Beschreibungsmodelle für die Ablaufplanung für den Einsatz in Datenbanken

Birthe Gebhardt
b.gebhardt@scriptsite.de

Übersicht Nicht jedes Planungssystem kann bei jeder Planungsdomäne eingesetzt werden. Die Zielsetzungen der einzelnen Domänen, wie Dienstleistung oder Produktion, sind unterschiedlich. Wird ein Planungssystem neu entwickelt, müssen neben der Klassifizierung des Problems auch Punkte über die semantische und syntaktische Darstellung erörtert werden. In diesem Artikel werden die drei Bereiche angesprochen.

Einleitung

Unternehmen aller Wirtschaftszweige haben eins gemeinsam: sie bilden ihre Abläufe in Plänen ab. Diese Darstellungsform bietet einen einfachen Überblick über die Zuordnung von Ressourcen zu Aktionen. Die Reihenfolgen der Aktionen werden durch die Balkendarstellung in Plänen in grafischer Form wiedergegeben. Für die Darstellung von Plänen werden Informationen über Ressourcen, Aktionen und Regeln benötigt.

Der Einsatz eines EDV-gestützten Systems benötigt die Informationen der Planungsdomäne übersetzt in ein Modell. Die Modellierung kann mit einem mathematischen Ansatz aus dem Operation Research (OR) Bereich oder einen Ansatz aus der Künstlichen Intelligenz (KI) realisiert werden. OR Modelle reduzieren das Planungsproblem auf wohl definierte statische Optimierungsprobleme, die mit mathematischen Programmiertechniken gelöst werden können. In der KI wird die Ausdrucksmächtigkeit einer Domäne zwischen Aktionsformalismus, wo der Schwerpunkt in einer ausdrucksstarken Sprache liegt, und der effizienten Umsetzung für Planungsmethoden unterschieden.

Forschungen in der domänen-unabhängigen KI Planung und Ablaufplanung befasst sich traditionell mit der Entwicklung von Algorithmen, die eine effiziente Lösung für das Planungsproblem in der Domäne suchen (Simpson et al.).

Dieser Artikel diskutiert Möglichkeiten für den Einsatz von Modellen unter dem Gesichtspunkt der Verwendung einer Datenbank. Im nachfolgenden wird auf die Planungsdomäne mit deren unterschiedlichen Bereichen eingegangen. Anschließend werden Beschreibungsmodelle aus den Bereichen der mathematischen Modelle, der Objekt Modelle und der Sprachen Modelle vorgestellt. Der dritte Teil befaßt sich mit der Auswahl eines geeigneten Datenbanksystems.

Planungsdomäne

Eine Planungsdomäne bildet das Umfeld des Einsatzgebietes für die Ablaufplanung ab. In (PINEDO, 2005) werden zwei Bereiche von Modellen im einzelnen unterschieden. Ein Bereich beschreibt die Klassenmodelle für das produzierende Gewerbe, der zweite Bereich befasst sich mit Modellen für den Dienstleistungs Sektor. Die Modelle unterscheiden sich im wesentlichen durch die Handhabung von Jobs bei Verspätungen und Ausfällen. In dem produzierenden Gewerbe werden Jobs selten gelöscht. Verspätungen von Aufträgen werden bei Problemen in den Ablaufplänen als Lösung angewandt. Auf dem Dienstleistung Sektor verhält es sich genau andersherum. Kunden akzeptieren selten Verzögerungen. Streichungen von Dienstleistungen werden eher vorgenommen, zum Beispiel durch Flugausfälle. In den nachfolgenden Abschnitten werden die beiden Domänen ausführlicher beschreiben.

Produzierendes Gewerbe

Die Klassenmodelle für das produzierende Gewerbe sind unterteilt in:

1) Projektplanung
2) Job Shop
3) Produktionssystem mit automatischer Material Beförderung
4) Massenherstellung
5) Lieferkette

Die Klassen nehmen an Komplexität zu.

In der Projektplanung wird das Projekt/der Job in Teilschritte/Operationen unterteilt. Die einzelnen Operationen sind voneinander abhängig. Ziel in der Projektplanung ist die Reduzierung der Projektzeit. Die Abbildung in der Projektplanung ist Job orientiert, d.h. dass eine Ressource einem Job zugeordnet wird.

Im Gegensatz dazu wird beim Einsatz eines Job Shops die Darstellung Maschinen orientiert angezeigt. Die Maschinen werden als Ressource neben der Zeitachse als Spalte definiert. Die Jobs werden als Balken in dem Gantt-Diagramm angezeigt. In dieser Klasse werden die Operationen eines Jobs auf einer oder mehreren Maschinen ausgeführt. Es können mehrere Ziele definiert werden, wie zum Beispiel die Reduzierung der Herstellungszeit und die Reduzierung von verspäteten Jobs.

Eine Erweiterung des Job Shops bildet ein Produktionssystem mit automatischer Material Beförderung. Der Einsatz eines Förderbands beeinflusst die Beförderungszeit des Material und die Bearbeitungszeit an den Maschinen. Das Ziel bei dem Einsatz dieser Klasse ist es, die Durchsatzzeit zu erhöhen.

Der Einsatz einer Massenfertigungs Klasse wird für mittelfristige und langfristige Produktionsplanung verwendet. Der Produktionsprozess ist fortlaufend. In dieser Klasse werden unterschiedliche Produkte hergestellt. Das führt zu einer Erhöhung der Durchlaufzeit durch den zusätzlichen Zeitverlust für die Umrüstung von Maschinen. Das Ziel bei dem Einsatz dieser Klasse ist es, die Lagerzeit von Produkten zu verringern und die Umrüstzeit der Maschinen zu minimieren.

Die letzte Klasse bildet die Lieferkette ab. Hierbei handelt es sich um ein hierarchisches Modell, dass auf den Klassen 2 und 4 basiert.

Dienstleistungsbereich

Die Klassen in dem Dienstleistungsbereich sind in (PINEDO, 2005) wie folgt definiert:

1) Projektplanung
2) Zeitplanung
3) Gruppen-Planung
4) Transport-Planung
5) Personal-Planung

Die Projektplanung wird in beiden Bereichen des produzierenden Gewerbes und der Dienstleistung eingesetzt. Die Handhabung unterscheidet sich hierbei nicht.

Im Unterschied zu der Projektplanung ist in der Zeitplanung die Ausführungszeit fest vorgegeben und kann nicht verändert werden. Unterbrechungen werden nicht eingesetzt. Die Stopzeit ist durch die definierte Ausführungszeit und einer Startzeit bestimmt. Die Startzeit kann variieren, ist jedoch in der Regel fest, wie zum Beispiel in einem Schulstundenplan. Eine variable Startzeit findet man unter anderem in der Autovermietung.

