Seminar: Kognitive Modellierung animierter Agenten
Bernhard Jung, Stefan Kopp, Nadine Leßmann

BDI-Architektur (Beliefs – Desires – Intentions)

Frank Schönmann*

SS 2003

*fschoenm@techfak.uni-bielefeld.de

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung/Motivation

Ziel der Entwicklung kognitiver Modelle für Agenten ist es, im jeweiligen Agenten deliberatives Verhalten zu implementieren. Dies steht im Gegensatz zu reaktivem Verhalten bei einfacheren Systemen.

Reaktives Verhalten erfordert keine komplexen Planungen: hierbei werden normalerweise Sensoreneingaben direkt miteinander verknüpft und führen zu einer unmittelbaren Reaktion des Agenten auf seine Umwelt. Beispielsweise verhält sich ein mobiler Roboter reaktiv, der in einer Blockwelt gegen ein Hindernis fährt und als Folge davon zurücksetzt und seinen Kurs leicht korrigiert.

Deliberatives Verhalten dagegen ist nötig, wenn man einem Agenten ein oder mehrere Ziele vorgibt, die er erreichen soll. Da die als gewünschte Weltzustände angegebenen Ziele nicht direkt in Relation zu Aktionen des Agenten stehen, muss dieser selbst durch Planen und Schlussfolgern in einer modellierten Welt geeignete Aktionen finden, um seinem Ziel näherzukommen. Diese Vorgänge nennt man auch *practical reasoning*; es handelt sich dabei um den Prozess der Entscheidungsfindung, der sich aus zwei Teilen zusammensetzt:

Die *deliberation* versucht zu ermitteln, **welche** Ziele der Agent erreichen soll. Dazu gehört neben mindestens einem Hauptziel auch das Aufwerfen und die Auswahl neuer Teilziele, die während der Bearbeitung anfallen.

Das *means-end reasoning* dagegen beschäftigt sich mit der Frage, **wie** die aufgeworfenen Ziele erreicht werden sollen. Ganz allgemein bezeichnet *means-end reasoning* die Auswahl von Aktionen, die den Abstand zwischen dem Ist-Zustand A und dem Soll-Zustand B verkleinern.

BDI ist ein Modell, das sich mit diesem Prozess des *practical reasoning* beschäftigt: es gibt Antworten auf die obigen Fragen nach der Auswahl von (Teil-)Zielen und deren Lösung.

1.1 Entwicklung von BDI

Der Ursprung der BDI-Architektur liegt im *Rational Agency Project* am Stanford Research Institute in den 1980'er Jahren. BDI basiert auf der Arbeit von MICHAEL E. BRATMAN, Professor in Stanford für Philosophie, der sich damit beschäftigt hat, wozu Menschen Intentionen nutzen und sich Ziele vorgeben – nämlich zur Entscheidungsfindung, also dem *practical reasoning* (siehe [GPP+99]).

Zur Übertragung der Ergebnisse auf deliberative Agenten wurden diese mit *mental attitudes* ausgestattet: Weltwissen (*beliefs*), Wünsche bzw. Ziele (*desires*) und aktuellen Absichten (*intentions*). Die folgenden Abschnitte betrachten diese Aspekte genauer.

Zuerst implementiert wurde das BDI-Modell von BRATMAN selbst in IRMA, der *intelligent resource-bounded machine architecture*. Dieser Name spiegelt die Auffassung wider, dass ein rationaler Agent das *practical reasoning* mit beschränkten Ressourcen durchführen muss – das ist auch ein Grund für die für BDI typische Konzentration auf ein aktuelles Ziel. Eine weitere frühe Implementation ist PRS, das *procedural reasoning system* von GEORGEFF und LANSKY. Näher erläutert werden diese beiden Implementationen des BDI-Konzepts in [HS96].

