Association for Information Systems
AIS Electronic Library (AISeL)

Wirtschaftsinformatik Proceedings 1999 | Wirtschaftsinformatik

February 1999

Ein temporales Regel-Repository zur Unterstützung evolutionärer Workflow-Modellierung

Holger Hoheisel
Universität Bern, hoheisel@ie.iwi.unibe.ch

Marcel Pfahrer
Universität Bern, pfahrer@ie.iwi.unibe.ch

Follow this and additional works at: http://aisel.aisnet.org/wi1999

Recommended Citation
Hoheisel, Holger and Pfahrer, Marcel, "Ein temporales Regel-Repository zur Unterstützung evolutionärer Workflow-Modellierung" (1999). *Wirtschaftsinformatik Proceedings 1999*. 30.
http://aisel.aisnet.org/wi1999/30

This material is brought to you by the Wirtschaftsinformatik at AIS Electronic Library (AISeL). It has been accepted for inclusion in Wirtschaftsinformatik Proceedings 1999 by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

Ein temporales Regel-Repository zur Unterstützung evolutionärer Workflow-Modellierung

Holger Hoheisel
Universität Bern (hoheisel@ie.iwi.unibe.ch)

Marcel Pfahrer
Universität Bern (pfahrer@ie.iwi.unibe.ch)

Inhalt

Electronic Business Engineering / 4. Internationale Tagung Wirtschaftsinformatik 1999.
Hrsg.: August-Wilhelm Scheer; Markus Nüttgens. – Heidelberg: Physica-Verlag, 1999

Abstract

Zwischen der Modellierung von Geschäftsprozessen und deren Automatisierung in Workflows besteht normalerweise eine zeitliche Verzögerung. Diese entsteht zwangsläufig aus dem Zeitunterschied der Planung von Prozessen und deren Realisierung. Inkonsistenzen zwischen modellierten Prozessen und aktiven Workflows können nur dann vermieden werden, wenn die Gültigkeit von Prozessen bzw. deren Komponenten in die Modellierung einbezogen werden kann.

Im Rahmen einer Modellierung von Prozessen auf Grundlage von Geschäftsregeln stellt ein Regel-Repository eine zentrale Architekturkomponente dar. In diesem Repository werden die modellierten Prozesse und deren Komponenten strukturiert abgebildet. Eine bitemporale Erweiterung dieses Regel-Repositories ermöglicht neben der versionserhaltenden Speicherung der Prozeßmodelle auch die Berücksichtigung deren Gültigkeitszeiten. Dadurch können zukünftige Gültigkeiten von Prozessen modelliert und zur Startzeit der Gültigkeit automatisch in Workflows umgesetzt werden. Mit diesem Konzept können die oben dargestellten Inkonsistenzen vermieden werden und darüber hinaus durch eine versionserhaltende Speicherung zusätzliche betriebswirtschaftliche Informationen gewonnen werden.

1 Einleitung

Die Abwicklung von Geschäften in Unternehmen kann durch Geschäftsprozesse beschrieben werden. Sowohl für die Darstellung der betriebswirtschaftlichen Sicht dieser Prozesse als auch für die daraus abzuleitenden technischen Spezifikationen automatisierbarer Teilprozesse (Workflows) (vgl. Galler 1997) ist eine Vielzahl von Methoden und Software-Produkten kommerziell verfügbar. Die dabei verwendeten Modellkonstrukte und Notationen weisen große Unterschiede auf und eine durchgängige Modellierung von Prozessen und Workflows wird nur unzulänglich unterstützt.

Im Rahmen des Forschungsprojektes SWORDIES[1] wird ein auf Geschäftsregeln basierender Ansatz entwickelt, welcher eine durchgängige und strukturierte Modellierung und Transformation von Geschäftsprozessen in Workflows unterstützen soll. Dabei werden die Geschäftsregeln der Unternehmen, welche teilweise auch in Prozeßmodellen abgebildet sind, zunächst umgangssprachlich

[1] Die hier vorgelegten Ergebnisse wurden teilweise durch den Schweizerischen Nationalfonds im Projekt SWORDIES (Swiss WORkflow management in Distributed EnvironmentS) unterstützt. Detailliertere Informationen zu diesem Projekt sind erhältlich unter "http://www.ifi.unizh.ch/groups/dbtg/SWORDIES".

beschrieben (Geschäftsprozeßmodell) und anschließend schrittweise soweit detailliert und strukturiert (konzeptionelles Workflow-Modell), daß eine Ableitung von Workflow-Spezifikationen für unterschiedliche Zielumgebungen möglich wird. Auf diese Weise erlaubt der regelbasierte Ansatz nicht nur eine auf allen Ebenen strukturell gleichbleibende Darstellung der Geschäftsprozesse, sondern dient auch als Integrationsschicht zwischen unterschiedlichen Prozeß-modellierungstechniken und Zielsystemen (Abbildung 1) (Endl et al. 1998; Knolmayer 1998).

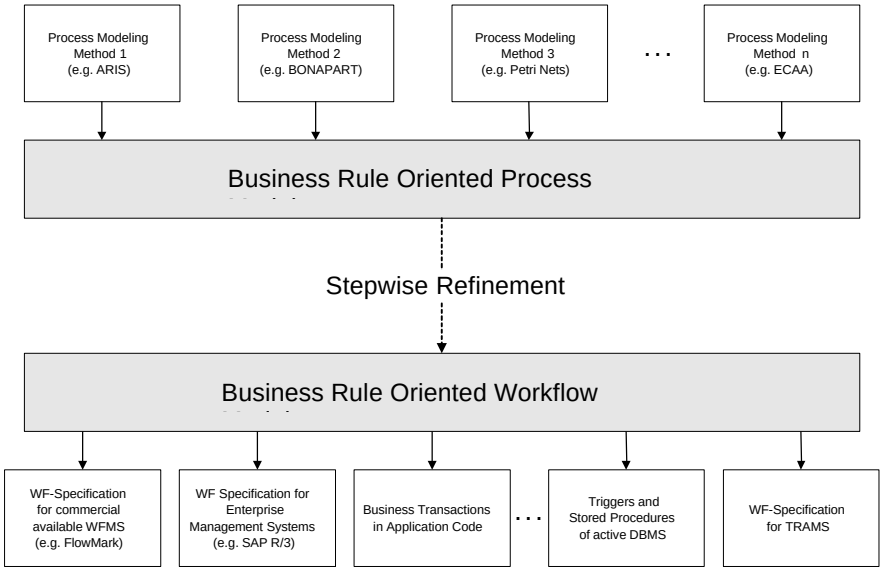

Abbildung 1: Geschäftsregeln als Integrationsschicht (Knolmayer 1998)

Für Geschäftsprozesse entsteht durch den kontinuierlichen Wandel im Umfeld jeder Unternehmung immer wieder die Notwendigkeit zur Anpassung. Für dieses "Business Process Reengineering" ist eine Unterstützung durch geeignete Modellierungswerkzeuge von großer Bedeutung. Im Zentrum steht dabei eine strukturierte Speicherung der Modelle in einem Repository. Für einen auf Geschäftsregeln basierenden Modellierungsansatz bedarf es demnach eines Regel-Repositories, welches eine strukturierte und über alle Abstraktionsebenen konsistente Speicherung der Geschäftsregeln ermöglicht.