Die Gruppen-Planung wird bei Wettkämpfen oder auch in der Schichtplanung, die auf Gruppen basiert, eingesetzt. Mehrere Gruppen müssen den gleichen Job ausführen. Vergleichbar ist es mit dem Einsatz von parallelen Maschinen in dem produzierenden Gewerbe. Zusätzlich zu der Zeitplanung der Gruppen und den Jobs müssen noch Regeln beachtet werden, die Einfluß auf die Erstellung des Plans haben. So müssen sich, zum Beispiel im Fußball Heimspiele und Auswärtsspiele im Plan abwechseln.

Die Durchführung eines Fluges oder einer Bahnfahrt wird in der Klasse für die Transport-Planung abgebildet. Der Job muss in einem definierten Zeitfenster mit einer Ressource durchgeführt werden. Für die Aneinanderreihung von Jobs müssen Ausführungsrestriktionen und Regeln beachtet werden. Durch die dadurch entstehende Rundreise von Flügen oder Bahnfahrten können die Kosten minimiert und der Profit maximiert werden. Das Problem des Handlungsreisenden ist ein bekanntes Beispiel für diese Klasse.

Die Personal-Planung ist eine wichtige Klasse in dem Dienstleistungbereich. Die Personalplanung ordnet einem Zeitinterval eine Anzahl von Personen zu oder Personen werden einer Aufgabe zugeordnet. Im ersten Fall wird die Schichtplanung abgebildet, die unter anderem im Einzelhandel oder Service Bereich verwendet wird. Der zweite Fall stellt die Aufgabe in den Mittelpunkt. Ein Beispiel wäre die Dienstleistung eines Beraters für ein Projekt oder die Erstellung eines Gewerks von einem Entwickler.

Fazit

Die Unterteilung und Klassifizierung der Planungsdomäne ist für die Auswahl eines geeigneten Hilfsmittels für die Erstellung und Reparatur eines Plans unablässig. Die dargestellten Klassen bieten eine Übersicht und Einsortierung des zu bearbeitenden Problems an. Es wurde aufgezeigt, dass Klassen in den beiden Bereichen - produzierendes Gewerbe und Dienstleistungen - ableitbar sind und sich in der Klasse Projektplanung überschneiden.

Die Klassifizierungen arbeiten mit Aufträge (Jobs), Regeln und Ressourcen (Maschinen, Material, Personal, etc.), die zu jeder Zeit einen Zustand besitzen. Aufträge führen eine Aktion auf bzw. mit einer Ressource aus.

Zum Beispiel, eine Maschine hat den Auftrag eine Stahlplatte zu walzen. Die Ressource *Maschine* führt den Auftrag mit der Aktion *Walzen* auf der Ressource *Stahlplatte* aus. Ein Beispiel aus dem Dienstleistungsbereich, der Auftrag *Flug* kann mit der Ressource *Flugzeug* und der Aktion *Fliegen* auf die Ressource *Fluggast* angewendet werden.

Regeln können sich signifikant in der Komplexität unterscheiden. Eine Einheitlichkeit für eine Domänen-unabhängige Lösung zu erzielen ist kompliziert.

Eine Domänen-unabhängige Lösung zu finden ist das Ziel des automatischen Planens (GHALLAB ET AL., 2004). Die Aufwände jedes einzelnen Domänproblems können durch den Einsatz bzw. die Adaption eines allgemeinen Werkzeugs reduziert werden. Das automatische Planen schließt das Domänen-spezifische Planen nicht aus. Allgemeine Ansätze werden in ein spezifisches Domänproblem integriert, sodass in einigen Bereichen eine Wiederverwendbarkeit erfolgen kann.

Beschreibungsmodelle

Die Darstellung der Eigenschaften für die Ablaufplanung ist entscheidend für die Abbildung als Modell in einem System. In den nachfolgenden Abschnitten werden unterschiedliche Ansätze der Beschreibungsmodelle aufgezeigt.

Mathematisches Modell

Mathematische Modelle werden für die Optimierung von Planungsabläufen in dem OR Bereich eingesetzt. Die Tripel Notation $\alpha|\beta|\gamma$ mit der Zuordnung für α als Maschinenbeschreibung, β als Jobbeschreibung und γ als Optimierungkriterium wird hierbei verwendet (BRUCKER, 2007)(GRAHAM ET AL., 1979). Nachfolgend werden die drei Gruppen näher beschrieben.

Machinenbeschreibung α

Die Maschinenbeschreibung α setzt sich aus zwei Parametern α_1 und α_2 zusammen. Der erste Parameter beschreibt die Maschineneigenschaft, der zweite gibt die Anzahl der Maschinen an. Wenn $\alpha_2 > 0$ und ganzzahlig ist, wird der Parameter als Konstante behandelt. α_2 wird als Variable angesehen, wenn keine Informationen vorliegen und α_1 angegeben ist, d.h. $\alpha_1 \neq \circ$.
α_1 kann die folgenden Eigenschaften annehmen:

Parameter α_1	Beschreibung	Ausführungszeit p Jobs(1..j), Maschinen(1..i)
∘	einzelne Maschine In diesem Fall ist α_2=1	$p_{1j} = p_j$
P	identische, parallele Maschinen	$p_{ij} = p_j$

Parameter α_1	Beschreibung	Ausführungszeit p Jobs(1..j), Maschinen(1..i)
Q	einheitliche, parallele Maschinen	$p_{ij} = q_i * p_j$ q_i : Maschinengeschwindigkeit
R	unabhängige, parallele Maschinen	$p_{ij} = q_{ij} * p_j$ q_{ij} : Maschinengeschwindigkeit pro Job j
PMPM	identische Mehrzweck Maschinen	
QMPM	einheitliche Mehrzweck Maschinen	
G	General Shop Es besteht eine Reihenfolgenbedingung zwischen den Operationen eines Jobs	Bei Verwendung dieser Parameter bestehen die Jobs aus mehreren Operationen.
O	Open Shop Alle Jobs verwenden die Maschinen in beliebiger Reihenfolge	
F	Flow Shop Alle Jobs verwenden die Maschinen in gleicher Reihenfolge	
J	Job Shop Jeder Job verwendet seine eigene Reihenfolge für die Maschinen	
X	Mixed Shop Kombination von Open Shop und Job Shop	

Tabelle 1: Maschinenbeschreibung α

Jobbeschreibung β

Die Jobbeschreibungen, zusammengefasst zu der Gruppe β, geben die Eigenschaften von Jobs an. Bei β unterscheidet man 6 wichtige Charakteristiken. Wird ein Parameter $\beta_2 - \beta_6$ nicht verwendet, wird dieser in der Gruppe β nicht aufgenommen.

Parameter	Beschreibung
β_1	Unterbrechungen von Jobs
pmtn	Unterbrechungen von Jobs sind erlaubt
β_2	Vorrangsbeziehung zwischen den Jobs
prec	Abhängigkeiten von Jobs werden in einem azyklisch gerichteten Graphen dargestellt G=(V,A). **prec** wird gesetzt, wenn G ein beliebiger azyklisch gerichteter Graph ist.