2 Aufbau eines BDI-Agenten

Abbildung 1 zeigt die grundlegende Struktur eines BDI-Agenten. Unterschieden wird dabei zwischen dem Input, also der Wahrnehmung durch Sensoren wie Kameras und Mikrofone, aber möglicherweise auch Nachrichten von anderen Agenten, die in den Entscheidungsprozess einfließen können. Beim Output handelt es sich um Aktionen, die unmittelbar die Umwelt beeinflussen, z. B. die direkte Ansteuerung von Motoren oder Mitteilungen an andere Agenten.

Im Folgenden werden sowohl die zugrundeliegenden Datenstrukturen, nämlich *beliefs*, *desires*, *intentions* und Plan-Bibliothek, als auch die Interpreter-Funktion, die aus mehreren Teilfunktionen zusammengesetzt ist und auf den zur Verfügung stehenden Daten arbeitet, genauer betrachtet.

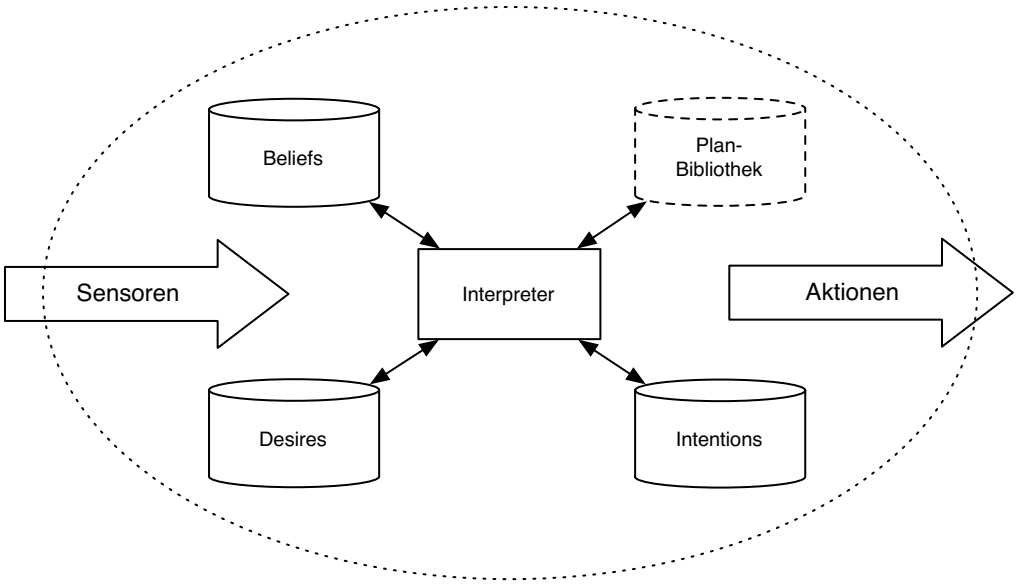

Abb. 1: Grundlegender Aufbau eines BDI-Agenten (nach [HS96])

2.1 Weltwissen – Beliefs

Wichtig für alle Arten von KI-Systemen ist die Information über den aktuellen Zustand der Welt, in der gearbeitet und geschlussfolgert wird. Diese wird in einer Wissensdatenbank (*knowledge base*) gespeichert und enthält Fakten über die aktuelle Umgebung, den internen Zustand des Agenten, Nachrichten von anderen Agenten und Hintergrundwissen, das für Schlussfolgerungen notwendig sein kann.

Die Speicherung erfolgt in einer beliebigen Wissensrepräsentation. Verbreitet ist dabei beispielsweise die Prädikatenlogik, aber auch andere Arten zur Repräsentation von Wissen sind denkbar.

Die Wissensdatenbank wird kontinuierlich aktualisiert, beeinflusst von der Wahrnehmung des Agenten und internen Schlussfolgerungen. Dadurch ist gewährleistet, dass das *practical reasoning* ständig in einem weitestmöglich gültigen Modell der Welt stattfindet.

2.2 Ziele – Desires

In dieser Datenstruktur sind die Hauptziele des Agenten angegeben, die das weitere Verhalten grundlegend beeinflussen. Die *desires* bleiben während der Ausführung normalerweise konstant. Es wird ein Ziel aus allen verfügbaren Optionen ausgewählt, das für eine bestimmte Zeit verfolgt wird. *Desires* sind ein wichtiger Bestandteil des deliberativen Verhaltens, da der Agent ohne vorgegebene Ziele zu keiner weiteren Aktion veranlasst wird.

