Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

**Document**
D-97-07

# PIMaS: Ein objektorientiert-regelbasiertes System zur Produkt-Prozeß-Transformation

Harold Boley,
Bernd Bachmann, Christian Blum, Christian Embacher,
Andreas Lorenz, Jamel Zakraoui

**October 1997**

**Deutsches Forschungszentrum für Künstliche Intelligenz GmbH**

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341

# PIMaS:
# Ein objektorientiert-regelbasiertes System zur Produkt-Prozeß-Transformation

Harold Boley,
Bernd Bachmann, Christian Blum, Christian Embacher,
Andreas Lorenz, Jamel Zakraoui

DFKI-D-97-07

This work has been supported within the project PRODUKTION 2000, GIPP-S1 by a grant from the German Federal Ministry of Education, Science, Research and Technology (FKZ 02PV41092).


*ISSN 0946-0098*

# PIMaS: Ein objektorientiert-regelbasiertes System zur Produkt-Prozeß-Transformation

Harold Boley
Bernd Bachmann
Christian Blum
Christian Embacher
Andreas Lorenz
Jamel Zakraoui

30. Oktober 1997

**Abstract**

PIMaS[1] permits an interactive object-oriented modelling of product data, which are then automatically transformed into production processes by if-then rules. The SmallTalk-implemented prototype is exemplified via products from the automobile and water-purification sectors.

[1]Process Information Management System. Diese Arbeit wurde innerhalb des Rahmenkonzeptes PRODUKTION 2000, GIPP-S1 mit Mitteln des BMBF unter dem Förderkennzeichen 02PV41092 gefördert.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# 1 Einleitung

Das Ziel von PIMaS ist, Transformationen von hierarchischen Produktdaten auf ihre Abwicklungsprozesse zu ermöglichen. Jede Transformation erfolgt durch eine Auswahl von Instanzen von Produktklassen, von denen ein neu zu definierender Prozeß abhängt. Dies hat zur Folge, daß jede auftretende Änderung bei den Produktdaten dem Benutzer sichtbar wird. Solche transparente Änderbarkeit ermöglicht eine Modellierung zur interaktiven Optimierung von Zeit und Kosten bei der Abwicklung von Produkten. Somit ist PIMaS ein Unterstützungssystem zur effizienten Produkt-Prozeß-Transformation. Es ist in SmallTalk [TV94] (VisualWorks 2.5 [HH95]) implementiert. Das System wird im folgenden anhand eines Beispiels aus dem Automobilbereich detailliert erklärt und in einer Fallstudie über die Abwicklung von Trinkwasseraufbereitungsanlagen (TWAs) praxisnah benutzt. Für einen ersten PIMaS-Überblick anhand des TWA-Beispiels siehe Anhang A.

# 2 Die Produktbeschreibung

## 2.1 Abstraktionsebenen

Im Sinne des objekt-orientierten Modellierens [RBP+94] und der wiederverwendbaren ontologischen Wissensrepräsentation [Bac97] wird das Produkt in PIMaS auf drei Abstraktionsebenen beschrieben.

- **Das Metamodell**
  Das Metamodell besteht aus generischen, fest definierten Klassen mit Attributen und ist für alle Produktarten gleich.

  1. Produkt: seriennummer name (müssen innerhalb eines Modells eindeutig sein)
  2. Komponente
  3. Baugruppe: hatTeile
  4. Typ: hatTeile

- **Das Produktmodell**
  Das Produktmodell besteht aus produktspezifischen, frei definierbaren Klassen und Attributen (z.B. Auto, Motor, Trinkwasseraufbereitungsanlage, Filteranlage...)

- **Die Produktdaten**
  Die Produktdaten sind Instanziierungen des Produktmodells durch Spezifizierung von Attributwerten.

Die vollständige Syntax zur Definition der SmallTalk-Pseudoklassen für das Meta-Produktmodell und die Produktdaten ist in Anhang D zu finden und wird durch das Auto-Beispiel (Anhang B) erläutert. In den nächsten Abschnitten wird die Vorgehensweise bei Bearbeitung und Laden der Produktdaten beschrieben. Es werden im wesentlichen die Funktionalitäten der Benutzungsoberfläche erläutert, um dem Leser einen ersten Eindruck von der Arbeitsweise und dem Umgang mit dem System zu vermitteln.

## 2.2 Einfilen der Produktdaten

Beim Anklicken des *PIMaS-Symbols* erscheint das in Abbildung 1 gezeigte Fenster. Dieses Fenster wird benötigt, um den Pfad, wo die Daten gespeichert sind, festzulegen. Hier besteht, die Möglichkeit, in dem Dateisystem zu navigieren bis der gewünchte Pfad gefunden ist. Nachdem der Pfad gesetzt ist, erscheinen zwei

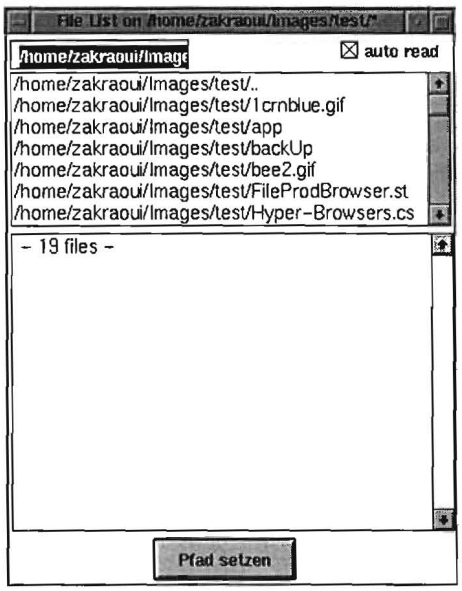



Abbildung 1: *Pfad setzen*

weitere Fenster mit den Titeln *Produkt Beschreibung* und *Prozeß Hierarchie* wie in Abbildung 2 gezeigt. Diese zwei Fenster sind miteinander gekoppelt d.h. beide Fenster werden automatisch gemeinsam geschlossen.

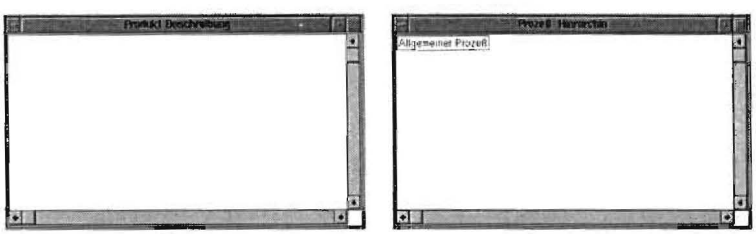


Abbildung 2: *Produkt Beschreibung* und *Prozeß Hierarchie* Fenster

In dem *Produkt Beschreibung*-Fenster werden die Klassen-Hierarchie des Produktmodells und ihre Instanzen dargestellt. Dies geschieht in zwei Schritten:

1. Auswahl des Menü-Items *Produkte laden* im *Produkt Beschreibung*-Fenster, worauf ein File-Browser mit den Produktdaten-Files erscheint:

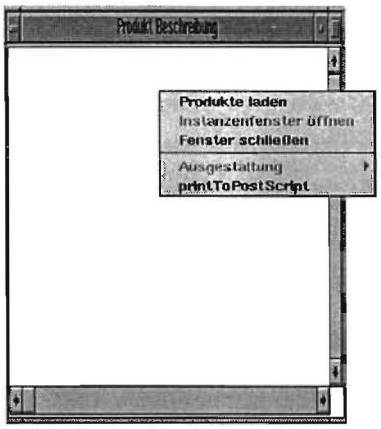



Abbildung 3: *Produkte laden (1)*

2. Auswahl des Menü-Items *Produkte laden* im FileBrowser:


Abbildung 4: *Produkte laden (2)*

Nach dem Selektieren dieses Items werden die notwendigen Files geparst (durch Instanziierung der jeweiligen Klassen und Bindung der Attributwerte) und eine Produkt-Hierarchie wie in Abbildung 5 erzeugt.

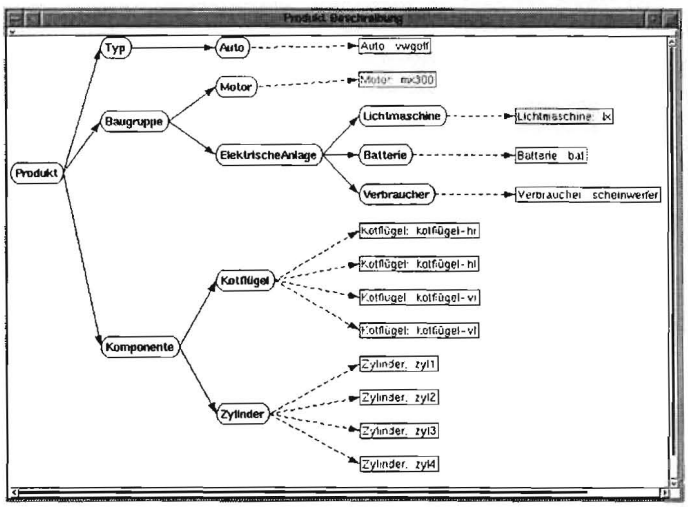



Abbildung 5: *Produkt-Hierarchie*

Die Darstellung der Produktdaten in Abbildung 5 ermöglicht auch eine Sicht auf die vorhandenen Attribute und Attributwerte der Klassen bzw. Instanzen. Dies geschieht, indem man eine bestimmte Klasse oder Instanz selektiert und das Item *Klasse/Instanz zeigen* auswählt. Die Attribute werden dann in einem Fenster dargestellt (Abbildung 6).

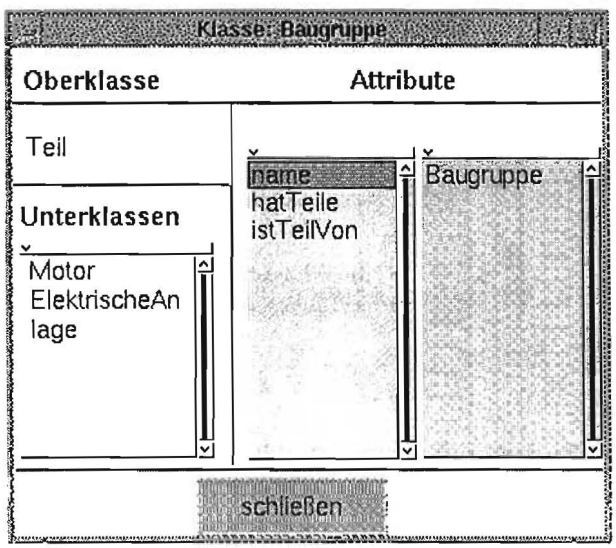



Abbildung 6: *Klassen/Instanzen-Inspektor*

# 3 Von Produktdaten zu ihren Prozessen

Die Kopplung von Produktdaten zu den Prozessen ihrer Abwicklung bildet die Basis für eine optimale Bearbeitung von Aufträgen im Anlagenbau. Dadurch werden folgende Ziele erreicht:

- **Zeit- und kostenoptimale Erstellung:** Ableitung von Vorgaben zur Abwicklung aus den Produktdaten
- **Hohe Flexibilität in Bezug auf Änderungen:** Änderungen in den Produktdaten müssen automatisch Änderungen in der Abwicklung bewirken (evtl. auch umgekehrt)

Hier wird die Transformation vom Produkt in Prozesse mit Hilfe von Ableitungsregeln vom Benutzer definiert und kann durch Auswertung der festgelegten Bedingungen aus dem File *product.rules* oder durch Selektieren der verwendeten Instanzen ausgeführt werden. [Obe94]

