Formales System und Vorgehensweise zur Modellierung von Geschäftsprozessen

Georg V. Zemanek
USU Softwarehaus
Unternehmensberatung GmbH
Spitalhof
71693 Möglingen

Hans W. Nissen
Informatik V
RWTH Aachen
Ahornstrasse 55
52074 Aachen

Zusammenfassung

Der vorliegende Beitrag beschäftigt sich mit den frühen Phasen des Requirements Engineering. Diese Phasen sind durch die Erhebung, Analyse und Konsolidierung unterschiedlicher Sichtweisen und Perspektiven auf den Problembereich gekennzeichnet. Es wird ein formales Verfahren entwickelt, welches die Repräsentation und die Analyse heterogener Perspektiven unterstützt. Die Grundlage dieser Entwicklung bildet die von der Firma USU verwendete Analysemethode PFR. Zur Repräsentation der Perspektiven wird ein Metamodell entwickelt, das über ein Höchstmaß an Überlappungen eine Vielzahl von potentiellen Konflikten zwischen den Perspektiven forciert. Konflikte werden in dieser Phase der Anforderungsanalyse als sehr hilfreich auf dem Weg zu einer gemeinsamen Vision für das zu entwerfende Informationssystem angesehen. Zudem bietet das Verfahren eine flexible Methode zur Erkennung, Analyse und Resolution von Konflikten. Das Verfahren ist in der Wissensrepräsentationssprache Telos realisiert und in der deduktiven Objektbank ConceptBase implementiert. Es hat sich in mehreren kommerziellen Projekten der Firma USU bewährt.

1 Einleitung

Anforderungen an Informationssysteme werden von vielen verschiedenen Standpunkten/Perspektiven/Sichtweisen aus erhoben. Diese Perspektiven unterscheiden sich in der Art der Erhebung, in dem gewählten Schwerpunkt der Betrachtung, als auch in den verwendeten Notationen. Gleichzeitig überlappen diese Perspektiven in ihren Inhalten und enthalten somit potentielle Konflikte. Gerade in den frühen Phasen des Requirements Engineering, in denen das Verstehen und die Herstellung einer von aller Beteiligten unterstützte Vision im Vordergrund stehen, sind Konflikte sehr produktiv und sollten nicht unterdrückt, sondern eher forciert werden.

Während eine Vielzahl von Methoden für die Erhebung dieser Perspektiven existieren, ist eine Werkzeug-Unterstützung für die weitere Verarbeitung und Aufbereitung der Informationen nur bedingt verfügbar. Vorhandene Werkzeuge enthalten in der Regel ein fest vorgegebenes Weltmodell und beeinflussen damit gewollt oder ungewollt die Analyse. Sie erweisen sich oft als zu unflexibel, da sie auf ganz bestimmte Einsatzbereiche und

Methoden ausgerichtet sind. Abfragemöglichkeiten bestehen lediglich für die vom Hersteller vorgesehenen Einsatzbereiche. Eine Anpassung auf sich ändernde Anforderungen ist nicht möglich. Aber gerade in der Analysephase wird man mit sich ständig ändernden Randbedingungen konfrontiert: Kunden wollen nur eine ihnen bereits vertraute Notation verwenden oder Ergebnisse der Anwendung einer eigenen Methodik einbeziehen.

Aufbauend auf der Analysemethode PFR für diese frühe Phase wird in diesem Beitrag eine formale Unterstützung zur Erkennung von Konflikten zwischen unterschiedlichen Perspektiven vorgestellt. Dieses umfaßt eine spezielle Methode zur Repräsentation von Perspektiven, die eine einfache Erkennung von Konflikten ermöglicht. Die Methode basiert auf einem speziellen Metamodell, das einen hohen Grad von Redundanz enthält. Die flexible Handhabung von Konflikten wird durch eine Kombination von Regeln, Konsistenzbedingungen und Anfragen ermöglicht.

Da die PFR Methodik von der Firma USU während des Requirements Engineering hauptsächlich zur Analyse der Geschäftsprozesse einer Unternehmung eingesetzt wird, stellt diese Anwendung ebenfalls den Rahmen dieses Beitrags dar. Ziel dieser Analyse ist die Generierung von Anforderungen an ein den Prozeß unterstützendes Informationssystem. Realisiert ist der Ansatz mit Hilfe der Objektbank ConceptBase und der Wissensrepräsentationsprache Telos.

