell eine Wertänderung möglich gewesen wäre, wird zwar keine neue Version angelegt, die Tatsache, dass es keine Änderung gegeben hat, muss aber abgebildet werden. Ein Beispiel für ein zeitabhängiges zyklisches Attribut wäre etwa das Gehalt eines Mitarbeiters. Bekommt ein Mitarbeiter keine Gehaltserhöhung, obwohl prinzipiell eine Gehaltserhöhung möglich gewesen wäre, so bleibt das Gehalt konstant, die Information über die nicht erfolgte Gehaltserhöhung wird aber ebenfalls abgebildet. Andere Beispiele für solche Attribute wären die Verwendungsgruppe von Mitarbeitern, der Kompetenzbereich von Mitarbeitern, die Kategorie von Beherbergungsbetrieben oder die Kategorie von Wohnungen.

Mit dem in [Kai00] entwickelten und dargestellten Mapping-Algorithmus können Schemata aus dem temporal erweiterten ER-Modell in das relationale Modell umgesetzt werden und so mit bereits bestehenden Applikationen verbunden und mit weit verbreiteten DBMS implementiert werden. Temporale Integritätsbedingungen wie etwa der PRIMARY KEY TEMPORAL oder der FOREIGN KEY TEMPORAL gewährleisten dabei eine konsistente Umsetzung zeitbezogener Daten.

3 Temporale Modellierung im objektorientierten Bereich

Im Datenbankbereich findet sich die „Objektorientierung“ in zweierlei Ausprägungen, nämlich in den objektorientierten Systemen und in den objektrelationalen Systemen. Fast alle kommerziellen Anbieter relationaler Datenbanksysteme haben ihre Produkte um die „Objektorientierung“ erweitert und bieten ihre Datenbankmanagementsysteme unter Begriffen wie „objektrelationale“ oder „postrelationale“ Datenbanken an. Im folgenden wird vor allem auf das *objektrelationale Modell* eingegangen (vgl. dazu etwa [Sto99] oder [EN00] Seite 435 ff.).

Für die Entwicklung von Informationssystemen wurden in den letzten Jahren viele Vorgehensmodelle[2] entwickelt. Die Modelle unterscheiden sich zwar in der Ablaufplanung und den Details, die Grundzüge - vom konzeptionellen über den logischen zum physischen Entwurf – sind jedoch in allen Modellen zu finden. Die nachstehende Tabelle gibt einen Überblick der Entwicklungsschritte und der eingesetzten Hilfsmittel.

[2] z.B.: DSDM (Dynamic System Development Method); Oracle CDM (Custom Development Method); i.w.s. auch XP (Extreme Programming); RUP (Rational Unified Process); actiF (von microTOOL), u.v.a.m.

Schritte des Datenbankentwurfs	**objektorientierte Informationssysteme**	**relationale Informationssysteme**
Konzeptioneller Entwurf	UML Typenmodell (bzw. Klassen)	Entity Relationship Modell
Konzeptionelles Schema	*(temporal erweitertes) UML Klassendiagramm*	*(temporal erweitertes) ER-Diagramm*
logischer Entwurf	objektrelationales Modell	relationales Modell
logisches Schema	*(zeitbezogenes) Objekttypen-Diagramm*	*(zeitbezogene) Relationen*
physischer Entwurf	Objekttypen, Objekttabellen, etc.	Tabellen, etc.
physisches Schema	*konkretes DBMS (z.B. Oracle)*	*konkretes DBMS (z.B. Oracle)*
Implementierung	*SQL (DDL)*	*SQL (DDL)*

Tabelle 1: (temporal erweiterte) Entwurfsschritte

In dieser Arbeit wird gezeigt, wie der konzeptionelle Entwurf bei objektorientierten Informationssystemen temporal erweitert werden kann und wie die Überleitungsregeln in den logischen Entwurf um zeitbezogene Regeln ergänzt werden können.

Die Unified Modeling Language (UML[3]) ist eine *graphische Modellierungssprache* zur Visualisierung, Spezifikation, Konstruktion und Dokumentation eines Softwaresystems. Die UML gibt einen Standard für das Entwickeln von Entwürfen, die sich mit *konzeptuellen Aspekten*, wie Geschäftsprozessen und Systemfunktionen auseinandersetzen, vor. Darüber hinaus regelt die UML aber auch ganz *konkrete Aspekte*, wie etwa Klassen, die in einer bestimmten Programmiersprache geschrieben sind, Datenbankschemata und wiederverwendbare Software-Komponenten.[5] Die UML versucht, den gesamten Entwicklungszyklus – d.h. den konzeptionellen, logischen und physischen Entwurf - abzudecken.

Bei der Modellierung von zeitbezogenen objektorientierten bzw. objektrelationalen Informationssystemen mit Hilfe der UML werden in weiterer Folge der Arbeit die einzelnen Entwurfsschritte klar voneinander getrennt. Ziel ist es, die Diagramme der UML so zu erweitern, dass die Zeitdimension realitätskonform abgebildet werden kann, gleichzeitig jedoch die Bedeutung bereits bestehender Elemente der UML beibehalten

[3] Für eine detaillierte Darstellung der UML sei an dieser Stelle auf [BRJ99] bzw. die Spezifikation der UML in [OMG01] verwiesen.

[5] siehe [BRJ99] Seite XV

werden kann. Dadurch wird eine lückenlose Aufwärtskompatibilität und Weiterverwendung existierender Schemata ermöglicht.