Darüber hinaus ermöglicht das zielorientierte Verhalten nach einem Fehlschlag einer auf das jeweils ausgewählte Ziel ausgerichteten Aktion die Wiederaufnahme des Ziels, z. B. durch einen alternativen Lösungsansatz oder einen späteren Versuch, nachdem sich die Umgebung des Agenten geändert hat. Im Gegensatz dazu werden bei taskorientierten Programmen, wie sie für konventionelle Software üblich sind, zu jedem Zeitpunkt nur Aktionen durchgeführt, die der Programmierer explizit vorgesehen hat. Dort ist keine automatische Auswahl des günstigsten Lösungsweg möglich (vgl. [GPP+99]).

Ein weiterer wichtiger Aspekt, v. a. zur Abgrenzung zu Intentionen, ist die Tatsache, dass angegebene Ziele sich widersprechen können, da der Agent jeweils nur eines zu jeder Zeit auswählt, das er bis zu einem bestimmten Punkt verfolgt.

2.3 Pläne

Um sinnvolle Aktionen in der Umgebung durchzuführen, muss der Agent wissen, welche Aktionen in welcher Situation angebracht sind und zu welchem Ergebnis sie führen. Diesem Zweck dienen Pläne in der Planbibliothek. Es handelt sich dabei um Abarbeitungsvorschriften, mit denen sich ein bestimmter Teilzustand der Welt in einen anderen überführen lässt.

Ein Plan besteht dabei laut [Hub99] aus folgenden Teilen (Unterschiede im Detail sind je nach BDI-Implementation möglich): eine Vorbedingung (*pre-condition*) gibt an, in welcher Situation der jeweilige Plan ausgeführt werden kann; die hier angegebenen Fakten müssen erfüllt sein, d. h. in der *knowledge base* vorhanden sein.

Daneben existiert eine Nachbedingung (*post-condition*), die einen erreichten Zustand nach erfolgreicher Abarbeitung des Plans darsstellt. Durch Vergleich des gewünschten Ziels mit der Nachbedingung und des aktuellen Weltzustands mit der Vorbedingung, ist es möglich, einen adäquaten Plan in der aktuellen Lage des Agenten auszuwählen.

Evtl. existiert eine weitere Bedingung, die *context condition*. Dort ist beschrieben, welche Voraussetzungen während der gesamten Ausführung eines Plans gelten müssen.

Jeder Plan besitzt darüber hinaus natürlich einen Anweisungsteil, den *body*, der aus direkt ausführbaren Aktionen und neu aufzuwerfenden Teilzielen besteht. Durch diese Teilziele ist es möglich, Pläne zu staffeln, um ein Ziel zu erreichen; nicht jeder Plan muss eine detaillierte Auflistung aller einzelnen Aktionen besitzen, um eine komplexere Aufgabe zu erledigen.

Abbildung 2 zeigt einen Beispiel-Plan in der BDI-Implementierung JAM, der einen Agenten dazu befähigt, ein Objekt auf ein anderes zu stapeln. Zu sehen sind eine explizite Nachbedingung und die Abarbeitungsvorschrift, in der implizit einige Vorbedingungen angegeben sind.

Weiterhin sieht man einen *utility*-Wert, der dazu verwendet werden kann, aus mehreren konkurrierenden Plänen zur Erreichung eines bestimmten Ziels den günstigsten auszuwählen. Der Plan ist auf beliebige Objekte ausgelegt, da er mit Variablen arbeitet; zur Anwendung in einer konkreten Situation ist daher eine Art Unifikation nötig.

```
Plan: {
NAME: "Stack blocks that are already clear"
GOAL:
    ACHIEVE ON $OBJ1 $OBJ2;
CONTEXT:
BODY:
    EXECUTE print "Making sure " $OBJ1 " is clear (ACHIEVEing it).\n";
    ACHIEVE CLEAR $OBJ1;
    EXECUTE print "Making sure " $OBJ2 " is clear (ACHIEVEing it).\n";
    ACHIEVE CLEAR $OBJ2;
    EXECUTE print "Both clear. Moving " $OBJ1 " on top of " $OBJ2 ".\n";
    PERFORM move $OBJ1 $OBJ2
UTILITY: 10;
FAILURE:
    EXECUTE print "\n\nStack blocks failed!\n\n";
}
```