Das folgende Kapitel führt in den Aufbau und Ablauf der PFR Methode ein, während Kapitel 3 sich mit den Eigenschaften der Sprache Telos beschäftigt. In Kapitel 4 wird über die formale Repräsentation und Analyse der PFR Ergebnisse berichtet. Der Beitrag endet mit einer abschließenden Betrachtung des vorgestellten Verfahrens.

2 Die PFR Methodik

USU, ein mittelständisches Softwarehaus und Unternehmensberatung, organisiert die Analyse von Geschäftsprozessen entsprechend der PFR Methodik (Analysis of *P*resence and *F*uture *R*equirements) [1]. Sie bedient sich vieler, sehr unterschiedlicher Informationsquellen und umfaßt drei Phasen:

In der ersten Phase wird ein ein- bis zweitägiger Workshop unter Anwenderbeteiligung abgehalten mit dem Ziel, zunächst alle am Geschäftsprozeß beteiligten Unternehmenseinheiten zu identifizieren und ihre Interaktionen zu beschreiben. Dabei beschränkt man sich auf den Austausch von Dokumenten oder anderen Medien. Die Workshop-Teilnehmer erstellen selbst eine graphische Darstellung der ihnen bekannten Prozesse. Vom Moderator vorgegeben werden nur prototypische Basis-Elemente (Bildsymbole) zur Repräsentation von Akteuren und Pfeile zur Repräsentation der Interaktionen. Auf diese Weise entsteht ein grober Beziehungsgraph vom IST-Zustand des zu modellierenden Geschäftsprozesses, der weder vollständig noch widerspruchsfrei ist, aber mit Anmerkungen über die Schwachstellen der derzeitigen Situation versehen wurde. Nach kurzer Retrospektive auf die Geschäftsziele des Unternehmens wird dann ein hypothetischer SOLL-Zustand entwickelt, aus dem sich für die Anwender - aber auch für das Management - Veränderungspotential für die Geschäftsprozesse ableiten läßt.

In der zweiten Phase wird in parallelen Interviews eine detaillierte Ablaufanalyse mit Angehörigen der oben identifizierten Unternehmenseinheiten durchgeführt. Es wird beschrieben, mit welchen Daten die Aktionen versorgt werden müssen und welche neuen

Daten als Ergebnis einer Aktivität entstehen. Bei dieser sehr detaillierten Befragung treten häufig ungewollte Konflikte zu den in Phase 1 erhobenen übersichtsartigen Perspektiven auf. Parallel zu den Interviews werden die im Geschäftsprozeß fließenden Dokumente untersucht: Die auf den Dokumenten enthaltenen Informationseinheiten, auf die im Prozeß zugegeriffen wird, werden identifiziert. Diese strukturelle Analyse ermöglicht eine spätere Optimierung der Formularzusammenstellung.

Die dritte Phase umfaßt die Analyse der ermittelten Informationen zur Aufdeckung von Inkonsistenzen, Widersprüchen und Unvollständigkeiten. Diese erkannten Konflikte führen dann zu zielgerichteten weiteren Befragungen und Erhebungen. Das Ziel ist eine konsolodierte Menge von Perspektiven, die als Grundlage für die abschließenden Anforderungsdokumente dienen.

Die Erfahrung von USU hat gezeigt, daß eine Analyse dieser Perspektiven von Hand sowohl zeitaufwendig, als auch fehleranfällig und unvollständig ist. Ein unterstützendes Werkzeug muß die oft informell dargestellten Perspektiven in einer "natürlichen" Art und Weise repräsentieren, so daß einerseits die Aussagekraft wiedergegeben wird aber andererseits der Formalismus immer noch einfach gehalten ist. Weiterhin soll es den Vergleich der repräsentierten Perspektiven ermöglichen, um Widersprüche und "Lücken" aufzudecken.

3 Die Modellierungssprache Telos

ConceptBase [2] ist eine deduktive Objektbank, die auf der Wissensrepräsentationssprache Telos [3] basiert. Telos bietet für die hier betrachtete Anwendung zwei besondere Eigenschaften:

- Es enthält ein sehr einfach gehaltenes Kernmodell, welches leicht auf spezielle Anwendugsbedürfnisse angepaßt bzw. erweitert werden kann. Im Gegensatz zu vielen anderen Datenmodellen enthält es kein fest vorgegebenes Modell der Welt.
- Da logische Formeln als Teil eines Modells definiert werden können, erlaubt Telos die formale Beschreibung von Zusammenhänge zwischen verschiedenen Perspektiven. Desweiteren erlauben Anfrageklassen eine flexible Handhabung dieser Zusammenhänge.