Parameter	Beschreibung
intree	G entspricht einer Baumstruktur, wobei die Knoten mehrere ankommende Verbindungen besitzen und nur eine ausgehende.
outtree	G entspricht einer Baumstruktur, wobei die Knoten mehrere ausgehende Verbindungen besitzen und nur eine ankommende.
tree	G entspricht einer Baumstruktur, entweder ein **intree** oder ein **outtree**.
chain	Eine Kette entspricht einer Baumstruktur, wobei jeder Knoten genau eine ankommende und eine ausgehende Verbindung besitzt.
sp	Series-parallel wird ein Graph genannt, wenn - nur ein Knoten existiert (Base graph) - $G=(V_1 \cup V_2, A_1 \cup A_2)$ zusammengeformt werden kann aus G_1 und G_2 (Parallel composition) durch gemeinsame Knoten und Verbindungen. - $G=(V_1 \cup V_2, A_1 \cup A_2 \cup T_1 \times S_2)$ zusammengeformt werden kann aus G_1 und G_2 (Series composition) durch gemeinsame Knoten und Verbindung und Hinzufügen von allen Verbindungen (t,s), mit $t \in T_1$, T_1=Senken von G_1, z.B Knoten ohne abgehende Verbindung und $s \in S_2$, S_2=Quelle von G_2, z.B. Knoten ohne ankommende Verbindung
β_3	Startdatum muss für jeden Job angegeben werden
r_i	Startdatum für Job J_i
β_4	Angabe von Einschränkungen in der Ausführungszeit oder der Anzahl von Operationen
p_i	Jeder Job/Operation besitzt eine einheitliche Ausführungs-anforderung
β_5	Fertigstellungsdatum muss für jeden Job angegeben werden
d_i	Job J_i darf nicht später als d_i fertiggestellt werden.
β_6	Jobs werden zu einer Gruppe zusammengefasst und diese gemeinsam zu verarbeiten
p-batch	Die Zeitdauer der Gruppe ist gleich dem Maximum der Ausführungszeit der Gruppe.
s-batch	Die Zeitdauer der Gruppe ist gleich der Summe der Ausführungszeiten aller Jobs in der Gruppe.

Tabelle 2: Jobbeschreibung β

Optimierungkriterium γ

Das Optimierungskriterium kann als Zielfunktion in der Lösung eines Ablaufplanungsproblems angesehen werden. Die Optimierungsfunktion G_i für γ zeigt

den Engpass oder das Ergebnis auf. Für eine Engpass-Funktion wird das Maximum bzw. das gewichtete Maximum ermittelt. Die Ergebnisfunktion liefert die Summe bzw. die gewichtete Summe zurück.

$$\gamma := \begin{cases} \max G_i \\ \max \omega_i G_i \\ \sum G_i \\ \sum \omega_i G_i \end{cases}$$

Für die Funktion G_i werden unter anderem folgende Berechnungen verwendet:

Name		Funktion
Produktionsdauer	C_i	
Fertigstellungsdatum	d_i	
Produktionsdauer	makespan	$\max\{C_i \mid i=1,..,n\}$
Gesamtdurchlaufzeit	total flow time	$\sum C_i$, mit i=1,..,n
Verspätung	lateness	$L_i := C_i - d_i$
Frühzeitigkeit	earliness	$E_i := \max\{0, d_i - C_i\}$
Verzug	tardiness	$T_i := \max\{0, C_i - d_i\}$
absolute Differenz	absolute deviation	$D_i := \lvert C_i - d_i \rvert$
quadratische Differenz	squared deviation	$S_i := (C_i - d_i)^2$
Anzahl Verspätungen	unit penalty	$U_i := \begin{cases} 0; C_i \le d_i \\ 1; \text{sonst} \end{cases}$

Tabelle 3: Optimierungkriterium γ

Objekt Modell

Die Verwendung von Objekt-Modellen erlaubt es bekannte OO-Methoden zu verwenden. Neben einer einfachen und klaren Darstellung von Informationen in Kategorien, können bekannte grafische Darstellungsmodelle, wie zum Beispiel UML, verwendet werden. Das objektorientierte Modell lässt sich leicht in ein relationales überführen, was eine Implementierung in eine Datenbank ermöglicht.

In diesem Abschnitt werden zwei Möglichkeiten aufgezeigt.

In (SAUER, 1993) wird das Ablaufplanungsproblem durch die Modellierungs-möglichkeiten von Objekten und Regeln in einem 5-Tupel beschrieben. Grundlegende Objekte, die in einem Plan definiert werden müssen, sind Ressourcen, Aufträge und Produkte. Regeln bzw. Constraints beschreiben die Bedingungen, an die sich die Objekte in einem Plan halten müssen. Die Constraints werden in Hard und Soft

Constraints unterteilt, die sich darin unterscheiden, dass Hard Constraints unbedingt eingehalten werden müssen. Soft Constraints können, müssen aber nicht erfüllt werden.

Das 7-Tupel ist eine Erweiterung der 5-Tupel Darstellung. Neben den bereits beschriebenen Komponenten des 5-Tupels, besteht die Erweiterung aus der Hinzunahme der Zielfunktionen und Ereignissen (SAUER, 2004). Das 7-Tupel (R,P,A,HC,SC,Z,E) ist wie folgt aufgebaut:

1. Ressourcen $R=\{r_1, r_2, r_3, \ldots, r_n\}$
   Die einzelnen Ressourcen werden zu einer Menge zusammengefasst.
2. Produkte $P=\{p_1, p_2, p_3, \ldots, p_n\}$
   Die einzelnen Produkte werden zu einer Menge zusammengefasst. Ein Produkt wird beschrieben durch seinen Herstellungsablauf, der die auszuführenden Operationen mit den benötigten Ressourcen beinhaltet.
3. Aufträge $A=\{a_1, a_2, a_3, \ldots, a_n\}$
   Die einzelnen Aufträge werden zu einer Menge zusammengefasst. Ein Auftrag legt die Menge der herzustellenden Produkte, die Start- sowie die Endzeit fest.
4. Hard Constraints $HC=\{hc_1, hc_2, hc_3, \ldots, hc_n\}$
   Die einzelnen Hard Constraints werden zu einer Menge zusammengefasst.
5. Soft Constraints $SC=\{sc_1, sc_2, sc_3, \ldots, sc_n\}$
   Die einzelnen Soft Constraints werden zu einer Menge zusammengefasst.
6. Zielfunktion $Z=\{zf_1, zf_2, zf_3, \ldots, zf_n\}$
   Die einzelnen Ziel- bzw. Bewertungsfunktionen werden zu einer Menge zusammengefasst. Die Zielfunktion dient zur Bewertung des Ablaufplans.
7. Ereignisse $E=\{e_1, e_2, e_3, \ldots, e_n\}$
   Die einzelnen Ereignisse werden zu einer Menge zusammengefasst. Ereignisse lösen Zustandsänderungen in einem Ablaufplan aus.

