Technologie-Zentrum Informatik

Technical Report 38

Ontologie-basiertes Agentenmatching

- Diplomarbeit -

Sven Werner

TZI-Bericht Nr. 38
2006

Universität Bremen

TZI-Berichte

Herausgeber:
Technologie-Zentrum Informatik
Universität Bremen
Am Fallturm 1
28359 Bremen
Telefon: +49-421-218-7272
Fax: +49-421-218-7820
E-Mail: info@tzi.de
http://www.tzi.de

ISSN 1613-3773

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1.

Einführung

1.1. Einleitung und Motivation

Ein Ansatz der zunehmenden Komplexität von Informationssystemen entgegenzuwirken, ist das Forschungsfeld der *Multi-Agenten-Systeme* (MAS) aus dem Bereich der *Verteilten Künstlichen Intelligenz* (VKI). Sie bildet einen Ansatz die komplexeren Architekturen aktueller Softwaresysteme, in denen Einzelkomponenten dynamisch zusammenwirken und parallele Verarbeitungen durchführen, zu modellieren. Nach [Wooldridge 2002, S. 7] lässt sich eine zunehmende Verbreitung dieser komplexen und vernetzen Systemarchitekturen feststellen, die zu einem erhöhten Interesse am Begriff des Agenten führt. Ein weiteres Einsatzgebiet, in dem Ansätze aus der VKI vielversprechend erscheinen, ist nach [Wooldridge 2002, S. 7] die Modellierung und Simulation komplexer Gesellschaftsstrukturen.

Der Begriff *Agent* ist hierbei als Softwareeinheit zu verstehen, die mit ihrer Umgebung interagiert und eine gewisse Autonomie in der Wahl ihrer Entscheidungen besitzt. Der Begriff Agent kann zusätzlich um das Attribut *intelligent* erweitert werden, das einem Agent zusätzlich die Fähigkeit zuschreibt, bestimmte Ziele zu verfolgen und sein Handeln anhand bestimmter Kriterien zu optimieren [Weiß 1999, S. 1 f]. Der Ansatz ist dadurch gekennzeichnet, dass eine Aufteilung eines zu lösenden Gesamtproblems in Teilprobleme erfolgt, die mit Hilfe eines Systems von Agenten gelöst werden. Dies führt schließlich zur Lösung des Gesamtproblems. Innerhalb des MAS ist zur Durchführung dieser Aufgabe *Interaktion* zwischen den beteiligten Agenten nötig. Innerhalb der VKI wird dieser Bereich als *Koordination* bezeichnet. Ziel der Koordination zwischen den beteiligten Agenten muss es sein, die zu lösenden Aufgaben durch das MAS erfolgreich abzuarbeiten und damit eine Lösung des Gesamtproblems zu erreichen. Zu unterscheiden sind grundsätzlich zwei Arten der Koordination: Einerseits ist dies die *Kooperation* von Agenten, die eine Zusammenarbeit der beteiligten Agenten realisiert und andererseits der Wettbewerb oder *Competition* der Agenten, in dem jeder einzelne Agent in Konkurenz zu den übrigen Agenten handelt. In dieser Arbeit liegt der Fokus auf der Kooperation als eine Strategie zur Lösung eines Gesamtproblems durch Verteilung auf

Intelligente Agenten innerhalb eines MAS. Dieser Ansatz wird auch als *Cooperative Distributed Problem Solving* (CDPS) bezeichnet. Eine Form der Realisierung des CDPS ist der so genannte elektronische Marktplatz. Dieser virtuelle Marktplatz bietet die Möglichkeit Agenten, die die Lösung für bestimmte Arten von Problemen zur Verfügung stellen können, mit anderen Agenten, die nach der Lösung für bestimmte Probleme suchen, zusammenzubringen. Die Interaktion erfolgt hier in der Art, dass einzelne Agenten bestimmte Probleme zur Disposition stellen und dann in Form von Verhandlungen mit anderen Agenten eine Lösung des Problems erarbeiten. Eine der Kernherausforderungen bei diesen Agentenverhandlungen auf elektronischen Marktplätzen und der VKI allgemein besteht in dem Bedarf eine Zuordnung zwischen angefragten Problemen und deren Eigenschaften[1] sowie Problemlösungsfähigkeiten – den so genannten *Capabilities* – anderer Agenten durchzuführen. Dieser Prozess wird als *Matchmaking* bezeichnet. Aktuell besteht das Problem, dass dieser Matchmaking-Prozess in Agentenverhandlungen häufig fehlschlägt, da meist wenig Flexibilität im Entscheidungsverfahren vorhanden ist, aber die Zusammensetzung des Agentenumfelds und die angefragten Dienstleistungen bzw. Probleme und deren Eigenschaften sehr dynamisch sind. Die fehlende Flexibilität begründet sich durch größtenteils implizite Repräsentationen der Agenten-Capabilities, woraus sich eine statische Zuordnung zwischen Problemen und Problemlösungen ergibt. Es bestehen hierdurch wenig Möglichkeiten zur Reaktion auf Störungen, die z. B. bei Ausfall eines Agenten entstehen. Durch statisch Zuordnung mit Hilfe eines zentralen Yellow-Page-Service, in dem Agenten ihre Capabilities registrieren, wird an dieser Stelle eine Kompensation des Ausfalls durch andere Agenten verhindert. Die Capabilities sind nicht in den Entscheidungs- und Planungsprozess der Agenten integriert, die demzufolge nicht die Möglichkeit haben festzustellen, ob sie evtl. zur Problemlösung beitragen können. Diese Arbeit wird sich mit den geschilderten Problemen befassen und Lösungsmöglichkeiten zur Beseitigung bzw. Abschwächung dieser Probleme vorschlagen. Im Rahmen der Semantic Web Services [Berners-Lee u. a. 2001] gibt es eine ähnliche Problemlage, da hier ebenfalls die Eigenschaften von Web Services beschrieben werden müssen, um anderen Web Services das Auffinden von bestimmten benötigten Diensten zu ermöglichen. Hier gibt es eine Reihe von Ansätzen, die auf eine Ontologie-basierte Lösung des Problems setzen. Diese bieten die Möglichkeit zur Modellierung einer bestimmten Problemdomäne und verleihen der so erstellten Beschreibung semantische Aussagekraft. Ontologien werden deswegen im Rahmen der Arbeit näher vorgestellt und untersucht.

[1] Eine Eigenschaft wird auch als *Feature* bezeichnet.

1.2. Zielsetzung

Zur Lösung der im vorherigen Abschnitt erwähnten Probleme lassen sich folgende Ziele für die durchzuführende Arbeit ableiten:

- Es soll eine *explizite Modellierung* der Agenten-Capabilities ermöglicht werden. Hierdurch soll eine Erhöhung der Flexibilität im Matchmaking-Prozess erreicht werden.
- Weiterhin wird eine bessere Ausnutzung von vorhandenen Potenzialen durch Nutzung von *Inferenz über die Agenten-Capabilities* und eine *Verbindung der Capabilities mit der Agentenplanung* angestrebt.
- Hieraus soll sich schließlich die Spezifikation eines *Capability-Matchings* ergeben. Außerdem soll der Ansatz in die vorhandene Architektur des *Diskursagenten* integriert werden.
- Die Evaluation der Arbeit soll durch eine *Implementierung im Rahmen des IntaPS-Prototypen* und einem *Vergleich mit statischen Ansätzen* innerhalb der Domäne der Produktionslogistik erfolgen.

1.3. Aufbau der Arbeit

Der Aufbau der Arbeit gliedert sich in 6 Kapitel. Nachdem in Kapitel 1 die Motivation und Zielsetzung der Arbeit erläutert wurde, erfolgt im zweiten Kapitel eine Darstellung der Grundlagen der Agententechnologie sowie die Vorstellung eines Anwendungsszenarios, auf das im weiteren Verlauf der Arbeit an einer Reihe von Stellen zurückgegriffen werden wird. Außerdem wird der Bereich der Ontologien als ein Bereich der Wissensrepräsentation vorgestellt. Das Kapitel 2 schließt mit einer Definition der Anforderungen an einen eigenen Ansatz zur Definition eines flexiblen Matchmakings für Agentensysteme. Im dritten Kapitel erfolgt die Aufarbeitung des aktuellen und relevanten Stands der Forschung. Der Fokus liegt hier auf Arbeiten und Ansätzen, die zur Erreichung der definierten Zielstellung geeignet erscheinen.

Im vierten Kapitel erfolgt dann die Ausarbeitung eines eigenen Ansatzes. Dieser wird zunächst grundlegend erläutert und anschließend formalisiert. Im Rahmen des Kapitels 5 wird dann dieser Ansatz in ein bestehendes Agentensystem integriert und die Implementierung erläutert. Anschließend erfolgt die Formulierung eines konkreten Szenarios, das der Evaluation des Ansatzes dienen soll und abschließend die Darstellung der erzielten Ergebnisse und deren Interpretation. Die Arbeit schließt mit einer Zusammenfassung und einem Ausblick auf weitere Arbeiten.

Kapitel 2.

Grundlagen und Anforderungsdefinition

In diesem Kapitel werden zunächst die Grundlagen der Agententechnologie und weiterer Aspekte in einem Überblick dargestellt, um ein weiteres Verständnis der Problemstellung zu ermöglichen. Die Darstellung beinhaltet zum Einen eine Erläuterung einer Reihe von Agentenarchitekturen sowie andererseits einen Überblick über den Aufbau von Agentensystemen. Da der Bereich der Wissensrepräsentation ein wichtiger Teil der Arbeit ist, erfolgt außerdem eine Einführung in die Wissensrepräsentation mit dem Schwerpunkt auf der Vorstellung des Ontologie-Ansatzes. Daran schließt sich eine Beschreibung des Szenarios an, auf das im Weiteren an verschiedenen Stellen zurückgegriffen werden wird. Das Kapitel schließt mit der Definition der Anforderungen an den zu entwickelnden Ansatz.

2.1. Agententechnologie

Für die Definition des Begriffs *Agent* gibt es grundsätzlich zwei verschiedene Ansätze: Es wird zwischen der „schwachen“ und „starken“ Charakterisierung unterschieden.

Die schwache Charakterisierung geht nach [Wooldridge und Jennings 1995] von drei wesentlichen Merkmalen zur Definition eines Agenten aus: *Flexibilität*, *Interaktivität* und *Autonomie*. Die Eigenschaft Flexibilität verlangt zum Einen vom Agenten für ihn unbekannte Situationen zu erkennen und darauf angemessen zu reagieren. Andererseits soll der Agent auch die Möglichkeit besitzen selbst eine Handlung anzustoßen, um beispielsweise ein bestimmtes Ziel zu erreichen. Die erste Verhaltensweise bezeichnet man als *Reaktivität*. Die zweite Möglichkeit des Agierens wird *Proaktivität* genannt. Interaktivität bezeichnet die Möglichkeit eines Agenten mit seiner Umgebung in Kontakt zu treten und zu interagieren. Die Interaktion erfolgt dabei durch Wettbewerb mit anderen Agenten entweder mit dem Ziel in *Kooperation* gemeinsame Problemlösungen zu erreichen oder in *Kompetition* zur Erfüllung konkurrierender Ziele.

Die Eigenschaft der Autonomie verlangt vom Agenten selbstständiges Handeln ohne äußere Kontrolle. Das Verarbeiten der hierzu nötigen Informationen und das Treffen der nötigen Entscheidungen soll vom Agenten selbst vorgenommen werden.

Die starke Charakterisierung definiert hingegen den Agentenbegriff nicht über bestimmte Eigenschaften sondern als Einheit, die bestimmte Zustände besitzt. Dies sind nach [Mamdani 1998] einerseits *informationsbezogene Zustände*, wie beispielsweise Umgebungswissen, andererseits *konative Zustände*, die Absichten oder Pläne repräsentieren und schließlich *affektive Zustände*, die z. B. Ziele des Agenten umfassen.

Agenten, die die oben erwähnten Eigenschaften aufweisen, werden im Bereich der Künstlichen Intelligenz als *Intelligente Agenten* bezeichnet. Eine weitere Definition des Begriffs liefern [Russel und Norvig 2003, S. 4], nach denen sich ein Intelligenter Agent dadurch auszeichnet, dass er Autonomie besitzt, seine Umgebung wahrnimmt, persistent über eine anhaltende Zeitdauer ist, sein Verhalten anpasst und bestimmte Ziele verfolgt. Dies unterscheide ihn außerdem von gewöhnlichen Computerprogrammen. Ein wichtiger Aspekt bei der Definition eines Agenten ist außerdem die Wahrnehmung der Agentenumgebung durch *Sensoren* und die Veränderung seiner Umwelt durch *Aktoren*. Weiterhin wird bei [Russel und Norvig 2003, S. 4] noch das Konzept des *Rationalen Agenten* eingeführt, der versucht den erwarteten Erfolg seines Handels zu maximieren. Dies versucht er durch Anpassung und Optimierung seines Handelns. Eine weitere Einführung findet sich außerdem in [Burkhard 2003].