## 3.1 Struktur der Ableitungsregeln

Die Ableitungsregeln werden definiert durch eine endliche Menge von Bedingungen, die die betroffenen Produktdaten beschreiben, und eine Aktion. Die Aktion beschreibt die Auswahl eines bestimmten Prozesses aus der Bibliothek. Zur Zeit erfolgt die Zuordnung einfach durch den Namen des zu verwendenden Prozesses. Das folgende Beispiel beschreibt die Transformation von der Instanz *Auto.vwgolf* in den Prozeß *AuslieferungGrossBritannien*:

```
!Regel:    hatTeile#Auto.vwgolf
 Bedingung:
       Klasse:  Auto
       Kennzeichen:  ?var1
       Attribut:   name
       Symbol:  =
       Wert:   vwgolf
 Bedingung:
       Klasse:  Auto
       Kennzeichen:  ?var1
       Attribut: lenkradseite
       Symbol:   =
       Wert: rechts
Aktion: AuslieferungGrossBritannien!
```

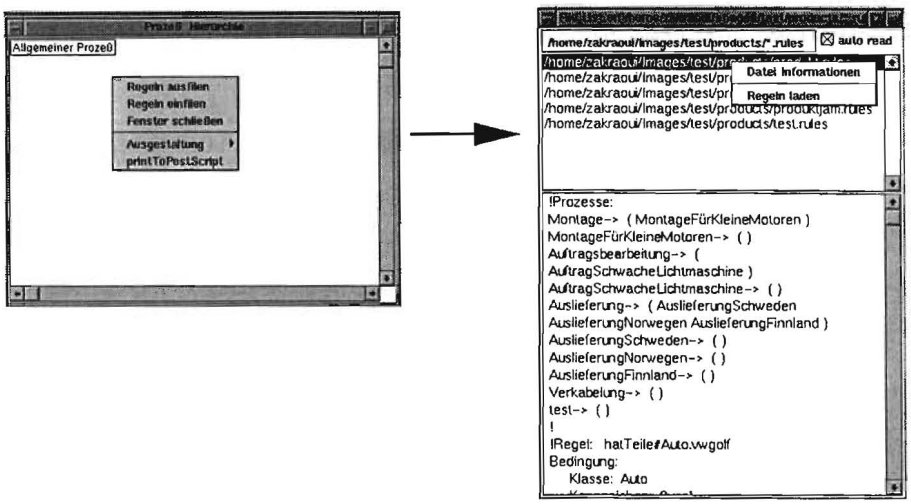



Abbildung 7: *Regeln laden*

Das Laden der Regeln geschieht in zwei Schritten:

1. Auswahl des Menü-Items *Regeln einfilen* (Abbildung 7, links) im *Produkt Beschreibung*-Fenster.

2. Auswahl des Menü-Items *Regeln laden* (Abbildung 7, rechts) im Filebrowser.

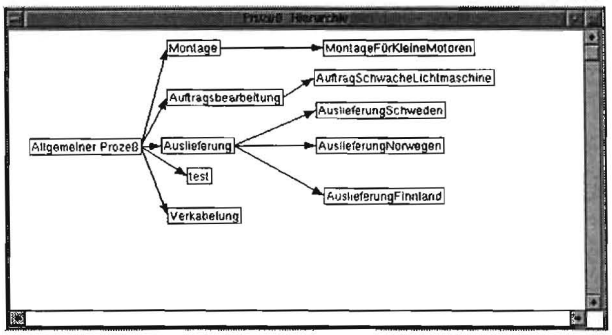



Abbildung 8: *Prozeß-Hierarchie*

Aufgrund der hierarchischen Anordnung der Prozesse (Abbildung 8), sind folgende Eigenschaften zu beachten :

- Die Kanten sind von allgemeineren zu spezielleren Prozessen gerichtet
- Die Semantik ergibt sich über die Prozesse und nicht über die Spezialisierung der Regeln
- Die Prozeßhierarchie muß explizit am Anfang des Files (*product.rules*) definiert sein

## 3.2 Auswertung von Ableitungsregeln

Die Ausführbarkeit der Regeln wird mit Hilfe eines Parsing-Algorithmus überprüft, bei welchem die Bedingungen der Regeln der Reihe nach bearbeitet werden. Wenn alle Bedingungen erfüllt sind (d.h. die Produktdaten vorhanden sind), wird ein Prozeß mit dem eingegebenen Namen (siehe Anhang B.2 bzw. C.2) kreiert. Konnte eine Regel nicht feuern, so wird sie in einem Attribut *rule* des Prozesses gespeichert und im nächsten Durchgang wieder validiert. Nachdem alle Regeln geparst sind, wird die Prozeß-Hierarchie aufgebaut und in zwei Fenstern wie in Abbildung 9 dargestellt. Das rechte Fenster zeigt die Liste der aktuellen Prozesse, d.h. die speziellsten Prozesse in der Prozeß-Hierarchie für die die Regeln gefeuert haben. Im linken Fenster werden die Produktdaten (Instanzen), von denen der selektierte Prozeß abhängt, gezeigt.

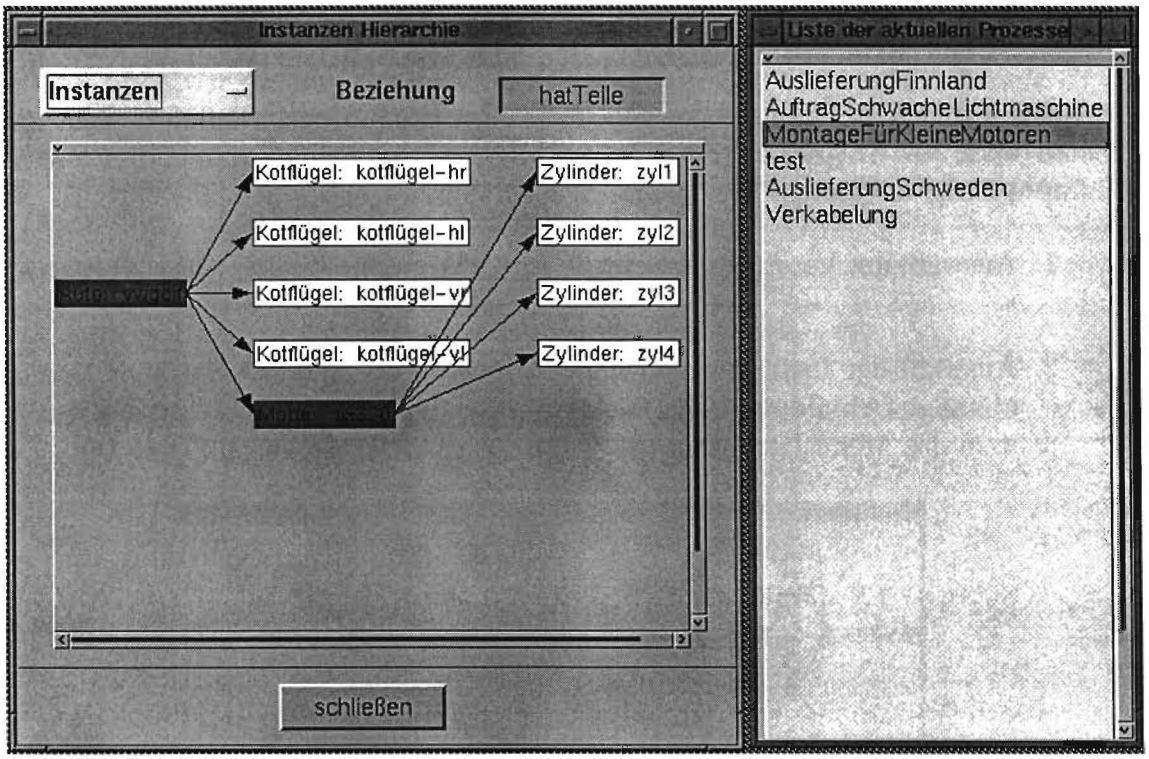



Abbildung 9: *Instanzen-Hierarchie* und *aktuelle Prozesse*

Im linken Fenster (*Instanzen-Hierarchie*) kann man sich durch Auswahl einer Wurzel-Instanz die Attributwerte anschauen (Abbildung 10).

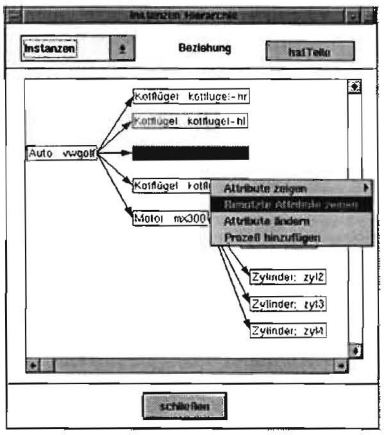



Abbildung 10: *Instanzen-Hierarchie*

## 3.3 Graphische Produkt-Prozeß-Kopplung

Außer mit den ASCII-Files, in denen die Abwicklungsregeln definiert sind, hat man auch die Möglichkeit, die Transformation von Produktdaten in Prozesse graphisch zu definieren. Dies geschieht nach dem folgenden Schema:

1. Auswahl des Vaterprozesses im *Prozeß-Hierarchie*-Fenster (siehe Abbildung 8)

2. Auswahl der Instanzen und der Attributwerte von denen der neue Prozeß abhängt (Abbildung 11). Die Attributwerte können gewählt werden indem man das Menü-Item *Attribute zeigen* (Abbildung 11, rechts) auswählt

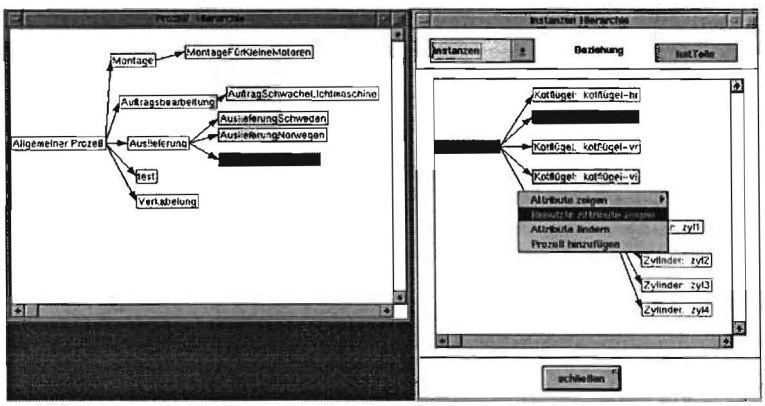



Abbildung 11: *Prozeß-Definition 1*

3. Nachdem die Produktdaten selektiert sind, wählt man das Menü-Item *Prozeß hinzufügen* (Abbildung 11, rechts) aus. Hier erscheint das in der Abbildung 12 dargestellte Fenster. Nach der Eingabe des Prozeß-Namens, wird die Liste der aktuellen Prozesse aktualisiert.

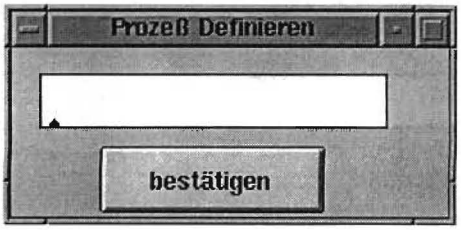



Abbildung 12: *Prozeß-Definition 2*

Nachdem die Transformation von Produktdaten in ihre Abwicklungsprozesse [KUW+94] abgeschlossen ist, kann man die Ableitungsregeln ausschreiben (d.h. ein ASCII-File *product.rules* erzeugen). Dies geschieht indem man das Menü-Item *Regeln ausfilen* auswählt (Abbildung 7).

# 4 Änderungsoperationen

Die Änderungsoperationen betreffen die Änderungen von Attributwerten. Beim Auswählen des Menü-Items *Attribut ändern* (Abbildung 10) erscheint ein Fenster (Abbildung 13) in welchem man das Attribut und seinen Wert auswählt. Nachdem

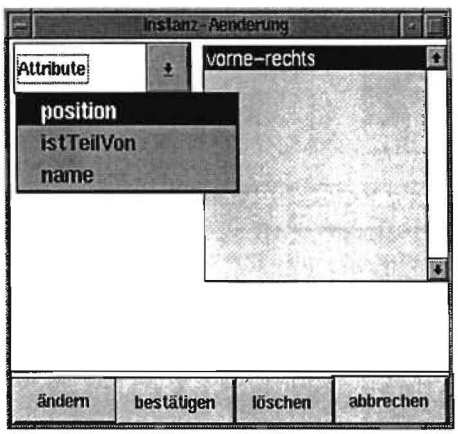



Abbildung 13: *Attributwert ändern (1)*

das geeignete Attribut selektiert ist, erscheint beim Anklicken des Felds *ändern* ein Eingabe-Feld, wo man den neuen Wert eingeben kann (Abbildung 14).