Eine Unterstützung des "Change Management" wird durch die Berücksichtigung zeitlicher Bezüge erreicht. Diese haben sowohl bei der Steuerung und Kontrolle eines Workflows, als auch bei der Modellierung der Prozesse große Bedeutung (vgl. Jasper et al. 1996; Kradolfer/Geppert 1998). So sind Geschäftsregeln oft nur in einem beschränkten Zeitraum gültig. In der vorliegenden Arbeit soll aufgezeigt werden, wie ein Regel-Repository mit Hilfe temporaler

Datenmodellierung im Hinblick auf diese Problematik erweitert werden kann. Im anschließenden zweiten Kapitel wird ein allgemeiner Überblick über die regelbasierte Geschäftsprozeß- und Workflow-Modellierung gegeben. Das dritte Kapitel beschreibt die Architektur und Implementierungsmöglichkeiten eines Regel-Repositories. Die temporalen Erweiterungen des Regel-Repositories werden im vierten Kapitel dargestellt. Die Arbeit wird mit einer Zusammenfassung und einem Ausblick (Kapitel 5) abgeschlossen.

2 Geschäftsregelbasierte Prozeß- und Workflow-Modellierung

2.1 Geschäftsregeln

Geschäftsregeln sind wesentlicher Bestandteil des Aufbaus von Organisationen bzw. der in ihnen stattfindenen Abläufe. Diese Regeln schreiben vor, wie die Geschäftsabwicklung zu erfolgen hat, d.h. sie beinhalten Richtlinien und Einschränkungen bezüglich der in Organisationen existierenden Zustände und Geschäftsprozesse (Herbst/Knolmayer 1995). Empirische Studien im Bereich von Workflowmanagement-Systemen zeigen, daß Geschäftsregeln eine wichtige Rolle bei der Beschreibung von Workflows einnehmen (Joosten et al. 1994). Mit Hilfe von Regeln können die unter bestimmten Bedingungen auszuführenden Aktivitäten und die sie auslösenden Ereignisse in Beziehung gesetzt werden. In diesem Sinne können Geschäftsprozesse auf der Grundlage von Geschäftsregeln beschrieben und ausgeführt werden (Knolmayer et al. 1997).

Im Bereich aktiver Datenbanksysteme werden Regeln oft durch die drei Komponenten *event* (Ereignis), *condition* (Bedingung) und *action* (Aktion) beschrieben, woraus sich die Bezeichnung als ECA-Regel ableitet (Dayal 1998). ECA- Regeln können nicht nur für die Spezifikation dynamischer Aspekte in Datenbanksystemen verwendet werden, sondern auch für die Formalisierung von Geschäftsregeln auf konzeptioneller Ebene. Hierzu eignen sich insbesondere ECAA-Regeln, welche um eine zweite Aktionskomponente ergänzt sind und somit die Spezifikation einer Selektion innerhalb einer Regel ermöglichen (Abbildung 2).

ON	***E****vent*
IF	***C****ondition*
Then DO	***A****ction*
Else DO	***A****lternative Action*

Abbildung 2: ECAA-Notation

Außerdem werden für die Modellierung auch EA-Regeln verwendet, welche keine Bedingungskomponenten enthalten (Endl et al. 1998). ECA- und EA-Regeln können als Sonderfall von ECAA-Regeln betrachtet werden. Deshalb beziehen sich die nachfolgenden Ausführungen jeweils auf ECAA-Regeln.

2.2 Ablaufmodellierung

Die Abbildung von Ablaufstrukturen mit Hilfe von ECAA-Regeln wird durch die Ereigniskomponente ermöglicht. Die Ausführung einer Aktionskomponente einer Regel führt zu einem Ereignis, welches in einer anderen Regel als auslösendes Ereignis deklariert werden kann. Auf diese Weise können mehrere ECAA-Regeln sequenziell verknüpft werden, wodurch auch eine sequenzielle Abfolge von Aktionen definiert wird (Endl et al. 1998).

Eine Selektion (XOR-Split) kann mit Hilfe einer ECAA-Regel abgebildet werden. Führen die beiden alternativen Aktionskomponenten der Regel zu verschiedenen Ereignissen, so können diese auch verschiedene Folgeregeln auslösen und somit zu alternativen Abläufen führen. Eine Zusammenführung alternativer Abläufe (XOR-Join) entsteht dadurch, daß die Abläufe früher oder später zum gleichen Ereignis führen.

Auf ähnliche Art kann mit Hilfe einer ECAA-Regel auch eine Iteration dargestellt werden. Dabei führt einer der alternativen Abläufe direkt oder indirekt zu einem Ereignis, welches eine Regel auslöst, die im gleichen Teilprozeß bereits vorgängig ausgelöst wurde (Endl et al. 1998).

Parallele Abläufe entstehen dadurch, daß das gleiche Ereignis mehrere verschiedene Regeln auslöst. Für die Zusammenführung bedarf es hier eines komplexen Ereignisses, welches eine disjunktive Verknüpfung der Endereignisse aller parallelen Abläufe darstellt. Eine Regel, in welcher dieses komplexe Ereignis als auslösendes Ereignis spezifiziert ist, wird somit ausgelöst, wenn alle parallelen Teilabläufe beendet sind (Endl et al. 1998).

Diese Ablaufstrukturen können beliebig miteinander kombiniert werden, wodurch auch komplexe Prozeßabläufe mit Geschäftsregeln problemlos darstellbar sind.

2.3 Erweiterung der ECAA-Notation um Aspekte der Workflow-Modellierung

Damit eine Durchgängigkeit vom Prozeßmodell über ein konzeptionelles Workflow-Modell bis hin zur konkreten Workflow-Spezifikation möglich wird, ist eine Erweiterung der herkömmlichen ECAA-Notation notwendig. Für die schrittweise Verfeinerung und Konkretisierung dieser Prozeßmodelle sind Konstrukte notwendig, welche die Darstellung von Regeln mit unterschiedlichen Strukturiertheits- und Detaillierungsgraden erlauben.

Für die Berücksichtigung weiterer Aspekte der Workflow-Modellierung ist zudem eine Erweiterung der Bedingungs- und der Aktionskomponente notwendig. Einerseits Bedarf es einer Möglichkeit zur Spezifikation von Akteuren (Verbindung zur Aufbauorganisation, "invoked applications"), andererseits müssen auch Input- und Output-Daten spezifiziert werden können (Verbindung zum Datenmodell) (vgl. Keller et al. 1992).