Diese Informationen beschreiben das Ablaufplanungsproblem und dienen der Suche nach einen konsistenten Plan.

Auf der ICKEPS(**I**CAPS **C**ompetition on **K**nowledge **E**ngineering for **P**lanning and **S**cheduling), die zusammen mit der ICAPS(**I**nternational **C**onference on **A**utomated **P**lanning & **S**cheduling) in Monterey/USA 2005 durchgeführt worden ist, wurde GIPO in der Kategorie 'General Tools' ausgezeichnet. GIPO(**G**raphical **I**nterface for **P**lanning with **O**bjects) ist ein experimentelles Entwicklungstool für die Darstellung einer Planungsdomäne (SIMPSON, 2005) (SIMPSON, 2007). Die Planungsdomäne wird mit Hilfe der objektorientierten Methode beschrieben. Der Ausgangpunkt bei diesem Ansatz ist, dass die Planausführung eine Manipulation bzw. Änderung des Zustandes von Objekten durchführt.

Der Aufbau einer Domäne erfolgt in 6 Schritten:

1. Beschreibung der Domäne
2. Anlegen von Kategorien und Objekten (Sorts). Die Objekte stellen die Ressourcen dar.
3. Festlegen der Prädikate (Predicates)
4. Erstellen aller Zustände (States)
5. Anlegen von Aktionen (Operator) mit Hilfe der erstellten Zustände
6. Aufgaben (Tasks) definieren, d.h. Anlegen eines initialen Zustandes und des Zielzustandes.

Einfache Constraints können durch Beziehungen der Prädikate spezifiziert werden. Komplexe Constraints können mit Fluents realisiert werden oder durch Anforderungen zum Quantifizieren von Eigenschaftswerten oder Beziehungen. Eine Unterscheidung zwischen Hard und Soft Constraints wird nicht vorgenommen.

Die Aktivierung des *Durative Action* Modus erlaubt es zu Aktionen Zeiten und Effekte zum Start und zum Ende hinzuzufügen. Der Zeitpunkt, wann ein Prädikat seinen Wert verändern kann, wird zu einem Constraint hinzugefügt.

Domänen im *Durative Action* Modus haben folgende Möglichkeiten:

- Objekte können nummerische Eigenschaften besitzen (*fluents*)
- Fluents können getestet und geändert werden, wenn eine Aktion durchgeführt wird.
- Aktionen können als Events definiert werden
- Ermittlung wie Fluents ein Event auslösen können
- Anlegen von Prozessen
- Prozesse können Fluents zeitabhängig testen und ändern

GIPO bietet die Möglichkeit, die Beschreibung der Domänen nach PDDL zu exportieren und ebenfalls PDDL Scripte zu importieren. Planer können über eine API eingebunden werden.

Sprachen Modell

Sprachen Modelle werden in unterschiedlichen Bereichen der Planung eingesetzt. Je nach Bereich überwiegt die Ausdrucksstärke -Aktionsformalismus- oder die Effizienz -Planungsmethoden- . Zu diesen beiden bekannten Bereichen wird durch die Arbeiten an Agenten in der Ablaufplanung ein weiterer Bereich hinzugefügt.

Der Aktionsformalismus und die Planungsmethoden besitzen mit den Arbeiten an dem Situationskalkül die gleichen Wurzeln (Green, 1969) . Durch das Planungssystem STRIPS (Fikes & Nilsson, 1971) wurden die Prioritäten Ausdrucksstärke und Effizienz getrennt weiter behandelt. In (S. Russell & Norvig, 2004) wird jedoch darauf hingewiesen, dass das Ziel einer Sprache beide Eigenschaften besitzen sollte, d.h. um eine Vielzahl von Domänen beschreiben zu können, sollte die Sprache ausdrucksstark sein, jedoch nur in einem gewissen Rahmen, sodass optimale Algorithmen eingesetzt werden können.

GOLOG abgeleitet von alGOL on LOGic (Levesque et al., 1997) ist eine der Sprachen basierend auf dem Aktionsformalismus. Die Erweiterung zu dem Situationskalkül basiert auf den zusätzlichen Möglichkeiten von Programm-konstrukten wie Rekursion, deterministische Verzweigungen und Prozeduren. Folgende Strukturen werden von GOLOG bereitgestellt :

1. Einfache Aktion: Die einfache Aktion wird mit dem Funktionsaufruf Do(a,s) aufgerufen. Die Funktion beschreibt die nachfolgende Situation, die von der Aktion a auf die Situation s ausgeübt wird.
2. Sequenz: Die Sequenz a;b besagt, das Do(b,s) nach Do(a,s') ausgeführt wird.

3. Test Aktion: Die Test Aktion prüft den Ausdruck p für die aktuelle Situation auf den Wahrheitswert. Dargestellt wird es mit p?.
4. Nicht deterministische Auswahl von zwei Aktionen: a|b bedeutet eine Oder Verknüpfung zwischen den Aktionen a und b.
5. Nicht deterministische Auswahl von Aktionsargumenten: (π x)a besagt, dass mindestens ein Wert für die Variable x existiert für die Ausführung der Aktion a.
6. Nicht deterministische Iteration: Mit dem Symbol a* wird die Aktion a mit einer unbestimmten Anzahl wiederholt.
7. Bedingung: Die Bedingung wird mit der Struktur **if** p **then** a **else** b **endIf** beschrieben.
8. Schleife: Als Schleife wird die **while** p **do** a **endWhile** verwendet.
9. Prozeduren

Erweiterungen der Sprache für neue Aufgabengebiete sind in Dialekten implementiert worden, unter anderem der zeitliche Aspekt (Reiter, 1998). Die Umsetzung von zeitabhängigen Aktionen erfolgt über den Einsatz von unmittelbaren Aktionen. Die Aktion Gehen(a,b) wird unterteilt in startGehen(a,b) und endGehen(a,b) und dem relationalen Fluent Gehen(a,b,s). Es wurden für die zeitliche Verarbeitung zwei Erweiterungen in der Sprache vorgenommen:

1) time(a): time(a) gibt die Zeit a, wann die Aktion a eintritt zum Beispiel time(startGehen(a,b,t))=t.
2) start(s): start(s) gibt die Startzeit für die Situation s an, start(do(a,s))=time(a)

Im Gegenzug zu GOLOG wurde 1998 eine Planungssprache entwickelt, die basierend auf STRIPS, den Fokus auf die Planungseffizienz und die Entwicklung einer Standardnotation für vergleichbare Probleme hat. PDDL ist die Abkürzung für **P**lanning **D**omain **D**escription **L**anguage (Ghallab et al., 1998). Aktuell liegt die Version 3.1 vor, vorgestellt auf der ICAPS 2008.