3.1 Konzeptioneller Entwurf

Inhalt des konzeptionellen Entwurfes ist es, eine Globalsicht der betreffenden Anwendung zu erstellen. Der Zweck von *konzeptionellen Modellierungsmethoden* ist die Beschreibung eines bestimmten Ausschnitts aus der realen Welt und damit eine Modellbildung. Durch die Modellbildung wird dieser Weltausschnitt vereinfacht, diskretisiert, idealisiert, andererseits aber auch für eine systematische Darstellung zugänglich gemacht.

Wurde im relationalen Bereich vor allem das *ER-Diagramm* als Hilfsmittel für diesen Entwurfsschritt eingesetzt, so findet in der „objektorientierten" Umgebung das UML *Klassendiagramm* seine Anwendung. Dass beide Hilfsmittel nicht nur ähnliche Ziele verfolgen, sondern einander gegenübergestellt werden können, zeigen u.a. [EN00] und [Hal98]: „Object modeling methodologies, such as UML … are becoming increasingly popular. ... Hence, an important part of these methodologies – namely, the *class diagrams* – are similar to EER diagrams in many ways."[6]

Die Terminologie und graphische Darstellung der beiden Modelle unterscheiden sich zwar, bezeichnen aber oft dasselbe.

[6] [EN00] Seite 93

ER-Diagramm	UML Klassendiagramm
Entitätstyp (entity type)	Klasse (class)
Entität (entity)	Objekt (object)
zusammengesetztes Attribut (composite attribute)	Typ (type)
mehrwertiges Attribut (multivalued attribute)	Kardinalität
Beziehungstyp (relationship type)	Assoziation, Verknüpfung (association)
Beziehung (relationship instance)	Verbindung, Instanzbeziehung (link)
attributierte Beziehung (relationship attribute)	Assoziationsklasse
rekursive Beziehung (recursive relationship)	reflexive Assoziation (reflexive association)
abhängige Entität (weak entity)	qualifizierte Assoziation (qualified association) bzw. qualifizierte Aggregation (qualified aggregation)

Tabelle 2: Modellierungsmöglichkeiten wesentlicher EER-Elementen in UML[7]

Basierend auf dieser Gegenüberstellung kann der temporale Erweiterungsmechanismus des Modells RETTE analog auf das UML Klassendiagramm angewendet werden.

[7] Diese Gegenüberstellung basiert auf [EN00] Seite 93 ff. und [Hal99]

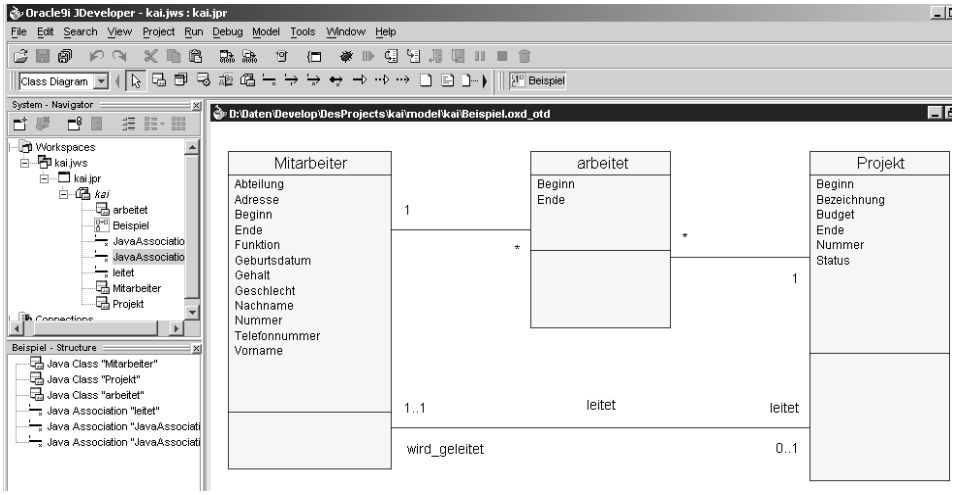

Abbildung 1: konzeptionelles UML Klassendiagramm[8]

Die folgende Tabelle gibt einen Überblick über die Grundelemente des temporal erweiterten UML Klassendiagramms. Die Grundtypen des Klassendiagramms bleiben dabei jedoch bestehen. Um diese von den temporal erweiterten Komponenten zu unterscheiden, werden diese in weiterer Folge unter dem Begriff konventionelle Komponenten zusammengefasst.

[8] Anm.: „association classes“ sind zwar Bestandteil der UML, werden jedoch nicht von allen Werkzeugen unterstützt. Aus diesem Grunde wurde eine „Hilfsklasse“ ins Diagramm aufgenommen.

Grund-komponenten	Komponenten	Erweiterung
Klasse	konventionelle	Differenzierung in konventionelle und zeitbezogene Elemente
	zeitbezogene	
Attribut	Zeitstempel	*explizite* Berücksichtigung der Zeitdimension
	zeitunabhängig	
	zeitabhängig im weiteren Sinn	
	zeitabhängig im engeren Sinn	
	zeitabhängig zyklisch	
Assoziation	konventionelle	Differenzierung in konventionelle und zeitbezogene Elemente
	zeitbezogene	
Methode	konventionelle	

Tabelle 3: Komponenten des temporal erweiterten Klassendiagramms

Eine explizite Kennzeichnung des temporalen Aspekts findet analog zur Vorgehensweise im Modell RETTE nur auf der Attribut-Ebene statt. Im folgenden werden die temporalen Erweiterungen der einzelnen Elemente näher beschrieben.