Im Rahmen der vorliegenden Arbeit wird zugunsten einer einfacheren Darstellung nicht weiter auf diese Erweiterungen eingegangen. Ein Vorschlag für eine erweiterte ECAA-Notation ist in (Endl et al. 1998) beschrieben.

2.4 Beispiel

Für ein besseres Verständnis der in den nachfolgenden Kapiteln beschriebenen Konzepte werden die Ausführungen mit Beispielen ergänzt. Der dabei zugrundegelegte Prozeß beschreibt ausschnittsweise die Bearbeitung eines Kreditantrages (KA) in einem Bankinstitut.

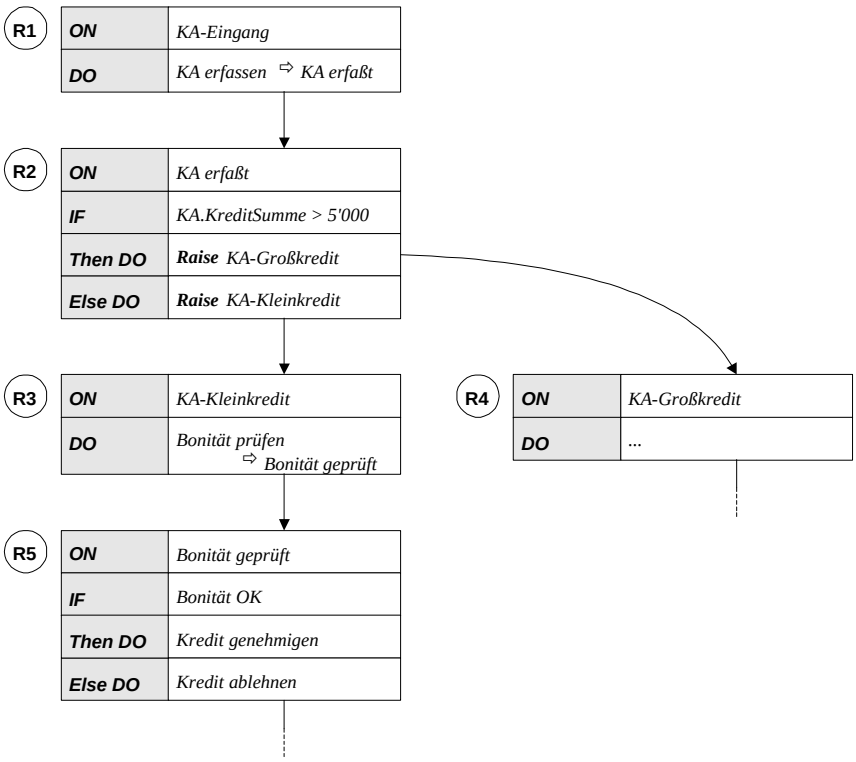

Abbildung 3: Regeln des Beispielprozesses

Nach dem Eingang eines Kreditantrages wird dieser zuerst durch einen Sachbearbeiter elektronisch erfaßt. Für die weitere Bearbeitung wird aufgrund der beantragten Kreditsumme unterschieden, ob es sich um einen Klein- oder Großkredit handelt. Während der Sachbearbeiter für Kleinkredite selbständig eine Bonitätsprüfung durchführen und über Annahme oder Ablehnung des Antrages entscheiden kann, ist für Großkredite ein umfangreicheres Prüfverfahren notwendig. Auf dieses Prüfverfahren sowie auf die weitere Bearbeitung des Antrages wird im Rahmen dieses Beispiels nicht weiter eingegangen. Die Beschreibung des Prozeßausschnittes mit Hilfe von ECAA-Regeln ist in Abbildung 3 dargestellt. In den nachfolgenden Abschnitten werden die einzelnen Regeln jeweils durch die Nummernbezeichnungen R1 bis R5 referenziert.

3 Geschäftsregel-Repository

3.1 System-Architektur

Zur Gewährleistung der Konsistenz der Modellierung über alle Abstraktionsebenen ist der Modellierungsprozeß durch geeignete Werkzeuge zu unterstützen. Von zentraler Bedeutung ist dabei die Verfügbarkeit eines Regel-Repositories (Herbst/Myrach 1997). Ein auf dem kommerziell verfügbaren Repository-Werkzeug *Rochade* basierendes System wurde in (Herbst 1997) beschrieben. Eine erweiterte Architektur eines Systems für die regelbasierte Prozeß- und Workflow-Modellierung ist in (Knolmayer et al. 1997) dargestellt (Abbildung 4).

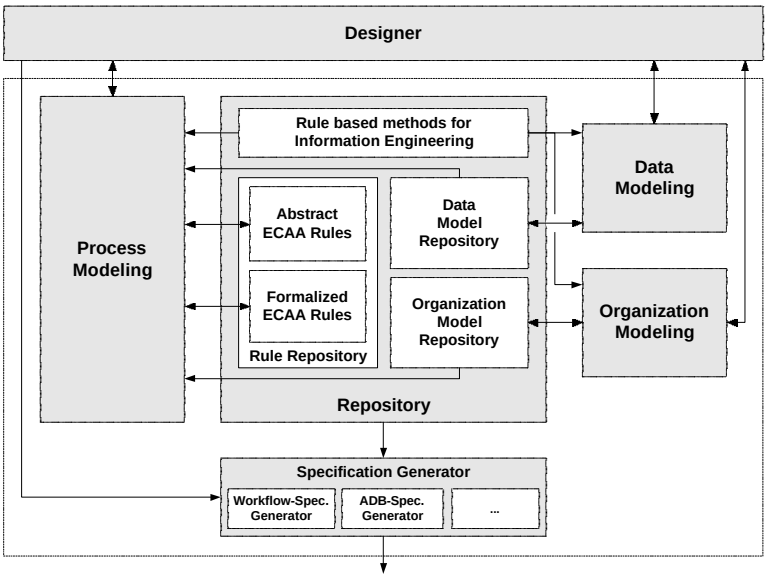

Abbildung 4: Architektur eines regelbasierten Werkzeugs zur Prozeß- und Workflow-Modellierung

3.2 Meta-Modell

Aufgrund der in den vorangegangenen Abschnitten beschriebenen Überlegungen kann die Struktur des Regel-Repositories in Form eines Meta-Modells definiert werden. Abbildung 5 zeigt den für diesen Beitrag relevanten Ausschnitt eines Meta-Modelles basierend auf (Herbst 1997, S. 97). Im Zentrum des Modells stehen die Entitätsmengen *BusinessRule* und *Process*. Ausschließlich anhand dieser beiden Entitätsmengen werden in den nachfolgenden Abschnitten auch die Überlegungen bezüglich einer möglichen Implementierung und temporalen Erweiterung des Repositories aufgezeigt. Bei der Realisierung in einer relationalen Datenbank wird zur Umsetzung der zwischen den beiden Entitätsmengen bestehenden "many to many"-Beziehung eine zusätzliche Beziehungsrelation notwendig. Während die Entitätsrelation *BusinessRule* die Struktur der Regeln beinhaltet, werden in der Beziehungsrelation *Process* die einzelnen Regeln bestimmten Prozessen zugewiesen. Die Entitätsrelation, in der die Prozeßbeschreibungen gespeichert werden, wird hier nicht betrachtet.