In dieser Arbeit wird im folgenden der Begriff „Agent“ in Anlehnung an [Weiß und Jakob 2005, S. 4–5] als Softwareeinheit verstanden, die die erwähnten Anforderungen an Flexibilität, Interaktivität und Autonomie erfüllt. Eine Verfeinerung dieser Definition wird in den folgenden Abschnitten vorgenommen.

2.1.1. Ansätze zur Agentenarchitektur

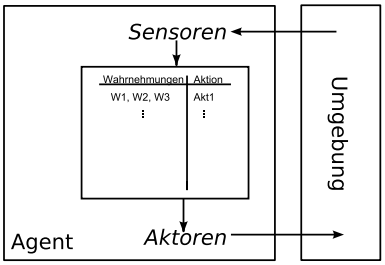


Abbildung 2.1.: Tabellen-basierender Agent

Zur Architektur von Intelligenten Agenten sind verschiedene Modelle entwickelt worden. Grundsätzlich wird zwischen simplen, reaktiven Agenten und deliberativen Agenten unterschieden. Simple, reaktive Agenten reagieren nur durch von vornherein festgelegte Verhaltensmuster auf äußere Wahrnehmungen. *Deliberative* und Ziel-basierte Agenten besitzen

dagegen Möglichkeiten zur Bewertung der aktuellen Situation und darauf aufbauenden Verhaltensanpassungen aufgrund früherer Erfahrungen.

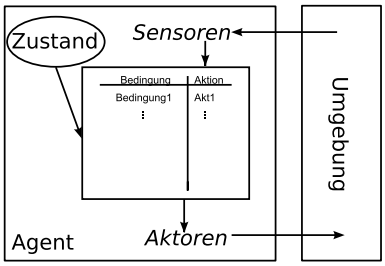


Abbildung 2.2.: Modell-basierender Agent

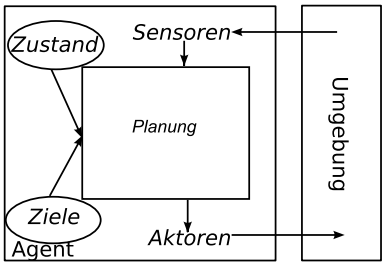


Abbildung 2.3.: Ziel-basierender Agent

Der einfachste Ansatz zur Architektur eines simplen und reaktiven Agenten ist der tabellenbasierende Agent (Abbildung 2.1). Dieser führt über eine Folge von Wahrnehmungen Abfragen in einer zur Designzeit des Agenten festgelegten Tabelle durch, die für eine bestimmte Folge von Wahrnehmungen eine durchzuführende Aktion definiert. Das Problem bei diesem Ansatz besteht darin, dass für jede mögliche Wahrnehmungsfolge ein Eintrag in der Wahrnehmungstabelle vorhanden sein muss, was durch die benötigte Größe einer solchen Tabelle einen praktischen Einsatz dieses Konzepts erschwert. Eine Beschränkung der Auswahl einer Aktion auf eine Abhängigkeit von der aktuellen Wahrnehmung des Agenten führt zum Konzept des Reflex-Agenten. Dieser wählt aufgrund der aktuellen Wahrnehmung unter Zuhilfenahme eines Regelsystems eine Auswahl seiner nächsten Aktion durch. Die Regeln legen dabei jeweils für eine bestimmte Wahrnehmung eine durchzuführende Aktion fest, nachdem über die Interpretation der aktuellen Wahrnehmung die passende Regel ausgewählt wurde. Dieses sehr einfache Konzept führt aber in Umgebungen, die für den Agenten nicht vollständig wahrnehmbar sind, zu Problemen und kann beispielsweise Endlosschleifen im Agentenverhalten auslösen. Ein Weg zur Lösung dieses Problems ist die Erweiterung des Agenten um ein Modell der Welt und führt zum so genannten Modell-basierenden Agenten. In dieser Agentendefinition besitzt der Agent die Möglichkeit seine Umgebung in einer Zustandsbeschreibung laufend zu aktualisieren. Hierbei berücksichtigt er einerseits die Entwicklung der Umgebung, die unabhängig von seinem Handeln ist und andererseits die Auswirkungen seiner eigenen Aktionen auf die Umwelt. Dieser Agentenansatz ist in Abbildung 2.2 dargestellt.

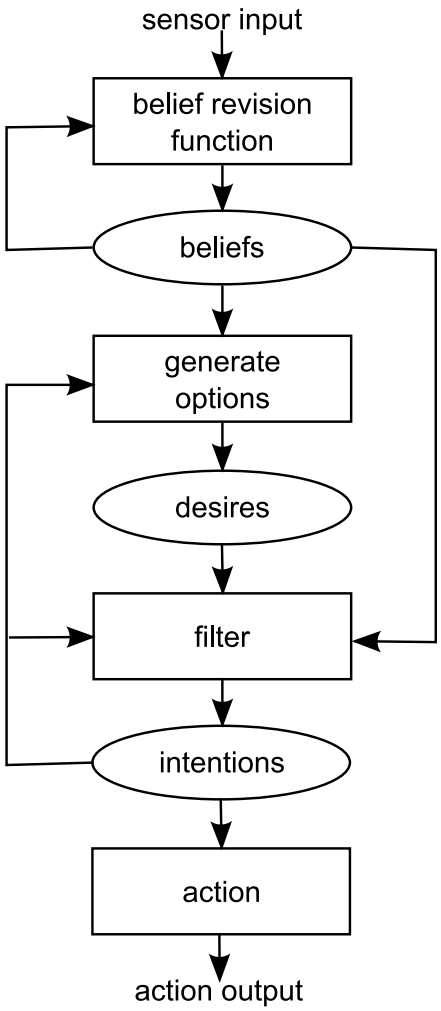


Abbildung 2.4.: Belief-Desire-Intention Deliberationszyklus nach [Bratman u. a. 1988]

Eine Erhöhung der Flexibilität des Agenten erhält man durch die Definition von Zielen, die durch das Handeln des Agenten erreicht werden sollen. Ziele können erfordern, dass der Agent mehrere Aktionen nacheinander ausführen muss, um den gewünschten Zielzustand zu erreichen. Dies erfordert die Durchführung einer Planung seiner Aktionen seitens des Agenten. Da die Zielsetzung der Arbeit eine Betrachtung der Agentenplanung einschließt ist dieses der Ansatz, der im Weiteren näher betrachtet wird und auf den aufgebaut wird.

Ein Ansatz einer Ziel-basierten Agentenarchitektur (Abbildung 2.3) ist die BDI-Architektur von [Bratman 1987]. Diese geht davon aus, dass ein Agent bestimmte Überzeugungen (*Beliefs*) seine Umgebung betreffend besitzt. Diese ergeben sich aus den Wahrnehmungen des Agenten. Zusätzlich besitzt der Agent bestimmte Ziele (*Desires*), die er erreichen möchte. Aus beiden ergeben sich im sog. *Deliberationszyklus* die Absichten (*Intentions*) des Agenten. Verdeutlicht wird dieser Deliberationszyklus in Abbildung 2.4.

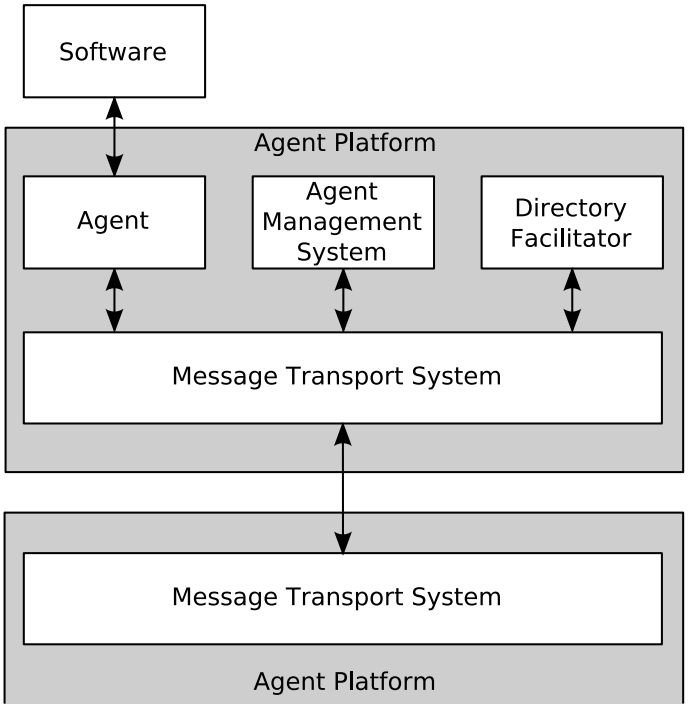


Abbildung 2.5.: FIPA Agent Management Referenz Modell

Eine formale Beschreibung dieser Architektur liefert [Wooldrigde 2000]. Hier definiert er die $\mathcal{LORA}$-Logik zur Beschreibung des internen Zustands eines BDI-Agenten.

2.1.2. Aufbau von Agentensystemen

Ein Ansatz zur Lösung komplexer Probleme, ist der Versuch durch die Verteilung der Aufgaben auf einzelne Problemlösungseinheiten der Komplexität der Probleme zu begegnen. Vielversprechend ist in dieser Hinsicht der Einsatz der in den vorherigen Abschnitten vorgestellten Agenten zur gemeinsamen Problemlösung. Diese werden hierzu zu einem so genannten Multiagentensystem (MAS) zusammengefasst, in dem jeder einzelne Agent versucht einen Teil des Problems zu bewältigen. D. h. es wird versucht, durch Interaktion verschiedener Agenten, die bestimmte Rollen repräsentieren, ein bestimmtes Problem abzubilden. Zur Koordination dieser Interaktionen gibt es verschiedene Ansätze. Einer dieser Ansätze ist die *markt-basierte Koordination*. In Anlehnung an reale Marktplätze bieten auf dem virtuellen Marktplatz des Agentensystems einzelne Agenten bestimmte Problemlösungen oder Dienstleistungen an, woraufhin andere Agenten Angebote abgeben. Die Verhandlungen können nach verschiedenen Schemata durchgeführt werden. Dieses sind beispielsweise Auktionsverhandlungen über den Preis einer bestimmten Dienstleistung. Durch die gemeinsame Lösung

verschiedener Teilprobleme, die über diese Art der Interaktion zwischen den Agenten koordiniert wird, erfolgt eine verteilte Lösung des Gesamtproblems.

Ein Standard, der die Architektur von Agenten bzw. Agentensystemen (Abbildung 2.5) spezifiziert, ist der FIPA-Standard der Foundation of Intelligent Physical Agents [FIPA 2002b]. Dieser definiert grundlegende Dienste, die auf einer Agentenplattform vorhanden sein müssen. Dies ist zum Einen das *Agent Management System* (AMS), das die Verwaltung der Adressierung eines Agenten durchführt. Desweiteren übernimmt der *Directory Facilitator* (DF) die Verwaltung der durch Agenten angebotenen Dienstleistungen (Services). Die Registrierungen ihrer Services nehmen die Agenten selbst vor. Die Abfrage erfolgt String-basiert. Unterhalb dieser beiden Einheiten übernimmt das *Message Transport System* (MTS) den Transport sämtlicher Nachrichten zwischen den Agenten, dem AMS und dem DF. Außerdem sorgt es für den Transport von Nachrichten zu Agenten auf anderen Agentenplattformen.