Klassen

Im Klassendiagramm werden zeitbezogene Klassen nicht gesondert[10] gekennzeichnet, sondern ebenso wie konventionelle Klassen dargestellt. Ob es sich um eine zeitbezogene oder eine konventionelle Klasse handelt, ergibt sich aus den zugeordneten Zeitstempelattributen.[11] Diese Darstellungsform für das konzeptionelle Modell wurde gewählt, um den Anforderungen der Verständlichkeit - die beiden Zeitstempelattribute kennzeichnen die Objektlebensdauer – gerecht zu werden.

[10] Dies ist einer der wesentlichen Unterscheidungsmerkmale zu anderen temporalen UML Erweiterungen wie z.B.: [SB97].

[11] entsprechend zu [Kai00] Seite 79

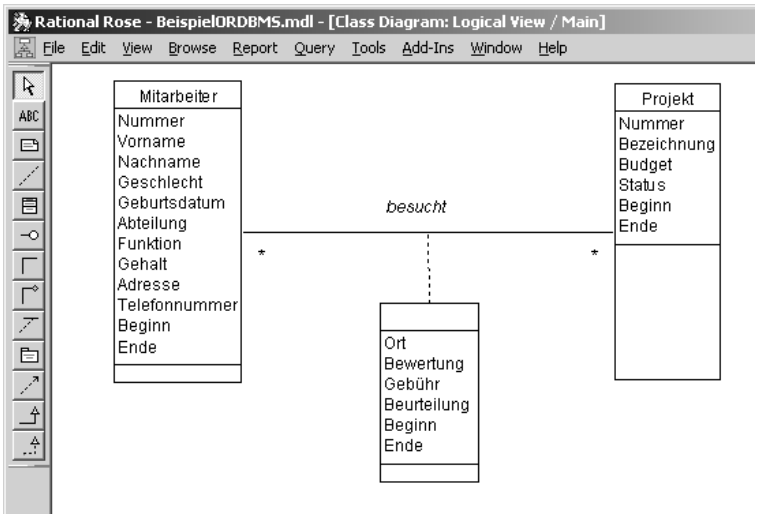

Abbildung 2: UML Klassendiagramm mit zeitbezogenen Klassen

Elemente	**Zeitbezug**
Innere Klasse	Kann Zeitbezug aufweisen
Unterklasse und Oberklasse	Zeitbezug wird von der Superklasse an die Subklasse vererbt. Die Subklasse kann jedoch zusätzlich einen Zeitbezug aufweisen.

Tabelle 4: Spezialfälle von temporalen Klassen

Attribute

Der unterschiedliche zeitliche Aspekt von Attributen wird im Modell RETTE durch fünf Attributarten berücksichtigt. Diese fünf Attributarten übernehmen alle Eigenschaften eines konventionellen Attributs, erhalten aber darüber hinaus zusätzliche semantische Informationen[12].

Die graphische Darstellung der Attributarten des Modells RETTE wird analog in das UML Klassendiagramm übernommen, wobei die Attribute dabei „inline“[13] durch verschiedenartiges Unterstreichen dargestellt werden. Das Unterstreichen von Bezeichnungen ist ein gängiges semantisches Ausdrucksmittel der UML.

[12] vgl. dazu Kapitel 2 in dieser Arbeit.
[13] Für eine ausführliche Diskussion bezüglich der Vor- und Nachteile siehe [Hal99].

Attributarten	**Graphische Darstellung**
Zeitstempelattribute	Beginn, Ende
zeitunabhängige Attribute	Geburtsdatum
zeitabhängige Attribute	
im weiteren Sinn	Adresse
im engeren Sinn	Funktion
zyklische Attribute	Verwendungsgruppe

Tabelle 5: Graphische Darstellung der *Attributarten*

Unterstreichungen werden zum Beispiel auch für den Geltungsbereich von Attributen verwendet. „...a feature that is classifier scoped is rendered by underlining the feature's name. No adornment means that the feature is instance scoped."[14] Außer der Unterstreichung mittels *einfacher Linie* existieren in der UML auf Attributebene keine weiteren semantischen Darstellungen dieser Art. Der temporale Aspekt kann daher durch andersartige Unterstreichungen auf Attributebene berücksichtigt werden.

Diese graphische Notation muss jedoch auch im UML Metamodell Berücksichtigung finden. Ein Vorteil der UML gegenüber dem ER-Modell ist der verfügbare Erweiterungsmechanismus, anhand dessen das Metamodell der UML erweitert werden kann.

Tagged Values

Um die temporale Information auf Attributebene ins Klassendiagramm einzubringen, werden „tagged values" verwendet. „ ... With stereotypes, you can add new things to the UML; with tagged values, you can add new properties."[15]

"Tagged values" können wie folgt in UML dargestellt werden: „Many kinds of elements have detailed properties that do not have a visual notation. In addition, users can define new element properties using the tagged value mechanism. A string may be used to display properties attached to a model element. This includes properties represented by attributes in the metamodel as well as both predefined and user-defined tagged values."[16]

[14] [BRJ99] Seite 124
[15] [BRJ99] Seite 81
[16] [OMG01] Punkt 3.17

Im folgenden ein Beispiel zur textuellen Darstellung von “tagged values”:

`{ temporal = zyklisch}` bzw. `{ zyklisch temporal }`

Assoziationen

Eine Assoziation beschreibt die Beziehung zwischen zwei Klassen. Eine zeitbezogene Assoziation bildet die *Historie* der Beziehung zwischen zwei Klassen ab. Diese besitzen jedenfalls die beiden Zeitstempelattribute, darüber hinaus können sie noch weitere zeitunabhängige und zeitabhängige Attribute im Sinne von Assoziationsklassen (association classes) besitzen.