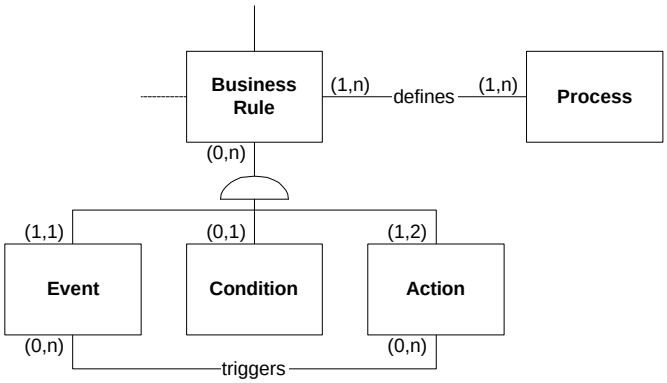

Abbildung 5: Ausschnitt aus dem Meta-Modell des Regel-Repositories

3.3 Implementierungsmöglichkeiten

Für die Implementierung des Regel-Repositories läßt sich das Meta-Modell in verschiedenen Datenmodellen abbilden. Neben kommerziell verfügbaren Repository-Systemen wäre auch eine Realisierung auf Basis eines relationalen, objektrelationalen oder objektorientierten Datenbanksystems denkbar. Aufgrund seiner weiten Verbreitung und seiner Einfachheit beziehen sich die nachfolgenden Ausführungen auf das relationale Datenbankmodell.

In einer konventionellen relationalen Datenbank (ohne temporale Erweiterung) können die beiden Relationen vereinfacht wie folgt definiert werden:

Process = { *P-ID*, *R-ID* (® *BusinessRule*) }
BusinessRule = { *R-ID*, *Event*, *Condition*, *Action*, *Alt-Action* }

Die unterstrichenen Attribute stellen die Primärschlüssel der jeweiligen Relation dar und mit dem Pfeil wird signalisiert, daß das Attribut einen Fremdschlüssel auf die nach dem Pfeil spezifizierte Relation darstellt.

Die ECAA-Struktur der einzelnen Geschäftsregeln wird in der Relation *BusinessRule* abgebildet. Während die Attribute *Event* und *Condition* die ON- bzw. IF-Komponente der Geschäftsregel beschreiben, bilden die Attribute *Action* und *Alt-Action* die beiden alternativen Aktionskomponenten der Geschäftsregel. Die Werte des Primärschlüssel-Attributs *R-ID* identifizieren die einzelnen Regeln. Die Beziehungs-Relation *Process* enthält die Regel-Identifikatoren *R-ID* und die Prozeß-Identifikatoren *P-ID*. Die Beziehung zwischen beiden Relationen wird über das Attribut *R-ID* hergestellt. Da eine Geschäftsregel in mehreren Prozessen enthalten sein kann, besteht zwischen beiden Relationen eine "one to many"-Beziehung.

4 Temporale Erweiterung des Regel-Repositories

4.1 Ziel und Grundlagen temporaler Datenbanken

Ziel von temporalen Datenbanken ist es, einmal in einer Datenbank erfaßte Daten auch dann rekonstruierbar bzw. abfragbar zu erhalten, wenn sie nicht mehr aktuell sind. Temporale Datenbanken zeichnen sich dadurch aus, daß ein physikalisches Löschen und Überschreiben von Daten unzulässig ist; statt dessen führen Datenmodifikations-Operationen zum Einfügen neuer Datensätze. Die Daten werden somit informationserhaltend über die Evolution von Objekten hinweg gespeichert.

Zur Umsetzung dieses Ziels werden bei der temporalen Datenmodellierung Fakten mit einer Zeitbindung verknüpft. Diese Zeitbindung wird durch Zeitstempel realisiert, die eine Zeitangabe darstellen. Zwei Typen von Zeitangaben, die durch Zeitstempel unterstützt werden, sind (Jensen et al. 1994):

Gültigkeitszeit: Durch die Gültigkeitszeit wird angezeigt, wann ein Faktum in der modellierten Realität tatsächlich vorliegt. Diese Zeiten müssen vom Benutzer spezifiziert werden. Wird mit einem Zeitstempel eine Periode der Gültigkeit ausgedrückt (d.h. das verbundene Faktum ist über eine bestimmte Zeitdauer gültig) und steht das Ende der Gültigkeit des Faktums zum jetzigen Zeitpunkt nicht fest, dann ist als Intervall-Obergrenze der unbestimmte Zeitwert *forever*

einzufügen. Dieser optimistische Wert drückt aus, daß zum jetzigen Zeitpunkt angenommen wird, das Faktum sei für die gesamte Zukunft gültig.

Transaktionszeit: Durch die Transaktionszeit wird angezeigt, zu welcher Zeit die im System abgelegten Daten als aktuell geführt werden. Diese Zeitangaben werden vom System automatisch bei den entsprechenden Datenmodifikationsoperationen gesetzt. Der Wert des Transaktionszeitstempels, der das Ende des Intervalls darstellt, bleibt solange auf UC (until changed), bis eine Änderung des zugehörigen Faktums erfolgt (d.h. wenn dem unbestimmten Ende der Gültigkeitszeit ein bestimmter Zeitwert zugewiesen wird) (Barnert/Schmutz 1997, S. 48).

Datenbanken, die beide Zeitkategorien berücksichtigen, werden bitemporal genannt. Wird nur eine der beiden Zeitkategorien genutzt, handelt es sich zwar auch um eine temporale Datenbank, jedoch kommt es in solchen Systemen zu Informationsverlusten (Myrach 1997, S. 36).

4.2 Relevanz für ein Regel-Repository

In einer konventionellen Datenbank könnte der in Abbildung 3 beschriebene Geschäftsprozeß folgendermaßen in die beiden zu betrachtenden Relationen umgesetzt werden (Abbildung 6):

Process	
P-ID	*R-ID*
P001	R1
P001	R2
P001	R3
P001	R4
P001	R5
...	...

BusinessRule				
R-ID	*Event*	*Condition*	*Action*	*Alt-Action*
R1	KA-Eingang		KA erfassen	
R2	KA erfaßt	Sum(KA) > 5000	KA-Großkredit	KA-Kleinkredit
R3	KA-Kleinkredit		Bonität prüfen	
R4	KA-Großkredit		...	
R5	Bonität geprüft	Bonität OK	KA genehmigen	KA ablehnen
...	...	...	...	...