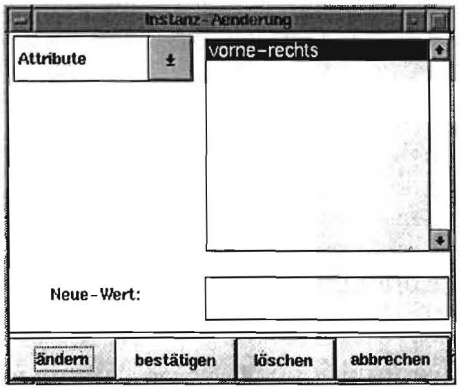



Abbildung 14: *Attributwert ändern (2)*

Nachdem die Änderung durchgeführt ist, werden automatisch die betroffenen Prozesse angezeigt (Abbildung 15).

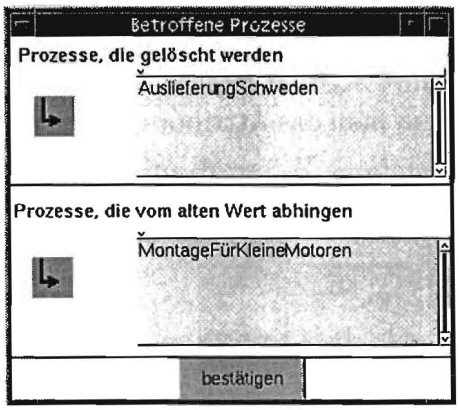



Abbildung 15: *Attributwert ändern (3)*

Die Prozesse, die von dem Attributwert abhängen und im oberen Teil der Abbildung 15 aufgelistet sind, werden gelöscht. Für die Prozesse, die von der Instanz selbst abhängen und im unteren Teil der Abbildung 15 angezeigt werden, kann der Benutzer selbst entscheiden, ob sie direkt aus der Liste der aktiven Prozesse gelöscht oder aber mit den neuen Produktdaten weiterbenutzt werden sollen.

# 5 Eine industrielle Fallstudie: TWA Amstand-ort

Die folgendenden Abschnitte beschreiben die Modellierung eines industriellen Produktes in PIMaS bis zur PIMaS-Anwendung in der Produktion anhand einer Trinkwasseraufbereitungsanlage (TWA). Die in der Fallstudie modellierte Struktur einer solchen Anlage von Siemens ist in Abbildung 16 zu sehen.

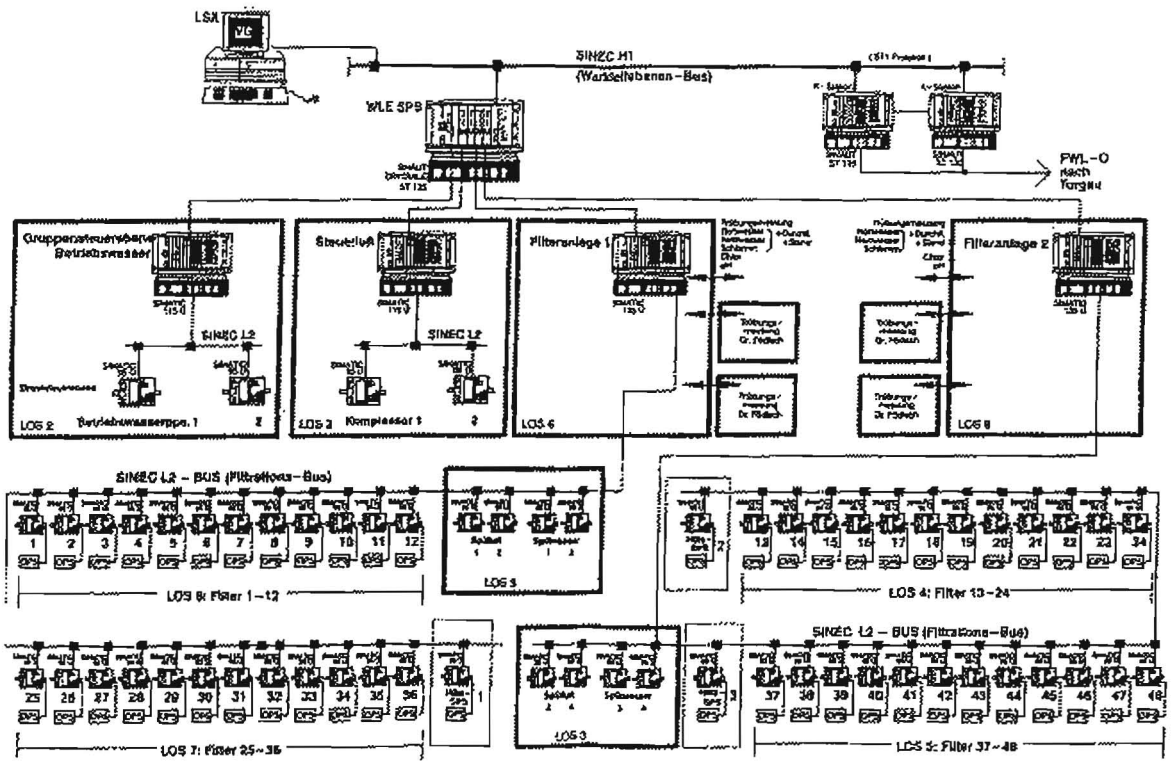



Abbildung 16: *Strukturplan einer TWA*

## 5.1 Produktmodellierung

Am Anfang einer solchen Modellierung steht die Analyse der vorgegebenen Struktur des Produktes. Um diese Struktur im objektorientierten PIMaS modellieren zu können, zerlegt man den Produktprototyp in seine abstrakten Komponenten, die dann intern als Klassen realisiert werden. Diese Klassen werden als Baum

in einer Klassenhierarchie dargestellt; für eine TWA zu sehen in Abbildung 17. Um ein konkretes Produkt aus diesen Klassen zusammenzustellen, werden In-

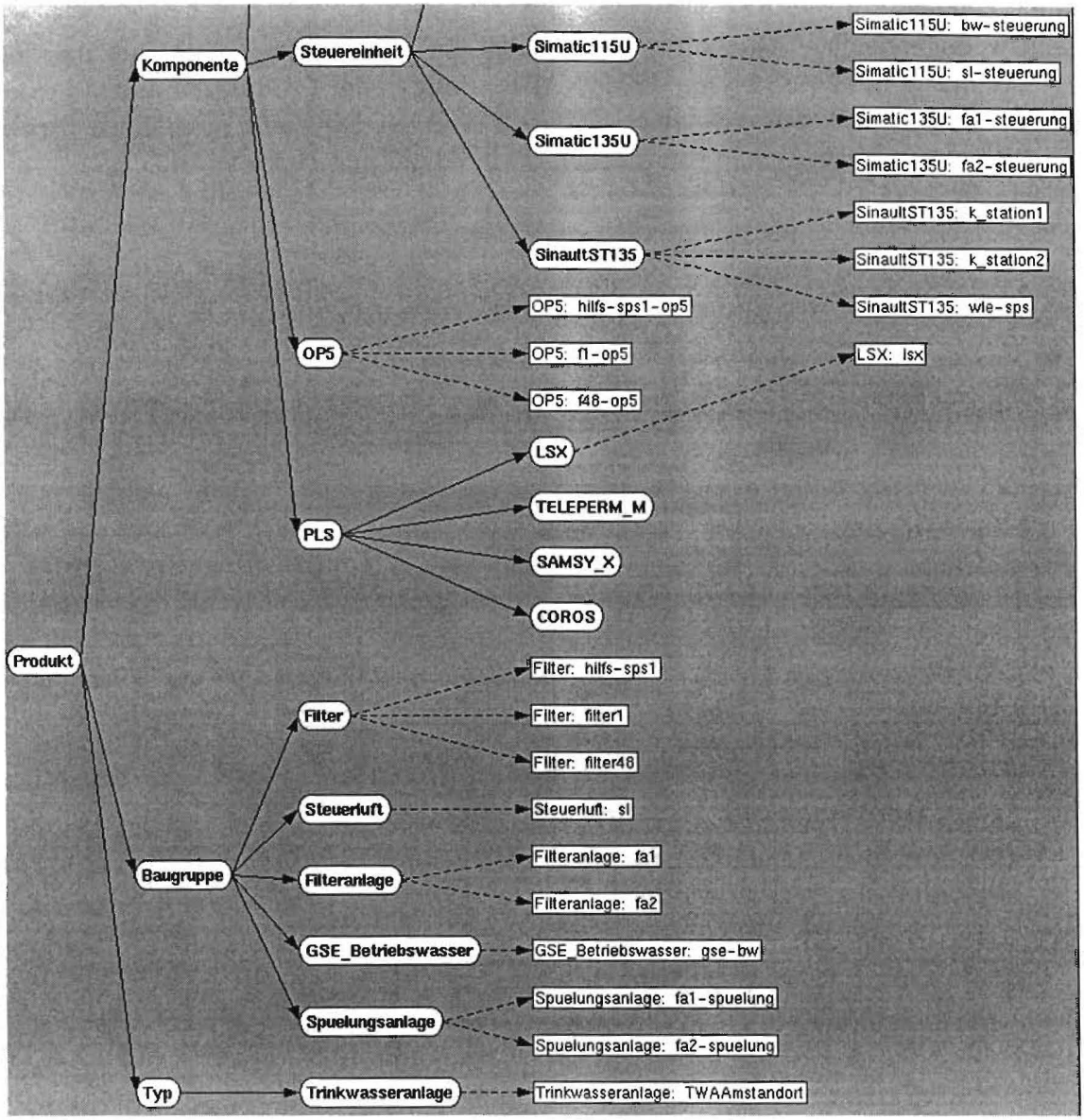



Abbildung 17: *Ausschnitt aus der Klassenhierarchie einer TWA*

stanzen der benötigten Komponenten erzeugt. Dabei ist es möglich, verschiedene Beziehungen dieser Instanzen untereinander darzustellen. Typisch sind z.B. topologische *has-as-part* - bzw. *is-part-of* - Beziehungen; in dem hier vorgestellten Beispiel handelt es sich um elektronische *steuert*-Beziehungen. Im Anhang C.1 befindet sich ein komplettes *TWA.produkt*-File in PIMaS-Syntax. In ihm ist die Realisierung derartiger Beziehungen vorgestellt. Anhand dieser Beziehungen erhält man eine hierarchisch gegliederte Struktur des Produktes, welche in Form einer Instanzenhierarchie repräsentiert wird. Ein Ausschnitt aus einer solchen Instanzenhierarchie ist in Bild 18 zu sehen.