Folgende Erweiterungen wurden zu den International Planning Competition (IPC) Konferenzen in den Jahren hinzugefügt:

IPC1998	PDDL1.2	Initiale PDDL Version
IPC2002	PDDL2.1	Zeiten und Nummern, *numeric fluent*
IPC2004	PDDL2.2	Abgeleitete Prädikate und zeitlich initiale Literale
IPC2006	PDDL3.0	Präferenzen und *trajectory constraints*
IPC2008	PDDL3.1	*Object fluent* und neues *Requirement*

PDDL beschreibt die Beschaffenheit einer Domäne mit Hilfe von Eigenschaften, Aktionen und den Effekten. Das Planungsproblem wird separat zu der Domänenbeschreibung mit PDDL definiert. Die Notation der PDDL Definitionen basiert auf der erweiterten Backus Naur Form (EBNF).

Die erste Erweiterung PDDL2.1 fügt die Verarbeitung von Zeiten hinzu, um die Möglichkeit zu schaffen, realistischere Planungsprobleme zu verarbeiten wie zum Beispiel aus der Logistik oder der Fertigungswirtschaft. Die Erweiterungen werden in (Fox & Long, 2003) beschrieben. Die Zeiten werden mit durativen Aktionen in einer diskreten oder kontinuierlichen Form dargestellt.

Durative Aktionen besitzen ein weitere Angabe `:duration` für die Angabe der Dauer der Aktion. Ergänzt wird die `:duration` durch drei Annotationen, angewendet auf die Bedingungen und Effekte :

1) `at start` : Am Anfang
2) `at end` : Am Ende
3) `over all` : Unveränderliche Bedingung/Effekt während der gesamten Dauer der Aktion

Das nachfolgende Beispiel aus (FOX & LONG, 2003) zeigt den Einsatz der durativen Aktion bei der Aktion *Wasser kochen* :

```
(:durative-action heat-water
     :parameters (?p - pan)
     :duration (= ?duration (/ (- 100 (temperature ?p)) (heat-rate)))
     :condition (and (at start (full ?p))
                     (at start (onHeatSource ?p))
                     (at start (byPan))
                     (over all (full ?p))
                     (over all (onHeatSource ?p))
                     (over all (heating ?p))
                     (at end (byPan)))
     :effect (and   (at start (heating ?p))
                    (at end (not (heating ?p)))
                    (at end (assign (temperature ?p) 100)))
)
```

PDDL2.2 wurde um die zeitlich festgelegten initialen Literale ergänzt. Diese Literale erlauben es Fakten zu einem definierten Zeitpunkt WAHR oder FALSCH werden zu lassen. Ein einfaches Beispiel ist in (EDELKAMP & HOFFMANN, 2004) dargestellt :

```
(:init
     (at 9 (shop-open))
     (at 20 (not (shop-open)))
)
```

In dem Bereich der Agentensprachen wird GOAL in diesem Abschnitt behandelt. Weitere Agentsprachen sind Jason, Jade, 3APL oder auch JACK Intelligent Agents. GOAL ist ein Akronym für *G*oal-*O*riented *A*gent *L*anguage und wurde auf der ATAL 2000 (Agent Theories, Architectures, and Languages) vorgestellt (HINDRIKS ET AL., 2001). Die Programmiersprache wurde für rationale Agenten entwickelt, die Zustände des deklarativen Glaubens und Ziele pflegen, um daraus die Wahl der Aktionen abzuleiten.

Die wichtigsten Merkmale dieser Sprache sind:

- Deklarative Annahme: Die Informationen, die Agenten benötigen, werden in einer symbolischen logischen Sprache dargestellt. Die Wissensrepräsentations-Sprache ist in GOAL nicht fest definiert.
- Deklarative Ziele: Agenten können mehrere Ziele haben. Die Ziele definieren, was der Agent zu welchem Moment zu erreichen hat und damit den Zustand der Umgebung. Es werden nicht die Aktionen spezifiziert, die zum Erreichen der Zustände benötigt werden.
- *Blind Commitment* Strategie: Die Agenten verpflichten sich ihre Ziele zu erreichen und erst dann diese zurückzugeben.

- Regel basierte Aktionsauswahl: Aktionsregeln werden für die Auswahl von Aktionen von den Agenten verwendet, basierend auf den gegebenen Zielen und Annahmen.
- Rechte basiertes Intention-Modul: GOAL stellt Module für die Zusammenstellung von geeigneten Regelaktionen und Wissensinformationen bereit, die für das Erreichen eines spezifischen Ziels benötigt werden.
- Kommunikation auf der Wissensebene: Die Agenten kommunizieren mit Hilfe der Wissensrepräsentations-Sprache, die ebenfalls für die deklarativen Ziele und Annahmen verwendet wird.

Ein GOAL Agenten Programm beinhaltet sechs Abschnitte: *knowledge*, *belief*, *goals*, *action rules*, *action specification* und *percept rules*. Die Abschnitte *knowledge*, *belief* und *goals* werden in einer Wissensrepräsentation Sprache wie zum Beispiel Prolog oder PDDL dargestellt. Für die Darstellung von zeitlichen Aspekten wird in GOAL die lineare temporale Logik (LTL) verwendet (HINDRIKS ET AL., 2009). Die Verwendung von einer linearen zeitlichen Logik kann das Erfolgsziel als ◇ (eventually) Operator, das Erhaltungsziel als □ (always) definieren. In beiden Fällen wird der unbegrenzte Aspekt behandelt. Agenten besitzen eine limitierte Anzahl von Zeiteinheiten und Ressourcen für Schlussfolgerungen und sind daher begrenzt rational. Das heißt, sie können nicht weit in die Zukunft sehen, um die bestmöglichste Aktion auszuwählen, daher muss ein begrenzter Horizont definiert werden, was eine eingeschränkte Schlussfolgerung zur Folge hat.

Für den Einsatz eines eingegrenzten Erfolgsziels wird anstelle des ◇ Operator die Syntax **before** verwendet, z.B. *atWork* **before** *09:00*. Eine äquivalente Syntax zu dem □ Operator wird **until** verwendet, Beispiel *fuel* **until** *atGasStation*.

Die drei dargestellten Sprachen werden in drei unterschiedlichen Bereichen eingesetzt, im Repräsentation Bereich, in einem Planer und im Agenten Bereich. Unterschiede zwischen den drei Sprachen wurden in der Forschung untersucht.

(EYERICH ET AL., 2006) hat die Ausdrucksfähigkeit zwischen GOLOG und PDDL mit Hilfe des Kompilations-Ansatz untersucht. Bei diesem Ansatz werden zwei Sprachen mit gleicher Ausdrucksfähigkeit betrachtet, um die Planungs Domäne und die Pläne in beiden Formalismen ohne erhöhten Aufwand abbilden zu können. Um die Ausdrucksfähigkeit zwischen GOLOG und PDDL gleichzusetzen, wird GOLOG auf das ADL Fragment in PDDL reduziert, wie auch PDDL. Die Reduzierung auf ADL ergab, dass das ADL Fragment von PDDL die gleiche Ausdrucksfähigkeit besitzt, wie ein syntaktisches Fragment in GOLOG. Die Diskrepanz zwischen einer Planungssprache und einem Aktionsformalismus kann überbrückt werden. Aus praktischer Sicht hat das Ergebnis die Möglichkeit geschaffen, einen effizienten Planungsalgorithmus in GOLOG zu implementieren.