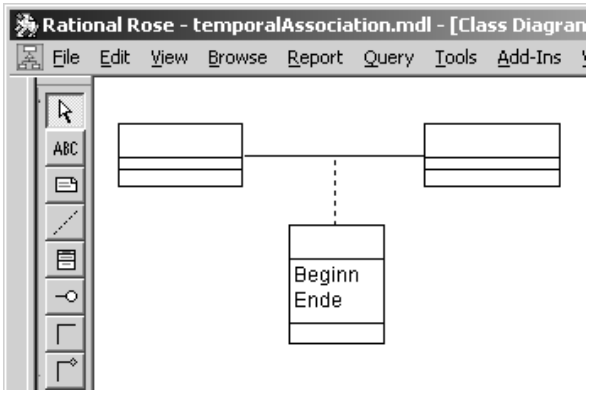

Abbildung 3: Graphische Darstellung einer temporalen Assoziation

Assoziationsklassen, das sind Elemente mit Eigenschaften sowohl von Assoziationen als auch von Klassen, werden anhand von Beziehungen, denen eine Klassendefinition angehängt wird, dargestellt.

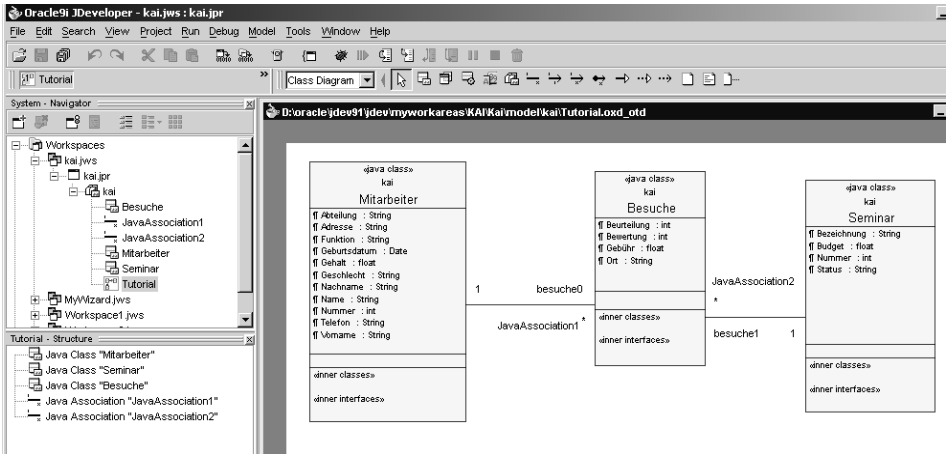

Abbildung 4: Klassendiagramm mit „Hilfsklasse“[17] für eine Assoziationsklasse

Eine Assoziation ist dann zeitbezogen, wenn sie die beiden Zeitstempelattribute „Beginn“ und „Ende“ besitzt.

Zeitstempelattribute

Die Zeitstempelattribute des Modells RETTE werden im relationalen Modell vorteilhaft als zwei Attribute abgebildet. Für die objektorientierte und objektrelationale Systementwicklung können die beiden jedoch auch als ein einzelner, eigener Typ dargestellt werden. Gelangt eine solche Umsetzung zur Anwendung, sind die Attribute „Beginn“ und „Ende“ im Text sinngemäß zu ersetzen.

3.2 Logischer Entwurf

Inhalt der Phase des logischen Entwurfs ist die Abbildung des konzeptionellen Schemas auf ein Schema in einem logischen Datenmodell. *Logische Datenmodelle* bestehen aus drei Hauptkomponenten: Datenstrukturen, Operationen, Integritätsbedingungen.[18]

Der Schritt vom konzeptionellen zum logischen Modell erfolgt bei der Entwicklung für objektorientierte Umgebungen oft nahtlos. Das konzeptionelle Modell wird so lange verfeinert, bis dieses den Konventionen der Zielumgebung entspricht.

---

[17] Da der Oracle JDeveloper in der Version 9i keine Assoziationsklassen unterstützt, wurde analog zur Auflösung von attributierten Beziehungen des ER-Modells eine Hilfsklasse eingeführt.

[18] Vgl. etwa [StNo99] Seite 2

Dem Modell RETTE folgend wird der Übergang hier jedoch transparent nachvollziehbar gemacht. Als nächster Schritt ist daher das konzeptionelle Modell in ein objektrelationales Modell überzuführen.

objektrelationale Elemente	**elements**	**Postfix**
Objekttyp	object type	_objtyp _ntabtyp (Typ für nested table)
Attribut	attribute	 _obj (basieren auf einen Objkettyp) _ref (Referenz) _ntab (für nested table)
Methode	method	
Link durch Einbettung (Komposition)	embedded link	
Link durch Referenz (Aggregation)	ref link	
Kollektion von Verbindungen durch Einbettung (Komposition)	embedded collection link	
Kollektion von Verbindungen durch Referenz (Aggregation)	collection of refs link	
Objekttabelle	object table	
Objektview	object view	

Tabelle 6: objektrelationale[19] Elemente

[19] Mangels einer allgemeinen Definition wird ein pragmatischer Ansatz gewählt – unter objektrelational im Sinne dieses Beitrags wird im folgenden der Funktionsumfang von Oracle9i verstanden. Siehe dazu [Gie01], [PR01] und [Lor01]

3.3 Überleitung des konzeptionellen UML Klassendiagramms

Wurde die Überleitung des ER-Modells in das relationale Modell in der Literatur ausgiebig behandelt und bereits in vielen Werkzeugen realisiert, so sind für die Ableitungen konzeptioneller Modelle in das objektrelationale Datenbankmodell derzeit keine detaillierten Abbildungsvorschriften[20] zu finden. Die Überleitung lässt ähnlich wie die Ableitung von Sub-/Supertypen im relationalen Modell Freiräume für Designentscheidungen. Die folgende Tabelle gibt eine grobe Übersicht der wichtigsten Überleitungen.