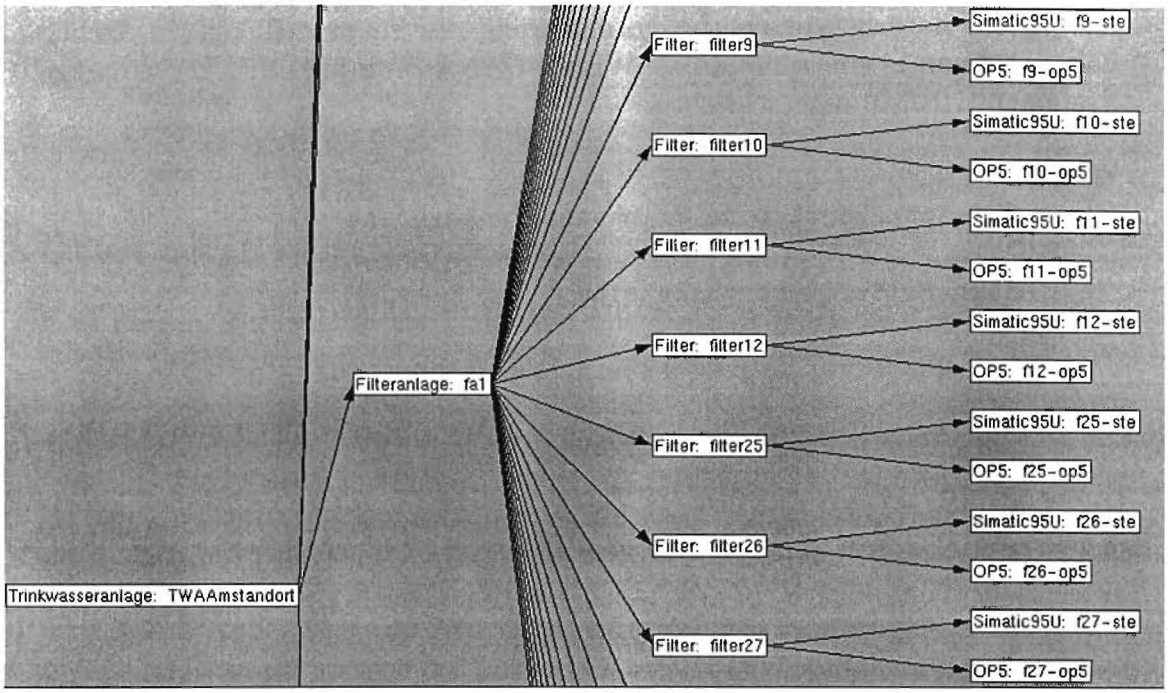



Abbildung 18: *Ausschnitt aus einer Instanzenhierarchie der TWA*

## 5.2 Transformation der Produktdaten in Abwicklungsprozesse

Nachdem die Produktdaten wie in Abschnitt 5.1 vorgestellt erfaßt und in der PIMaS-Syntax modelliert sind, werden die notwendigen Produkt-Prozeß-Regeln für eine Regel-Bibliothek in (semi)formaler WENN-DANN Form akquiriert und anschließend in einer PIMaS-verständlichen Syntax formalisiert. Hier als Beispiel eine Produkt-Prozeß-Abhängigkeit, die mit PIMaS beschrieben werden soll:

```
"Sind die Filter der TWA Amstandort verbraucht, und kann die
Wasserversorgung (kurzzeitig) auch von einer anderen TWA
uebernommen werden, so sind die Filter alle gleichzeitig
durch neue Filter zu ersetzen."
```

Diese natürlichsprachliche Formulierung des zu modellierenden Sachverhalts wird nun in eine semiformale WENN-DANN Notation i.S.v. Vorwärtsregel-Systemen [DBS93] umgeschrieben:

```
WENN
      TWA-Amstandort eine Trinkwasseranlage ist
   UND
      ihre Filter verbraucht sind
   UND
      die Wasserversorgung extern abgesichert ist
DANN
   tausche die Filter parallel aus
```

Ohne Berücksichtigung der Klassen- und Instanzenhierarchien ergibt sich folgende logische Umschreibung mit Rückwärtsregeln, hier als Prolog-Klausel:

```
paralleler Filteraustausch (X) :-
   Trinkwasseranlage (X),
   name (X, TWAAmstandort),
   altAnlage (X, vorhanden),
   filterZustand (X, verbraucht),
   externVersorgung (X, sicher).
```

Diese PIMaS-unabhängigen Darstellungen der Regel werden schließlich in die PIMaS-verständliche Syntax übersetzt; nach dem Schlüsselwort `Regel` wird auf eine Instanz der Klassenhierarchie zugegriffen, bei jeder `Bedingung` die betroffene Klasse referenziert und dann ggf. eine `Aktion` ausgeführt:

```
!Regel:  dummy#Trinkwasseranlage.TWAAmstandort
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: name
     Symbol: =
     Wert: TWAAmstandort
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: altAnlage
     Symbol: =
     Wert: vorhanden
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: filterZustand
     Symbol: =
     Wert: verbraucht
```

```
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: externVersorgung
     Symbol: =
     Wert: sicher
Aktion: parallelerFilteraustausch!
```

Eine Regelbasis, welche als Fallstudie mit Siemens-Experten im Rahmen der GiPP-Projektgemeinschaft S1 erstellt wurde [BKRW97], wird im Anhang C.2 in PIMaS-Syntax dargestellt. Aus der Gesamtheit all diesen Produkt- (vgl. Abschnitt 5.1) und Regelwissens ergibt sich eine komplette Wissensbasis zur Produkt-Prozeß-Transformation, wie in Abbildung 19 gezeigt, in welcher die obige Beispielregel im mittleren Fenster enthalten ist. In dieser kompletten Wissensbasis ist es dann auch möglich, Veränderungen an den Produktdaten vorzunehmen und die daraus resultierenden Prozeßänderungen mittels der Regeln zu verfolgen. Dies wird durch Änderungen an Attributen der einzelnen Instanzen erreicht und ist bereits im Kapitel 4 anhand des Beispiels aus der Automobilindustrie beschrieben. Ein Ziel solcher Modifikationen kann sein, Auswirkungen von unterschiedlichen Produktdaten auf Abwicklungsprozesse zu durchschauen. Im obigen Beispiel ergibt sich etwa bei der Einzelmodifikation `filterZustand = unverbraucht` die Aktion `keinFilteraustausch`, bei `externVersorgung = unsicher` jedoch die Aktion `sequentiellerFilteraustausch`.

# 6 Zusammenfassung

PIMaS erlaubt eine interaktive, objekt-orientierte Modellierung von Produktdaten. Diese werden durch WENN-DANN Regeln automatisch auf Produktionsprozesse abgebildet und erlauben die Verfolgung der Abhängigkeiten zwischen Produktdaten und Abwicklungsprozessen. Der hier vorgestellte Prototyp, der in SmallTalk verfügbar ist, wurde am einführenden Beispiel der Automobilproduktion getestet und in der industriellen Fallstudie des Trinkwasseraufbereitungsanlagenbaus erprobt. Der Prototyp ist auf Basis der virtuellen Maschine von SmallTalk gut auf andere Plattformen portierbar. Möglichkeiten einer Reimplementierung des Systems in Java wurden vorläufig evaluiert.

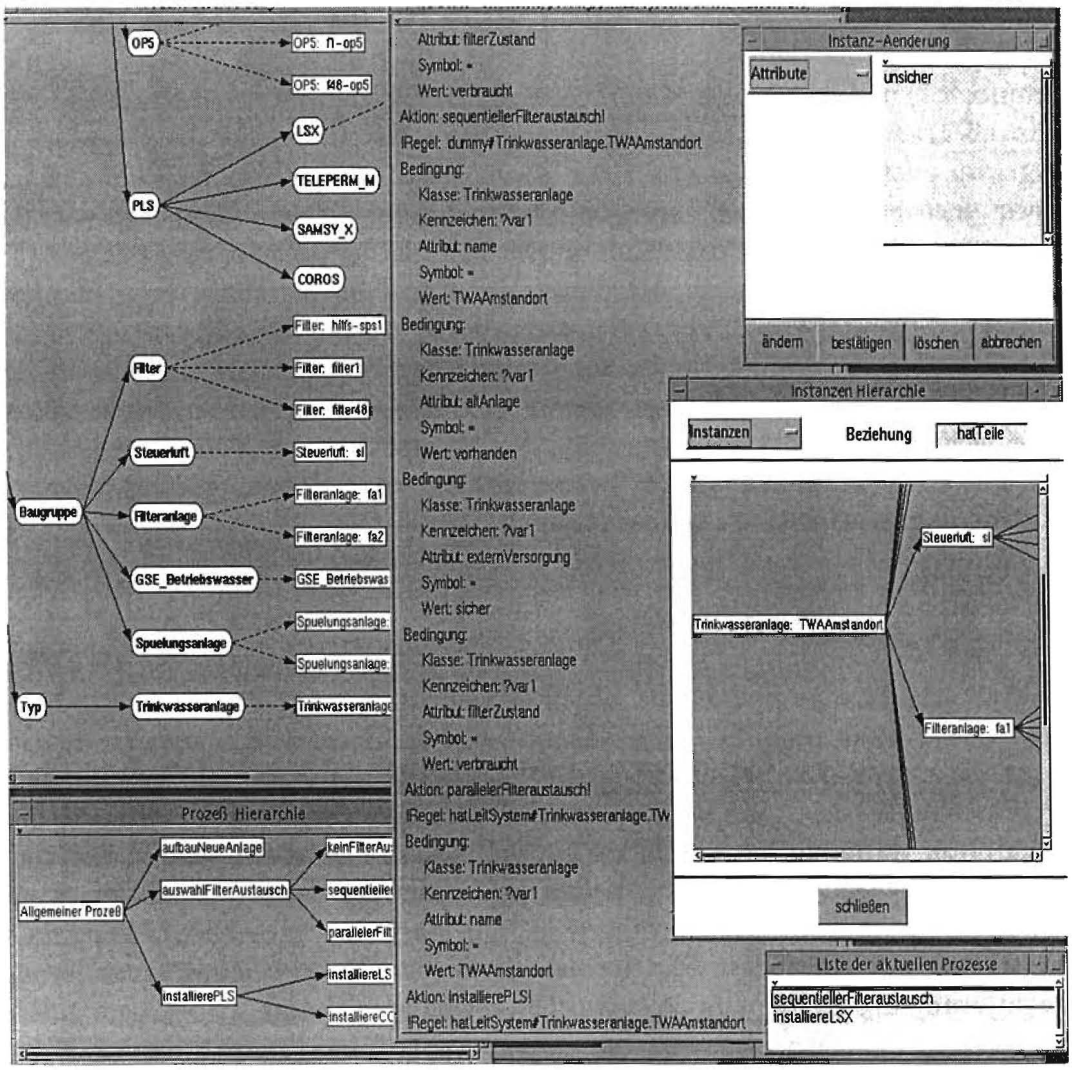



Abbildung 19: *komplette Regelumgebung*

# A PIMaS im Überblick

**P**rocess **I**nformation
**Ma**nagement-**S**ystem

Der Einsatz von PIMaS am Beispiel des Anwendungsbereichs Trinkwasseranlagen

Christian Embacher
Andreas Lorenz
Harold Boley

DFKI Kaiserslautern
24. Juli 1997

- Möglichkeiten von PIMaS
- Beschreibung von Produkten
- Beispiel: Die Beschreibung einer Trinkwasseranlage
- Kopplung von Abwicklungsprozessen an Produkte durch Zuordnungsregeln
- Beispiel: Eine Zuordnungsregel für den Anwendungsbereich Trinkwasseranlagen

## Möglichkeiten von PIMaS

PIMaS ist ein Softwaresystem mit dem

- Produktklassen und Produktinstanzen beschrieben werden können.

- Zuordnungsregeln von einem Produkt zu seinen Abwicklungsprozessen definiert werden können.

- die Menge der von einem Produkt abhängigen Prozesse bestimmt werden kann.

## Beschreibung von Produkten

Ein Produkt wird in PIMaS mit einem objektorientierten Ansatz auf drei Abstraktionsebenen beschrieben:

- Das **Metamodell** beschreibt die Produktklasse eines bestimmten *Typs* als Ansammlung von *Komponenten* und *Baugruppen*.

  Das Metamodell besteht aus *abstrakten* Klassen mit Attributen; diese sind durch PIMaS fest vorgegeben und für alle Arten von Produkten gleich.

- Das **Produktmodell** beschreibt die Produktklasse als hierarchisch aufgebautes Artefakt.