Der Vergleich zwischen PDDL und GOAL wurde in (HINDRIKS & ROBERTI, 2006) untersucht. Eine Einschränkung erfolgte nur auf PDDL Seite, indem Fragmente bestehend aus den Axiomen, ADL und temporalen Plan Constraints verwendet wurden. Dieser Vergleich verbindet die Stärken eines Planers mit der Flexibilität einer Agenten Programmierung. Das Resultat zeigt, dass GOAL als Planungsformalismus verwendet werden kann, unter Zuhilfenahme eines Kompilations-Schemas, welches bereits in dem Vergleich zwischen GOLOG und PDDL verwendet worden ist (RÖGER ET AL., 2008).

Abschließend lässt sich feststellen, dass die Sprachen sich untereinander ergänzen, was auch die Forschungsergebnisse widerspiegeln.

Fazit

In diesem Abschnitt wurden drei unterschiedliche Beschreibungsansätze dargestellt, der mathematische Ansatz, der objektorientierte Ansatz und die Verwendung einer Sprache.

Ein mathematischer Ansatz vereinfacht eine Implementierung in ein Datenbanksystem gegenüber einer Sprache, da hierbei bereits eine einfache feste Struktur vorliegt.

Zu einem Problem wird ein mathematisches Modell erstellt, das eine optimale Lösung erzeugt. Das mathematische Modell wird in einem hohen Abstraktionsgrad erstellt, welches der realen Problemstellung nicht vollständig entspricht (Sauer, 1993). Eine Sprache in eine Datenbank zu implementieren erfordert einen erhöhten Aufwand. Eine Erweiterung von PDDL für Ontologien -Web-PDDL- (Dou) wurde mit Hilfe eines Wrappers zwischen der Ontologie-Abfrage Sprache und der datenbankseitigen integrierten Abfragesprache SQL in ein relationales Datenbanksystem integriert (LePendu et al., 2007).

Durch die Möglichkeit des Einsatzes von OO-Methoden bei einer Verwendung eines objektorientierten Modells ist bei dieser Wahl eine Integration in eine Datenbank unkomplizierter. In (Pinedo, 2005) dargestellten Systemen setzen auf dem objektorientierten Ansatz auf. Der Ansatz basiert überwiegend auf der Verwendung von objektorientierten Entwicklungen, nicht so sehr in der Beschreibung eines Plans. Ein Plan mit seinen Parametern ist ebenso ein Objekt wie Schaltflächen und Felder auf einer Benutzeroberfläche.

Darstellungsmodelle

Im Nachfolgenden werden unterschiedliche Darstellungsmodelle für die Speicherung von Informationen vorgestellt. Ziel ist es, ein geeignetes Modell zu finden, um alle bekannten semantischen Informationen abspeichern zu können, sodass ein nahezu vollständiges Abbild in einer strukturierten Form zur Verfügung steht.

Die Darstellungsmodelle lehnen sich an die Datenbanksysteme an. Die Entwicklung der Datenmodelle begann mit dem hierarchischen Modell über das Netzwerkmodell und das relationale Modell zu dem semantischen Modell. Das semantische Modell gliedert sich in ein objekt-relationales Datenmodell und ein objektorientierte Datenmodell auf.

Ältere Systeme wie das hierarchische Datenmodell oder auch das Netzwerkmodell werden hier nicht berücksichtigt.

Relationales Darstellungsmodell

Relationale Datenbanken werden in jedem größeren Unternehmen für die Speicherungen der Geschäftsinformationen eingesetzt. Das Konzept des relationalen Datenmodells basiert auf der Mengentheorie.

Die Informationen werden für die semantische Datenmodellierung mit Hilfe eines Entity-Relationship Modells (ER Modell) in ein Datenbank Design überführt. Zu den bekanntesten Notationen für die Erstellung eines Entity-Relationship Diagramm (ERD) gehört die UML Notation und die Barker und Bachman Notation.

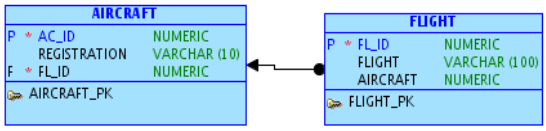

Abbildung 1: Bachman Notation

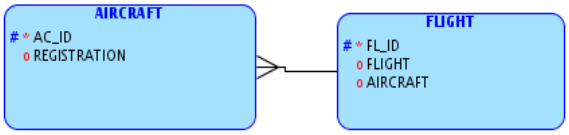

Abbildung 2: Baker Notation

Die relationale Darstellung sieht wie folgt aus:

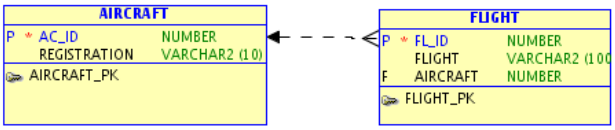

Abbildung 3: Relationale Darstellung

Die erstellten Relationen beinhalteten die Tupel. In einem Tupel wird genau ein Wert jedem Attribut zugeordnet. Die Tupel besitzen in einer Relation keine Ordnung.

Der Vorteil bei dieser Art der Informationsspeicherung ist es, dass Erweiterungen an dem Modell vorgenommen werden können, ohne dass die bestehenden Anwendungen beeinträchtigt werden. Unterschiedliche logische Strukturen können mit Hilfe von Views erstellt werden, basierend auf mehreren Tabellen. Die Datenbankstruktur muss hierfür nicht verändert werden. Dadurch wird eine hohe Flexibilität erreicht.

EAV Modell

Das Entity-Attribute-Value (EAV) Modell (Grechenig et al., 2009) bietet eine große Flexibilität. Anstelle einer relationalen Verarbeitung von Informationen wird

das logische Schema durch Entitäten, Attribute und Werte abgebildet. Das erlaubt einen hohen Grad an Flexibilität. Das bedeutet auch, dass nicht alle Informationen bei der Erstellung des Modells vorliegen müssen.

Die Umsetzung des EAV Modells erfolgt durch den Einsatz einer Tabelle. Folgende Spalten beinhaltet die Tabelle:

i) Entität – Speicherung des Objekt Namens
ii) Attribut – Eigenschaften des Objekts
iii) Wert – Wert des Attributes für das Objekt

Die Spaltendarstellung des relationalen Modells wandelt sich um in eine Zeilendarstellung.

Name	Spalte1	Spalte2	Spalte3
Name1	Wert1	Wert2	Wert3

→

Name	Spalte	Wert
Name1	Spalte1	Wert1
Name1	Spalte2	Wert2
Name1	Spalte3	Wert3

Abbildung 4: EAV Modell - Tabellendarstellung

In (ANHØJ, 2003) wurde das EAV Modell für die Darstellung einer Klinik Domäne in einer relationalen Datenbank eingesetzt. Das damalige Design machte eine schnelle Reaktion auf Erweiterungen der Domäne schwierig. Ziel der Modelländerung war es, bei Erweiterungen nicht das physische Datenmodell ändern zu müssen.