Klassendiagramm	**objektrelationales Schema**	**Postfix**
Klasse	object type (ohne Attribute)	_objtyp
Attribut	attribute	
	constructor	
Klassen mit Aggregation und limitierter Kardinalität (wobei die Aggregation bei der anderen Klasse liegt) (im Sinne von multivalued Attribute)	object type als varray (ordered, no need for querying)	_vartyp
Klasse mit Assoziation und einer Kardinalität von 1:1 (im Sinne eines zusammengesetzten Attributs)	object type	_objtyp
Klassen mit Aggregation und limitierter Kardinalität (wobei die Aggregation bei der anderen Klasse liegt) (im Sinne von multivalued Attribute)	object type als nested table (unordered)	_ntabtyp
Assoziation	Attribut (des Typs REF) (entsprechend der Navigierbarkeit)	_ref
Aggregation (im Sinne eines multivalued Attributes)	Attribut (des Typs des anderen Endes der Aggregation)	_var
Aggregation (im Sinne einer weak Entity)	Attribut (des Typs des anderen Endes der Aggregation)	_ntab

[20] Ansätze sind u.a. zu finden in [Gie01] Seite 9-10ff bzw. [EN00] Seite 415.

Klassendiagramm	objektrelationales Schema	Postfix
Aggregation mit Kardinalität von 1:1 (im Sinne eines zusammengesetzten Attributs)	Attribut (des Typs des anderen Endes der Aggregation)	_obj
Attribut	Attribut	

Tabelle 7: Übersicht der wichtigsten Ableitungsschritte

Basierend auf der nachfolgend angegebenen Vorschrift wird die Verfeinerung des konzeptionellen temporalen Modells in ein logisches Modell durchgeführt.

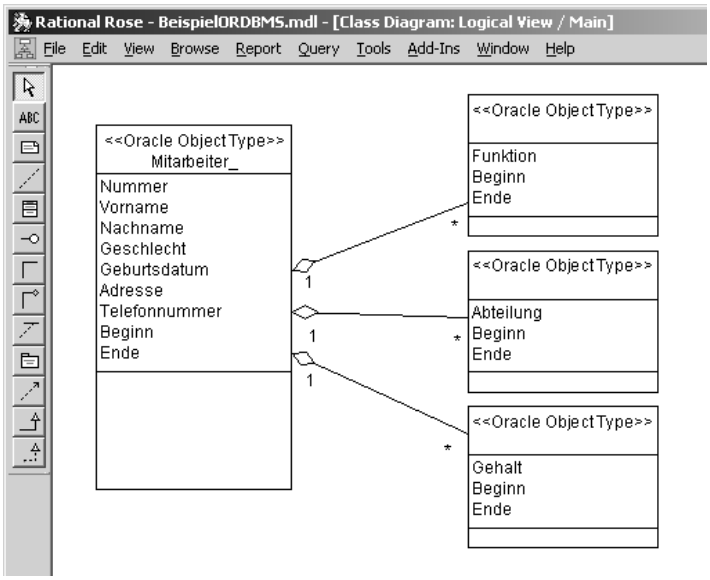

Abbildung 5: Zerlegung in Teilobjekttypen (explizit dargestellt)[21]

Die temporalen Attribute werden in diesem Ableitungsschritt vorerst ebenfalls als Attribute übertragen und erst dann mit dem Zerlegungsalgorithmus weiter behandelt.

[21] Die Darstellung kann auch auf Attributebene durch Angabe der Kardinalität erfolgen.

Zerlegungsalgorithmus für den temporaler Aspekt

Der Zerlegungsalgorithmus lehnt sich sehr eng an den Zerlegungsalgorithmus des Modells RETTE (vgl. [Kai00],S.99ff) an.

Schritt Z1: In dem abgeleiteten Objekttyp verbleiben alle zeitunabhängigen Attribute, alle zeitabhängigen Attribute im weiteren Sinn, die Zeitstempelattribute *Beginn* und *Ende* der Klasse.

Schritt Z2: Gruppiere alle zeitabhängigen Attribute im engeren Sinn nach Synchronitätsklassen.

Schritt Z3: Erstelle einen zeitbezogenen Objekttyp mit folgenden Attributen: alle zeitabhängigen Attribute einer Synchronitätsklasse, den Zeitstempelattributen.

Schritt Z3 (optional): Erstelle entsprechende Zugriffsmethoden für jedes Attribut das in einen zeitbezogenen Objekttyp „ausgelagert“ wurde. Diese Methode gestaltet den Zugriff auf das temporale Attribut einfacher.

Eine Methode sollte gleich dem Attribut bezeichnet werden und als Rückgabewert den Typ des Attributs liefern. Durch die Eigenschaft der Snapshot-Reduzierbarkeit des Modells RETTE ist gewährleistet, dass diese Methode nur einen Wert für einen angegebenen Zeitpunkt zurückliefert. Normalerweise wird dieser Zeitpunkt die Systemzeit sein bzw. in einem Applikationskontext ein vom Benutzer einstellbarer Zeitpunkt.