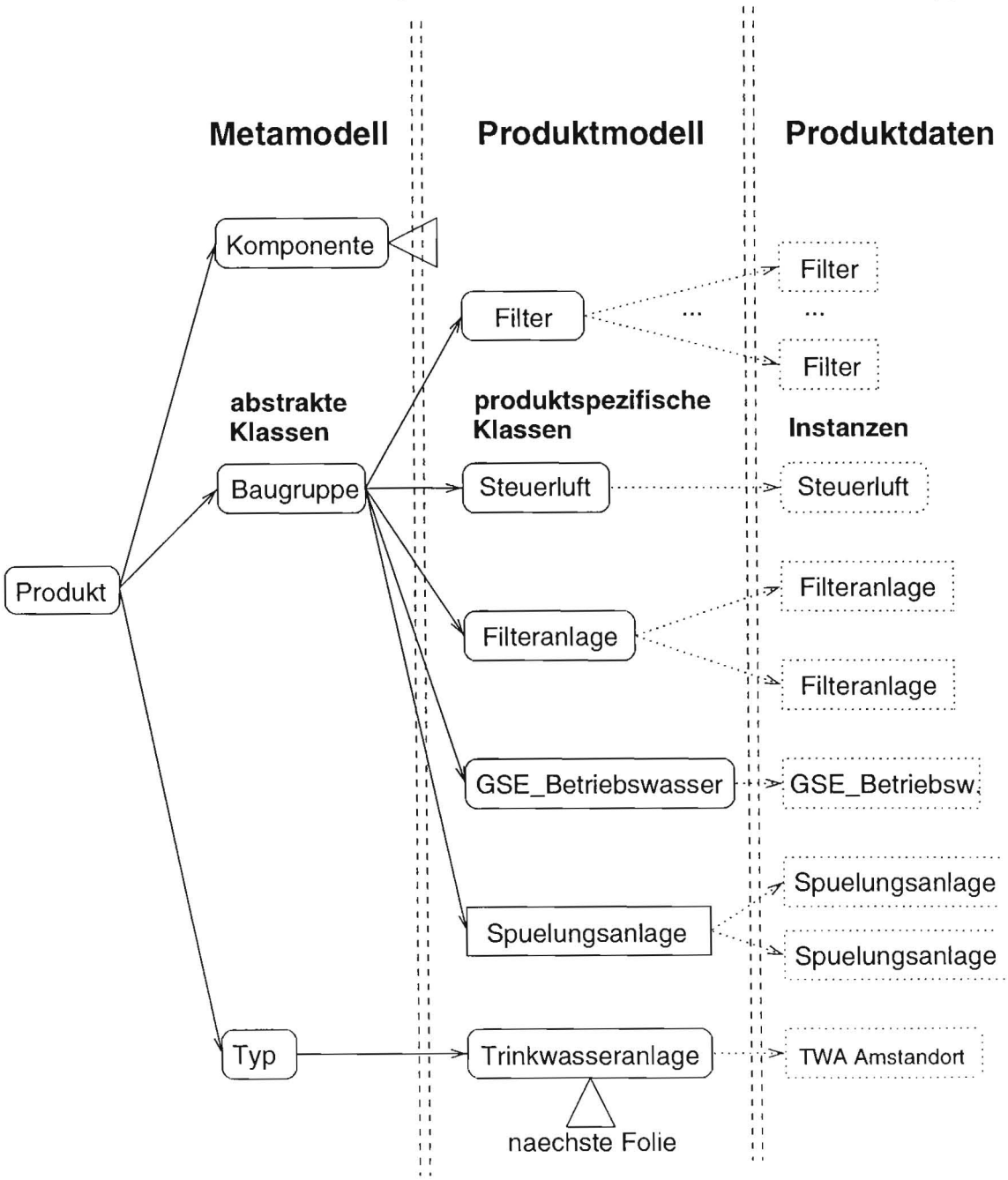
  Das Produktmodell besteht aus Klassen mit Attributen; diese Klassen sind *produktspezifisch* durch den Nutzer definierbar.

  Einem Teil der Attribute werden Werte zugewiesen, um gewisse Eigenschaften festzulegen, die jede Instanz des Produktmodells besitzt.

- Die **Produktdaten** beschreiben eine Produktinstanz; diese entsteht durch eine Belegung der noch freien Attribute des Produktmodells mit Werten (also durch Instanziierung des Produktmodells).

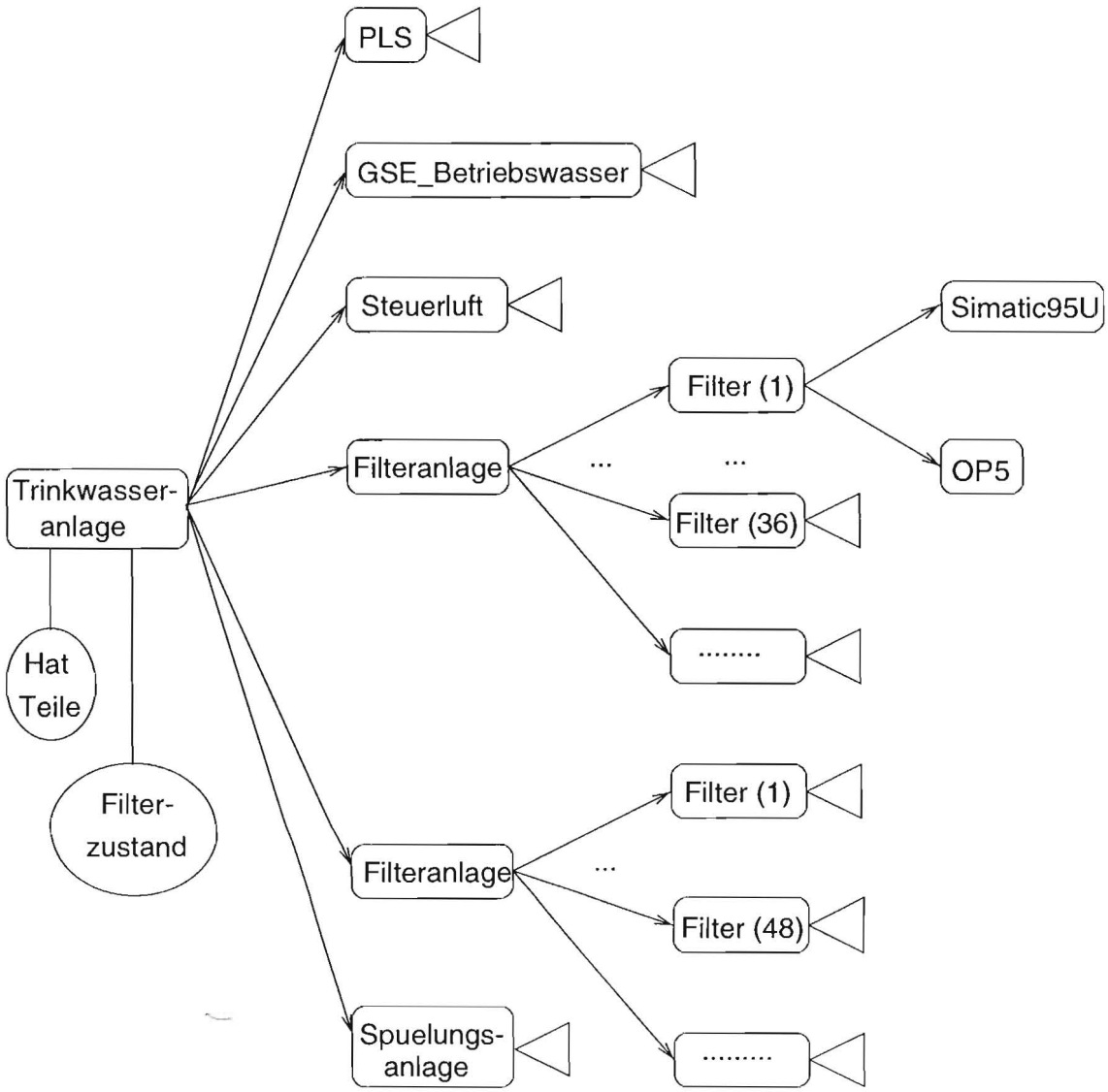
# Beispiel:

## Die Beschreibung einer Trinkwasseranlage (I)



## Beispiel:

## Die Beschreibung einer Trinkwasseranlage (II)



## Kopplung von Abwicklungsprozessen an Produkte durch Zuordnungsregeln

Durch eine Menge von Zuordnungsregeln werden Abwicklungsprozesse an Produkteile gekoppelt.

- **Aufbau und Bedeutung einer Regel**

  Eine Regel wird durch Bedingungen und eine Aktion definiert.

  Die Bedingungen beschreiben Produktteile.

  Die Aktion beschreibt einen Abwicklungsprozeß für die Produktteile.

  WENN die Bedingungen alle erfüllt sind,
  DANN wähle den durch die Aktion beschriebenen Abwicklungsprozeß aus.

- **Auswertung einer Bedingung**

  Eine Regelbedingung wird mit Hilfe der Wissensbasis ausgewertet; diese enthält neben den Regeln auch die Produktdaten, die die involvierten Produktinstanzen beschreiben.

  Eine Bedingung ist genau dann erfüllt, wenn die in der Bedingung geforderten Produktdaten in der Wissenbasis vorhanden sind.

## Beispiel: Eine Zuordnungsregel für den Anwendungsbereich Trinkwasseranlagen (I)

Sachverhalt, der in einer Regel dargestellt werden soll:

"Sind die Filter der TWA Amstandort verbraucht, und kann die Wasserversorgung (kurzzeitig) auch von einer anderen TWA übernommen werden, so sind die Filter alle *gleichzeitig* durch neue Filter zu ersetzen."

### Darstellung der Regel in *semiformaler Notation*

WENN es die TWA Amstandort gibt
UND
ihre Filter verbraucht sind
UND
die Wasserversorgung extern abgesichert ist
DANN
tausche die Filter parallel aus

## Beispiel: Eine Zuordnungsregel für den Anwendungsbereich Trinkwasseranlage (II)

Sachverhalt, der in einer Regel dargestellt werden soll:

"Sind die Filter der TWA Amstandort verbraucht, und kann die Wasserversorgung (kurzzeitig) auch von einer anderen TWA übernommen werden, so sind die Filter alle *gleichzeitig* durch neue Filter zu ersetzen."

**Darstellung der Regel in *Prolog-Notation***

```
parallelerFilteraustausch(X) :-
    trinkwasserAnlage(X),
    name(X, twaAmstandort),
    altAnlage(X, vorhanden),
    filterZustand(X, verbraucht).
    externVersorgung(X, sicher),
```

Beispielhafte Produktdaten (als Prolog-Fakten):

```
trinkwasserAnlage(twaAmstandort).
name(twaAmstandort, twaAmstandort).
altAnlage(twaAmstandort, vorhanden).
filterZustand(twaAmstandort, verbraucht).
externVersorgung(twaAmstandort, sicher).

[ externVersorgung(twaAmstandort, unsicher). ]
```

# Beispiel: Eine Zuordnungsregel für den Anwendungsbereich Trinkwasseranlage (III)

Sachverhalt, der in einer Regel dargestellt werden soll:

"Sind die Filter der TWA Amstandort verbraucht, und kann die Wasserversorgung (kurzzeitig) auch von einer anderen TWA übernommen werden, so sind die Filter alle *gleichzeitig* durch neue Filter zu ersetzen."

**Darstellung der Regel in *PIMaS-Notation***

**!Regel:**
dummy#Trinkwasseranlage.TWAAmstandort

**Bedingung:**
*Klasse:* Trinkwasseranlage
*Kennzeichen:* ?var1
*Attribut:* name
*Symbol:* =
*Wert:* TWAAmstandort

**Bedingung:**
*Klasse:* Trinkwasseranlage
*Kennzeichen:* ?var1
*Attribut:* altAnlage
*Symbol:* =
*Wert:* vorhanden

# Beispiel: Eine Zuordnungsregel für den Anwendungsbereich Trinkwasseranlage (IV)

Sachverhalt, der in einer Regel dargestellt werden soll:

"Sind die Filter der TWA Amstandort verbraucht, und kann die Wasserversorgung (kurzzeitig) auch von einer anderen TWA übernommen werden, so sind die Filter alle *gleichzeitig* durch neue Filter zu ersetzen."