Abb. 2: Beispiel-Plan (JAM) zur Stapelung von Objekten (enthalten in der JAM-Distribution unter `http://www.marcush.net/IRS/irs_downloads.html`)

2.4 Intentionen – Intentions

Hat der Agent mehrere Pläne oder (Teil-)Ziele zur Auswahl, muss er sich auf eines davon festlegen. Dies bezeichnet man als *intention*, der letzten grundlegenden Datenstruktur für BDI-Systeme. Durch eine Verpflichtung auf ein Ziel wird dieses verfolgt, bis es erfüllt oder nicht mehr sinnvoll bzw. erfüllbar ist. Diese Festlegung nennt man *commitment*.

Intentionen spielen eine wichtige Rolle im Prozess des *practical reasoning*: die Festlegung auf ein Ziel entspricht dabei der *deliberation*. Intentionen führen außerdem zu direkten Aktionen, z. B. der Ansteuerung von Motoren, oder der weiteren rekursiven Auswahl von Plänen, nämlich wenn im aktuellen Plan ein Teilziel aufgeworfen wird. Auf diese Weise findet *means-end reasoning* bei BDI-Agenten statt.

Eine Form der Implementierung kann man sich als *intention stack* vorstellen: zuoberst liegt dabei jeweils das aktuell verfolgte Ziel. Handelt es sich um eine direkt ausführbare Aktion, wird diese einer Ausführungsfunktion übergeben. Andernfalls sucht der Agent nach Plänen, um dieses Teilziel zu erreichen, und wählt den nützlichsten davon aus, der dann auf den Stack gelegt wird. Ein erreichtes Ziel wird jeweils herunter genommen, so dass danach wieder Zugriff auf übergeordnete Ziele besteht.

Einige Eigenschaften der Intentionen sind (vgl. [Wei99]):

- Beeinflussung des *means-end reasoning*: die Intentionen führen zu Aktionen, um dem ausgewählten Ziels näherzukommen
- Beschränkung der weiteren Planung: Intentionen beeinflussen die Auswahl von Teilzielen; kontraproduktive Pläne werden auf diese Weise nicht weiter betrachtet, wenn sie den momentanen Zielen zuwider laufen

- Persistenz: Intentionen bleiben bestehen, solange es sinnvoll ist; ist das Ziel erreicht oder dessen Verfolgung nicht mehr sinnvoll, wird es verworfen
- Beeinflussung der *beliefs* über die Zukunft: möglicherweise ist es hilfreich, den Agenten Schlussfolgerungen durchführen zu lassen, die auf der Annahme basieren, dass das Ziel erreicht wird

2.5 Desires vs. Intentions

Um *desires* und *intentions* voneinander abzugrenzen, unterscheidet man diese besonders in zwei Eigenschaften:

Während der BDI-Agent beliebig viele *desires* haben kann, die er nicht unbedingt erreichen muss, legt er sich nach Auswahl auf eine Intention fest (*commitment*). Anschließend wird diese bis zur Erfüllung weiter verfolgt. Wann und ob der Agent die Intention überhaupt wieder verwirft, ist implementations- und parameterabhängig.

Direkt damit hängt zusammen, dass bei den aktuell ausgewählten Intentionen keine Widersprüche auftreten dürfen, damit der Agent sich in der Erreichung seiner Ziele nicht selbst behindert. Bei den *desires* dagegen sind Inkonsistenzen möglich, da sie nicht zwingend ausgeführt werden müssen.