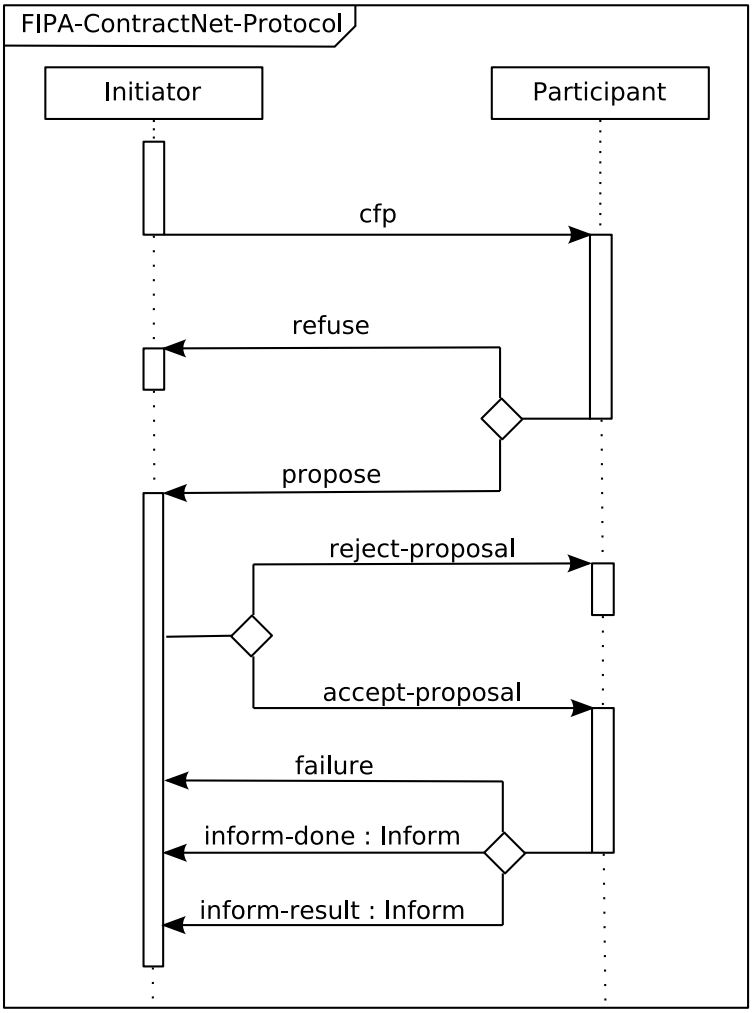


Abbildung 2.6.: FIPA Contract-Net-Protocol

Der FIPA-Standard legt außerdem eine Reihe von Interaktionsprotokollen für die Agentenkommunikation fest. Ein Beispiel hierfür ist das *Contract-Net-Protocol* (siehe Abb. 2.6) [FIPA 2002a]. Die Notation wird in [Odell u. a. 2001] definiert. Anhand dieses Beispiels lässt sich der grundsätzliche Aufbau der Agentenkommunikation erkennen. Er gliedert sich in eine Sender- und Empfängerseite sowie verschiedene mögliche Nachrichten, die zwischen beiden Seiten ausgetauscht werden. Häufig ist das Verschicken alternativer Nachrichten möglich, um beispielsweise Zustimmung bzw. Ablehnung auszudrücken. Beim Contract-Net-Protokoll fragt der Sender[1] mit Hilfe des Sprechaktes *Call-for-Proposal* zunächst andere Agenten nach Angeboten (Sprechakt *Propose*) für eine bestimmte Dienstleistung oder Aktion, die von mehreren möglichen Teilnehmern (Participants) eingehen können. Alternativ können Teilnehmer der Interaktion eine Zusammenarbeit auch ablehnen (Sprechakt *Refuse*). Der Initiator kann die eingegangenen Angebote dann entweder mit Hilfe der Sprechakte *Accept-* und *Reject-Proposal* annehmen bzw. ablehnen. Über die abgeschlossene Durchführung der Aktion informiert der teilnehmende Agent schließlich unter Zuhilfenahme des *Inform*-Sprechaktes. Sollte ein Fehler aufgetreten sein, erfolgt die Fehlermitteilung durch den Sprechakt *Failure*. Zur formalen Beschreibung des Zustands eines Multiagentensystems eignet sich die Multi-Modal-Logik $\mathcal{VSK}$. Sie wurde durch [Wooldridge und Lomuscio 2001] eingeführt.

2.2. Wissensrepräsentation

In diesen Abschnitt erfolgt ein Überblick über die Grundlagen der Wissensrepräsentation mit einem Schwerpunkt auf dem Bereich der Ontologien. In der aktuellen Forschung gibt es verschiedene Ansätze zur Repräsentation von Wissen. Sie alle verfolgen das Ziel einer bestimmten Darstellung von Informationen eine Bedeutung (*Semantik*) zu geben. Bestimmte Informationen sollen eine eindeutige Aussage bekommen, um so eine einheitliche Verarbeitung und Bewertung von Informationen und deren Beziehungen untereinander zu ermöglichen. Wichtig ist dies beispielsweise dann, wenn verschiedene Informationssysteme zur Lösung eines Problems herangezogen werden sollen und diese das Problem aus unterschiedlichen Perspektiven betrachten [Maedche und Motik 2003, S. 45]. Um eine Zusammenarbeit zu ermöglichen muss zunächst die Möglichkeit zur Integration der Informationen über das Problem geschaffen werden. Das Ziel aller Ansätze ist es, ein *semantisches Modell* der verfügbaren und zur Problemlösung benötigten Informationen aufzubauen. Nach [Maedche und Motik 2003] zeichnet sich ein semantisches Modell dadurch aus, dass es folgende drei Eigenschaften besitzt:

Gemeinsame Begrifflichkeit: Verschiedene Systeme, die beispielsweise eine Interaktion vornehmen wollen, müssen ein gemeinsames Vokabular besitzen, um sich beispielsweise über Interaktionsinhalte zu verständigen.

[1] Im FIPA-Standard auch *Inititator* genannt.

Beziehungen zwischen Dingen: Um komplexere Sachverhalte zu repräsentieren, ist es nötig die Beziehungen zwischen den Dingen und deren Rollen zu definieren.

Mathematisch fundierte Semantik: Um die Eindeutigkeit einer Semantik zu gewährleisten, muss diese mathematisch korrekt definiert sein.

Ein Ansatz zur Erfüllung dieser Eigenschaften sind Ontologien. Auf diese wird im nächsten Abschnitt näher eingegangen.

2.2.1. Ontologien

Ontologien werden im Bereich der Informatik in verschiedenen Bereichen eingesetzt. Dies ist zum Einen der Bereich der Kommunikation, in der Ontologien zur Repräsentation einer gemeinsamen Begrifflichkeit verwendet werden, dann der Bereich der Repräsentation von Wissen, über das das Schließen ermöglicht werden soll und schließlich dem Bereich der Sicherung von Informationen zur späteren Wiederverwendung [Gruninger und Lee 2002]. Nach [Gruber 1993] versteht man unter dem Begriff Ontologie folgendes:

> „An ontology is an explicit specification of a conceptualization.“ [Gruber 1993]

Eine Ontologie ist also ein Versuch der Einteilung der Welt in einzelne Konzepte, die bestimmte Objekte der Welt repräsentieren. Sie bieten die Möglichkeit zur Definition taxonomischer Beziehungen zwischen den Konzepten und außerdem der Repräsentation von Konzepteigenschaften. Zur Definition der Ontologien wurden verschiedene Ontologiesprachen entwickelt, die eine Definition und Notation von Ontologien ermöglichen und dabei unterschiedliche Ausdrucksmächtigkeit bieten. Hier sind beispielsweise KL-ONE [Brachman und Schmolze 1985] und Oil [Fensel u. a. 2000] zu erwähnen. Zu beachten ist hier einerseits, dass eine hohe Ausdrucksmächtigkeit zur Repräsentation komplexer Domänen nötig ist, dies aber andererseits grundsätzlich mit Anforderungen an Berechnungsverfahren auf diesen Weltrepräsentationen konkurriert. Weitere Informationen zum Begriff der Ontologien und deren Erstellung finden sich in [Gómez-Pérez u. a. 2004].

Mit der Zeit wurden verschiedene Repräsentationsansätze im Bereich der Ontologien entwickelt, auf die im Folgenden näher eingegangen wird. Zunächst wird auf den Bereich der *Beschreibungslogiken* eingegangen, die eine wichtige Grundlage aktueller Ontologiesprachen bilden.

Beschreibungslogik

An dieser Stelle wird eine an [Baader und Nutt 2003] angelehnte Einführung in den Bereich der Beschreibungslogiken gegeben. Beschreibungslogiken sind aus semantischen Netzen entwickelte Fragmente der Prädikatenlogik erster Stufe. Sie zeichnen sich durch ihre Ausdrucksstärke aus und sind entscheidbar. Das grundlegende Konzept dieser Beschreibungssprachen

ist die Einteilung der Wissensbasis in eine TBox (Terminological Box) und eine ABox (Assertional Box). Die TBox enthält hierbei das Wissen über Konzepte und Rollen innerhalb einer Domäne. Konzepte sind hierbei als Mengen von Individuen definiert und deren Beziehungen untereinander als Rollen. Eine Rolle ist also eine binäre Relation zwischen zwei Individuen. Neben atomaren Konzepten und Rollen lassen sich beliebig komplexe Beschreibungen von Konzepten und Rollen erstellen. Die TBox enthält somit das konzeptionelle Wissen der Wissensbasis. In der ABox ist hingegen das Wissen über Instanzen dieser Konzepte und damit die Daten der Domäne repräsentiert. In dieser Einführung wird auf die Familie der $\mathcal{AL}$-Sprachen eingegangen werden. Die Sprache $\mathcal{AL}$ definiert grundlegende Elemente einer Beschreibungslogik. Vorgestellt wurde sie in [Schmidt-Schauß und Smolka 1991]. Zu dieser grundlegenden Beschreibungslogik existieren verschiedene Erweiterungen, die im Rahmen dieser Einführung ebenfalls integriert werden.

Konzepte können innerhalb der Beschreibungslogik durch die folgenden Konstruktoren erzeugt werden. Dabei steht A für ein atomares Konzept, R für eine atomare Rolle und C und D für beliebige Konzeptbeschreibungen. n ist definiert als nicht-negative Ganzzahl.

$C,\ D$	$\longrightarrow$	$A \mid$	(atomares Konzept)
		$\top \mid$	(universelles Konzept)
		$\bot \mid$	(leeres Konzept)
		$\neg A \mid$	(atomare Negation)
		$\neg C \mid$	(Negation)
		$C \sqcap D \mid$	(Schnittmenge)
		$C \sqcup D \mid$	(Vereinigung)
		$\forall R.C \mid$	(Wertbeschränkung)
		$\exists R.\top \mid$	(beschränkte Existenzquantifikation)
		$\exists R.C \mid$	(Existenzquantifikation)
		$\geqslant nR \mid$	(*minimale* Kardinalitätsbeschränkung)
		$\leqslant nR \mid$	(*maximale* Kardinalitätsbeschränkung)

Mit Hilfe der folgenden terminologischen Axiome lassen sich Aussagen über die Beziehungen von Konzepten bzw. Rollen untereinander treffen. C und D bezeichnen Konzepte; R und S sind definiert als beliebe Relationen zwischen Individuen (Rollen).

$C \sqsubseteq D$	$(R \sqsubseteq S)$	(Inklusion)
$C \equiv D$	$(R \equiv S)$	(Äquivalenz)

Im Folgenden soll formal die Semantik der Beschreibungslogik $\mathcal{AL}$ und ihrer Erweiterungen definiert werden. Dazu seien die Interpretationen $\mathcal{I}$ als nicht leere Menge $\Delta^{\mathcal{I}}$ definiert. Die Menge $\Delta^{\mathcal{I}}$ wird die Domäne der Interpretation genannt. Durch eine Interpretationsfunktion wird jedem atomaren Konzept A eine Menge $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ zugewiesen. Einer atomaren Rolle R wird die binäre Relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ zugeordnet. Durch die folgenden Definitionen wird die Interpretationsfunktion für die Konzeptdefinitionen festgelegt:

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \forall b.\, (a, b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\
(\exists R.\top)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b.\, (a, b) \in R^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b.\, (a, b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \\
(\geqslant nR)^{\mathcal{I}} &= \left\{ a \in \Delta^{\mathcal{I}} \;\middle|\; |\{b \mid (a, b) \in R^{\mathcal{I}}\}| \;\geq\; n \right\} \\
(\leqslant nR)^{\mathcal{I}} &= \left\{ a \in \Delta^{\mathcal{I}} \;\middle|\; |\{b \mid (a, b) \in R^{\mathcal{I}}\}| \;\leq\; n \right\}
\end{aligned}
$$

Aufbauend auf diesen grundlegenden Sprachelementen sind noch folgende Definitionen Bestandteil der Beschreibungslogik $\mathcal{SHOIN}(\mathcal{D})$, die eine konkrete Erweiterung der Beschreibungslogik $\mathcal{AL}$ ist. Zunächst sind dies die so genannten *Nominals*, die eine Konzeptdefinition durch die dem Konzept zugeordneten Individuen erlauben. Formal ist dies definiert als der Konstruktor $\{a_1, \ldots, a_n\}$ mit der Semantik:

$$
\{a_1, \ldots, a_n\} \;=\; \{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}
$$

Zusätzlich sind die transitiven Hüllen für Rollen und inverse Rollen definiert:

$$
\begin{aligned}
(R^+)^{\mathcal{I}} &= \textstyle\bigcup_{i \geq 1} (R^{\mathcal{I}})^i, \text{ d. h. } (R^+)^{\mathcal{I}} \text{ ist transitive Hülle von} (R)^{\mathcal{I}} \\
(R^-)^{\mathcal{I}} &= \{(b, a) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (a, b) \in R^{\mathcal{I}}\}
\end{aligned}
$$

Eine weitere wichtige Erweiterung der Beschreibungslogik $\mathcal{AL}$ sind die so genannten konkreten Domänen (*Concrete Domains*). Die Einbeziehung der concrete domains bietet die Möglichkeit beispielsweise die Menge der natürlichen Zahlen $\mathbb{N}$ in die Definition von Konzepten einzubeziehen. Sachverhalte der Domäne, wie z. B. die Beschränkung einer Eigenschaft auf bestimmte Zahlen, können so direkt formuliert werden und müssen nicht durch eine abstrakte Darstellung modelliert werden. Hierdurch ergeben sich zusätzlich die Konstruktoren:

$$
\begin{aligned}
C,\, D \longrightarrow\; & \leqslant_n R && \text{(maximale Beschränkung auf konkreter Domäne)} \\
& \geqslant_n R && \text{(minimale Beschränkung auf konkreter Domäne)} \\
& =_n R && \text{(exakte Beschränkung auf konkreter Domäne)}
\end{aligned}
$$

Mit Hilfe dieser Definitionen lassen sich nun Konzepthierarchien aufbauen. Ein Beispiel für Familienbeziehungen ist in Abbildung 2.2.1 dargestellt.

$$
\begin{aligned}
\text{Frau} &\equiv \text{Person} \sqcap \text{Weiblich} \\
\text{Mann} &\equiv \text{Person} \sqcap \neg\text{Woman} \\
\text{Mutter} &\equiv \text{Frau} \sqcap \exists\text{hatKind.Person} \\
\text{Vater} &\equiv \text{Mann} \sqcap \exists\text{hatKind.Person} \\
\text{Elternteil} &\equiv \text{Vater} \sqcup \text{Mutter} \\
\text{Großmutter} &\equiv \text{Mutter} \sqcap \exists\text{hatKind.Parent} \\
\text{MutterMitVielenKindern} &\equiv \text{Mutter} \sqcap \geqslant 3\ \text{hatKind} \\
\text{MutterOhneTocher} &\equiv \text{Mutter} \sqcap \forall\text{hatKind.}\neg\text{Frau} \\
\text{Ehefrau} &\equiv \text{Frau} \sqcap \exists\text{hatEhemann.Mann}
\end{aligned}
$$

Abbildung 2.7.: Eine bespielhafte TBox aus Konzeptdefinitionen für Familienbeziehungen (nach [Baader und Nutt 2003, S. 52]).