**Darstellung der Regel in *PIMaS-Notation***
(Fortsetzung)

**Bedingung**:
*Klasse:* Trinkwasseranlage
*Kennzeichen:* ?var1
*Attribut:* filterZustand
*Symbol:* =
*Wert:* verbraucht
**Bedingung:**
*Klasse:* Trinkwasseranlage
*Kennzeichen:* ?var1
*Attribut:* externVersorgung
*Symbol:* =
*Wert:* sicher

**Aktion:** parallelerFilteraustausch!

# Beispiel: Eine Zuordnungsregel für den Anwendungsbereich Trinkwasseranlage (V)

Vollständige Modellierung der Regeln für "Filteraustausch" in Prolog-Notation:

parallelerFilteraustausch(X) :-
    trinkwasserAnlage(X),
    name(X, twaAmstandort),
    altAnlage(X, vorhanden),
    filterZustand(X, *verbraucht*).
    externVersorgung(X, **sicher**),

sequentiellerFilteraustausch(X) :-
    trinkwasserAnlage(X),
    name(X, twaAmstandort),
    altAnlage(X, vorhanden),
    filterZustand(X, *verbraucht*).
    externVersorgung(X, **unsicher**),

keinFilteraustausch(X) :-
    trinkwasserAnlage(X),
    name(X, twaAmstandort),
    altAnlage(X, vorhanden),
    filterZustand(X, *unverbraucht*).

# B Wissensbasis des Automobil-Beispiels

## B.1 Produktdaten

```
!vwgolf: Auto
lieferland: Schweden
hatTeile: kotfl"ugel-hr kotfl"ugel-hl kotfl"ugel-vr kotfl"ugel-vl mx30(
!
!kotfl"ugel-hr: Kotfl"ugel
istTeilVon: vwgolf
position: hinten-rechts
!
!kotfl"ugel-hl: Kotfl"ugel
istTeilVon: vwgolf
position: hinten-links
!
!kotfl"ugel-vr: Kotfl"ugel
istTeilVon: vwgolf
position: vorne-rechts
!
!kotfl"ugel-vl: Kotfl"ugel
istTeilVon: vwgolf
position: vorne-links
!
!mx300: Motor
istTeilVon: vwgolf
hatTeile: zyl1 zyl2 zyl3 zyl4
leistung: 50ps
hubraum: 2000ccm
!
!zyl1: Zylinder
istTeilVon: mx300
nummer: #1
!
!zyl2: Zylinder
istTeilVon: mx300
nummer: #2
!
!zyl3: Zylinder
istTeilVon: mx300
nummer: #3
!
!zyl4: Zylinder
istTeilVon: mx300
nummer: #4
!
```

```
!lx: Lichtmaschine
leistung: 16Amph
!
!bat: Batterie
leistung: 10Amph
istElektrischVerbundenMit: lx
!
!scheinwerfer: Verbraucher
leistung: 2A
istElektrischVerbundenMit: bat
!
```

## B.2 Regeln

```
!Prozesse:
Montage->  ( MontageFuerKleineMotoren )
MontageFuerKleineMotoren->  ( )
Auftragsbearbeitung->  ( AuftragSchwacheLichtmaschine )
AuftragSchwacheLichtmaschine->  ( )
Auslieferung->  ( AuslieferungSchweden AuslieferungNorwegen
 AuslieferungFinnland )
AuslieferungSchweden->  ( )
AuslieferungNorwegen->  ( )
AuslieferungFinnland->  ( )
Verkabelung->  ( )
test->  ( )
!
!Regel:   hatTeile#Auto.vwgolf
Bedingung:
     Klasse:  Auto
     Kennzeichen:  ?var1
     Attribut:   name
     Symbol:  =
     Wert:   vwgolf
Aktion:   Montage!
!Regel:   hatTeile#Auto.vwgolf
Bedingung:
     Klasse:  Auto
     Kennzeichen:  ?var1
     Attribut:   name
     Symbol:  =
     Wert:   vwgolf
Aktion:   Auftragsbearbeitung!
```

```
!Regel:   hatTeile#Auto.vwgolf
Bedingung:
     Klasse:  Auto
     Kennzeichen:  ?var1
     Attribut:   name
     Symbol:  =
     Wert:   vwgolf
Bedingung:
     Klasse:  Auto
     Kennzeichen:  ?var1
     Attribut: lieferland
     Symbol:   =
     Wert:  nil
Aktion:   Auslieferung!
!Regel:   dummy#Lichtmaschine.lx
Bedingung:
     Klasse:  Lichtmaschine
     Kennzeichen:  ?var1
     Attribut:   name
     Symbol:  =
     Wert:   lx
Bedingung:
     Klasse:  Lichtmaschine
     Kennzeichen:  ?var1
     Attribut: leistung
     Symbol:   =
     Wert:  nil
Aktion:   Verkabelung!
!Regel:   hatTeile#Auto.vwgolf
Bedingung:
     Klasse:  Motor
     Kennzeichen:  ?var1
     Attribut:   name
     Symbol:  =
     Wert:   mx300
Bedingung:
     Klasse:  Motor
     Kennzeichen:  ?var1
     Attribut: leistung
     Symbol:   =
     Wert:  nil
```

```
Bedingung:
     Klasse:  Auto
     Kennzeichen:  ?var2
     Attribut:   name
     Symbol:  =
     Wert:   vwgolf
Aktion:   MontageFuerKleineMotoren!
!Regel:   dummy#Lichtmaschine.lx
Bedingung:
     Klasse:  Lichtmaschine
     Kennzeichen:  ?var1
     Attribut:   name
     Symbol:  =
     Wert:   lx
Bedingung:
     Klasse:  Lichtmaschine
     Kennzeichen:  ?var1
     Attribut: leistung
     Symbol:   =
     Wert:  nil
Aktion:   AuftragSchwacheLichtmaschine!
!Regel:    hatTeile#Auto.vwgolf
Bedingung:
     Klasse:  Auto
     Kennzeichen:  ?var1
     Attribut:   name
     Symbol:  =
     Wert:   vwgolf
Bedingung:
     Klasse:  Auto
     Kennzeichen:  ?var1
     Attribut: lieferland
     Symbol:   =
     Wert:  Schweden
Aktion:   AuslieferungSchweden!
!Regel:    hatTeile#Auto.vwgolf
Bedingung:
     Klasse:  Auto
     Kennzeichen:  ?var1
     Attribut:   name
     Symbol:  =
     Wert:   vwgolf
```

```
Bedingung:
     Klasse:  Auto
     Kennzeichen:  ?var1
     Attribut: lieferland
     Symbol:   =
     Wert:  Norwegen
Aktion:   AuslieferungNorwegen!
!Regel:   hatTeile#Auto.vwgolf
Bedingung:
     Klasse:  Auto
     Kennzeichen:  ?var1
     Attribut:   name
     Symbol:  =
     Wert:   vwgolf
Bedingung:
     Klasse:  Auto
     Kennzeichen:  ?var1
     Attribut: lieferland
     Symbol:   =
     Wert:  nil
Aktion:   AuslieferungFinnland!
```

# C Wissensbasis der TWA-Amstandort-Fallstudie

## C.1 Produktdaten

```
!TWAAmstandort: Trinkwasseranlage
standort: Amstandort
hatLeitSystem: lsx
hatTeile: lsx wle-bus wle-sps gse-bw sl fa1 fa2 k_station1 k_station2
!
!k_station1: SinaultST135
!
!k_station2: SinaultST135
!
!lsx: LSX
!
!wle-bus: SinecH1
!
!wle-sps: SinaultST135
!
!gse-bw: GSE_Betriebswasser
hatTeile: bw-steuerung bw-bus bw-ste1 bw-ste2
!
!bw-steuerung: Simatic115U
!
!bw-bus: SinecL2
!
!bw-ste1: Simatic95U
!
!bw-ste2: Simatic95U
!
!sl: Steuerluft
hatTeile: sl-steuerung sl-bus sl-kompressor1 sl-kompressor2
!
!sl-steuerung: Simatic115U
!
!sl-bus: SinecL2
!
!sl-kompressor1: Simatic95U
!
!sl-kompressor2: Simatic95U
!
!fa1: Filteranlage
hatTeile: fa1-steuerung fa1-spuelung fa1-filtrationsbus filter1 hilfs-sps1
!
!fa2: Filteranlage
hatTeile: fa2-steuerung fa2-spuelung fa2-filtrationsbus filter48
!
```

```
!fa1-steuerung: Simatic135U
!
!fa2-steuerung: Simatic135U
!
!fa1-filtrationsbus: SinecL2
!
!fa2-filtrationsbus: SinecL2
!
!hilfs-sps1: Filter
hatTeile: hilfs-sps1-ste hilfs-sps1-op5
!
!hilfs-sps1-ste: Simatic95U
!
!hilfs-sps1-op5: OP5
!
!fa1-spuelung: Spuelungsanlage
hatTeile: spl-ste1 spw-ste1
!
!fa2-spuelung: Spuelungsanlage
hatTeile:
!
!spl-ste1: Simatic95U
!
!spw-ste1: Simatic95U
!
!filter1: Filter
hatTeile: f1-ste f1-op5
!
!f1-ste: Simatic95U
!
!f1-op5: OP5
!
!filter48: Filter
hatTeile: f48-ste f48-op5
!
!f48-ste: Simatic95U
!
!f48-op5: OP5
!
```

## C.2 Regeln

```
!Prozesse:
aufbauNeueAnlage-> ( )
auswahlFilterAustausch-> ( keinFilterAustausch sequentiellerFilteraustausch
 parallelerFilteraustausch )
keinFilterAustausch-> ( )
sequentiellerFilteraustausch-> ( )
parallelerFilteraustausch-> ( )
installierePLS-> ( installiereLSX installiereCOROS )
installiereLSX-> ( )
installiereCOROS-> ( )
!
!Regel:  dummy#Trinkwasseranlage.TWAAmstandort
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: name
     Symbol: =
     Wert: TWAAmstandort
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: altAnlage
     Symbol: =
     Wert: keineVorhanden
Aktion: aufbauNeueAnlage!
!Regel:  dummy#Trinkwasseranlage.TWAAmstandort
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: name
     Symbol: =
     Wert: TWAAmstandort
Aktion: auswahlFilterAustausch!
!Regel:  dummy#Trinkwasseranlage.TWAAmstandort
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: name
     Symbol: =
     Wert: TWAAmstandort
```

```
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: altAnlage
     Symbol: =
     Wert: vorhanden
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: filterZustand
     Symbol: =
     Wert: unverbraucht
Aktion: keinFilterAustausch!
!Regel:  dummy#Trinkwasseranlage.TWAAmstandort
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: name
     Symbol: =
     Wert: TWAAmstandort
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: altAnlage
     Symbol: =
     Wert: vorhanden
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: filterZustand
     Symbol: =
     Wert: verbraucht
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: externVersorgung
     Symbol: =
     Wert: unsicher
Aktion: sequentiellerFilteraustausch!
!Regel:  dummy#Trinkwasseranlage.TWAAmstandort
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: name
     Symbol: =
     Wert: TWAAmstandort
```

```
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: altAnlage
     Symbol: =
     Wert: vorhanden
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: filterZustand
     Symbol: =
     Wert: verbraucht
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: externVersorgung
     Symbol: =
     Wert: sicher
Aktion: parallelerFilteraustausch!
!Regel: hatLeitSystem#Trinkwasseranlage.TWAAmstandort
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: name
     Symbol: =
     Wert: TWAAmstandort
Aktion: installierePLS!
!Regel: hatLeitSystem#Trinkwasseranlage.TWAAmstandort
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: name
     Symbol: =
     Wert: TWAAmstandort
Bedingung:
     Klasse: LSX
     Kennzeichen: ?var2
     Attribut: name
     Symbol: =
     Wert: lsx
Aktion: installiereLSX!
```

```
!Regel: hatLeitSystem#Trinkwasseranlage.TWAAmstandort
Bedingung:
     Klasse: Trinkwasseranlage
     Kennzeichen: ?var1
     Attribut: name
     Symbol: =
     Wert: TWAAmstandort
Bedingung:
     Klasse: COROS
     Kennzeichen: ?var2
     Attribut: name
     Symbol: =
     Wert: coros
Aktion: installiereCOROS!
```

# D Syntax der Files: Meta-Modell, Produkt-Modell und Produkt-Instanzen Files

- **BNF-Regeln**

*BeschreibungsFormat* ::= *[Meta-Modell.st*
*Produkt-Modell.st*
*Produkt-Instanzen.produkt ]*