Schritt Z4: Wiederhole Schritt 3 für alle Synchronitätsklassen der zeitabhängigen Attribute im engeren Sinn.

Ein zum Modell RETTE analoges Überleiten ist für die zyklischen Attribute und Synchronitätsklassen zu beschreiten.

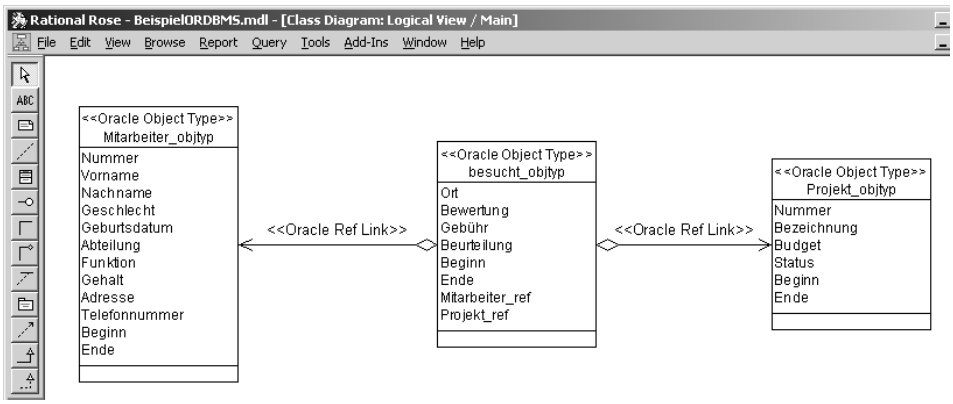

Abbildung 6: Logisches Modell in UML Notation

3.4 Physischer Entwurf

Um die vorgestellte Modellierung des temporalen Aspekts mittels UML zu vervollständigen, findet sich in diesem Abschnitt ein kurzer Überblick der Implementierungsschritte.

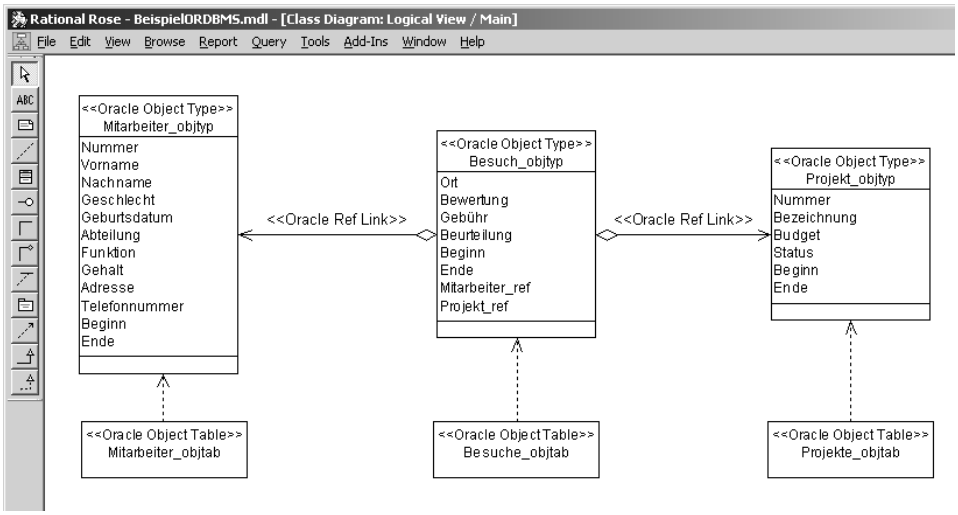

Abbildung 7: temporale Objekttypen (object type) und Objekttabellen (object table)

Umsetzung des Zerlegungsalgorithmus

Durch Anwendung des Zerlegungsalgorithmus auf den abgeleiteten temporalen Objekttyp entstehen weitere Definitionen von Objekttypen. Diese Objekttypen werden als Kompositionen (compositions) mit dem Objekttyp verbunden und bei der Umsetzung wird eine eingebettete Tabelle (nested table) erzeugt. Diese scheint im Objekttyp als Attribut (z.B.: ABTEILUNG__HNTAB) auf.

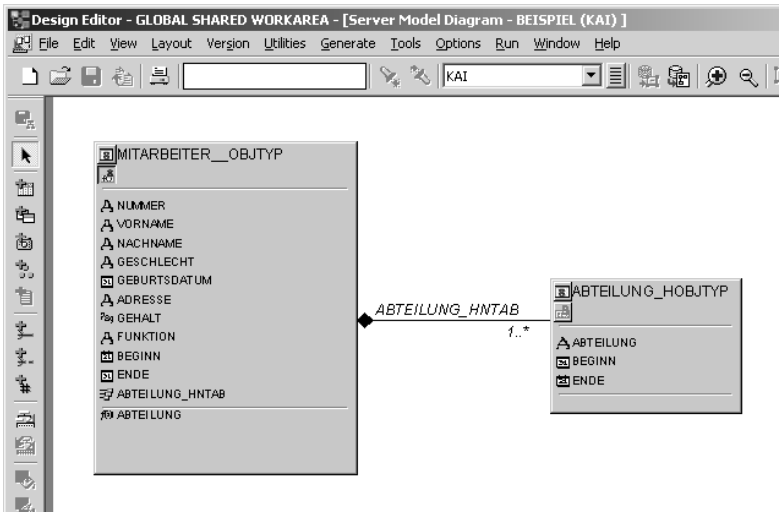

Abbildung 8: temporaler Objekttyp mit eingebetteter Verbindung (embedded link)