Nachteil der Realisierung mittels dem EAV Datenmodells war es, dass es bei größeren Datenbanken zu Performance Problemen führen kann. Durch die Flexibilität werden SQL-Anweisungen umfangreicher, was eine Beeinträchtigung der Ausführungszeit mit sich bringt. Ein weiterer Nachteil ist es, dass vorhandene DBMS Funktionen wie zum Beispiel Constraints und Indexes nicht verwendet werden können.

Für den Einsatz eines EAV Modells in einem relationalen Modell kann die Tabelle „gedreht“ werden, d.h. die Pivot Funktion wird auf diese Daten angewendet.

Für einen effizienten und flexiblen Einsatz des EAV Modells ist es sinnvoll, nur einen Teil des Datenmodells variabel zu halten und das restliche Modell in ein relationales Modell zu überführen.

Objektorientiertes Darstellungsmodell

Das objektorientierte Darstellungsmodell ermöglicht die Verwaltung von Daten als Objekte in einer Datenbank. Konzepte aus der objektorientierten Programmierung wurden in die objektorientierten Datenbanken übernommen. Dazu zählen die Definition von Objekten als Instanzen von Klassen, die Kapselung und Vererbung sowie die Polymorphie. Diese Umsetzung in ein Datenbanksystem ermöglicht die Speicherung und Handhabung der Daten auf die selbe Art und Weise wie in der Programmierung. Dies verringert den Programmieraufwand und verbessert die Performance.

Aus der relationalen Tabelle FLIGHT aus dem vorherigen Abschnitt wird das folgende Objekt definiert. Neben den Attributen können Methoden hinzugefügt werden.

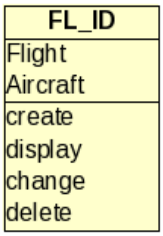

Abbildung 5: FLIGHT Objekt

Die Umsetzung von Planungssystemen erfolgt mit Hilfe einer objektorientierten Programmiersprache wie C++ oder Java. In (PINEDO, 2005) wurde aufgezeigt, dass überwiegend relationale Datenbanken im Einsatz sind. Die Aussage spiegelt sich in der Praxis wieder, da überwiegend mit einfachen strukturierten Daten gearbeitet wird und die gesteigerte Komplexität bei dem Einsatz von Objekten eher ein Nachteil darstellt.

Von 1993 bis 2001 hat die ODMG(Object Data Management Group) an einem Standard für die objektorientierte Datenbank gearbeitet. Aktuell liegt die Version 3.0 vor (CATTELL ET AL., 2000) .

Die zweite Möglichkeit eines objektorientierten Datenbanksystems bietet das objekt-relationale Modell an. Hierbei wird ein relationales Datenbanksystem durch objektorientierte Eigenschaften erweitert. Objekt-relationale Datenbankmanagement-systeme (ORDBMS) ermöglichen die Speicherung selbstdefinierter komplexer Datentypen in einzelnen Tabellenzellen einer relationalen Datenbank. Dies ermöglicht eine Objekt-Schicht, die zwischen dem Business Objekt und der relationalen Tabelle implementiert ist.

XML Darstellungsmodell

Die eXtensible Markup Language (XML) ist eine Auszeichnungssprache zur Darstellung hierarchischer strukturierter Informationen. XML Texte können in Textdateien und Datenbanken abgespeichert werden. XML Datenbanken werden unterschieden in zwei Kategorien. Die erste Kategorie speichert Informationen in XML in einer relationalen Datenbank. Die zweite Kategorie speichert die Informationen in ein XML basiertes Dateisystem ab.

XML hat den Vorteil, dass diese Art der Datenstrukturierung für Anwendungen und Schnittstellen sehr verbreitet ist. Hingegen ist die mangelnde Performance bei einer großen Datenbasis der Nachteil.

(VARELA ET AL., 2005) stellt ein Modellierungskonzept für die α|β|γ Klassifizierung auf XML Basis vor. XML wurde aufgrund des Einsatzes für Applikationen im Internet und Intranet sowie für die Web Service Technologie ausgewählt. Der Vorteil der Verbreitung von XML wird hier genutzt, um eine flexible Kommunikation zwischen unterschiedlichen Scheduling Applikationen zu erreichen. Als Datenbasis werden ebenfalls XML Dokumente verwendet. Die Kommunikation erfolgt via XML-RPC Protokoll.

Fazit

In diesem Abschnitt wurden Darstellungsmodelle auf relationaler, objektorientierter und XML Basis vorgestellt. Aktuelle Datenbanksysteme unterstützen alle drei Modelle.

XML wird überwiegend als Datenaustauschformat verwendet. Die Informationen werden in einer relationalen Datenbank gespeichert. Für reine XML Anwendungen ist eine XML Datenbank eine interessante Alternative.

Das relationale Datenmodell ist das meist verbreitetste Datenbanksystem (LEAVITT, 2000). RDBMS besitzen einfache Datentypen und ermöglichen eine einfachere Optimierung der Abfragen für ein effizienteres Verhalten des Systems. Diese Einfachheit reduziert die Vielseitigkeit gegenüber einem objekt-relationalen Datenbanksystem. Die objektorientierte Erweiterung in einem RDBMS ermöglicht die Vorteile von beiden Systemen RDBMS und OODBMS. Aus der RDBMS Welt wird die standardisierte Abfragesprache SQL in erweiterter Form verwendet. Das OODBMS stellt dem ORDBMS die objektorientierte Funktionalität zur Verfügung. Wo hingegen das OODBMS eine nahtlose Integration einer objektorientierten Programmiersprache ermöglicht und dadurch Performance Vorteile besitzt.

Zusammenfassung

Es ist eine Herausforderung ein geeignetes Planungssystem in einem bestehenden Informationssystem zu integrieren. In (PINEDO, 2005) werden unterschiedliche Typen von Modulen aufgelistet:

i. Datenbanken als Wissensdatenbanken und für die Speicherung der Objekte
ii. Verfahren für die Erstellung von Plänen und
iii. das graphische User Interface (GUI)

In diesem Artikel wurden drei wichtige Punkte für die Erstellung eines Planungssystems aufgezeigt. Zum einen muss eine Analyse bzw. Klassifizierung der Planungsdömäne durchgeführt werden, um alle Parameter zu erfassen. Hierbei wurde unterschieden, ob es sich um ein Problem aus dem Dienstleistungsbereich oder aus dem Bereich des produzierenden Gewerbes handelt. Bereits diese Abgrenzung kann Einfluß auf das später einzusetzende Lösungsverfahren haben, da unterschiedliche Ziele verfolgt werden.