*Meta-Modell.st* ::= **!NeueKlasse: Objekt cr OberKlasse: AbstractClass**
**cr Attribute:** { *attrName* }* **cr ! {!NeueKlasse:**
*KlassenName* **cr OberKlasse:** *KlassenName*
**cr Attribute:** { *attrName* }* **cr ! }+**

*Produkt-Modell.st* ::= {**!NeueKlasse:** *KlassenName* **cr OberKlasse:**
*KlassenName* **cr Attribute:** { *attrName* }* **cr ! }+**

*Produkt-Instanzen.produkt*::= {**!***instName***:** *InstKlassenName* **cr** { *attrName***:** { *attrWert*
}* **cr** }* **!** }*

- **Support**

*KlasseName* ::= eine Zeichenkette
*attrName* ::= eine Zeichenkette
*instName* ::= eine Zeichenkette
*InstKlassenName* ::= Bezeichnet einen Klassen Name der
in der Produkt-Modell.st
definiert wurde

# E Syntax der Regeln

- **BNF-Regeln**

  **!prozesse: cr** { *ProcName*→ ({ *ProcName* }* ) **cr**}*
  **!Regel:** {**attrName** | **dummy**}♯*KlassName. InstName* **cr Bedingung: cr**
  **Klasse:***KlassName* **cr Kennzeichen:** *Kennzeischen* **cr**
  **Attribut:** *AttrName* **cr Symbol:** *Symbol* **cr Wert:** *AttrWert* **cr Aktion:** *Pr*

- **Support**

| | |
|---|---|
| *KlassName* | ::= eine Zeichenkette |
| *AttrName* | ::= eine Zeichenkette |
| *InstName* | ::= eine Zeichenkette |
| *InstKlassenName* | ::= Bezeichnet einen Klassen Namen der in der Produkt-Modell.st definiert wurde |
| *ProcName* | ::= *Name des Prozesses* |
| *AttrWert* | ::= *Wert des Attributes* |
| *Symbol* | ::= = |
| *AttrName* | ::= *Name des Attributes* |
| *Kennzeichen* | ::= **?var***i* |

# Literatur


[Bac97] Bernd Bachmann. *A Solution for the Semantic Unification Problem to Reuse Knowledge-Based Systems.* Infix Verlag, 1997.

[BKRW97] Harold Boley, Andreas Kodweiss, Christian Rupprecht, and Markus Wittmann. Eine Methodik zur semantischen Zusammenführung von Prozeß- und Produktmodellen im Anlagenbau. *IM – Information Management and Consulting*, (Sonderausgabe GiPP'97), Okt. 1997.

[DBS93] R. Davis, B. G. Buchanan, and E. H. Shortliffe. Retrospective on "production rules as a representation for a knowledge-based consultation program". *Artificial Intelligence*, 59(1-2):181–189, February 1993.

[HH95] Trevor Hopkins and Bernard Horan. *Smalltalk: An Introduction to Application Development using Visualworks.* Prentice-Hall, 1995.

[KUW+94] S. Kirn, R. Unland, U. Wanka, S. Abbas, and G. O'Hare. Flexible Organisationen durch Workflow Management? Oder: Zum Problem der Modellierung von Geschäftsprozessen. In U. Hasenkamp, editor, *Einfüuhrung von CSCW-Systemen in Organisationen - Tagungsband der D-CSCW 1994*, pages 13–27, Vieweg Wirtschaftsinformatik, Braunschweig, 1994.

[Li95] Tao Li. A rule-based and object-oriented AI programming language. *ACM SIGPLAN Notices*, 30(12):17–24, December 1995.

[Obe94] Andreas Oberweis. Workflow management in software engineering projects. Technical report, Karlsruhe, 1994.

[RBP+94] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Objektorientiertes Modellieren und Entwerfen.* Carl Hanser Verlag, Muenchen, 1994.

[TV94] S. Tietjen and E. Voss. *Objektorientierte Programmierung mit Smalltalk/V.* Vieweg, Braunschweig, 1 edition, 1994.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

-Bibliothek, Information
und Dokumentation (BID)-
PF 2080
67608 Kaiserslautern
FRG

Telefon (0631) 205-3506
Telefax (0631) 205-3210

e-mail
dfkibib@dfki.uni-kl.de
WWW
http://www.dfki.uni-sb.de/dfkibib

## Veröffentlichungen des DFKI

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse oder (so sie als per ftp erhaeltlich angemerkt sind) per anonymous ftp von ftp.dfki.uni-kl.de (131.246.241.100) im Verzeichnis pub/Publications bezogen werden. Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

## DFKI Publications

*The following DFKI publications or the list of all published papers so far are obtainable from the above address or (if they are marked as obtainable by ftp) by anonymous ftp from ftp.dfki.uni-kl.de (131.246.241.100) in the directory pub/Publications.*
*The reports are distributed free of charge except where otherwise noted.*

---

# DFKI Research Reports

## 1997

**RR-97-08**
*Stefan Müller*
Complement Extraction Lexical Rules and Argument Attraction
14 pages

**RR-97-07**
*Stefan Müller*
Yet Another Paper about Partial Verb Phrase Fronting in German
26 pages

**RR-97-06**
*Stefan Müller*
Scrambling in German – Extraction into the *Mittelfeld*
24 pages

**RR-97-05**
*Harald Meyer auf'm Hofe*
Finding Regions of Local Repair in Hierarchical Constraint Satisfaction
33 pages

**RR-97-04**
*Serge Autexier, Dieter Hutter*
Parameterized Abstractions used for Proof-Planning
13 pages

**RR-97-03**
*Dieter Hutter*
Using Rippling to Prove the Termination of Algorithms
15 pages

**RR-97-02**
*Stephan Busemann, Thierry Declerck, Abdel Kader Diagne, Luca Dini, Judith Klein, Sven Schmeier*
Natural Language Dialogue Service for Appointment Scheduling Agents
15 pages

**RR-97-01**
*Erica Melis, Claus Sengler*
Analogy in Verification of State-Based Specifications: First Results
12 pages

## 1996

**RR-96-06**
*Claus Sengler*
Case Studies of Non-Freely Generated Data Types
200 pages

**RR-96-05**
*Stephan Busemann*
Best-First Surface Realization
11 pages

**RR-96-04**
*Christoph G. Jung, Klaus Fischer, Alastair Burt*
Multi-Agent Planning
Using an *Abductive*
EVENT CALCULUS
114 pages

**RR-96-03**
*Günter Neumann*
Interleaving
Natural Language Parsing and Generation
Through Uniform Processing
51 pages

**RR-96-02**
*E.André, J. Müller , T.Rist:*
PPP-Persona: Ein objektorientierter Multimedia-Präsentationsagent
14 Seiten

**RR-96-01**
*Claus Sengler*
Induction on Non-Freely Generated Data Types
188 pages

## 1995

**RR-95-20**
*Hans-Ulrich Krieger*
Typed Feature Structures, Definite Equivalences, Greatest Model Semantics, and Nonmonotonicity
27 pages

**RR-95-19**
*Abdel Kader Diagne, Walter Kasper, Hans-Ulrich Krieger*
Distributed Parsing With HPSG Grammar
20 pages

**RR-95-18**
*Hans-Ulrich Krieger, Ulrich Schäfer*
Efficient Parameterizable Type Expansion for Typed Feature Formalisms
19 pages

**RR-95-17**
*Hans-Ulrich Krieger*
Classification and Representation of Types in TDL
17 pages

**RR-95-16**
*Martin Müller, Tobias Van Roy*
Title not set
0 pages

**Note:** The author(s) were unable to deliver this document for printing before the end of the year. It will be printed next year.

**RR-95-15**
*Joachim Niehren, Tobias Van Roy*
Title not set
0 pages

**Note:** The author(s) were unable to deliver this document for printing before the end of the year. It will be printed next year.

**RR-95-14**
*Joachim Niehren*
Functional Computation as Concurrent Computation
50 pages

**RR-95-13**
*Werner Stephan, Susanne Biundo*
Deduction-based Refinement Planning
14 pages

**RR-95-12**
*Walter Hower, Winfried H. Graf*
Research in Constraint-Based Layout, Visualization, CAD, and Related Topics: A Bibliographical Survey
33 pages

**RR-95-11**
*Anne Kilger, Wolgang Finkler*
Incremental Generation for Real-Time Applications
47 pages

**RR-95-10**
*Gert Smolka*
The Oz Programming Model
23 pages

**RR-95-09**
*M. Buchheit, F. M. Donini, W. Nutt, A. Schaerf*
A Refined Architecture for Terminological Systems: Terminology = Schema + Views
71 pages

**RR-95-08**
*Michael Mehl, Ralf Scheidhauer, Christian Schulte*
An Abstract Machine for Oz
23 pages

**RR-95-07**
*Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt*
The Complexity of Concept Languages
57 pages

**RR-95-06**
*Bernd Kiefer, Thomas Fettig*
FEGRAMED
An interactive Graphics Editor for Feature Structures
37 pages

**RR-95-05**
*Rolf Backofen, James Rogers, K. Vijay-Shanker*
A First-Order Axiomatization of the Theory of Finite Trees
35 pages

**RR-95-04**
*M. Buchheit, H.-J. Bürckert, B. Hollunder, A. Laux, W. Nutt,*
*M. Wójcik*
Task Acquisition with a Description Logic Reasoner
17 pages

**RR-95-03**
*Stephan Baumann, Michael Malburg, Hans-Guenther Hein, Rainer Hoch,*
*Thomas Kieninger, Norbert Kuhn*
Document Analysis at DFKI
Part 2: Information Extraction
40 pages

**RR-95-02**
*Majdi Ben Hadj Ali, Frank Fein, Frank Hoenes, Thorsten Jaeger,*
*Achim Weigel*
Document Analysis at DFKI
Part 1: Image Analysis and Text Recognition
69 pages

**RR-95-01**
*Klaus Fischer, Jörg P. Müller, Markus Pischel*
Cooperative Transportation Scheduling
an application Domain for DAI
31 pages

## 1994

**RR-94-39**
*Hans-Ulrich Krieger*
Typed Feature Formalisms as a Common Basis for Linguistic Specification.
21 pages

**RR-94-38**
*Hans Uszkoreit, Rolf Backofen, Stephan Busemann, Abdel Kader Diagne,*
*Elizabeth A. Hinkelman, Walter Kasper, Bernd Kiefer, Hans-Ulrich Krieger,*
*Klaus Netter, Günter Neumann, Stephan Oepen, Stephen P. Spackman.*
DISCO–An HPSG-based NLP System and its Application for Appointment Scheduling.
13 pages

**RR-94-37**
*Hans-Ulrich Krieger, Ulrich Schäfer*
TDL - A Type Description Language for HPSG, Part 1: Overview.
54 pages

**RR-94-36**
*Manfred Meyer*
Issues in Concurrent Knowledge Engineering. Knowledge Base and Knowledge Share Evolution.
17 pages

**RR-94-35**
*Rolf Backofen*
A Complete Axiomatization of a Theory with Feature and Arity Constraints
49 pages

**RR-94-34**
*Stephan Busemann, Stephan Oepen, Elizabeth A. Hinkelman,*
*Günter Neumann, Hans Uszkoreit*
COSMA – Multi-Participant NL Interaction for Appointment Scheduling
80 pages

**RR-94-33**
*Franz Baader, Armin Laux*
Terminological Logics with Modal Operators
29 pages

**RR-94-31**
*Otto Kühn, Volker Becker, Georg Lohse, Philipp Neumann*
Integrated Knowledge Utilization and Evolution for the Conservation of Corporate Know-How
17 pages