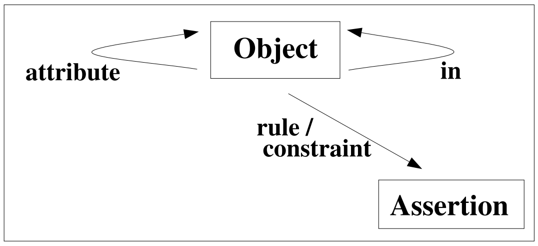

Abbildung 1: Der Kern von Telos

Das Kernmodell von Telos ist in Abbildung 1 in Form eines semantischen Netzes dargestellt. Es basiert auf der einfachsten vorstellbaren graphischen Notation: Knoten und Kanten. Es werden zwei Typen von eigenständigen Konzepten, dargestellt als Knoten, und zwei Typen von Beziehungen, dargestellt als Kanten, unterstützt:

- `Assertion` bezeichnet logische Formeln und erlaubt die Definition von Regeln, Integritätsbedingungen und Anfragen.
- Der Knoten `Object` dient zur Darstellung aller weiteren eigenständigen Entitäten der realen Welt.
- Beliebige Beziehungen zwischen Objekten ermöglicht die Kante `attribute`. Das Ziel eines Attributs kann wiederum ein Knoten oder eine Kante sein.
- Die Kante `in` beschreibt die Instanzierungsbeziehung von Objekten mit einer gemeinsamen Struktur zu einem abstrakteren Objekt (genannt Klasse), welches den gemeinsamen Typ und gemeinsame Eigenschaften beschreibt. Diese Kante realisiert das Abstraktionsprinzip der Klassifikation. In Telos ist die Klassifikationshierarchie nicht beschränkt, d.h. Klassen können wiederum (Meta-) Klassen instanzieren, die ihrerseits (Metameta-) Klassen instanzieren usw.

Von diesem Kern aus können alle weiteren Eigenschaften der Sprache, wie z.B. die Spezialisierungsbeziehung, in einer Art Bootstrapping Verfahren hergeleitet werden.

Die Anpassung von Telos auf bestimmte Notationen ist nicht nur syntaktisch, sondern auch semantisch möglich. Die Fähigkeit zur Spezifikation von Metamodellen erlaubt die Erzeugung eines konzeptuellen Modells der betrachteten Notation. Dieses Modell beschreibt die vorhandenen Sprachelemente und ihre syntaktisch korrekte Verwendung. Es können nun Modelle entsprechend dieser Notation als Instanz des Metamodells erstellt werden; Die Instanzierungseigenschaften garantieren ein entsprechend des Metamodells strukturell korrektes Modell. Semantische Eigenschaften der Sprachelemente, wie regelhafte Abhängigkeiten oder Konsistenzbedingungen, können so allerdings noch nicht ausgedrückt werden. Viele existierende Metamodellierungsansätze [4, 5] beschränken sich auf diese syntaktische Erklärung einer Notation. Dieses reicht jedoch nicht aus, um das dynamische Verhalten der Sprachkonstrukte zu erfassen. Das Konzept des Formel-Objekts (Assertion) erlaubt es nun, diese Bedingungen unmittelbar an die entsprechenden Objekte des Metamodells zu binden. Diese Fähigkeiten bilden das Grundgerüst für die formale Repräsentation und Konfliktanalyse unterschiedlicher Perspektiven.

4 Repräsentation und Analyse von Geschäftsprozessen

Das in Abbildung 2 dargestellte Metamodell dient zur Repräsentation und struktureller Integration der durch die PFR Methodik erzeugten Perspektiven auf die Geschäftsprozesse eines Unternehmens.

Das Meta-Modell kommt mit vier Objekten aus: `Akteure`, `Träger`, `Daten` und `Aktionen`. Zwischen diesen Objekten bestehen jeweils mehrfache Beziehungen: Akteure erzeugen und brauchen Datenträger, geben sie untereinander weiter; Aktionen nehmen Datenträger als Input und geben sie als Output aus, Aktionen erzeugen Daten, die auf Datenträger geschrieben werden müssen. Zwischen den Aktionen ist eine Abfolge-Relation definiert.