Der zweite Teil befasste sich mit der semantischen Abbildung. Kann ein Problem rein mathematisch beschrieben werden oder wird ein komplexeres Darstellungsmodell benötigt? Neben dem mathematischen Modell mit der Tripel Notation $\alpha|\beta|\gamma$ aus dem OR Bereich wurde das Objekt Modell und das Sprachen Modell vorgestellt. Für die Beschreibung des Objekt Modells wurde das 7-Tupel vorgestellt und die objekt-orientierte Methode aus GIPO. Die Sprachen PDDL und GOLOG stellten die dritte Möglichkeit, ein Problem in ein Modell abzubilden, vor.

Der letzte Teil gab einen Einblick in die Speicherung von Modellen in ein Datenbanksystem. Neben der relationalen Datenbank wurde auf die objektorientierte und objekt-relationale Datenbank eingegangen. Durch den vermehrten Einsatz des Internets für ein globales Planungssystem wurde ebenfalls die XML Datenbank in der Betrachtung mit aufgenommen.

Literaturverzeichnis

ANHØJ, J.: Generic Design of Web-Based Clinical Databases. In: *Journal of Medical Internet Research* vol. 5 (2003), Nr. 4. – PMID: 14713655 PMCID: 1550574

BRUCKER, P.: *Scheduling Algorithms*. 5. ed : Springer, Berlin, 2007 – ISBN 354069515X

CATTELL, R. G. ; BARRY, D. K. ; BERLER, M. ; EASTMAN, J. ; JORDAN, D. ; RUSSELL, C. ; SCHADOW, O. ; STANIENDA, T. ; VELEZ, F.: *The Object Data Standard: ODMG 3.0*. 1. ed : Morgan Kaufmann, 2000 – ISBN 1558606475

DOU, D.: The Formal Syntax and Semantics of Web-PDDL

EDELKAMP, S. ; HOFFMANN, J.: PDDL2. 2: The language for the classical part of the 4th international planning competition. In: *4th International Planning Competition (IPC'04), at ICAPS'04* (2004)

EYERICH, P. ; NEBEL, B. ; LAKEMEYER, G. ; CLASSEN, J.: GOLOG and PDDL: What is the Relative Expressiveness?, ACM, Proceedings of the 2006 international symposium on Practical cognitive agents and robots, 2006, p 104

FIKES, R. ; NILSSON, N.: STRIPS: A new approach to the application of theorem proving to problem solving. In: *Artificial intelligence* vol. 2 (1971), Nr. 3/4, pp 189-208

FOX, M. ; LONG, D.: PDDL2. 1: An extension to PDDL for expressing temporal planning domains. In: *Journal of Artificial Intelligence Research* vol. 20 (2003), Nr. 2003, pp 61-124

GHALLAB, M. ; HOWE, A. ; KNOBLOCK, C. ; MCDERMOTT, D. ; RAM, A. ; VELOSO, M. ; WELD, D. ; WILKINS, D.: PDDL: The Planning Domain Definition Language, Version 1.2. In: *Yale Center for Computational Vision and Control, Tech Report CVC TR98003/DCS TR1165* (1998)

GHALLAB, M. ; NAU, D. ; TRAVERSO, P.: *Automated Planning: theory and practice* : Morgan Kaufmann Publishers, 2004 – ISBN 1-55860-856-7

GRAHAM, R. ; LAWLER, E. ; LENSTRA, J. ; KAN, A.: Optimization and approximation in deterministic sequencing and scheduling: A survey. In: *Annals of Discrete Mathematics* vol. 5 (1979), Nr. 2, pp 287-326

GRECHENIG, T. ; BERNHART, M. ; BREITENEDER, R. ; KAPPEL, K.: *Softwaretechnik: Mit Fallbeispielen aus realen Entwicklungsprojekten*. 1. ed : Pearson Studium, 2009 – ISBN 3868940073

GREEN, C.: Application of theorem proving to problem solving (1969)

HINDRIKS, K. ; DE BOER, F. ; VAN DER HOEK, W. ; MEYER, J.: Agent programming with declarative goals. In: *Intelligent Agents VII Agent Theories Architectures and Languages* vol. 1986 (2001), pp 228-243

HINDRIKS, K. ; VAN DER HOEK, W. ; VAN RIEMSDIJK, M.: Agent programming with temporally extended goals, International Foundation for Autonomous Agents and Multiagent Systems, Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, 2009, pp 137-144

HINDRIKS, K. ; ROBERTI, T.: GOAL as a Planning Formalism, Springer, Multiagent System Technologies: 7th German Conference, MATES 2009 Hamburg, Germany, September 9-11, 2009 Proceedings, 2006, p 29

LEAVITT, N.: Whatever happened to object-oriented databases? In: *IEEE Computer* vol. 33 (2000), Nr. 8, pp 16-19

LEPENDU, P. ; DOU, D. ; ARIOLA, Z. ; WILSON, C.: Ontology-based Relational Databases (2007)

LEVESQUE, H. ; REITER, R. ; LESPERANCE, Y. ; LIN, F. ; SCHERL, R.: GOLOG: A logic programming language for dynamic domains. In: *The Journal of Logic Programming* vol. 31 (1997), Nr. 1-3, pp 59-83

PINEDO, M. L.: *Planning and Scheduling in Manufacturing and Services* : Springer, Berlin, 2005 – ISBN 0387221980

REITER, R.: Sequential, temporal GOLOG, Citeseer, PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING-INTERNATIONAL CONFERENCE-, 1998, pp 547-556

RÖGER, G. ; HELMERT, M. ; NEBEL, B.: On the Relative Expressiveness of ADL and Golog: The Last Piece in the Puzzle. In: *Proc. of KR-08* (2008), pp 544-550

RUSSELL, S. ; NORVIG, P.: *Künstliche Intelligenz: Ein moderner Ansatz*. 2. ed : Pearson Studium, 2004 – ISBN 3827370892

SAUER, J.: *Wissensbasiertes Lösen von Ablaufplanungsproblemen durch explizite Heuristiken*. vol. 37. St. Augustin, Germany : Infix Verlag, 1993 – ISBN 3-929037-37-8

SAUER, J.: *Intelligente Ablaufplanung in lokalen und verteilten Anwendungsszenarien*. Wiesbaden : Teubner, 2004 – ISBN 978-3-519-00473-8

SIMPSON, R.: Gipo graphical interface for planning with objects, Citeseer, Proceedings of the International Conference for Knowledge Engineering in Planning and Scheduling, 2005

SIMPSON, R.: Structural Domain Definition using GIPO IV, 2007

SIMPSON, R. ; MCCLUSKEY, T. ; LONG, D. ; FOX, M.: Generic types as design patterns for planning domain specification. In: *Knowledge Engineering Tools and Techniques for AI Planning: AIPS* vol. 2

VARELA, L. ; APARÍCIO, J. ; SILVA, C.: Production Scheduling Concepts Modeling through XML, XATA 2005, 2005