Abbildung 6: Umsetzung des Beispielprozesses in Relationen

In den Relationen wird deutlich, welche Geschäftsregeln in einem bestimmten Prozeß ausgeführt werden und welche Struktur die Geschäftsregeln haben. Mit der Abfrage

```
SELECT *
FROM   Process
WHERE  P-ID = 'P001'
```

werden beispielsweise sämtliche Geschäftsregeln des Prozesses P001 selektiert. Es wird jedoch nicht sichtbar, wann der Prozeß modelliert wurde, ob und wann der Prozeß in eine Workflow-Spezifikation umgesetzt wurde bzw. werden soll

und ob im Laufe der Zeit Geschäftsregeln dem Prozeß hinzugefügt wurden oder ob Regeln entfernt wurden.

Bleibt die gesamte Evolution der gespeicherten Objekte erhalten und abfragbar, stehen einerseits zusätzliche Informationen über die Evolution von Prozessen und Regeln zur Verfügung. Zum anderen werden Informationen darüber gewonnen, wann ein modellierter Prozeß mit welcher Struktur in eine Workflow-Spezifikation umgesetzt wurde oder werden soll, d.h. wann ein Prozeß von der Buildtime- in die Runtime-Komponente transferiert wurde oder werden soll. Dazu wird der Zeitpunkt, zu dem ein Prozeß in die Workflow-Spezifikation umgesetzt wurde bzw. werden soll, als Startzeitpunkt der Gültigkeitszeit eingetragen. Es ist also, ähnlich der Freigabezeit von Aufträgen im Fertigungsprozeß (Scheer 1994) möglich, die Umsetzung eines Prozesses oder einer Prozeßkomponente in der Zukunft zu bestimmen. Ist beispielsweise die Umstrukturierung eines Prozesses zu einem in der Zukunft liegenden Zeitpunkt geplant, so können die neuen Strukturen vorab definiert werden. Als Startzeitpunkt der Gültigkeitszeit wird dem neu strukturierten Prozeß das Datum der geplanten Einführung des Prozesses zugewiesen. Zu diesem durch die Gültigkeitszeit spezifizierten Zeitpunkt wird der modellierte Prozeß automatisch in die Workflow-Spezifikation umgesetzt. Als Voraussetzung für die zeitgerechte automatisierte Umsetzung muß also zum einen der modellierte Prozeß mit den neu auszuführenden Regeln in die Relation *Process* gespeichert werden und zum anderen als Startwert der Gültigkeitszeit das Datum der Umstrukturierung zugewiesen bekommen. Diese neuen Regeln werden automatisch zu dem in der Gültigkeitszeit bestimmten Startwert in die Workflow-Spezifikation umgesetzt. Die Regeln, welche nach der Umstrukturierung nicht mehr im Prozeß verbleiben sollen, erhalten das Datum der Umstrukturierung als Ende der Gültigkeitszeit zugewiesen. Zu diesem Zeitpunkt werden diese Regeln automatisch aus der Workflow-Spezifikation entfernt.

Die zusätzlichen ablauforganisatorischen Informationen (in welchem Zeitraum war ein Prozeß mit welcher Struktur in der Workflow-Spezifikation aktiv) können auch betriebswirtschaftlich genutzt werden. Zum einen kann die Evolution der Abläufe des modellierten Unternehmens nachvollzogen werden; zum anderen wird ein Vergleich von Abläufen möglich, die mit bestimmten Kennzahlen verknüpft werden. Beispielsweise könnten Durchlaufzeiten mit unterschiedlichen Prozeßstrukturen verglichen werden (z.B. nach Hinzufügen oder Entfernen einzelner Geschäftsregeln aus einem Prozeß).

Ähnliches gilt auch für die zweite betrachtete Relation des Regel-Repository. Werden die Strukturen von Geschäftsregeln in konventionellen Datenbanken modelliert, gehen zum einen durch Löschungen und Überschreibungen Informationen über Zustände der einzelnen Regeln verloren. Es kann nicht nachvollzogen werden, ob und zu welchen Zeitpunkten die bestehenden Regeln andere Komponenten enthalten haben als die Komponenten, die zum jetzigen Zeitpunkt gespeichert sind. Zum anderen ist es nicht möglich bereits modellierte

Geschäftsregeln, die jedoch erst in der Zukunft ihre Gültigkeit erlangen, jetzt zu speichern und mit der Information der zukünftigen Gültigkeit zu versehen.

Beispielsweise wird die Regel

ON	*KA erfaßt*
IF	*KA.KreditSumme > 5'000*
Then DO	***Raise*** *KA-Großkredit*
Else DO	***Raise*** *KA-Kleinkredit*

dahingehend geändert, daß ab Beginn des folgenden Jahres die Summe für Kreditanträge, die als Großkredit bearbeitet werden sollen, von DEM 5.000 auf DEM 10.000 erhöht wird. In einer konventionellen Datenbank kann diese zukünftige Änderung auf zwei Arten behandelt werden:

- Die Änderung der Kreditsumme wird sofort eingetragen. Bis zum tatsächlichen Eintreten der Änderung befindet sich die Datenbank in keinem integeren Zustand. Sie enthält falsche Informationen.
- Das Update erfolgt zum Zeitpunkt des tatsächlichen Eintretens der Änderung. Es besteht bei diesem Vorgehen die Gefahr, daß diese Änderung im Alltagsgeschäft vergessen wird. Auch dann befindet sich die Datenbank in keinem integeren Zustand.

Wird jedoch das Ende der Gültigkeitszeit der zum jetzigen Zeitpunkt gültigen Regel auf dieses Änderungsdatum gesetzt, verliert die Regel bei Eintreten des spezifizierten Datums ihre Gültigkeit. Gleichzeitig kann bereits zum Zeitpunkt der Entschlußfassung über die Änderung der Kredithöhe die neue Version der Regel modelliert werden und mit dem Datum der Einführung im Unternehmen als Startzeitpunkt der Gültigkeitszeit gestempelt werden. Die modellierte Regel wird automatisch zum spezifizierten Zeitpunkt gültig. Die Datenbank befindet sich weiterhin in einem integeren Zustand und es besteht nicht die Gefahr, daß zukünftig eintretende Änderungen vergessen werden.

Ein weiterer Vorteil der versionserhaltenden Speicherung von Geschäftsregeln besteht darin, daß bei Rechtsstreitigkeiten genau nachvollzogen werden kann, zu welchem Zeitpunkt eine bestimmte Regel gültig war. Entscheidet beispielsweise ein Mitarbeiter der Abteilung Kleinkredit über einen Kredit von DEM 7.000, so kann genau nachvollzogen werden, ob er zum Zeitpunkt seiner Entscheidung dazu berechtigt war oder nicht, d.h. ob er die Entscheidung vor Inkrafttreten der neuen Regelung getroffen hat oder danach.