3 Ablauf des practical reasoning bei BDI-Agenten

Abbildung 1 zeigte den grundlegenden Aufbau eines BDI-Agenten. Der Interpreter, der auf den bisher vorgestellten Datenstrukturen arbeitet, besteht dabei allerdings aus mehreren Einzelfunktionen, die im folgenden ausführlicher beschrieben werden. In Abbildung 3 ist der grundlegende Ablauf des *practical reasoning* bei BDI-Agenten dargestellt, wie er in [Wei99] beschrieben ist.

3.1 Verwaltung der Fakten – belief revision

Diese Funktion dient dazu, das gesamte Wissen, das der Agent besitzt, auf dem aktuellen Stand zu halten. Alle Eingaben, z. B. Wahrnehmung über Sensoren oder Nachrichten von anderen Agenten, werden ausgewertet, um die Menge der *beliefs* dem geänderten Weltzustand anzupassen. Dies ist notwendig, damit der Agent zu jedem Zeitpunkt günstige Pläne entwickeln kann und keine Schlussfolgerungen durchführt, die in einer möglicherweise veränderten Situation keinen Sinn mehr machen.

3.2 Aufwerfen von möglichen Plänen – generate options

Abhängig von dem aktuellen Zustand der Umgebung werden aus der Plan-Datenbank konkrete Pläne ausgewählt, die der Agent momentan anwenden kann. Die Auswahl wird dabei von mehreren Faktoren beeinflusst: aufgrund möglicherweise für bestimmte Einträge angegebenen Vorbedingungen, die mit den *beliefs* abgeglichen werden, wählt diese Funktion nur bestimmte Pläne aus. Darüber hinaus gehen die momentan ausgewählten *intentions* in die Entscheidung mit ein; dies bewirkt vor allem die Auswahl von Teilplänen, die dazu dienen, Unterziele zu erreichen, und verhindert die Festlegung auf widersprüchliche Absichten, die sich gegenseitig behindern oder ausschließen.

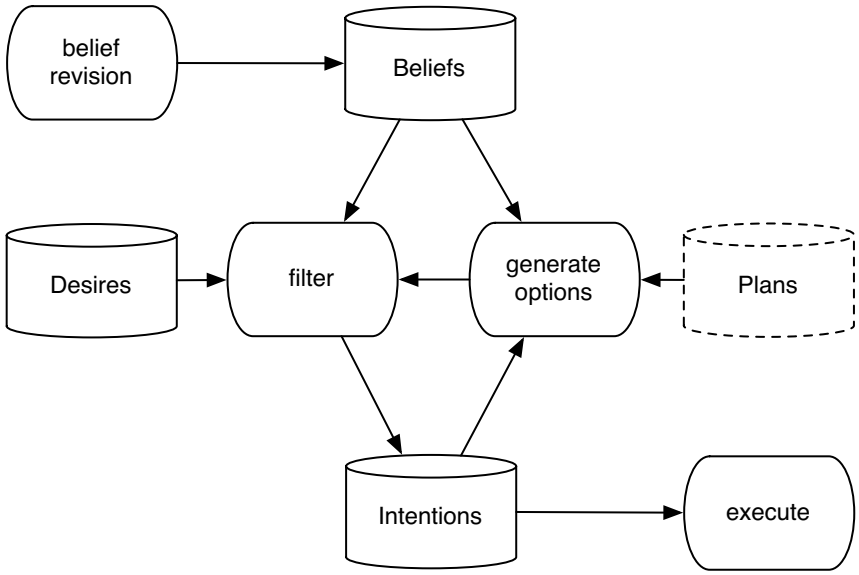

Abb. 3: Ablauf des *practical reasoning* bei BDI-Agenten unter Verwendung der Funktionen *belief revision*, *filter*, *generate options* und *execute* (nach [Wei99]). Die Pfeile stellen dar, wie sich Funktionen und Daten gegenseitig beeinflussen.

Hier findet auch eine Konkretisierung der in der Datenbank abgespeicherten, abstrakten Pläne statt. Diese können Variablen enthalten, die durch den Optionsgenerator mit konkreten Werten, d. h. mit Objekten aus der Welt des Agenten, gefüllt werden. Dazu wird eine Unifikation durchgeführt.