In Abbildung 3 sind die Repräsentationen der Perspektiven innerhalb des Metamodells dargestellt:

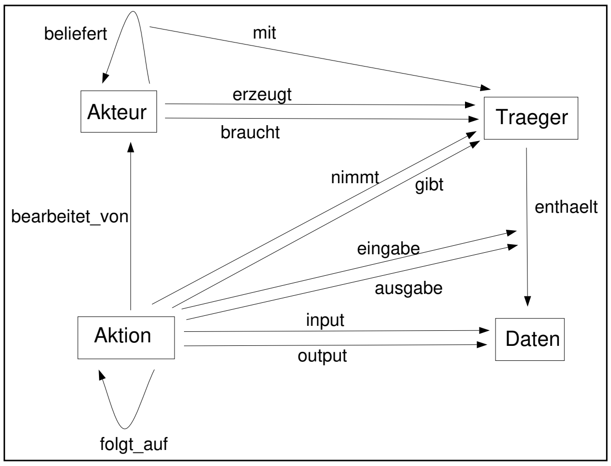

Abbildung 2: Meta-Modell zur Modellierung von Geschäftsprozessen

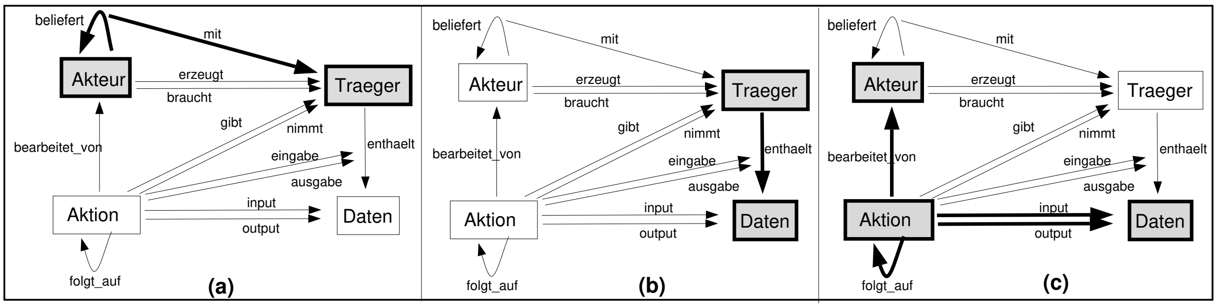

Abbildung 3: Die Repräsentation der Perspektiven

1. Der Teil (a) hebt den Teil des Metamodells hervor, der zur Repräsentation des *Dokumentenaustausches zwischen Unternehmensseinheiten* verwendet wird. Diese Perspektive ist in der ersten Phase erhoben worden. Eine Unternehmenseinheit wird als ein abstrakter `Akteur` dargestellt, der andere Akteure `mit` einem `Träger` `beliefert`. Ein `Träger` kann sowohl etwas persistent wie ein Blatt Papier oder eine Diskette sein, als auch etwas flüchtiges wie die Stimme, die Wörter transportiert.
2. Das Modell in Teil (b) repräsentiert den *strukturellen Aufbau der Dokumente* durch die explizite Unterscheidung zwischen einem `Träger` und den `Daten`, die er `enthält`. Diese Unterscheidung ist erforderlich, um über Phänomene wie leere und gefüllte Formulare, Lesen von und Schreiben auf einen Träger, Übertragung einer Information von einen Träger auf einen anderen und Akteure, die einen Träger erhalten, aber die enthaltenen Daten nie benutzen, zu sprechen
3. Der Teil (c) stellt das Modell für die *detaillierte Aktivitätsbeschreibungen der einzelnen Beteiligten* dar. Eine `Aktion` wird von einem `Akteur bearbeitet`. Die `folgt_auf` Relation beschreibt eine partielle Ordnung auf den Aktionen (Workflow). Eine `Aktion` ist eine atomare Aktivität, die existierende `Daten` als Eingabe nimmt (`input`) und neue Daten generiert (`output`). Daten können nur einmal durch `output` erzeugt werden, danach gehen sie nicht mehr verloren.

Das Metamodell deckt nicht nur alle Notationen des gegebenen Anwendungsgebietes ab, sondern ist derart aufgebaut, daß jedes Element des Metamodells in mindestens zwei unterschiedlichen Perspektiven enthalten ist. Mit anderen Worten, wir haben versucht, ein Höchstmaß an Überlappungen und somit an potentiellen Konflikten zu erzeugen. Besonderen Wert ist dabei auf die Ausgewogenheit zwischen mehrfach zu erstellender redundanter Information (d.h. einer Erhöhung des Modellieraufwandes) und dem daraus resultierenden Nutzen für die Modellanalyse gelegt worden.