Durch die Speicherung der Transaktionszeit wird festgehalten, wann sich die Daten in der Datenbank geändert haben. Sie wird automatisch bei Datenmodifikations-Operationen vom System in die entsprechenden Relationen einge-

fügt. Durch sie wird die versionserhaltende Speicherung möglich und ein physisches Überschreiben bzw. Löschen der Daten kann unterbleiben.

4.3 Temporale Erweiterungen des Regel-Repositories

Im Bereich temporaler Datenmodellierung wurden in den vergangenen 25 Jahren weit mehr als 1.000 Forschungsbeiträge veröffentlicht (für einen Überblick vgl. Lorentzos/Mitsopoulos 1997; Snodgrass 1990; Snodgrass 1995; Tansel et al. 1993 sowie die Literaturdokumentation unter "ftp.cs.arizona.edu"). Dennoch existieren noch keine kommerziellen Produkte, welche die Konzepte temporaler Datenbanken umsetzen. Deshalb ist es erforderlich, das temporale Regel-Repository in einem konventionellen Datenbanksystem umzusetzen. Die temporalen Aspekte der Datenerfassung und -speicherung müssen also in bestehende Datenmodelle - hier das relationale Modell - integriert werden.

Als Periode wird ein gerichteter Zeitraum mit festgelegtem Start- und Endzeitpunkt verstanden (Soo/Snodgrass 1995, S. 126ff.). Da es in konventionellen Datenbanken keinen Datentyp *Periode* gibt, der im Gegensatz zum Datentyp *Intervall* einen verankerten Zeitraum darstellt, müssen die Attribute der Zeitstempel zerlegt werden (Myrach 1997, S. 39f.). Für die Gültigkeitszeit sind die Attribute *gültig-von* und *gültig-bis* erforderlich, damit die Periode der Gültigkeit eines Tupels dargestellt werden kann. Zur Repräsentation des speziellen Zeitwerts *forever*, der für das Ende der Gültigkeitszeit genutzt werden kann, ist der Eintrag des größten vom System unterstützten Zeitwertes möglich (z.B. ist der größte mögliche Zeitwert bei Oracle 4712-12-31) (Clifford et al. 1994). Die Transaktionszeit wird durch die Attribute *erfaßt-am*, *ersetzt-am* und *korrigiert-am* dargestellt. Sämtliche Werte dieser Attribute werden vom System automatisch gesetzt und können vom Anwender nicht beeinflußt werden. Während die Werte des Attributs *erfaßt-am* anzeigen, wann ein Datensatz in die Datenbank gespeichert wurde, stellen die Werte des Attributs *ersetzt-am* den Zeitpunkt des logischen Ersetzens des Datensatzes dar, bzw. die Werte des Attributes *korrigiert-am* stellen den Zeitpunkt dar, zu dem der Datensatz nach einer falschen Eingabe korrigiert wurde. Die Datensätze, deren Attribut *korrigiert-am* den Wert *NULL* haben, geben die aktuellen Daten an (Ben-Zvi 1982; Barnert/Schmutz 1995).

In dieser bitemporalen Systematik wird ein konventionelles Tupel über eine Menge von Tupelversionen dargestellt (Tupelversionen-Menge). Diese Tupelversionen-Menge beschreibt die gesamte Evolution des gespeicherten Objekts. Die einzelnen Tupel einer bitemporalen Relation beschreiben dementsprechend eine Version des Objekts (Tupelversion). Durch eine Tupelversion wird der Zustand des gespeicherten Objekts während eines bestimmten Zeitraums definiert (Ben-Zvi 1982). Dieser Zeitraum wird durch die beiden Gültigkeitszeit-Werte definiert und kann in der Vergangenheit, Gegenwart oder in der Zukunft liegen. Es muß jedoch sichergestellt werden, daß über die Lebensspanne (d.h. die Summe der Gültigkeits-Perioden der Tupelversionen einer Tupelversionen-

Menge; vgl. Clifford/Croker 1993) einer Tupelversionen-Menge zu jedem Zeitpunkt nur eine korrekte Tupelversion existiert (non-overlap-property) (Myrach et al. 1996; Myrach 1995).

Das physische Überschreiben von Werten wird in einem Fall als Ausnahme zugelassen. Zur Verringerung von Redundanzen in den gespeicherten Werten kann der Wert von *gültig-bis* dann überschrieben werden, wenn er den Wert *forever* repräsentiert (Barnert/Schmutz 1997, S. 49). Auf diese Weise werden die Informationen einer Tupelversion in einem Tupel gehalten und es sind keine Korrekturen notwendig. Der Zeitpunkt, zu dem das Ende der Gültigkeitszeit mit einem Zeitwert ungleich *forever* gestempelt wird, wird als Zeitwert in *ersetzt-am* festgehalten.

Da der zeitunveränderliche Primärschlüssel-Wert innerhalb einer Tupelversionen-Menge (Tupelversionen-Mengen-Identifikator) immer gleich bleibt und sich somit über mehrere Tupelversionen erstrecken kann, muß er um weitere Werte ergänzt werden, damit eine Eindeutigkeit der einzelnen Tupel erlangt wird. (Barnert/Schmutz 1995) schlagen dafür einen Surrogat-Wert vor, der automatisch beim Einfügen einer neuen Tupelversion erhöht wird. Der Primärschlüssel ergibt sich dann aus den Werten des Tupelversionen-Mengen-Identifikators und des Surrogat-Wertes. Eine ausführliche Diskussion der Schlüsselproblematik in temporalen Datenbanken befindet sich in (Myrach 1995.)

Die beiden Beispiel-Relationen des Regel-Repositorie erhalten die in Abbildung 7 dargestellte Struktur.

Process							
P-ID	*R-ID*	*Surogat*	*gültig-von*	*gültig-bis*	*erfaßt-am*	*Ersetzt-am*	*korrigiert-am*
P001	R1	1	1998-09-01	forever	1998-05-15	UC	
P001	R2	2	1998-09-01	forever	1998-05-31	UC	...
P001	T3	3	1998-09-01	forever	1998-06-21	UC	1998-07-02
P001	R3	6	1998-09-01	forever	1998-07-02	UC	
R001	R4	4	1998-09-01	forever	1998-06-30	UC	
P001	R5	5	1998-09-01	forever	1998-07-01	UC	

BusinessRule										
R-ID	*Sur*	*Event*	*Condition*	*Action*	*Alt-Action*	*gültig-von*	*gültig-bis*	*Erfaßt-am*	*Ersetzt-am*	*korrigiert-am*
R1	1	KA-Eingang		KA erfassen		1997-01-13	forever	1997-01-03	UC	
R2	2	KA erfaßt	SUM (KA)> 5000	KA-Großkredit	KA-Klein-kredit	1997-12-01	1998-12-31	1997-12-01	1998-09-03	
R2	6	KA erfaßt	SUM (KA)> 10000	KA-Großkredit	KA-Klein-kredit	1999-01-01	forever	1998-09-03	forever	
R3	3	KA-Klein-kredit		Bonität prüfen		1998-05-12	forever	1998-05-03	UC	
R4	4	KA-Groß-kredit	...	...	...	1998-06-30	forever	1998-05-03	UC	
R5	5	Bonität geprüft	Bonität OK	KA genehmigen	KA ablehnen	1998-07-01	forever	1998-06-15	UC	
...	...	...	...	...	...	...	...	...	...	...