Ontologiesprachen

Konkrete Ontologiesprachen, die im Rahmen der Wissensrepräsentation Verwendung finden, basieren auf den im vorherigen Abschnitt eingeführten Beschreibungslogiken bzw. lassen sich auch darauf abbilden. Dieser Zusammenhang ermöglicht den Einsatz logischer Inferenzverfahren zur Ableitung von Informationen aus dem bestehenden Wissen. Die Ontologiesprache DAML+Oil ist eine Kombination der durch die *DARPA*[2] einwickelten DARPA Agent Markup Language (DAML) und des *Ontology Interference Layer* (Oil)[3] genannten Ansatzes für ein Repräsentations- und Inferenz-Framework [Fensel u. a. 2000]. Der Ansatz besteht in der Zusammenführung von Elementen aus Frame-basierten Ontologiesprachen und dem Einsatz von Beschreibenslogiken zur Wissensrepräsentation. Nähere Informationen finden sich in der Definition des *World Wide Web Consortium* (W3C) [Connolly u. a. 2001].

Der Nachfolger, der nicht mehr weiterentwickelten Sprache DAML+Oil, ist die Sprache OWL, deren Definition sich in [Smith u. a. 2004] findet. Grundsätzlich ist die Sprache in drei Unterfragmente eingeteilt, die sich durch ihre erlaubten Sprachelemente und damit in ihrer Ausdrucksmächtigkeit unterscheiden. Dies ist zunächst *OWL/Lite*, das ein eingeschränktes, grundlegendes Vokabular zur Wissensdarstellung bereitstellt. Eine Erweiterung hiervon bildet *OWL/DL*, dessen Ausdrucksmächtigkeit vergleichbar zur oben erwähnten Beschreibungslogik $\mathcal{SHOIN}(\mathcal{D})$ ist. Den vollen Sprachumfang stellt schließlich *OWL/Full* zur Verfügung. Dies schließt beispielsweise *Reifikation*[4] mit ein. Das Ausmaß der Ausdrucksmächtigkeit hat selbstverständlich Einfluss auf die Entscheidbarkeit des einzelnen Sprachdialekte. Während die erstgenannten Sprachfragmente (OWL/Lite und OWL/DL) entscheidbar sind, ist OWL/Full unentscheidbar. Als Serialisierung einer OWL-Ontologie ist die OWL-RDF-Syntax üb-

[2] siehe `http://www.darpa.mil/`

[3] siehe `http://www.ontoknowledge.org/oil/`

[4] Reifikation ist die Möglichkeit zur Formulierung von Aussagen über Aussagen

lich, d. h. die Ontologie wird in Form eines RDF-Graphens[5] abgelegt. Wichtig für das weitere Verständnis ist noch die Anmerkung, dass in der Definition von OWL Konzepte als *Classes* bezeichnet werden und die in Abschnitt 2.2.1 erwähnten Rollen in diesem Ansatz Properties heißen.

Weitere Information zu den Möglichkeiten, die sich durch die Repräsentation von Wissen in OWL bieten und deren praktische Anwendung bietet [Horridge u. a. 2004].

2.2.2. Weitere Arten der Wissensrepräsentation

Die bis jetzt erwähnten Wissensrepräsentationssprachen bedienen sich zur Darstellung von Wissen des Einsatzes von Beschreibungslogiken. Diese Logiken sind Fragmente der Prädikatenlogik erster Stufe, so dass es sinnvoll erscheinen könnte, diese direkt zur Repräsentation einzusetzen. Dafür würde sicherlich die große Ausdrucksmächtigkeit sprechen, da sich über die beliebige Formulierung der Prädikate beliebige Aussagen über die Welt formulieren lassen. D. h. durch den Einsatz der Prädikatenlogik erster Stufe wäre eine hohe Flexibilität in der Formulierung der Weltbeschreibungen möglich, der aber die Semi-Entscheidbarkeit als Problem gegenübersteht. Modallogiken sind Erweiterungen der Prädikatenlogik um weitere Modalitäten wie beispielsweise „möglich“ oder „notwendig“, die die Ausdrucksmächtigkeit erweitern. Sie sind verwandt zu den Beschreibungslogiken. Die Verwendung von Modallogiken ist im Bereich der Modellierung von Agentensystemen und -architekturen verbreitet da sich über die zusätzlichen Operatoren beispielsweise Konsequenzen von Aktionen in Form einer Zukunftsbetrachtung repräsentieren lassen.

2.2.3. Inferenzen auf Wissensbasen

Das Ziel der Wissensrepräsentation ist nicht nur die reine Erfassung und Darstellung von Wissen, sondern auch das Ermöglichen von Schlussfolgerungen über das erfasste Wissen. Aus vorhandenen Informationen sollen neue Fakten abgeleitet werden [Russel und Norvig 2003, S. 195]. Hierzu lassen sich verschiedene Verfahren einsetzen, die sich durch unterschiedliche Merkmale auszeichnen.

Zunächst soll kurz darauf eingegangen werden, welche Anfragen an eine Wissensbasis von Interesse sein können. Beispiele hierfür sind zunächst die Frage nach der Konsistenz der Wissensbasis oder darüber hinaus die Frage nach Äquivalenz zweier Konzepte. Wichtig ist außerdem die Zuordnung von Instanzen zu einer bestimmten Klasse von Konzepten (*Klassifikation*) und das Bilden einer Konzepthierarchie (*Subsumption*). Das Lösen von Inferenzproblemen wird im Allgemeinen auf den Nachweis der Unerfüllbarkeit einer Wissensbasis

[5] RDF ist eine formale Sprache, die semantische Annotationen zu Webseiten in Form von Tripeln speichert. Eine Einführung bietet [Manola und Miller 2004].

zurückgeführt. Da die Beschreibungslogiken, in denen das Wissen repräsentiert wird, Fragmente der Prädikatenlogik sind, erscheint zunächst der Einsatz der *prädikatenlogischen Resolution* sinnvoll. Hier ergibt sich das Problem, dass die existierenden Algorithmen nicht immer terminieren. Verschiedene auf Tableaux-Beweisen basierende Ansätze umgehen dieses Problem durch den Einsatz von Optimierungen und Einschränkungen. Beispiele hierfür sind Reasoning-Algorithmen *Racer* [Haarslev und Möller 2001], *FACT* [Horrocks 1998] und *Pellet* [Sirin und Parsia 2004] mit ihren entsprechenden Implementierungen.

Ein weiterer Algorithmus, der versucht Informationen aus einer in einer Ontologie abgelegten Taxonomie zu gewinnen ist der cobac*-Algorithmus [Timm und Woelk 2003]. Hier erfolgt eine Bewertung von Synergie- bzw. Konfliktpotenzial über die Entfernung von Konzepttermen in der Taxonomie.

Um die Einbindung einer Ontologie-Wissensbasis in ein Softwaresystem zu ermöglichen, ist eine Schnittstelle zwischen diesem und dem eingesetzten Reasoner nötig. Ein Standard, der diese Schnittstelle spezifiziert, ist das *DIG-Interface* [Bechhofer 2003]. Für Anfragen auf Wissensbasen gibt es verschiedene Abfragesprachen für in RDF-Graphen repräsentierte Informationen. Dies ist zum Einen *RDQL* [Seaborne 2004] und andererseits deren Weiterentwicklung SQARQL [Prud'hommeaux und Seaborne 2006].

Die Beispielanwendung *Wine Agent*[6] basiert auf einer weiteren Reasonerumsetzung (*JTP* [Fikes u. a. 2003b]) und der Abfragesprache *OWL-QL* [Fikes u. a. 2003a].

2.3. Planen

In diesem Abschnitt wird ein kurzer Überblick über den Bereich des Planens gegeben. Das Planen liegt aber nicht im Hauptfokus dieser Arbeit. In der Künstlichen Intelligenz wird das Finden einer Aktionssequenz zum Erreichen eines bestimmten Ziel als *Planen* bezeichnet. Diese Aktionssequenzen sind dann zur Ausführung durch autonome Systeme wie intelligente Agenten oder unbemannte Fahrzeuge vorgesehen. Ein Planungsproblem besteht üblicherweise aus drei Teilen:

- Einem definierten initialen Weltzustand,
- dem Ziel, das erreicht werden soll
- und einer Menge von möglichen Aktionen mit definierten Vor- und Nachbedingungen.

Es wird allgemein zwischen klassischen und nicht-klassischen Planungsproblemen unterschieden. In klassischen Planungsdomänen ist vollständiges Wissen über die Welt vorhanden, die außerdem endlich, statisch, deterministisch und diskret beobachtbar ist. Diese Voraussetzungen sind bei nicht-klassischen Planungsproblemen nicht gegeben [Russel und Nor-

[6] siehe `http://onto.stanford.edu:8080/wino/index.jsp`

vig 2003, S. 375]. Dies bedingt natürlich andere Anforderungen an die einzusetzenden Planungsalgorithmen. Die folgenden Abschnitte geben einen Überblick über die existierenden formalen Plansprachen zur Beschreibung von Planungsproblemen und die entwickelten Algorithmen zur Lösung derselben.

2.3.1. Plansprachen

Plansprachen dienen zur formalen Spezifikation eines Planungsproblems. Eine der zu diesem Zweck entwickelte Sprache ist *STRIPS*. Sie wurde ursprünglich zur Formulierung von Problemen für den gleichnamigen *Stanford Research Institute Problem Solver* [Fikes und Nilsson 1971] entwickelt. Ein Planungsproblem besteht aus genau den oben erwähnten Teilen. Der Initialzustand besteht aus einer Menge von Aussagen, die als wahr angenommen werden (*Closed World Assumption*). Für alle übrigen Annahmen gilt die Annahme, dass sie falsch sind. Für das Ziel des Plans ist spezifiziert welche Aussagen wahr und welche falsch sein sollen. Die Aktionen sind schließlich als Operatoren spezifiziert, die für ihre Ausführung bestimmte Aussagen voraussetzen und bestimmte Aussagen verändern.

Eine Erweiterung, die aufgrund der teilweise ungenügenden Ausdrucksmächtigkeit von STRIPS entwickelt wurde, ist die *Action Description Language* (ADL) [Pednault 1986]. Hier gilt im Gegensatz zu STRIPS die *Open World Assumption*, d. h. nicht ausdrücklich als wahr angenommene Aussagen gelten nicht als falsch sondern als unbekannt. Außerdem erlaubt ADL mehr Flexibilität in der Spezifikation von Zielen und Bedingungen. Es sind beispielsweise zusätzlich zu Konjunktionen auch Disjunktionen in der Zieldefinition erlaubt und Zustände dürfen negative Aussagen enthalten.

Ein weiterer Ansatz zur Definition einer Plansprache ist PDDL [Edelkamp und Hoffmann 2004]. Ihre hauptsächliche Motivation bestand darin, eine Vergleichbarkeit von Planern zu schaffen. PDDL stellt Untersprachen für STRIPS, ADL und zusätzlich für HTN-Planen (siehe **??**) bereit. Um diese Flexibilität zu ermöglichen, ist es nötig, bei jeder Problembeschreibung zu definieren, welche einzelnen Sprachelemente zur Lösung des Planungsproblems vorausgesetzt werden. PDDL wurde in verschiedenen Versionen weiterentwickelt, um Vergleichswettkämpfe zwischen verschiedenen Planalgorithmen zu ermöglichen. Die Version 2.2 beispielsweise für die 4th International Planning Competition, IPC4 [Edelkamp und Hoffmann 2004].

2.4. Produktionslogistik

Teil dieser Arbeit wird die Einführung und Definition eines realistisches Anwendungsszenarios sein. Als Domäne dieses Szenarios wurde die Produktionslogistik gewählt. Der nächste Abschnitt gibt zunächst einen kurzen Überblick über deren Eigenschaften, führt dann den Einsatz von Agenten in diesem Bereich ein und stellt schließlich ein konkretes Szenario vor.