Formal gesehen äußern sich Konflikte als Unvollständigkeiten und Widersprüche zwischen Perspektiven. Voraussetzung für die Möglichkeit eines Konflikts ist die Existenz von Beziehungen der Sichten untereinander. Überlappende Repräsentationen der Perspektiven machen Beziehungen zwischen einzelnen Perspektiven explizit. Diese Beziehungen "konnen dann als Integritätsbedingungen und deduktive Regeln formalisiert werden. Sie ermöglichen das formale Erkennen von Unvollständigkeiten, Inkonsistenzen und Verständnisfehlern.

Für das Zusammenspiel der obigen Perspektiven haben wir über 70 Konsistenzbedingungen formuliert. Diese definieren sowohl die Konsistenz einzelner Perspektiven, als auch die Konsistenz zwischen mehreren Sichten. In der praktischen Anwendung hat sich jedoch die Verwendung von Integritätsbedingungen als hinderlich erwiesen: Tritt bei der Eingabe einer Perspektive eine Verletzung auf, wird also ein potentieller Konflikt erkannt, so weist das System die Eingabe zurück. Dieses ist das übliche Verhalten einer Datenbank bei der Verletzung einer Integritätsbedingung. Dieses verlangt nun aber von dem Analytiker, zunächst den Konflikt zu beheben, und dann erst die Perspektive einzutragen. Dieses steht im krassen Gegensatz zum beabsichtigten Einsatz des Werkzeugs, welches alle Perspektiven zunächst aufnehmen und dann diese Informationen auf Konflikte hin untersuchen soll. Aus diesem Grund sind die Konsistenzbedingungen nicht als Integritätsbedingungen, sondern als Anfragen in ConceptBase realisiert worden. Anfragen werden in Telos durch Klassen repräsentiert, deren (virtuelle) Instanzen genau die Antworten auf die Anfrage darstellen [6]. Die Information wird nun so inkonsistent, wie sie von den Anwendern kommt, in das System eingegeben und erst in einem späteren Schritt analysiert und korrigiert. Zur Unterstützung eines Analytikers haben wir für jede Anfrage eine Menge von möglichen Interpretationen vorbereitet. So kann die Analysemethode auch von Personen durchgeführt werden, die sich selbst nicht mit der Formulierung von Anfragen auskennen.

Eine Anfrageklasse hat die folgende Form:

```
QueryClass <name> isA <superklassen> with
  attribute
    <antwort attribute>
  constraint
    <bedingung>
end
```

Sie enthält vier wichtige Teile:

1. Der Name der Anfrageklasse ist `<name>`.
2. Der `<superklassen>` Teil spezifiziert die Superklassen der Anfrageklasse. Es werden zur Ermittlung der Antworten der Anfrage nur die gemeinsamen Instanzen aller Superklassen herangezogen.

3. Der `<antwort attribute>` Teil definiert die Attribute der Antworten. Diese können sowohl von den Superklassen vererbt, als auch während der Anfrageauswertung neu berechnet werden.
4. Der `<bedingung>` Teil enthält eine beliebige geschlossene Formel, die von allen Antworten erfüllt werden muß.

Die folgende Anfrage realisiert eine Konsistenzbedingung für die Perspektive der detaillierten *Aktivitätsbeschreibungen*: Daten müssen erst als `output` einer Aktion auftreten, bevor sie als `input` von einer anderen Aktion verwendet werden können.

```
QueryClass Daten_BenutztBevorProduziert isA Daten with
  attribute
     benutzer : Aktion
  constraint
     c : $ (benutzer input this) and
           (produzent output this) and
           (produzent trans_folgt_auf this) $
end
```

The Anfrage berechnet alle Daten, die von einer Aktion (als Rückgabeparameter `benutzer`) benutzt werden, bevor sie produziert wurden. Die Antworten können unterschiedlich interpretiert werden:

1. als Eingabefehler: der Interviewte hat eine falsche Aktionenfolge angegeben.
2. als Eingabefehler: die `folgt_auf` Kante hat die falsche Richtung.
3. als Abstimmungsproblem: zwei Interviewte benutzen das gleiche Datenobjekt für unterschiedliche Informationen.
4. als durch unser Metamodell nicht ausdrückbaren Fall: Die Daten werden durch die `output` Beziehung nicht erzeugt, sondern verändert, d.h. es liegt eine fehlerhafte Interpretation der `output` Kante als Modifikation vor. Die Modifikation von Daten kann durch unser Modell nicht ausgedrückt werden.