Abbildung 7: Temporale Erweiterung der Relationen *Process* und *BusinessRule*

Die Beziehungen zwischen Relationen gestalten sich bei temporalen Relationen komplexer als bei konventionellen. Während im konventionellen Modell jedes Tupel der Frendschlüssel-Relation sich auf nur ein Tupel der Primärschlüssel-Relation bezieht, müssen bei temporalen Relationen Beziehungen zwischen Tupelversionen und Tupelversionen-Mengen betrachtet werden. Eine Tupelversion der referenzierenden Relation verweist auf eine Tupelversionen-Menge der referenzierten Relation. Damit Fremdschlüssel-Beziehungen zwischen den temporalen Relationen unter Beachtung der referenziellen Integrität hergestellt werden können, müssen aus den Relationen der Tupelversionen-Mengen Relationen abgeleitet werden, die nur aus dem Tupelversionen-Mengen-Identifikator und der Lebensspanne der Tupelversionen-Menge bestehen (Myrach 1995; Thyssen 1995). Zwischen der Basisrelation und der abgeleiteten Relation wird die Beziehung über den Tupelversionen-Mengen-Identifikator gewährleistet. Damit eine vollständige Beziehungsintegrität erzielt werden kann, muß sichergestellt werden, daß die Lebensspanne der Tupelversionen-Menge tatsächlich die Vereinigung der Gültigkeitszeit-Perioden der nicht korrigierten Tupelversionen dieser Tupelversionen-Menge ist. Die Fremdschlüssel-Beziehung wird dementsprechend zwischen einer Tupelversion der referenzierenden Relation und dem abgeleiteten Tupel der referenzierten abgeleiteten Relation hergestellt (Abbildung 8).

Process							
P-ID	*R-ID*	*Surogat*	*gültig-von*	*gültig-bis*	*Erfaßt-am*	*Ersetzt-am*	*korrigiert-am*
P001	R1	1	1998-09-01	forever	1998-05-15	UC	
P001	R2	2	1998-09-01	forever	1998-05-31	UC	...
P001	R3	3	1998-09-01	forever	1998-06-21	UC	1998-07-02
P001	R3	6	1998-09-01	forever	1998-07-02	UC	
R001	R4	4	1998-09-01	forever	1998-06-30	UC	
P001	R5	5	1998-09-01	forever	1998-07-01	UC	

LS-BusinessRule		
R-ID	*Gültig-von*	*gültig-bis*
R1	1997-01-13	forever
R2	1997-12-01	forever
R3	1998-05-12	forever
R4	1998-06-30	forever
R5	1998-07-01	forever
...	...	...

Abbildung 8: Beziehung zwischen den temporalen Relationen

Die Beziehung zwischen den temporal erweiterten Beispiel-Relationen wird über die Attribute *R-ID* der Relationen *Process* (referenzierende Relation) und *LS-BusinessRule* (referenzierte Relation) hergestellt. Da diese Beziehung außerhalb der Gültigkeit der referenzierten Relation nicht zulässig ist, muß eine der folgenden temporalen Intervall-Beziehungen zwischen der referenzierenden Tupelversion und der referenzierten Tupelversionen-Menge bestehen (VT kennzeichnet hier die Periode der Gültigkeit) (vgl. Allen 1983):

- VT (Tupelversion(BusinessRule)) equal VT (Tupelversionen-Menge(Process))
- VT (Tupelversion(BusinessRule)) during VT (Tupelversionen-Menge(Process))
- VT (Tupelversion(BusinessRule)) starts VT (Tupelversionen-Menge(Process))
- VT (Tupelversion(BusinessRule)) finishes VT (Tupelversionen-Menge(Process))

Vereinfachend können diese Beziehungen so zusammengefaßt werden, daß die Gültigkeitszeit-Perioden der referenzierenden Tupelversionen von der Gültigkeitszeit der referenzierten Tupelversionen-Menge vollständig überdeckt werden müssen (contain property) (Myrach et al. 1996; Barnert/Schmutz 1995).

Ein weiterer Mechanismus, der bei der temporalen Erweiterung des Regel-Repository berücksichtigt werden muß, ist die Umsetzung der konzeptionellen Modelle in ihre Workflow-Spezifikation. Die Gültigkeitszeit der Relation *Process* muß deshalb Zugriff auf die Systemuhr haben. Ist der Wert der Systemuhr identisch mit dem Wert des Attributs *gültig-von*, wird die Umsetzung des modellierten Prozesses bzw. einzelner Komponenten in die Workflow-Spezifikation automatisch initiiert. Ist dagegen der Wert des Attributs *gültig-bis* identisch mit der Systemuhr, dann wird der Prozeß bzw. eine Komponente des Prozesses aus der Workflow-Spezifikation entfernt. Dieser Initiierungsmechanismus könnte z.B. über folgende Trigger-Logik in Datenbanken implementiert werden:

(1) ON gültig-von = current-date
DO transfer-to-workflow-specification
(2) ON gültig-bis = current-date
DO remove-from-workflow-specification.

5 Zusammenfassung und Ausblick

Durch ein temporales Regel-Repository kann die zeitliche Lücke zwischen der Modellierung von Geschäftsregeln und deren Umsetzung in eine Workflow-Spezifikation dahingehend geschlossen werden, daß modellierte Prozesse bzw. deren Komponenten mit dem Zeitpunkt ihrer geplanten Umsetzung gestempelt werden. In einem Unternehmen geplante Änderungen der Prozeßabläufe können somit im voraus modelliert werden und automatisch zum Zeitpunkt der geplanten Inkraftsetzung der neuen Strukturen in Workflow-Spezifikationen transformiert werden. Zudem können durch Verknüpfung einzelner Versionen eines Prozesses (bzw. seiner Komponenten) mit betriebswirtschaftlichen Kennzahlen weitere Erkenntnisse über die Effizienz der Abläufe im Unternehmen gewonnen werden. Aufgrund der Wichtigkeit der zeitlichen Bezüge - nicht nur in der Modellierung, sondern auch während der Ausführung von Workflows - sollten temporale Ansätze in zukünftigen Forschungsvorhaben stärkere Beachtung finden. Anregungen dazu könnten beispielsweise aus den Bereichen Produktionsplanung und -steuerung oder aus der Projektplanung bezogen werden.