Nested Table

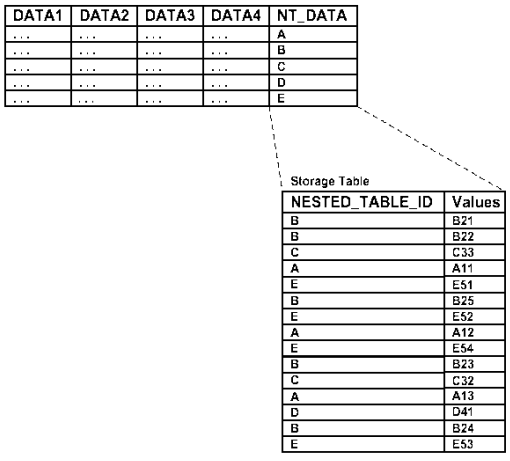

Abbildung 9: verschachtelte Tabelle

Verschachtelte Tabellen (nested table) werden hier als eine der Möglichkeiten, Kollektionen (collections) in das physische Modell überzuführen, verwendet.

Implementierung

```
CREATE OR REPLACE TYPE MITARBEITER_OBJTYP AS OBJECT
 (NUMMER VARCHAR2(240)
 ,VORNAME VARCHAR2(240)
  ...
 ,FUNKTION VARCHAR2(240)
 ,BEGINN DATE
 ,ENDE DATE
 );

CREATE OR REPLACE TYPE BESUCH_OBJTYP AS OBJECT
 (ORT VARCHAR2(240)
 ,BEURTEILUNG NUMBER(1)
 ...
 ,BEGINN DATE
 ,ENDE DATE
 ,MITARBEITER_REF REF MITARBEITER
 ,SEMINAR_REF REF SEMINAR
 ,MEMBER PROCEDURE ADD_FUNKTION_HT
 (FUNKTION VARCHAR2
 ,BEGINN DATE
 ,ENDE DATE
 );
```

Um die Historie speichern zu können, muss eine Kollektion (collection type) angelegt werden.

```
CREATE TABLE MITARBEITER_OBJTAB OF MITARBEITER_OBJTYP;
CREATE TABLE BESUCHE_OBJTAB OF BESUCH_OBJTYP;
CREATE TABLE PROJEKTE_OBJTAB OF PROJEKT_OBJTYP;

CREATE TYPE ABTEILUNG_HNTAB AS TABLE OF
 ABTEILUNG_HOBJTYP;
```

Die Kollektion kann nun in den Objekttyp Mitarbeiter als Attribut inkludiert werden.

```
CREATE OR REPLACE TYPE MITARBEITER_OBJTYP AS OBJECT
 (NUMMER VARCHAR2(240)
 ,VORNAME VARCHAR2(240)
  ...
 ,FUNKTION VARCHAR2(240)
 ,BEGINN DATE
 ,ENDE DATE
 ,ABTEILUNG_HNTAB ABTEILUNG_HNTAB
 ,MEMBER FUNCTION ABTEILUNG
    RETURN VARCHAR2
 );
```

Um Werte der zuvor definierten Objekttypen auch persistent halten zu können, sind im objektrelationalen Schema Objekttabellen zu erzeugen.

```
CREATE TABLE MITARBEITER_OBJTAB OF MITARBEITER_OBJTYP
 NESTED TABLE ABTEILUNG_HNTAB STORE AS
 MITARBEITER_OBJTAB_NT112;
```

Operationen

Im folgenden wird ein Beispiel für das Einfügen eines Mitarbeiters mit seiner Abteilungshistorie ausgeführt. Im Gegensatz zum relationalen Modell - bei dem zum Einfügen zwei getrennte Anweisungen nötig sind - kann hier die Historie in einer einzelnen Anweisung angegeben werden.

```
insert into mitarbeiter__objtab (
 NUMMER,VORNAME,NACHNAME,GESCHLECHT,GEBURTSDATUM,
 ADRESSE, GEHALT,FUNKTION,BEGINN,ENDE,ABTEILUNG_HT
) values (
 2,'Maria','Huber','M','01-APR-1978','Hauptplatz 10',
 23000,'Sekretärin','05-APR-1988','15-APR-2002',
 ABTEILUNG_HNTAB(ABTEILUNG_HOBJTYP('CONS',
                        '05-APR-1988','10-APR-1993'),
                ABTEILUNG_HOBJTYP('PROD',
                        '10-APR-1993','15-APR-2002'))
);
```

Abfragen

Durch die objektrelationale Umsetzung ist die Abfrage und Navigation der temporalen Attribute einfacher als im relationalen Schema, bei dem für Teilrelationen explizite Verknüpfungen erstellt werden müssen.