2.4.1. Einführung in die Produktionslogistik

Die Produktionslogistik beschäftigt sich mit der Planung und Steuerung von Prozessen in der betrieblichen Produktion. Nach [Glaser u. a. 1992] ist dieser Bereich wie folgt definiert:

> „Das Gebiet der Produktionsplanung und -steuerung umfasst die Gesamtheit von Dispositionen, die auf die Festlegung eines Absatz- bzw. Produktionsprogramms und die Bestimmung des Vollzugs dieses Programms in mengemäßiger und zeitlicher Hinsicht ausgerichtet sind.“ [Glaser u. a. 1992, S. 1]

Der betrachtete Teilaspekt der Produktion ist die innerbetriebliche Fertigung[7] und zwar die Arbeitsplanung auf Shop-Floor-Ebene. Gegenwärtig ist hier eine starke Trennung zwischen der zunächst zu leistenden Arbeitsplanung und der nachfolgenden Fertigungssteuerung festzustellen. Diese Trennung verhindert eine Berücksichtigung aktueller Informationen aus der Fertigung in der Arbeitsplanung. Durch aktuelle Entwicklungen ergibt sich die Anforderung, dass individuelle Kundenwünsche bei der Auftragsabwicklung stärker zu berücksichtigen sind. Darüber hinaus führt die momentane Entwicklung zu einer Verkleinerung der Losgrößen und einer Steigerung der Komplexität des Fertigungsprozesses durch den Einsatz verschiedener neuer Fertigungsmethoden. Aktuelle Forschungsarbeiten beschäftigen sich in diesem Bereich mit der Verbesserung der Verteilungs- und Zuordnungsverfahren der vorhandenen Aufträge, um eine möglichst optimale Ausnutzung der Maschinenkapazität zu erreichen. Gewünscht ist zusätzlich eine Überwindung der oben aufgezeigten Grenzen der aktuellen Systeme.

Aus produktionslogistischer Sicht erfolgt in diesem Bereich eine Kapazitätsplanung von Aufträgen zu Maschinen, woraus sich die beiden Elemente der Domäne – Aufträge und Maschinen – ergeben. Es handelt sich bei der zu lösenden Aufgabe um ein typisches *Job-Shop-Scheduling*-Szenario[8], das die Produktion einer Mischung von Aufträgen über verschiedene Fertigungsschritte auf unterschiedlichen Maschinen umfasst. Die Abfolge der Bearbeitungsschritte ist dabei vom Typ des Auftrags – und damit dem zu fertigenden Produkt – abhängig. Bei der Annahme von Aufträgen und deren Zuweisung zu bestimmten Produktionszeitpunkten und Maschinen gilt es verschiedene Faktoren zu berücksichtigen. Ein Beispiel hierfür sind

[7] Der Begriff Produktion umfasst zusätzlich die Montage vorgefertigter Teile.
[8] nähere Erläuterungen hierzu finden sich in [Blanchard 1992, S. 299]

andere Werkzeugbestückungen und -einstellungen für verschiedene Arten von Aufträgen, die auf einer bestimmten Maschine gefertigt werden sollen. Die Zeit, die zum Ändern dieser Bestückungen und Einstellungen benötigt wird, bezeichnet man als *Rüstzeit*. Durch geschickte Einplanung von Aufträgen kann nun versucht werden, das Auftreten von Rüstzeiten zu minimieren. Dies kann zum Beispiel dadurch geschehen, dass versucht wird, Aufträge, die die gleiche Werkzeugbestückung erfordern, nacheinander zu fertigen. Zusätzlich gibt es verschiedene andere Kriterien, deren Optimierung angestrebt wird.

2.4.2. Agenten in der Fertigung

Ein Ansatz zur Optimierung der Fertigung ist der Einsatz von Agenten. Ein Projekt, dass sich mit dieser Thematik beschäftigt, ist das Projekt „Intaps“ [Tönshoff u. a. 2002]. Im Rahmen dieses Projektes wurde ein Ansatz entwickelt, in dem Aufträge und Maschinen durch Agenten repräsentiert werden und durch einen dezentralen Lösungsansatz die in Abschnitt 2.4.1 genannten Einschränkungen überwunden werden. Um eine Zuordnung von Aufträgen zu Maschinen zu erreichen, finden zwischen Auftrags- und Maschinenagenten Verhandlungen über zu fertigende Produkteigenschaften und Freikapazitäten statt. Ergebnis des Verhandlungsprozesses ist eine Zuordnung der Aufträge zu Maschinen, die versucht, die oben erwähnten Optimierungskriterien zu berücksichtigen.

Ein Problem, dass sich häufig bei diesen Verhandlungen ergibt ist die Identifikation von Maschinen, welche die nötigen Fähigkeiten (engl. *Capabilities*) besitzen, um einen bestimmten Auftragstyp zu fertigen. Dieser Prozess wird als *Matchmaking* bezeichnet. Aktuell besteht wie schon erwähnt das Problem, dass dieser Match-Making-Prozess in Agentenverhandlungen häufig fehlschlägt, da meist wenig Flexibilität im Zuordnungsprozess zwischen Problemen und Problemlösungen vorhanden ist, aber die Zusammensetzung des Agentenumfelds und die angefragten Dienstleistungen bzw. Probleme und deren Eigenschaften sehr dynamisch sind. Die fehlende Flexibilität entsteht durch eine größtenteils statische Zuordnung zwischen Problemen und Problemlösungen, die keinen Raum für flexible Verteilung und Zuordnung lässt. Es bestehen hierdurch wenig Möglichkeiten zur Reaktion auf Störungen, die z. B. bei Ausfall eines Agenten entstehen. Engpässe, die durch ungünstige Konstellationen von Auftragseigenschaften und Fähigkeiten der Maschinen entstehen, sind die Folge.

2.4.3. Szenario „Sterlingmotor“

Als durchgängiges Anwendungszenario der Arbeit wird ein im Rahmen des Projektes IntaPS verwendetes Szenario genutzt. Das Szenario dient zum Einen der Erklärung einer Reihe von Aspekten bei der Darstellung des aktuellen Stands der Forschung und des eigenen Ansatzes und zum Anderen auch der abschließenden Evaluation des entwickelten Ansatzes. Dieses Szenario definiert die zu fertigende Baugruppe „Sterlingmotor“, die sich aus verschiedenen Unterbaugruppen zusammensetzt. Eine schematische Darstellung ist in Abbildung 2.8

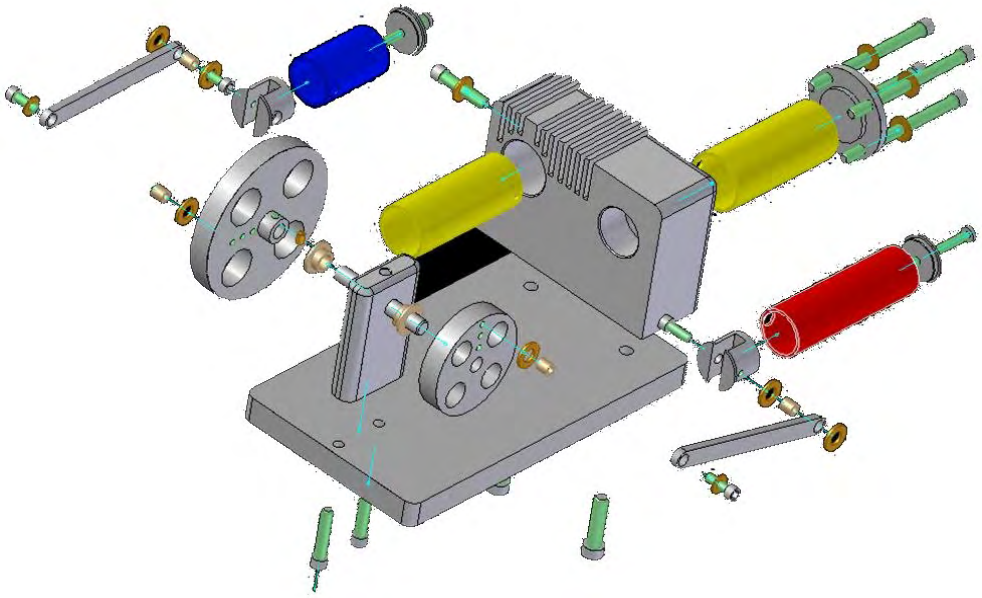


Abbildung 2.8.: Explosionsdarstellung des zu fertigenden Stirlingmotors

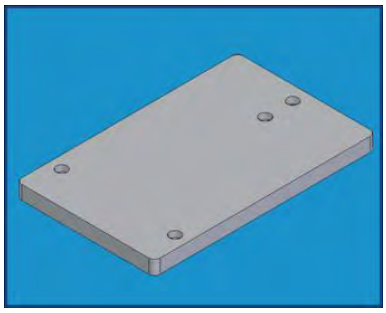
illustriert. Für jede Unterbaugruppe müssen mehrere Bauteile entweder in Eigenfertigung hergestellt oder zugekauft werden. In Eigenfertigung[9] müssen unter anderem folgende 14 in Tabelle 2.1 aufgeführten Bauteile gefertigt werden.

Tabelle 2.1.: Zur Herstellung des Stirlingmotors zu fertigende Bauteile

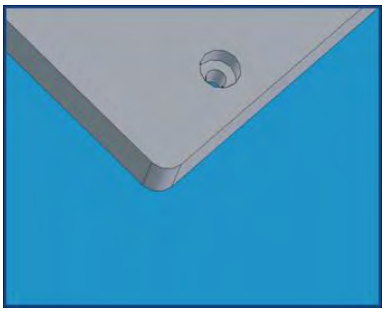
Grundplatte	Zylinderkopf
Motorblock	Kolbendeckel
Lagerblock	Kolbenboden
Arbeitszylinder	Pleularbeitskolben
Verdrängerzylinder	Pleulverdrängerkolben
Arbeitskolben	Kurbelrad
Verdrängerkolben	Schwungrad

Diese Bauteile erfordern wiederum verschiedene Bearbeitungsmerkmale, die so genannten *Features*. Ein Feature stellt ein geometrisch definiertes Element eines Bauteils dar. Aus verschiedenen Elementen lassen sich beliebig komplexe Bauteile zusammensetzen. Zusätzlich zu den geometrischen Eigenschaften eines Features lassen sich auch Informationen über eine bestimmte Werkstoffart zur Featuredefinition hinzufügen. Eine Festlegung dieser Features erfolgt durch [ISO 14649-10 2004]. Im Szenario Sterlingmotor ist beispielsweise für das Bauteil „Grundplatte“ das Feature „Edge Round“ mit einem bestimmten Radius erforderlich. Der

[9] Nur diese sind für die Arbeitsplanung und Fertigungssteuerung relevant.



(a) Bauteil „Grundplatte“



(b) Feature „Counterbore Hole“

Abbildung 2.9.: Verschiedene Elemente des Szenarios

Tabelle 2.2.: Der Fertigungsplan für das Bauteil „Grundplatte“

Bauteilbezeichnung		**Fertigungsnummer**	
Grundplatte		CB–WS 05–01	
Fertigungsfeature	**Variable**	**Wert in mm**	**Arbeitsplannummer**
Counterbore Hole	Durchmesser 1	8,0	30–40
	Durchmesser 2	4,5	
	Tiefe des Absatzes	4,5	
	Tiefe gesamt	10,0	
Rectangular Closed Shape Profil	Tiefe	1,0	20
	Länge	100,0	
	Breite	10,0	
Edge Round	Radius	4,0	50

vollständige Fertigungsplan ist in Tabelle 2.2 dargestellt.

Um dieses Feature zu fertigen, kommen verschiedene Bearbeitungsverfahren in Frage: In diesem Fall wären das „Profilfräsen“ oder „Profilschleifen“, die alternativ zur Anwendung kommen könnten. Eine Klassifikation der Bearbeitungs- bzw. Fertigungsverfahren ist durch [DIN 8580 2003] erfolgt. Hier werden die verschiedenen Fertigungsverfahren in mehre Hauptgruppen, Gruppen und Untergruppen eingeteilt. Für dieses Szenario ist die Hauptgruppe 3 „Trennen“ mit den das „Spanen“ betreffenden Untergruppen 3.2 und 3.3 relevant. In der weiteren Unterteilung nach [DIN 8589-0 2003] findet sich dann beispielsweise das Verfahren „Fräsen“ in der Untergruppe 3.2 mit der Gruppennummer 3.2.2. Diese Einteilung nach dem zu verwendenden Werkzeug kann schließlich noch durch eine Unterteilung nach der zu erzeugenden Fläche ergänzt werden, so dass sich das Bearbeitungsverfahren 3.2.2.1 „Planfräsen“ ergibt. Die Bearbeitungsverfahren werden von verschiedenen Maschinen bereitgestellt. Für das Verfahren „Planfräsen“ kommen zum Beispiel die Maschinen „Heller PFV-1“, „FST-CL-160“ oder „MC 16“ zum Einsatz.

2.5. Anforderungen an einen flexiblen Ansatz zum Ontologie-basierten Agentenmatching

In diesem Abschnitt erfolgt eine systematische Aufstellung der Anforderungen an ein flexibles Capability-Matching. Die Anforderungen lassen sich grundsätzlich in die folgenden Bereiche gliedern:

Explizite, semantische Modellierung der Agenten-Capabilities Zunächst muss eine Modellierung der Capabilities durch einen geeigneten Repräsentationsansatz erfolgen, um ein Schließen über die Fähigkeiten eines Agenten zu ermöglichen.