Literaturverzeichnis

Allen, J.F. (1983): Maintaining Knowledge about Temporal Intervals. In: Communications of the ACM 26 (1983) 11, S. 832 - 843.

Barnert R./Schmutz, G. (1995): Die zeitbezogene Datenhaltung in einer Oracle-Umgebung bei den Schweizer Regionalbanken. In: Berücksichtigung zeitbezogener Daten bei der Realisierung betrieblicher Informationssysteme, 1. Workshop des Arbeitskreises Zeitorientierte betriebliche Informationssysteme (ZobIS) der Gesellschaft für Informatik (GI), Düsseldorf 1995.

Barnert, R./Schmutz, G. (1997): Die zeitbezogene Datenhaltung bei den Schweizer Regionalbanken. In: Wirtschaftsinformatik 39 (1997) 1, S. 45 - 53.

Ben-Zvi, J. (1982): The Time Relational Model, Ph.D. Diss., University of California, Los Angeles 1982.

Clifford, J./Croker, A. (1993): The Historical Relational Data Model /HRDM) Revisited. In: A.U. Tansel/J. Clifford et al. (Hrsg.): Temporal Databases:

Theory, Design and Implementation, Redwood City et al. 1993, S. 6 - 27.

Clifford, J. et al. (1994): On the Semantics of ”Now” in Temporal Databases, Working Paper R-94-2047, Institute for Electronic Systems, Aalborg University 1994.

Dayal, U. (1988): Active Database Management Systems, in: C. Beeri/J.W. Schmidt/U. Dayal (Hrsg.), Proceedings of the 3rd International Conference on Data and Knowledge Bases, San Matheo: Morgan Kaufmann 1988, S. 150 - 169.

Endl, R./Knolmayer, G./Pfahrer, M. (1998): Modeling Processes and Workflows by Business Rules, 1st European Workshop on Workflow and Process Management (WPM'98), Zürich 1998, (http://www.inf.ethz.ch/department/IS/iks/Other/wpm98/program.html)

Galler, J. (1997): Vom Geschäftsprozeßmodell zum Workflow-Modell, Wiesbaden 1997.

Herbst, H. (1997): Business Rule-Oriented Conceptual Modeling, Heidelberg 1997.

Herbst, H./Knolmayer, G. (1995): Ansätze zur Klassifikation von Geschäftsregeln, in: Wirtschaftsinformatik, 37 (1995) 2, S. 149 - 159.

Herbst, H./Myrach, T. (1997): A Repository System for Business Rules, in: R. Meersman/L. Mark (Hrsg.), Database Application Semantics, London: Chapman & Hall 1997, S. 119 - 138.

Jasper, H./Zukunft, O./Behrends, H. (1996): Time Issues in Advanced Workflow Management Applications of Active Databases. In: M. Berndtsson/J. Hansson (Hrgs.): Active and Real-Time Database Systems (ARTDB-95), Proceedings of the First International Workshop on Active and Real-Time Database Systems (Skövde, Sweden), Berlin et al.: Springer-Verlag 1996.

Jensen, C.S./Clifford, J./Gadia, S./Segev, A./Snodgrass, R. (1994): A Consensus Glossary of Temporal Database Concepts. In: SIGMOD Record 23 (1994) 1, S. 52 - 64.

Joosten, S./Aussems, G./Duitshof, M./Huffmeijer, R./Mulder, E. (1994): Wa-12: an empirical study about the practice of workflow management, Technical Report, University of Twente 1994.

Keller, G./Nüttgens, M./Scheer, A.-W. (1992): Semantische Prozeßmodellierung auf der Grundlage "Ereignisgesteuerter Prozeßketten (EPK), Institut für Wirtschaftsinformatik (IWi), Universität des Saarlandes, Saarbrücken 1992.

Knolmayer, G./Endl, R./Pfahrer, M./Schlesinger, M. (1997): Geschäftsregeln als Instrument zur Modellierung von Geschäftsprozessen und Workflows, Swordies-Report Nr. 97-8, Bern 1997.

Knolmayer, G. (1998): A Business Rules Layer Between Process and Workflow Modelling, ECOOP'98 Workshop on Tools and Environments for Business Rules, Brussels 1998.

Kradolfer, M./Geppert, A. (1998): Dynamic Workflow Schema Evolution based on Workflow Type Versioning and Workflow Migration. Technical Report 98.02, Department of Computer Science, University of Zurich 1998.

Lorentzos, N.A./Mitsopoulos, Y.G. (1997): SQL Extension for Interval Data. In: IEEE Transactions on Knowledge and Data Engineering 9 (1997) 3, S. 480 – 499.

Myrach, T. (1995): Die Schlüsselproblematik bei der Umsetzung temporaler Konzepte in das relationale Datenmodell. In: Berücksichtigung zeitbezogener Daten bei der Realisierung betrieblicher Informationssysteme, 1. Workshop des Arbeitskreises Zeitorientierte betriebliche Informationssysteme (ZobIS) der Gesellschaft für Informatik (GI), Düsseldorf, 1995, S. 13 – 15.

Myrach, T. (1997): Realisierung zeitbezogener Datenbanken: Ein Vergleich des herkömmlichen relationalen Datenmodells mit einer temporalen Erweiterung. In: Wirtschaftsinformatik 39 (1997) 1, S. 35 - 44.

Myrach, T./Knolmayer, G./Barnert, R. (1996): On Ensuring Keys and Referential Integrity in the Temporal Database Language TSQL2. In: Databases and Information Systems, Proceedings of the 2nd International Baltic Workshop, Vol. 1: Research Track, Tallinn 1996, S. 171 - 181.

Scheer, A.-W. (1994): Wirtschaftsinformatik - Referenzmodelle für industrielle Geschäftsprozesse, 4. Auflage, Berlin et al. 1994.

Snodgrass, R. (1990): Temporal Databases Status and Research Directions. In: SIGMOD Record 19 (1990) 4, S. 83 - 89.

Snodgrass, R. (Hrsg.) (1995): The TSQL2 Temporal Query Language. Boston et al. 1995.

Soo, M.D./Snodgrass, R. (1995): Temporal Data Types. In: R. Snodgrass (Hrsg.): The TSQL2 Temporal Query Language, Boston et al. 1995, S. 123 - 152.

Tansel, A.U. et al. (Hrsg.) (1993): Temporal Databases: Theory, Design and Implementation. Redwood City et al. 1993.

Thyssen, U.M. (1995): Ein Bewirtschaftungssystem für bitemporale Tabellen. In: Berücksichtigung zeitbezogener Daten bei der Realisierung betrieblicher Informationssysteme, 1. Workshop des Arbeitskreises Zeitorientierte betriebliche Informationssysteme (ZobIS) der Gesellschaft für Informatik (GI), Düsseldorf, 1995.