Zusätzlich zu Anfragen, die die Konsolidierung der Perspektiven unterstützen, haben wir Anfragen formuliert, die den modellierten Geschäftsprozeß hinsichtlich seiner Eigenschaften untersuchen. Insgesamt entstanden so über 80 Anfrageklassen für diese Anwendung.

Als Beispiel betrachten wir den Dokumentenaustausch innerhalb des modellierten Prozesses. Die folgende Anfrage untersucht, ob sich der vorliegende Dokumentenaustausch optimieren läßt. Dazu ermittelt die Anfrage alle Akteure, die ein Formular erhalten, welches aber nur Daten enthält, die ihm bereits durch andere Formulare bereitgestellt werden.

```
QueryClass MultipleDataSupply isA Agent with
  attribute
    superfluous : Medium
  constraint
     c : $ (supply in Agent!supplies) and (supply to this) and
           (supply with superfluous) and
```

```
            forall data (superfluous contains data)
            exists other_medium (other_medium <> superfluous) and
             other_medium contains data) $
end
```

Antworten können folgendermaßen interpretiert werden:

1. Die Belieferung mit dem ermittelten Formular ist überflüssig, da alle enthaltenen Daten bereits durch andere Formulare geliefert werden.
2. Der Akteur nimmt einen Vergleich der Daten auf verschiedenen Formularen vor.
3. Es wird die gleiche Aktion auf den gleichen Daten auf unterschiedlichen Formularen ausgeführt. Dieses kann durch unser Modell nicht explizit ausgedrückt werden.

5 Abschließende Bemerkungen

Wir haben in diesem Beitrag ein Verfahren zur formalen Unterstützung der Analysemethode PFR vorgestellt. Das Verfahren ist in der Wissensrepräsentationssprache Telos realisiert und in ConceptBase implementiert. Die wichtigsten Merkmale dieses Ansatzes sind ein Metamodell, welches potentielle Konflikte zwischen Perspektiven durch eine sich stark überlappende Repräsentation der Perspektiven forciert, und eine flexible Analysemöglichkeit der erhobenen Perspektiven durch die Verwendung von Anfrageklassen anstatt von Integritätsbedingungen.

Die in diesem Beitag vorgestellte Analyseumgebung ist bei der Firma USU bereits in mehreren kommerziellen Projekten erfolgreich eingesetzt worden [7]. Das Metamodell hat sich nach einer relativ kurzen Evolutionszeit durch die Projekterfahrungen stabilisiert. Eine Anpassung an besondere Kundenwünsche ist jedoch weiterhin möglich. Bei dem Entwurf des Metamodells wurde bewußt vermieden, alle denkbaren Phänomene der realen Welt darzustellen, wie z.B. die Organisationsstruktur des betrachteten Unternehmens durch eine hierarchische Beziehung über `Akteur`. Daraus ergeben sich einige Unzulänglichkeiten in der Modellierung, aber der Vorteil des kleinen und leicht zu verstehenden Metamodells wiegt diese wieder auf.

Literatur

[1] P. Abel. Kurzbeschreibung der Analysemethode USU-PFR. Technical report, USU GmbH, Möglingen, 1995.

[2] M. Jarke, R. Gallersdörfer, M.A. Jeusfeld, M. Staudt, and S. Eherer. ConceptBase - a deductive object base for meta data management. *Journal of Intelligent Information Systems*, 4(2):167–192, March 1995.

[3] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: Representing knowledge about information systems. *ACM Transactions on Information Systems*, 8(4):325–362, October 1990.

[4] M. Saeki, K. Iguchi, K. Wen-yin, and M. Shinohara. A meta-model for representing software specification & design methods. In N. Prakash, C. Rolland, and B. Pernici, editors, *Proc. of the IFIP WG8.1 Working Conference on Information System Development Process*, Como, 1993.

[5] K. Smolander, K. Lyytinen, V.-P. Tahvanainen, and P. Marttiin. MetaEdit - a flexible graphical environment for methodology modelling. In R. Andersen, J.A. Bubenko, and A. Solvberg, editors, *Proc. of 3rd Intl. Conf. on Advanced Information Systems Engineering (CAiSE'91)*, Trondheim, Norway, May 1991.

[6] M. Staudt, H.W. Nissen, and M.A. Jeusfeld. Query by class, rule and concept. *Applied Intelligence, Special Issue on Knowledge Base Management*, 1994.

[7] G.V. Zemanek. Projektergebnisse des USU-Projekts ConceptBase. Technical report, USU GmbH, Möglingen, 1995.