**RR-94-23**
*Gert Smolka*
The Definition of Kernel Oz
53 pages

**RR-94-20**
*Christian Schulte, Gert Smolka, Jörg Würtz*
Encapsulated Search and Constraint Programming in Oz
21 pages

**RR-94-19**
*Rainer Hoch*
Using IR Techniques for Text Classification in Document Analysis
16 pages

**RR-94-18**
*Rolf Backofen, Ralf Treinen*
How to Win a Game with Features
18 pages

**RR-94-17**
*Georg Struth*
Philosophical Logics—A Survey and a Bibliography
58 pages

**RR-94-16**
*Gert Smolka*
A Foundation for Higher-order Concurrent Constraint Programming
26 pages

**RR-94-15**
*Winfried H. Graf, Stefan Neurohr*
Using Graphical Style and Visibility Constraints for a Meaningful Layout in Visual Programming Interfaces
20 pages

**RR-94-14**
*Harold Boley, Ulrich Buhrmann, Christof Kremer*
Towards a Sharable Knowledge Base on Recyclable Plastics
14 pages

**RR-94-13**
*Jana Koehler*
Planning from Second Principles—A Logic-based Approach
49 pages

**RR-94-12**
*Hubert Comon, Ralf Treinen*
Ordering Constraints on Trees
34 pages

**RR-94-11**
*Knut Hinkelmann*
A Consequence Finding Approach for Feature Recognition in CAPP
18 pages

**RR-94-10**
*Knut Hinkelmann, Helge Hintze*
Computing Cost Estimates for Proof Strategies
22 pages

**RR-94-08**
*Otto Kühn, Björn Höfling*
Conserving Corporate Knowledge for Crankshaft Design
17 pages

**RR-94-07**
*Harold Boley*
Finite Domains and Exclusions as First-Class Citizens
25 pages

**RR-94-06**
*Dietmar Dengler*
An Adaptive Deductive Planning System
17 pages

**RR-94-05**
*Franz Schmalhofer, J. Stuart Aitken, Lyle E. Bourne jr.*
Beyond the Knowledge Level: Descriptions of Rational Behavior for Sharing and Reuse
81 pages

**RR-94-03**
*Gert Smolka*
A Calculus for Higher-Order Concurrent Constraint Programming with Deep Guards
34 pages

**RR-94-02**
*Elisabeth André, Thomas Rist*
Von Textgeneratoren zu Intellimedia-Präsentationssystemen
22 Seiten

**RR-94-01**
*Elisabeth André, Thomas Rist*
Multimedia Presentations: The Support of Passive and Active Viewing
15 pages

---

# DFKI Technical Memos

## 1997

**TM-97-03**
*Hans-Jürgen Bürckert, Klaus Fischer, Gero Vierke*
TeleTruck: A Holonic Fleet Management System
10 pages

**TM-97-02**
*Christian Gerber*
Scalability of Multi-Agent Systems - Proposal for a Dissertation
49 pages

**TM-97-01**
*Markus Perling*
GeneTS: A Relational-Functional Genetic Algorithm for the Traveling Salesman Problem
26 pages

## 1996

**TM-96-02**
*Harold Boley*
Knowledge Bases in the World Wide Web:
A Challenge for Logic Programming
(Second, Revised Edition)
10 pages

**TM-96-01**
*Gerd Kamp, Holger Wache*
CTL — a description Logic with expressive concrete domains
19 pages

## 1995

**TM-95-04**
*Klaus Schmid*
Creative Problem Solving
and
Automated Discovery
— An Analysis of Psychological and AI Research —
152 pages

**TM-95-03**
*Andreas Abecker, Harold Boley, Knut Hinkelmann, Holger Wache,*
*Franz Schmalhofer*
An Environment for Exploring and Validating Declarative Knowledge
11 pages

**TM-95-02**
*Michael Sintek*
FLIP: Functional-plus-Logic Programming
on an Integrated Platform
106 pages

**TM-95-01**
*Martin Buchheit, Rüdiger Klein, Werner Nutt*
Constructive Problem Solving: A Model Construction Approach towards Configuration
34 pages

## 1994

**TM-94-05**
*Klaus Fischer, Jörg P. Müller, Markus Pischel*
Unifying Control in a Layered Agent Architecture
27 pages

**TM-94-04**
*Cornelia Fischer*
PAntUDE - An Anti-Unification Algorithm for Expressing Refined Generalizations
22 pages

**TM-94-03**
*Victoria Hall*
Uncertainty-Valued Horn Clauses
31 pages

**TM-94-02**
*Rainer Bleisinger, Berthold Kröll*
Representation of Non-Convex Time Intervals and Propagation of Non-Convex Relations
11 pages

**TM-94-01**
*Rainer Bleisinger, Klaus-Peter Gores*
Text Skimming as a Part in Paper Document Understanding
14 pages

---

# DFKI Documents

## 1997

**D-97-08**
*Christoph G. Jung, Klaus Fischer, Susanne Schacht*
Distributed Cognitive Systems
Proceedings of the VKS'97 Workshop
50 pages

**D-97-07**
*Harold Boley, Bernd Bachmann, Christian Blum, Christian Embacher,*
*Andreas Lorenz, Jamel Zakraoui*
PIMaS:
Ein objektorientiert-regelbasiertes System zur Produkt-Prozeß-Transformation
45 Seiten

**D-97-06**
*Tilman Becker, Stephan Busemann, Wolfgang Finkler*
DFKI Workshop on Natural Language Generation
67 pages

**D-97-05**
*Stephan Baumann, Majdi Ben Hadj Ali, Jürgen Lichter, Michael Malburg,*
*Harald Meyer auf'm Hofe, Claudia Wenzel*
Anforderungen an ein System zur Dokumentanalyse im Unternehmenskontext
— Integration von Datenbeständen, Aufbau- und Ablauforganisation
42 Seiten

**D-97-04**
*Claudia Wenzel, Markus Junker*
Entwurf einer Patternbeschreibungssprache
für die Informationsextraktion
in der Dokumentanalyse
24 Seiten

**D-97-03**
*Andreas Abecker, Stefan Decker, Knut Hinkelmann, Ulrich Reimer*
Proceedings of the Workshop „Knowledge-Based Systems for Knowledge Management in Enterprises" 97
167 pages

**D-97-02**
*Tilman Becker, Hans-Ulrich Krieger*
Proceedings of the Fifth Meeting on Mathematics of Language (MOL5)
168 pages

**Note:** This document is available for a nominal charge of 25 DM (or 15 US-$).

**D-97-01**
*Thomas Malik*
NetGLTool Benutzeranleitung
40 Seiten

## 1996

**D-96-07**
*Technical Staff*
DFKI Jahresbericht 1995
55 Seiten

**Note:** This document is no longer available in printed form.

**D-96-06**
*Klaus Fischer (Ed.)*
Working Notes of the KI'96 Workshop on Agent-Oriented Programming and Distributed Systems
63 pages

**D-96-05**
*Martin Schaaf*
Ein Framework zur Erstellung verteilter Anwendungen
94 pages

**D-96-04**
*Franz Baader, Hans-Jürgen Bürckert, Andreas Günter, Werner Nutt (Hrsg.)*
Proceedings of the Workshop on Knowledge Representation and Configuration WRKP'96
83 pages

**D-96-03**
*Winfried Tautges*
Der DESIGN-ANALYZER - Decision Support im Designprozess
75 Seiten

**D-96-01**
*Klaus Fischer, Darius Schier*
Ein Multiagentenansatz zum Lösen von Fleet-Scheduling-Problemen
72 Seiten

## 1995

**D-95-12**
*F. Baader, M. Buchheit, M. A. Jeusfeld, W. Nutt (Eds.)*
Working Notes of the KI'95 Workshop:
*KRDB-95 - Reasoning about Structured Objects:*
Knowledge Representation Meets Databases
61 pages

**D-95-11**
*Stephan Busemann, Iris Merget*
Eine Untersuchung kommerzieller Terminverwaltungssoftware im Hinblick auf die Kopplung mit natürlichsprachlichen Systemen
32 Seiten

**D-95-10**
*Volker Ehresmann*
Integration ressourcen-orientierter Techniken in das wissensbasierte Konfigurierungssystem TOOCON
108 Seiten

**D-95-09**
*Antonio Krüger*
PROXIMA: Ein System zur Generierung graphischer Abstraktionen
120 Seiten

**D-95-08**
*Technical Staff*
DFKI Jahresbericht 1994
63 Seiten

**Note:** This document is no longer available in printed form.

**D-95-07**
*Ottmar Lutzy*
Morphic - Plus
Ein morphologisches Analyseprogramm für die deutsche Flexionsmorphologie und Komposita-Analyse
74 Seiten

**D-95-06**
*Markus Steffens, Ansgar Bernardi*
Integriertes Produktmodell für Behälter aus Faserverbundwerkstoffen
48 Seiten

**D-95-05**
*Georg Schneider*
Eine Werkbank zur Erzeugung von 3D-Illustrationen
157 Seiten

**D-95-04**
*Victoria Hall*
Integration von Sorten als ausgezeichnete taxonomische Prädikate in eine relational-funktionale Sprache
56 Seiten

**D-95-03**
*Christoph Endres, Lars Klein, Markus Meyer*
Implementierung und Erweiterung der Sprache $\mathcal{ALCP}$
110 Seiten

**D-95-02**
*Andreas Butz*
BETTY
Ein System zur Planung und Generierung informativer Animationssequenzen
95 Seiten

**D-95-01**
*Susanne Biundo, Wolfgang Tank (Hrsg.)*
PuK-95, Beiträge zum 9. Workshop „Planen und Konfigurieren", Februar 1995
169 Seiten

**Note:** This document is available for a nominal charge of 25 DM (or 15 US-$).

## 1994

**D-94-15**
*Stephan Oepen*
German Nominal Syntax in HPSG
— On Syntactic Categories and Syntagmatic Relations —
80 pages

**D-94-14**
*Hans-Ulrich Krieger, Ulrich Schäfer*
TDL - A Type Description Language for HPSG, Part 2: User Guide.
72 pages

**D-94-12**
*Arthur Sehn, Serge Autexier (Hrsg.)*
Proceedings des Studentenprogramms der 18. Deutschen Jahrestagung für Künstliche Intelligenz KI-94
69 Seiten

**D-94-11**
*F. Baader, M. Buchheit, M. A. Jeusfeld, W. Nutt (Eds.)*
Working Notes of the KI'94 Workshop: KRDB'94 - Reasoning about Structured Objects: Knowledge Representation Meets Databases
65 pages

**Note:** This document is no longer available in printed form.

**D-94-10**
*F. Baader, M. Lenzerini, W. Nutt, P. F. Patel-Schneider (Eds.)*
Working Notes of the 1994 International Workshop on Description Logics
118 pages

**Note:** This document is available for a nominal charge of 25 DM (or 15 US-$).

**D-94-09**
*Technical Staff*
DFKI Wissenschaftlich-Technischer Jahresbericht 1993
145 Seiten

**D-94-08**
*Harald Feibel*
IGLOO 1.0 - Eine grafikunterstützte Beweisentwicklungsumgebung
58 Seiten

**D-94-07**
*Claudia Wenzel, Rainer Hoch*
Eine Übersicht über Information Retrieval (IR) und NLP-Verfahren zur Klassifikation von Texten
25 Seiten

**Note:** This document is no longer available in printed form.

**D-94-06**
*Ulrich Buhrmann*
Erstellung einer deklarativen Wissensbasis über recyclingrelevante Materialien
117 Seiten

**D-94-04**
*Franz Schmalhofer, Ludger van Elst*
Entwicklung von Expertensystemen: Prototypen, Tiefenmodellierung und kooperative Wissensevolution
22 Seiten

**D-94-03**
*Franz Schmalhofer*
Maschinelles Lernen: Eine kognitionswissenschaftliche Betrachtung
54 Seiten

**Note:** This document is no longer available in printed form.

**D-94-02**
*Markus Steffens*
Wissenserhebung und Analyse zum Entwicklungsprozeß eines Druckbehälters aus Faserverbundstoff
90 pages

**D-94-01**
*Josua Boon (Ed.)*
DFKI-Publications: The First Four Years
1990 - 1993
75 pages

**PIMaS:**
**Ein objektorientiert-regelbasiertes System zur Produkt-Prozeß-Transformation**

**D-97-07**
Document

Harold Boley,
Bernd Bachmann, Christian Blum, Christian Embacher,
Andreas Lorenz, Jamel Zakraoui