Das folgende Beispiel zeigt die „Neueinsteiger“ in einer Abteilung nach dem 09.04.1993.

```
select NUMMER,NACHNAME, a.ABTEILUNG, a.BEGINN, a.ENDE,
       a.NESTED_TABLE_ID
from mitarbeiter_objtab, TABLE(ABTEILUNG_HT) a
where a.BEGINN > to_date('09-APR-1988');

NUMMER     NACHNAME   ABTEILUNG      BEGINN   ENDE
---------- ---------- -------------- -------- --------
2          Huber      PROD           10.04.93 15.04.02
4          Simpl      CONS           05.06.01 10.06.02
4          Simpl      PROD           10.06.01 15.06.93

select m.NUMMER,NACHNAME
from mitarbeiter_objtab m
where
  exists( select 'X'
          from TABLE(m.ABTEILUNG_HT) x
          where x.BEGINN > to_date('10-APR-1993')
  );

NUMMER     NACHNAME
---------- ----------
2          Huber
4          Simpl
```

4 Zusammenfassung

Die vorliegende Arbeit hat untersucht, inwieweit die Anwendung des Modells RETTE auf das objektrelationale Modell praktikabel und sinnvoll erscheint. Es wurde ein grober Leitfaden zur Umsetzung des temporalen Aspekts in objektrelationalen Systemumgebungen, im speziellen mit einer Oracle Umgebung gegeben. Inwieweit sich

diese auf Systeme anderer Hersteller umsetzen lassen ist Gegenstand weiterführender Arbeiten. Aufbauend auf dieser Darstellung können in zukünftigen Arbeiten Spezialfälle und Details vor allem im Bereich der Operationen bzw. Methoden und Bedingungen (constraints), die sich auf temporale Attribute beziehen, behandelt werden. Spezialfälle des UML Klassendiagramms sowie des temporalen Aspekts wurden in dieser Arbeit nicht im Detail behandelt.

Literaturverzeichnis

[BRJ99] Booch, G.; Rumbaugh, J.; Jacobson, I.: *„The Unified Modeling Language – User Guide“,* Addison Wesley, 1999

[BS98] Bergamaschi,S.; Sartori,C.: Chrono: a conceptual design framework for temporal entities. In: Proceedings of the 17th International Conference on Conceptual Modeling (ER´98), S.35-50, Springer,1998

[EN00] Elmasri, R.; Navathe, S.: *“Fundamentals of Database Systems*”, Third Edition, The Benjamin/Cummings Publishing Company, Inc., 2000

[Gen96] Genz, H.: “*Wie die Zeit in die Welt kam – Die Entstehung einer Illusion aus Ordnung und Chaos*”, Rowohlt Taschenbuch Verlag, 1996

[Gie01] Gietz, B.: “*Oracle9i Application Developer’s Guide - Object-Relational Features*”, Release 1 (9.0.1), Oracle Corporation, 2001

[GJ97] Gregersen,H.; Jensen,C.S.: TR-3: Temporal Entity-Relationship Models – A Survey. Technical report, TimeCenter, 1998

[GJ98] Gregersen,H.; Jensen,C.S.: TR-35: Conceptual Modeling of Time-Varying Information. Technical report, TimeCenter, January 1997

[GMJ98] Gregersen,H.; Mark,L.; Jensen,C.S.: TR-39: Mapping Temporal ER Diagrams to Relational Schemas. Technical report, TimeCenter, 1998

[Hal99] Halpin, T.: “*UML Data Models From An ORM Perspective*”, in Journal of Conceptual Modeling, Information Conceptual Modeling, Inc, 1998-1999

[JS98] Jensen, C. S.; Snodgrass, R. T.; et al.: “*A Consensus Glossary of Temporal Database Concepts*” February 1998 Version, in Temporal Databases: Research and Practice, O. Etzion, S. Jajodia, and S. Sripada, Eds.: Springer-Verlag, 1998.

[Kai00] Kaiser, A.: *Die Modellierung zeitbezogener Daten,* Peter Lang Verlag, 2000

[Kai98] Kaiser, A.: *Neuere Entwicklungen auf dem Gebiet temporaler Datenbanken – eine kritische Analyse*, in Proceedings des 6. Internationalen Symposiums für Informationswissenschaft (ISI’98), Universitätsverlag Konstanz, S. 329-343, 1998

[Kai99] Kaiser, A.: „*Eine temporale Erweiterung des Entity Relationship Modells*“, 5. ZoBIS Workshop, 1999

[KW00] Kaiser, A.; Wurglitsch, R.: „*Die Umsetzung zeitbezogener Daten in betrieblichen Informationssystemen mit einer Oracle-Umgebung unter Berücksichtigung des Modells RETTE*“, 6. ZoBIS Workshop, 2000

[Lor01] Lorentz, D.: “*Oracle9i SQL Reference*”, Release 1 (9.0.1), Oracle Corporation, 2001

[OMG01] Object Management Group: „ *OMG Unified Modeling Language Specification*", Version 1.4, OMG, 2001

[PR01] Portfolio, T.; Russell, J.: "*PL/SQL User's Guide and Reference*" Release 9.0.1, Oracle Corporation, 2001

[SB97] Marianthi Svinterikou, Babis Theodoulidis: „*The Temporal Unified Modelling Language (TUML)*", Technical Report des TimeLab, 1997

[Sno00] Snodgrass, R.: *Developing Time-Oriented Database Applications in SQL*, Morgan Kaufmann Publishers, 2000

[StNo99] Steiner, A.; Norrie, M. C.: "*Implementing Temporal Databases in Object-Oriented Systems*", in Procceedings of the Fifth International Conference on Database Systems for Advanced Applications, 1997

[Sto99] Stonebraker, M.; Moore, D.: "*Objektrelationale Datenbanken: Die nächste große Welle*", Carl Hanser Verlag, 1999

[SW99] D'Souza, D. F.; Cameron Willis, A.: *„Objects, Components, and Frameworks with UML – The Catalysis Approach"*, Addisson-Wesley, 1999 www.catalysis.org