3.3 Auswahl eines Plans – filter

Die ermittelten Pläne werden von der *filter*-Funktion weiter verarbeitet. Aufgrund der aktuellen Situation und der Ziele des Agenten wird aus der Menge der Optionen ein bestimmter Plan ausgewählt. Zu diesem Zweck benötigt man eine Bewertungsfunktion, mit der die Nützlichkeit der zur Verfügung stehenden Pläne analysiert wird. Danach werden die Intentionen des Agenten aktualisiert.

3.4 Ausführung eines Plans – execute

Intentionen, d. h. ausgewählte Pläne, geben dem Agenten an, was zu tun ist: entweder muss erst ein Unterziel erreicht werden oder der nächste Schritt in der Ausführung des Plans ist eine Anweisung, die direkt ausgeführt werden kann. Dies könnte beispielsweise eine konkrete Bewegung eines Roboters sein, die in ein Kommando an Motoren weitergegeben wird. Diesen Zweck erfüllt die *execute*-Funktion, die eine direkt ausführbare Anweisung in eine Aktion des Agenten umsetzt.

4 Diskussion

4.1 Problem: Verwaltung der Intentionen

Ein grundlegendes Problem bei der Verwendung der BDI-Architektur ist die Verwaltung von Intentionen. Führt man sich den *intention stack* vor Augen, so ist es meistens nicht sinnvoll, stur das oberste Teilziel zu verfolgen, bis es erreicht ist. Möglicherweise hat sich zu einem Zeitpunkt in der Vergangenheit die Welt des Agenten derart verändert, dass der aktuell ausgewählte Lösungsweg nicht mehr der bestmögliche ist oder sogar überhaupt nicht mehr zu einem Ergebnis führt. In dem Fall sollte der Agent einige der Intentionen verwerfen und eine günstigere Lösung finden.

Daher ist bei der Implementation eine Abwägung nötig zwischen einer ständigen und einer seltenen Überprüfung der Intentionen. Man bezeichnet das auch als *commitment*-Strategie, da dieses Vorgehen die Festlegung auf Teilziele beeinflusst.

Überprüft der Agent ständig den gerade eingeschlagenen Lösungsweg, so gehen Zeit und möglicherweise auch andere Ressourcen verloren. Wird ein Ziel zu lange verfolgt, kann der Agent z. B. eine günstige Gelegenheit für einen schnelleren Weg zum Erfolg verpassen oder durch nicht mehr erreichbare Ziele Zeit verschwenden.