Einsatz eines Inferenzverfahrens zum Matchmaking Das Schließen über die Agenten-Capabilities ist durch den Einsatz eines geeigneten Inferenzverfahrens im Matchmaking-Prozess zu ermöglichen.

Flexibles Matchmaking Dieses ist die zentrale Anforderung der Arbeit und soll ein flexibles Matchmaking zwischen Problemen und Problemlösungen ermöglichen.

Integration in Agentenplanung Die Capabilities müssen schließlich in die Planung des Agenten integriert werden.

Integration in den Diskursagentenansatz Das flexible Matchmaking ist abschließend in den Diskursagentenansatz zu integrieren.

Diese zentralen Anforderungen werden im Folgenden näher ausgeführt und verfeinert. Die Anforderung der expliziten Modellierung der Agentencapabilities formuliert die Forderung nach einer Zuordnung von Problemlösungsfähigkeiten zu bestimmten Agenten. Für jeden Agenten muss genau festgelegt sein, welche Dienste er anderen Agenten zur Verfügung stellen kann. Diese Formulierungen benötigen zusätzlich semantische Aussagekraft und müssen außerdem die Möglichkeit zur Definition von Beziehungen zwischen den einzelnen Capabilities erlauben. Sie müssen weiterhin eine ausreichend hohe Ausdrucksmächtigkeit besitzen, um die Modellierung komplexer Domänen zu ermöglichen, was z. B. die Berücksichtigung konkreter Werte und Wertebereiche einschließt. Weiterhin muss der zu wählende Repräsentationsansatz das Schließen bzw. die Inferenz über die Repräsentationseinheiten ermöglichen, um die Entscheidungsfindung bei der Zuordnung von Problemen zu Problemlösungen zu ermöglichen bzw. zu unterstützen.

Diese Entscheidungsfindung wird durch die zweite Forderung nach dem Einsatz eines geeigneten Inferenzverfahrens weiterführt. Dieses Inferenzverfahren muss einerseits eine flexible Zuordnung von Problemlösungen zu Problemen ermöglichen, aber andererseits auch insoweit effizient arbeiten, dass ein Einsatz im Rahmen der Entscheidungsfindung eines Agenten bzw. eine Einbettung in die Verhandlung zwischen Agenten in einem Agentensystem sinnvoll möglich ist. Es muss zusätzlich die Möglichkeit besitzen die Inferenz auch über konkrete Werte und Wertebereiche durchzuführen, um z. B. in technischen Domänen, wie der vorgestellten

Produktionslogistik zum Einsatz zu kommen. Ein realistisches Szenario in diesem Bereich erfordert die Formulierung von konkreten Maßen von Aufträgen und minimalen und maximalen Fertigungsgrößen auf Maschinenseite. Selbstverständlich soll das eingesetzte Verfahren auch die Anforderung an Korrektheit und Vollständigkeit erfüllen. Durch die Ermöglichung der flexiblen Zuordnung ist die zentrale Anforderung des flexiblen Matchings umzusetzen. Flexibilität bedeutet an dieser Stelle, dass beispielsweise Verallgemeinerungen bzw. Spezialisierungen von Agenten-Capabilities möglich sind. Der Ansatz muss zusätzlich den Aufbau von Subsumptionshierarchien ermöglichen.

Die beiden letztgenannten Anforderungen verlangen schließlich nach einer Integration der Agentencapabilities in die Planung des Agenten und in den Diskursagentenansatz. Als Planung ist in diesem Fall das Verfahren zu verstehen, nach dem der Agent bestimmt, wie er zur Lösung eines Problems vorzugehen hat. Eine Integration in diesem Schritt ist insoweit nötig, als beispielsweise in dem Fall, dass ein Agent mehrere Capabilities besitzt, ein Wechsel zwischen diesen Zwischenschritte erfordert, die durch diesen Wechsel bedingt werden. Der Diskursagentenansatz ist ein schon vorhandener Ansatz einer BDI-basierten Agentenarchitektur. Sie führt Ansätze aus dem Bereich der Agentenarchitekturen einerseits und dem der Multi-Agenten-Systeme andererseits zusammen. Ein flexibles Matchingverfahren fehlt im Rahmen der Architektur und ist durch diese Arbeit hinzuzufügen.

Kapitel 3.

Stand der Forschung

In diesem Kapitel wird der aktuelle Stand der Forschung dargestellt. Dabei wird ausgehend von den in Abschnitt 2.5 definierten Anforderungen ein Überblick über verschiedene Bereiche gegeben, aus denen der Einsatz von Techniken zur Erfüllung der Anforderungen sinnvoll erscheint. Da das zentrale Ziel der Arbeit die Flexibilisierung des Agenten-Matchmakings ist, liegt der Fokus bei den zu untersuchenden Arbeiten im Bereich des Matchmakings. Nach der Vorstellung eines Ansatzes aus dem Bereich der Semantic Web Services erfolgt die Untersuchung weiterer Ansätze zum Matchmaking und abschließend eine Bewertung der Ansätze und die Auswahl eines Ansatzes zur Integration in den eigenen Ansatz.

3.1. Überblick über den Bereich des Matchmakings

Das Matchmaking bzw. Matching ist eine wichtige Operation, die immer dann anfällt, wenn Informationen auf unterschiedliche Art repräsentiert werden und anschließend verglichen werden sollen. In dieser Arbeit wird der Fall betrachtet, dass eine Zuordnung zwischen einem bestimmten Problem und verschiedenen Problemlösungen stattfinden muss. Genauer gesagt muss festgestellt werden, ob die jeweilige Problemlösung zu einem Problem passt. Zunächst soll in diesem Abschnitt eine Darstellung des aktuellen Stands der Forschung im Bereich des Matchingmakings mit einer Vorstellung verschiedener Ansätze aus diesem Bereich erfolgen. Dieser Überblick ist an [Shvaiko und Euzenat 2005] angelehnt. Diese Arbeit entwickelt eine Klassifizierung der verschiedenen, vorhandenen Matchingverfahren. Sie fokussiert dabei auf Ansätze, die auf Schema-Ebene ansetzen. Ansätze, die ein Matching auf Instanzebene vornehmen, werden nicht beachtet. Generell kann es sich bei den Daten, zwischen denen ein Matching vorgenommen werden soll, um sehr unterschiedliche Arten der Repräsentation handeln. Im Bereich der Informationsintegration sind dies häufig Datenbankschemata, im Internet XML-repräsentierte Daten sowie Ontologien im Bereich der Wissensrepräsentation. Aber auch andere beispielsweise Beschreibungslogik-basierte Ansätze verlangen nach einem Matching zwischen verschiedenen Konzepten. Zu unterscheiden sind nach [Shvaiko und Euzenat 2005] einerseits reine Schema-basierte Ansätze und andererseits Ontologie-basierte Ansätze. Der wichtigste Unterschied zwischen beiden Ansätzen liegt darin, dass Ontologien eine

formale Semantik besitzen und diese im Rahmen des Matching zum Vergleich verschiedener Konzepte herangezogen werden kann, wohingegen Schema-basierte Ansätze die Bedeutung der Konzepte aus den Daten selbst folgern müssen. Im Folgenden werden durch [Shvaiko und Euzenat 2005] einige Beispiele des Einsatzes von Matchingverfahren gegeben. Zu nennen sind hier die Integration von Firmenkatalogen auf Datenbankebene, der Einsatz im Rahmen der Agentenkommunikation sowie der Integration von Web Services[1]. Diese Einsatzgebiete unterscheiden sich stark in den Anforderungen an ein geeignetes Matchingverfahren. Im ersten Fall ist das Einsatzgebiet des Matching als eher statisch anzusehen und die Durchführung des Matchings braucht zunächst nur einmal bei Beginn eines Geschäftsverhältnisses zu erfolgen. Die beiden letztgenannten sind hingegen als dynamische Umgebungen anzusehen, in denen sehr viel häufiger ein Matching-Prozess durchzuführen ist. Zu berücksichtigen ist außerdem, ob vom Matchingverfahren exakte Ergebnisse verlangt werden oder ob zugunsten einer höheren Performanz der Verfahren, beispielsweise in einer hochdynamischen Umgebung auch „ungefähre“ Ergebnisse akzeptiert werden. In aktuellen Matchingsystemen werden verschieden Ansätze kombiniert. Diese kann Kombination kann entweder sequentiell oder parallel aufgebaut sein. Sequentielle Ansätze werden als *hybride* Verfahren bezeichnet, wohingegen parallele Verfahren auch als *composite* Verfahren benannt werden. Allgemein gilt, dass bei der Kombination verschiedener Matchingverfahren die Einzelergebnisse in einer geeigneten Weise kombiniert werden müssen. Möglich ist dies beispielsweise über Mittelwertbildung über die Ergebnisse der Einzelverfahren.

Zur Erörterung des Problems des Matchings wird durch [Shvaiko und Euzenat 2005] ein so genanntes *mapping element* als Quintupel $\langle id,\ e,\ e',\ n, R\rangle$ definiert, dessen Elemente folgende Bedeutung haben:

- *id* identifiziert das *Mapping-Element* eindeutig,
- e und e' sind Elemente der Schemas bzw. der Ontologien, zwischen denen ein Match bestimmt werden soll.
- n wird als so genanntes *Confidence-Measure* eingeführt, das einen numerischen Gleichheitswert im Intervall [0,1] der Elemente e und e' angibt.
- R ist eine Relation zwischen den Elementen e und e'. Typisch sind
  - *Äquivalenz* ($=$),
  - *Allgemeiner* ($\sqsupseteq$),
  - *Disjunkt* ($\perp$)
  - und *Überlappend* ($\sqcap$).

Es wird außerdem ein *Alignment* als Menge einer Reihe von Mapping-Elements eingeführt. Der Prozess des *Matchings* liefert dann zu einem Paar zweier Schemas bzw. Ontologien o und o' unter Berücksichtigung etwaiger Parameter (p) und evtl. vorhandener weiterer Ressourcen (r) ein Alignment A' der Elemente der gegebenen Schemas/Ontologien. Dieser Ablauf ist in Abbildung 3.1 dargestellt.

[1] Auf diesen Bereich wird im Folgenden noch näher eingegangen

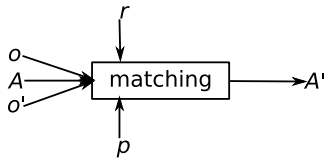


Abbildung 3.1.: Der Matching-Prozess nach [Shvaiko und Euzenat 2005]

Die durch [Shvaiko und Euzenat 2005] vorgenommene Klassifikation des Matching führt eine zweigeteilte Betrachtung des Matchings ein. Zum Einen findet Beachtung ob das Verfahren nur die Daten selbst oder zusätzlich ihre Struktur verarbeitet und wie diese interpretiert werden. Zum Anderen wird die Art der Daten selbst zur Klassifikation herangezogen. Diese Unterscheidung wird in so genannte *Layer* eingeteilt. Diese werden im Folgenden näher erläutert:

Granularity/Input Interpretation Layer Auf dieser Ebene findet Berücksichtigung, ob das Verfahren nur auf den Elementen oder zusätzlich auf ihrer Struktur arbeitet und wie die Elemente interpretiert werden. Dieser Layer unterteilt sich demnach in folgende *Level*:

- **Element-Level/Structure-Level** An dieser Stelle wird unterschieden, ob das zu klassifizierende Verfahren nur die Elemente, auf denen es arbeitet beachtet oder ob zusätzlich deren Beziehungen untereinander berücksichtigt werden.
- **Syntactic/External/Semantic** Die Unterscheidung auf diesem Level gliedert sich nach der Art, wie die zu matchenden Elemente interpretiert werden. Verfahren können hier einerseits rein auf der Ebene der syntaktischen Struktur der Elemente arbeiten oder andererseits externe Quellen zur Interpretation der Daten heranziehen. Der Fall einer semantischen Interpretation liegt vor, falls die Ermittlung der Bedeutungen der Elemente anhand einer formal definierten Semantik durchgeführt wird.

Kind of Input Layer Auch auf dieser Ebene findet eine Einteilung der bestehenden Ansätze anhand zweier Kriterien nach dem Eingabetyp der zu matchenden Elemente statt.

- **Terminological/Strutural/Semantics** Auf dieser Schicht wird unterschieden, ob die Daten nur terminologisch oder hierarchisch strukturiert sind. Zusätzlich können sie auch eine formal definierte Semantik besitzen.
- **String-based/Linguistic/Internal/Relational** Auf dieser Schicht erfolgt die Klassifikation der Eingabe des Matchings anhand der Methode ihrer Interpretation. Stringbasierte Verfahren betrachten eine Eingabe nur als Zeichenkette wohingegen linguistische Verfahren deren Aussage im linguistischen Sinne berücksichtigen. Der Unterschied zwischen interner und relationeller Verarbeitung liegt darin, dass einerseits nur interne Attribute der Elemente berücksichtigt werden oder andererseits auch Beziehungen zwischen Elementen beachtet werden.