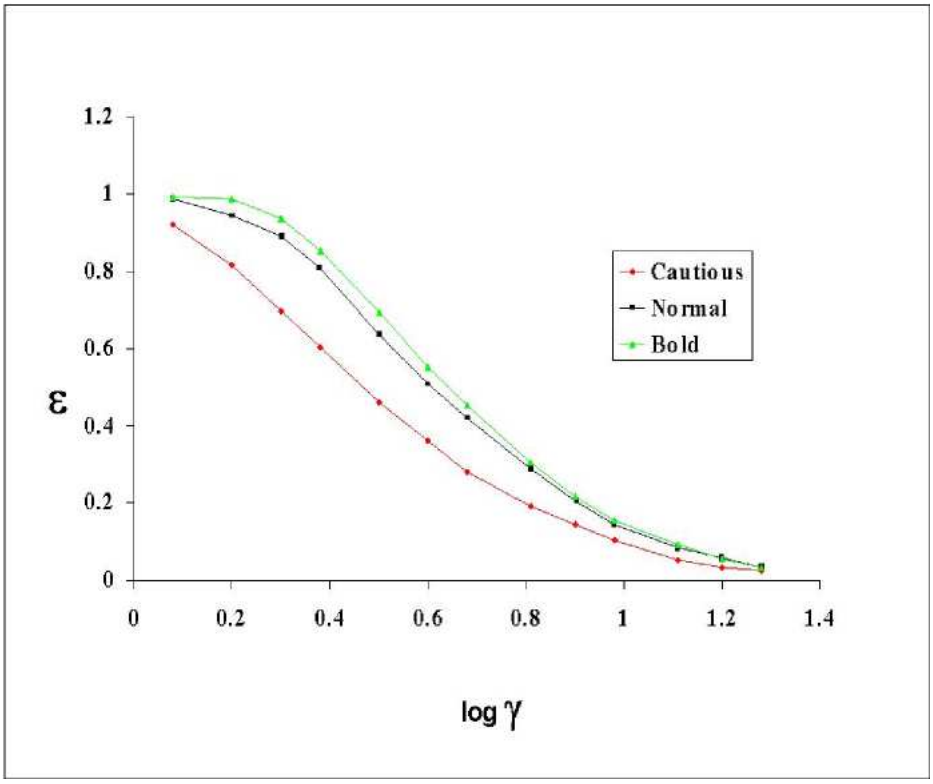

Abb. 4: Vergleich verschiedener *commitment*-Strategien. ε ist dabei die Erfolgsrate, γ die Veränderungsrate der Welt, in der sich der Agent bewegt (siehe [GPP+99]).

In [GPP+99] findet sich ein kurzer Vergleich verschiedener *commitment*-Strategien. Teile des Ergebnisses sind in Abbildung 4 dargestellt. Man sieht, dass der Erfolg stark abhängig ist von der Veränderungsrate der Welt: natürlich wird der Agent in der Durchführung seiner

geplanten Aktionen durch große Veränderungn in der Welt gestört.

Man kann ebenfalls erkennen, dass die *bold*-Strategie, bei der der Agent fast nie seine Intentionen überprüft, zwar in einer relativ gleichbleibenden Umgebung erfolgreicher ist als andere Verfahren. Dieser Vorteil nimmt aber mit steigender Veränderungsrate ab und wird später gegenüber anderen Verfahren sogar zum Nachteil.

4.2 Vergleich mit SOAR

SOAR ist ebenfalls eine Architektur, die dazu dient, deliberative Agenten zu modellieren (siehe [New90]). Dabei liegt im Gegensatz zu BDI ein großes Augenmerk auch auf einem Lernverfahren, das dem Agenten erlaubt, geschlussfolgerte Lösungswege zu speichern und später erneut anzuwenden.

Ansonsten scheinen SOAR und BDI aber kompatibel zu sein (»*the Soar model seems fully compatible with the BDI architectures*«, [GPP+99]): den *beliefs* in BDI entspricht in SOAR der aktuelle Zustand, der ebenfalls das Wissen des Agenten über seine Umwelt enthält. *Desires* oder Ziele finden sich in ähnlicher Form in beiden Architekturen. *Plans* und *intentions* entsprechen den verfügbaren bzw. ausgewählten Operatoren. Und ähnlich wie beim *commitment* gibt es in SOAR eine Strategie, die sich um die Operator-Terminierung kümmert.

Da SOAR und BDI das gleiche Problem auf unterschiedliche Weisen versuchen zu lösen, kann es in einigen Anwendungen sinnvoll sein, Teilaspekte beider Konzepte miteinander zu kombinieren, um bessere Ergebnisse zu erhalten.

4.3 Vorteile und Kritikpunkte

Der Hauptvorteil von BDI liegt klar darin, dass diese Architektur sehr intuitiv ist. Grund dafür ist, dass es auf den Prinzipien der Entscheidungsfindung beim Menschen basiert; BDI hat darin sogar ihren Ursprung. Auch das Konzept der *mental attitudes* ist leicht einsichtig, daher ist ein Verständnis der Architektur nicht weiter schwierig.

Plant man den Einsatz von BDI, ist es außerdem möglich, bereits bestehende Implementationen und Systeme zu verwenden, z. B. JAM, IRMA, PRS, dMARS und andere. Einige davon wurden bereits erfolgreich in größeren Projekten verwendet oder getestet, beispielsweise bei der Fehlerdiagnose für das Space Shuttle (vgl. [HS96]) oder zur Anflugkontrolle eines Flughafens (vgl. [RG95]).