Aus der Kombination der oben vorgestellten Klassifikations-Layer ergibt sich die durch [Sh-

vaiko und Euzenat 2005] als *Basic Techniques Layer* bezeichnete Schicht, auf der eine Einordnung vorhandener Matchingverfahren erfolgen kann.

Nach dieser Vorstellung der Klassifikation des Matchings werden im Folgenden einige Gruppen von Matchingverfahren mit ihren spezifischen Eigenschaften beschrieben. Eine Auswahl konkreter Matchmaking-Ansätze wird daran anschließend im nächsten Abschnitt beschrieben.

3.1.1. String-basierte Verfahren

String-basierte Verfahren sind Matching-Ansätze, die auf der Ebene der Elemente arbeiten und dabei das Matching durch Verarbeitung der zu diesen gehörenden Zeichenketten durchführen. Zum Einsatz kommen können an dieser Stelle eine Reihe verschiedener Verfahren, die Gleichheit bzw. Übereinstimmung von Strings bestimmen. Diese Ansätze gehen davon aus, dass übereinstimmende bzw. sich ähnelnde Zeichenketten ähnliche Dinge beschreiben. Die Auswahl an Verfahren reicht von relativ simplen Verfahren, die ihre Aussage über die Beachtung von Wortanfängen bzw. -enden treffen, über Verfahren, die bewerten wie stark eine Zeichenkette bearbeitet werden muss, um in eine andere Zeichenkette umgewandelt zu werden, bis hin zu komplexeren Verfahren, die eine Zerlegung der Zeichenketten vornehmen und dann eine Bewertung der Ähnlichkeit durchführen.

3.1.2. Constraint-basierte Verfahren

Constraint-basierte Verfahren, die ebenfalls unter die Element-Level-Verfahren einzuordnen sind, berücksichtigen Einschränkungen, die auf Eigenschaften oder Attributen der Elemente definiert werden. Diese Einschränkungen können sich beispielsweise auf bestimmte Datentypen beziehen oder auch die Kardinalität bestimmter Attribute. Beispielsweise ließe sich ein Attribut mit einer Menge von Bearbeitungsmerkmalen vom Typ *Bohrloch* definieren, das bei einem konkreten Werkstück die Kardinalität 3 besitzt. Dies bedeutet, dass bei diesem Werkstück drei Bohrlöcher zu fertigen sind. Das Matchingverfahren muss im Prozess des Findens eines passenden Bearbeitungsangebots diese Beschränkung berücksichtigen.

3.1.3. Modell-basierte Verfahren

Die Modell-basierten Verfahren fallen in die Kategorie der Struktur-berücksichtigenden Verfahren. Sie können bei der Berechnung des Matches zwischen Elementen neben einer festgelegten Struktur zusätzlich auf eine formal definierte Semantik zurückgreifen. Verfahren, die beschreibungslogische Modelle verarbeiten, können neben der Überprüfung, ob sich die Beschreibungen zweier Elemente auf beschreibungslogischer Ebene widersprechen, über die Berücksichtigung der formal definierten Beziehungen der Elemente ihren Zusammenhang im

Hinblick auf die Subsumption der Elemente untereinander überprüfen. Dies ermöglicht eine genaue Beurteilung des Matches im Hinblick auf die eingeführten Relationen *Äquivalenz*, *Allgemeiner*, *Disjunkt* und *Überlappend*. Der Ansatz der beschreibungslogischen Verfahren wird im Weiteren durch einen vorzustellenden Ansatz noch näher erläutert werden.

3.2. Ansätze zum Matchmaking

In diesem Abschnitt werden eine Reihe konkreter Arbeiten zum Matchmaking vorgestellt, die ebenfalls eine Zuordnung zwischen Agenten, die die Lösung eines Problems verlangen und anderen Agenten, die Probleme lösen können, ermöglichen wollen. Zunächst wird ein kurzer Überblick über das zu dieser Arbeit verwandte Forschungsfeld der Semantic Web Services gegeben und daran anschließend ein Ansatz zum Matchmaking aus diesem Bereich sowie eine Reihe weiterer Matchmakingansätze aus anderen Bereichen vorgestellt.

3.2.1. Semantic Web Services

Im Bereich der *Semantic Web Services* lassen sich einige Entsprechungen zum Matchmaking-Problem bei Agenten sehen. Das Ziel ist hier ebenfalls eine Zuordnung von problemlösenden Web Services zu einem bestimmten Problem zu finden. Ein Ansatz, um diese Zuordnung zu automatisieren ist die Erweiterung der *Web Services Description Language* (WSDL)[2] [Chinnici u. a. 2006] um Semantik. Durch maschinenverstehbare Beschreibungen soll z. B. das Auffinden von Informationen durch autonome Informationsagenten ermöglicht werden. Die Idee des Semantic Web wurde ursprünglich durch [Berners-Lee u. a. 2001] vorgeschlagen.

Ein Ansatz, um die Eigenschaften von Web Services zu beschreiben, ist die OWL Ontologie OWL-S [Martin u. a. 2004]. Durch diese Ontologie werden verschiedene Konzepte zur Klassifikation eines Web Services standardisiert. Zunächst erfolgt eine Aufteilung der Beschreibung in drei Unterkategorien bzw. Unterontologien:

Profile Das Profile eines Services definiert dessen Aufgabe, d. h. hier wird definiert zur Lösung welcher Aufgabe der Service in Anspruch genommen werden kann. Neben der Angabe des Anbieters erfolgt hier die Spezifikation der *Preconditions*, des *Inputs*, der *Effects* und der *Outputs* des Services. Die Preconditions (Vorbedingungen) legen fest, welche Bedingungen gegeben sein müssen, damit der Service arbeiten kann. Durch die Angabe der Inputs erfolgt die Festlegung der Eingaben, die ein Service benötigt. Die Effects eines Service definieren, welche Bedingungen durch den Service verändert werden; die *Results* geben schließlich an, was durch den Service produziert wird.

[2] WSDL ist eine standardisierte Sprache, die beispielsweise Zugriffsmöglichkeiten zu Web Services beschreibt.

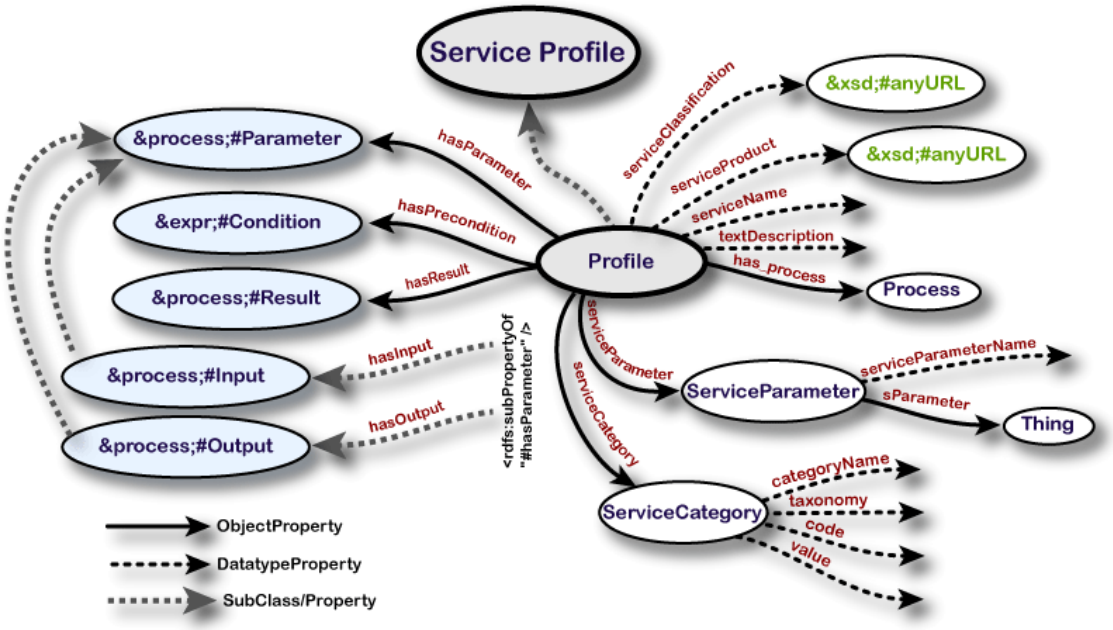


Abbildung 3.2.: Einige Klassen und Eigenschaften des Profile-Konzepts (Quelle: `http://www.ai.sri.com/daml/services/owl-s/1.2/overview`) Mit freundlicher Genehmigung der OWL-S Coalition.

Process Modell Mit Hilfe des Process-Modells erfolgt die Definition des genauen Aufbaus eines Services. Da sich hier die Vor- und Nachbedingungen eines Services festlegen lassen, ist dieses beispielsweise bei der Zusammenstellung verschiedener Web Services heranzuziehen. Für die Modellierung eines Services stehen verschiedene Prozessmodelle zur Verfügung.

Atomic Process Dieser Prozess bildet eine einfache Aktion eines Services ab, die innerhalb eines einzelnen Handlungsschritts erfolgt.

Composite Process Durch dieses Konzept erfolgt die Modellierung komplexer Prozesse, die die Durchführung mehrerer Handlungsschritte erfordern. Diese lassen sich jeweils durch atomare Prozesse abbilden.

Simple process Dieser Prozess soll die abstrakte Modellierung eines atomaren Prozesses einerseits bzw. eine stark vereinfachte Darstellung eines zusammengesetzten Prozesses andererseits bieten.

Grounding Das Grounding eines Services definiert, wie dieser in seine Umgebung eingebettet ist. Mit Hilfe dieses Konzepts erfolgt unter anderem die Definition der Protokolle, mit Hilfe derer der Service erreicht werden kann.

Im folgenden wird ein Ansatz zum Matchmaking für Semantic Web Services vorgestellt, der auf dem Profile-Konzept der OWL-S Ontologie aufsetzt. Das Profile-Konzept ist in Abbil-

dung 3.2 dargestellt. Es besitzt eine Reihe von Properties, die die oben erwähnten Aspekte abbilden.

A Software Framework For Matchmaking Based on Semantic Web Technology [Li und Horrocks 2004]

Li und Horrocks untersuchen in dieser Arbeit, wie mit Hilfe des Einsatzes von Technologien der Semantic Web Services das Anbieten und Auffinden von Dienstangeboten im Bereich des E-Commerce unterstützt werden kann. Sie beschreiben das Design und die Implementierung eines Prototypen zum Matchmaking, der mit der DAML-S-Ontologie und dem Einsatz eines Reasoners angebotene Web Services vergleicht. DAML-S ist ein auf DAML+Oil aufbauender Vorläufer von OWL-S. Sie begründen die Auswahl der DAML-S Ontologie zum Einen dadurch, dass diese durch ihre in DAML+OIL formulierten Konzepte den Einsatz eines Reasoners zum Berechnen der Subsumptionbeziehungen zwischen verschiedenen Service Profilen erlaube und zum Anderen durch die explizite Spezifikation der *Inputs* und *Outputs* ermögliche, die Attribute und Bedingungen eines Services zu spezifizieren. Zusätzliche Gründe für die Entscheidung seien die Schaffung einer gemeinsamen Semantik um Interoperabilität zu ermöglichen und die im Bereich des E-Commerce wichtige Unterstützung von konkreten Datentypen. Zusätzlich könne Flexibilität aufgrund der Möglichkeit der Repräsentation von wenig strukturierten Servicebeschreibungen erreicht werden.