Als Kritikpunkte lassen sich anführen, dass BDI-Agenten nicht explizit darauf ausgelegt sind, zu lernen und danach ihr Verhalten anzupassen. Auch ist der grundlegende Ansatz nicht explizit auf das Zusammenspiel mehrerer Agenten ausgelegt (vgl. [GPP+99]). Diese Nachteile sind aber durch Erweiterungen am BDI-Konzept zu beheben: beispielsweise könnte ein Agent Pläne, die zusammen einen Lösungsweg für ein größeres Problem bilden, zwischenspeichern, um ein SOAR-ähnlichen Lernverfahren zu realisieren. Soziales Verhalten ließe sich durch Hinzunahme von Sensoren und Aktoren für Kommunikation nachbilden.

5 Zusammenfassung

Die BDI-Architektur erreicht ihr Ziel, Agenten mit deliberativem Verhalten zu modellieren, wie man auch an den vielen verschiedenen Implementationen und den erfolgreichen Einsätzen unter realen Bedingungen sehen kann.

Dem Verständnis dieses Konzepts kommt besonders zugute, dass der philosophische Hintergrund der *mental attitudes* sehr intuitiv ist, denn diese basieren gerade darauf, wie Schlussfolgerungen und Entscheidungsfindungen beim Menschen ablaufen.

Auf der technischen Seite wird dies realisiert durch die verschiedenen Daten, auf denen der Agent arbeitet: das Zusammenspiel zwischen Zielen und Absichten bringt dabei Ordnung in eine möglicherweise große Zahl von verfügbaren Plänen. Die *deliberation* wird dabei durch Filterung der Optionen erreicht, das *means-end reasoning* durch die hierarchische Generierung von Intentionen.

Den Kritikpunkten der mangelnden Lernfähigkeit und des fehlenden Zusammenspiels mit anderen Agenten lässt sich begegnen mit Erweiterungen des Grundkonzeptes von BDI, z. B. durch SOAR-ähnliche Anpassungen.

Literatur

[BDKTV97] Frances M. T. Brazier, Barbara Dunin-Keplicz, Jan Treur, and Rineke Verbrugge. Modelling Internal Dynamic Behaviour of BDI Agents. In *ModelAge Workshop*, pages 36–56, 1997.

[Eli02] Renée Elio. Belief-Desire-Intention Agency in a General Cognitive Architecture. *Cognitive Science Quarterly*, 2(3-4/2002):320–339, 2002.

[GPP+99] Mike Georgeff, Barney Pell, Martha Pollack, Milind Tambe, and Mike Wooldridge. The Belief-Desire-Intention Model of Agency. In Jörg Müller, Munindar P. Singh, and Anand S. Rao, editors, *Proceedings of the 5th International Workshop on Intelligent Agents V: Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555, pages 1–10. Springer-Verlag: Heidelberg, Germany, 1999.

[HS96] Afsaneh Haddadi and Kurt Sundermeyer. Belief-Desire-Intention Agent Architectures. In G. M. P. O'Hare and N. R. Jennings, editors, *Foundations of Distributed Artificial Intelligence*, chapter 5, pages 169–185. John Wiley & Sons Inc., New York, 1996.

[Hub99] Marcus J. Huber. JAM: A BDI-Theoretic Mobile Agent Architecture. In *Proceedings of the Third International Conference on Autonomous Agents*, Seattle, 1999.

[New90] Allen Newell. *Unified Theory of Cognition*. Harvard University Press, 1990.

[RG95] Anand S. Rao and Michael P. Georgeff. BDI Agents: From Theory to Practice. In *Proceedings of the First Intl. Conference on Multiagent Systems*, San Francisco, 1995.

[TPH02] John Thangarajah, Lin Padgham, and James Harland. Representation and Reasoning for Goals in BDI Agents. In Michael J. Oudshoorn, editor, *Twenty-Fifth Australasian Computer Science Conference (ACSC2002)*, Melbourne, Australia, 2002. ACS.

[Wei99] Gerhard Weiss, editor. *Multiagent Systems: A modern Approach to Distributed Artificial Intelligence*, chapter Belief-Desire-Intention Architectures, pages 54–61. MIT Press, 1999.