Zur Repräsentation der Services entwickeln Li und Horrocks eine Bespielontologie, die den Verkauf von Computern modelliert. Dazu definieren sie eine Reihe von Unterkonzepten des Konzepts `ServiceProfile`:

$$\begin{aligned} \mathit{ServiceProfile} &\sqsubseteq \top \\ \mathit{Advertisement} &\sqsubseteq \mathit{ServiceProfile} \\ \mathit{Query} &\sqsubseteq \mathit{ServiceProfile} \\ \mathit{Template} &\sqsubseteq \mathit{ServiceProfile} \\ \mathit{Proposal} &\sqsubseteq \mathit{ServiceProfile} \end{aligned}$$

Außerdem werden die Services `Sales` und `Delivery` definiert. `Sales` erhält beispielsweise folgende Definition:

$$\begin{aligned} \mathit{Sales} \sqsubseteq\ & (=\ 1\ \mathit{providedBy.Actor}) \sqcap \\ & (=\ 1\ \mathit{requestedBy.Actor}) \sqcap \\ & (=\ 1\ \mathit{item.EEquipment}) \sqcap \\ & (=\ 1\ \mathit{hasQuantity.positiveInteger}) \sqcap \\ & (=\ 1\ \mathit{hasUnitPrice.nonNegInteger}) \sqcap \\ & (=\ 1\ \mathit{canDeliver.Delivery}) \end{aligned}$$

`Delivery` und `Actor` erhalten Definitionen nach dem gleichen Schema, wobei bei `Delivery` eine Spezifikation des Ortes und der Lieferzeit und bei `Actor` eine Namens- und Kreditwürdigkeitsangabe erfolgt. Der Verhandlungsgegenstand `PC` mit seiner Komponente `Processor` wird folgendermaßen festgelegt:

$$\begin{aligned} PC &\sqsubseteq EEquipment \sqcap \\ &\quad (= 1\ hasProcessor.Processor) \sqcap \\ &\quad (= 1\ memorySize.positiveInteger) \\ Processor &\equiv PentiumIII \sqcup Pentium4 \sqcup Athlon \end{aligned}$$

Die Parameter der Services gliedern sich dabei in dieser Domäne in die Inputs, wie beispielsweise die zu liefernde Menge oder den Preis pro Einheit und den Output, der in diesem Fall die bestellte Computereinheit ist. Formal ist dies festgelegt als:

$$\begin{aligned} inputs &\sqsubseteq parameter \\ outputs &\sqsubseteq parameter \\ \\ hasQuantity &\sqsubseteq intputs \\ hasUnitPrice &\sqsubseteq inputs \\ \\ item &\sqsubseteq outputs \end{aligned}$$

Die Definition eines Angebots und einer Anfrage spezifizieren Li und Horrocks anhand eines Beispiels mit konkreten Werten für die Parameter des `Advertisement`- und `Query`-Konzepts. Sie weisen darauf hin, dass sie eine Anpassung dieser Spezifikation vorgenommen haben. Diese wurde nötig durch Einschränkungen, die sich durch die DAML-S-Spezifikation ergeben und die die Flexibilität des Ansatzes eingeschränkt hätten. Die Einschränkung besteht in der geforderten Angabe des anbietenden und anfragenden `Actors` im `ServiceProfile`. Um die Einschränkung zu umgehen, definieren sie das `Advertisement` um und verbinden das ServiceProfile über eine Variable *Profile* mit dem Advertisement. Die Angabe des anfragenden und anbietenden `Actors` erfolgt außerhalb des `ServiceProfile`. Das Beispiel stellt sich dann wie folgt dar:

$$
\begin{aligned}
Advert1 \;=\; & (providesBy.(Actor \sqcap hasName.Georgia),\\
& requestedBy(Actor \sqcap \geq 5 hasCreditLevel),\\
& profile.(ServiceProfile \sqcap\\
& \forall item.(PC \sqcap \geq 128 memorySize) \sqcap\\
& \leq 700\ hasUnitPrice \sqcap\\
& \leq 200\ hasQuantity \sqcap\\
& \forall delivery.(Delivery \sqcap\\
& \leq 20020915\ date \sqcap \forall location.Bristol)))
\end{aligned}
$$

$$
\begin{aligned}
Query2 \;=\; & (providedBy(Actor \sqcap \forall hasName.Alan),\\
& requestedBy(Actor = 5 hasCreditLevel),\\
& profile(ServiceProfile \sqcap\\
& \forall item.(PC \sqcap = 256\ memorySize) \sqcap\\
& = 500\ hasUnitPrice))
\end{aligned}
$$

Ausgehend hiervon wird dann der Algorithmus zur Bestimmung einer Übereinstimmung – eines *Matches* – zwischen einer eingehenden Anfrage und einer Auswahl von Angeboten definiert. Dabei wird α als die Menge aller vorhandenen Angebote angenommen und *matches*(Q) als die Funktion, die die zur Anfrage Q passenden Angebote zurückgibt. Formal ist dies:

$$matches(Q) = \{A \in \alpha | compatible(A, Q)\}$$

$compatible(A, Q)$ wird dabei defniert als Erfüllbarkeit der Schnittmenge der beiden Beschreibungen $D1$ und $D2$:

$$compatible(D1, D2) \Leftrightarrow \neg(D1 \sqcap D2 \sqsubseteq \bot)$$

Die Ausgaben des Matching-Algorithmus werden im Folgenden weitergehender klassifiziert, um zu einer genaueren Definition des Matches zu kommen. Li und Horrocks bauen folgende Bewertungsskala für den Grad eines Matches auf:

Exact Für den Fall, dass das Angebot A und Anfrage B äquivalente Konzepte sind ($A \equiv B$), soll der Grad des Matches *exact* heißen.

PlugIn Falls der Request R ein Unterkonzept der Anfrage A ist ($R \subseteq A$), handelt es sich um einen *PlugIn*-Match.

Subsume Ist der Request R ein Oberkonzept des Angebots A ($A \subseteq R$), dann heißt der Match *Subsume*.

Intersection Um einen *Intersection*-Match handelt es sich, falls die Schnittmenge zwischen Angebot A und Anfrage R erfüllbar ist ($\neg(A \sqcap R \sqsubseteq \bot)$).

Disjoint Schließlich soll der Match *Disjoint* heißen, falls die Schnittmenge unerfüllbar ist ($A \sqcap R \sqsubseteq \bot$)).

Nach dieser Klassifikation ist der höchste Grad eines Matches der Exact-Match und der niedrigste Grad der Disjoint-Match, der die Nichterfüllbarkeit der Anfrage wiedergibt. Die dazwischenliegenden Matchtypen geben eine weitere Abstufung der Grade des Matches wieder. In ihrem Softwareprototypen verwenden Li und Horrocks den Reasoner Racer [Haarslev und Möller 2001], um die Subsumptionsbeziehungen zwischen Angeboten und Anfragen zu berechnen. Dazu wird zunächst die Hierarchie der Angebote klassifiziert. Anschließend wird das `ServiceProfile` der Anfrage R klassifiziert und bestimmt mit welchen Angeboten ein Exact-, PlugIn- oder Subsume-Match vorliegt. Durch die darauffolgende Klassifikation von $\neg R$ kann bestimmt werden, mit welchen Angeboten ein Intersection-Match vorliegt (falls ein Angebot $\neg R$ subsumiert aber nicht gleich ist) bzw. ein Disjoint-Match (falls ein Angebot von $\neg R$ subsumiert wird).

Die weitere Architektur des Prototypen, der zur Evaluation dieses Ansatzes mit Blick auf die Skalierbarkeit implementiert wurde, sieht neben anbietenden und anfragenden Agenten als zentrale Instanz den Agenten mit der Rolle *Host* vor, der die Verwaltung von Anfragen und Angeboten sowie das Matchmaking vornimmt.

3.2.2. LARKS

In [Sycara u. a. 2002] wird die Sprache *LARKS* (Language for Advertisement and Request for Knowledge Sharing) zur Repräsentation von Agentangeboten und -anfragen beschrieben. Ausgehend von der Feststellung, dass das Matchmaking zwischen Angeboten und Anfragen meist dynamisch erfolgt und daher effizient sein muss, wird außerdem ein auf der Sprache LARKS aufsetzender Match-Making-Algorithmus definiert, der, um dieser Anforderung gerecht zu werden, verschiedene Filter definiert, die zur Feststellung eines Matches herangezogen werden können. Das Matchmaking kann einerseits rein textuell erfolgen, es kann aber auch durch die Verbindung der Wörter der Textbeschreibungen mit Konzepten einer Konzeptsprache erweitert werden, um den Anforderungen komplexerer Domänen gerecht zu werden. Zunächst werden drei Rollen definiert, die innerhalb eines Agentensystem durch Agenten eingenommen werden können:

Provider Dieser Agententyp bietet seine Fähigkeiten anderen Agenten zur Nutzung an.

Requester Der Requester nimmt die Dienste der Provider in Anspruch und richtet entsprechende Anfragen an einen Matchmakeragent.

Matchmaker Diese Agenten bilden zentrale Instanzen, die zum Einen die Angebote der Provider entgegennehmen und zum Anderen versuchen, die Angebote den von Requestern empfangenen Anfragen zuzuordnen. Dazu übermitteln die Provider zunächst ihre Fähigkeiten an den Matchmaker und dieser hält diese vor bis eine Anfrage nach einer

`Context`	Kontext der Spezifikation
`Types`	Angabe der benutzen Variablentypen
`Input`	Angabe der Eingabevariablen
`Output`	Angabe der Ausgabevariablen
`InConstraints`	Beschränkungen der Eingabevariablen
`OutConstraints`	Beschränkungen der Ausgabevariablen
`ConcDescriptions`	Ontologische Beschreibung der benutzen Wörter
`TextDescription`	Textuelle Beschreibung der Spezifikation

Abbildung 3.3.: Der Frame einer LARKS-Spezifikation mit den zugehörigen Slots nach [Sycara u. a. 2002].

bestimmten Fähigkeit eingeht. Nachdem der Matchmaker mit Hilfe seines Matchmaking-Verfahrens eine bewertete Liste von Angeboten ermittelt hat, sendet er diese an den Requester.

Mit der weiteren Spezifikation der Sprache LARKS und dem darauf basierenden Matchmaking-Verfahren verfolgt der Ansatz das Ziel, die Heterogenitäten zwischen Providern und Requestern zu überwinden.

Die Spezifikation eines Angebots oder einer Anfrage erfolgt in der Frame-basierten Sprache LARKS durch einen Frame mit den in Abbildung 3.3 angegeben Slots. Die einzelnen Slots der Spezifikation haben die folgenden Bedeutungen:

Context Dieser Slot gibt den Kontext der Beschreibung in der Domäne des Agenten an.

Types Durch diesen Slot wird angegeben, welche Datentypen in der Spezifikation Verwendung finden. Diese Angabe ist optional.

Input und Output Hier werden die Eingabe- und Ausgabevariablen der Spezifikation angegeben. Es können zusätzlich Konzepte zur Unterscheidung von Datentypen des gleichen Namens angegeben werden, die selbst im Slots *ConcDescriptions* definiert werden müssen.

InConstraints und OutConstraints In Hornklauseln[3] formulierte Beschränkungen der Eingabe- und Ausgabevariablen.

ConcDescriptions Optional können in diesem Slot Konzepte zur Beschreibung der Wörter, die in der Spezifikation genutzt werden, angeben werden. Die Zuordnung erfolgt in der Form $w * C$ wobei das Konzept C dem Wort w zugeordnet wird und dessen ontologische Beschreibung enthält.

TextDescription Dieser Slot kann optional eine Beschreibung der Bedeutung dieser Beschreibung für das Matchmaking zwischen Angeboten und Anfragen sowie eine weitere Beschreibung der Angaben in übrigen Slots enthalten.

[3]Die Definition einer Hornklausel erfolgt beispielsweise in [Schöning 2000, S. 31].

FindComputerInfo	
Context	Computer*Computer;
Types	InfoList = ListOf(model: Model*ComputerModel, brand: Brand*Brand price: Price*Money, color: Color*Colors);
Input	brands: SetOf Brand*Brand; areas: SetOf State; processor: SetOf CPU*CPU; priceLow*LowPrice: Integer; priceHigh*HighPrice: Integer;
Output	Info: InfoList;
InConstraints	
OutConstraints	sorted(Info).
ConcDescriptions	Computer = (and Product (exists has-processor CPU) (all has-memory Memory) (all is-model ComputerModel)); LowPrice = (and Price (ge 1800) (exists in-currency aset(USD))); HighPrice = (and Price (le 50000) (exists in-currency aset(USD))); ComputerModel = aset(HP-Vectra,PowerPC-G3,Thinkpad770,Satellite315); CPU = aset(Pentium,K6,PentiumII,G3,Merced) [Product, Colors, Brand, Money]

Abbildung 3.4.: Ein Beispiel für eine LARKS-Spezifikation für die Suche nach Computern mit per Konzept definierten Wörtern nach [Sycara u. a. 2002].

Die Angabe der Konzepte, die zur Wortbeschreibung herangezogen werden sollen, erfolgt dabei in der Konzeptsprache *ITL* (Information Terminological Language), die in [Sycara u. a. 1998] spezifiziert wird. Dies ermöglicht durch den Aufbau einer lokalen Ontologie seitens des Anbieters bzw. des Anfragers eine genauere Angabe des gesuchten bzw. des angebotenen Services und außerdem das Inferieren über die Servicebeschreibungen durch den Matchmaker, der dadurch eine Verfeinerung seines Matchmakingsprozesses erreicht. Zu beachten ist dabei, dass die lokalen Ontologien den jeweils anderen Agenten nicht bekannt sein müssen und auch durch den Matchmaker nur beachtet werden müssen, falls mehrere Ontologien definiert wurden. In diesem Fall ist nicht sichergestellt, dass gleiche Konzeptnamen die selbe Semantik haben und der Matchmaker muss die Konzeptdefinitionen zur Durchführung des Matchmaking-Prozesses heranziehen. Er baut hierfür dynamisch eine Konzepthierarchie durch Berechnung der Subsumptionsbeziehungen zwischen den ihm zur Verfügung stehenden Konzeptdefinitionen auf. Abbildung 3.4 enthält ein Beispiel für eine Spezifikation, die Konzepte zur Beschreibung eines Angebots zur Suche nach Computern beinhaltet. Das Beispiel zeigt, wie beispielsweise das Konzept *Computer* durch die Angabe einer Konzeptdefinition im Slot `ConcDescriptions` spezifiziert wird und wie weiterhin im Slot `OutConstraints` für die Ausgabeliste *Info* die Anforderung der Sortiertheit festgelegt wird.

Der Matchmaking-Prozess des LARKS-Ansatzes gliedert sich in fünf Phasen, die jeweils strenger werdende Filter auf die Spezifikationen anwenden. Zunächst werden aber die Typen eines Matches definiert.

Exact match Dies ist der beste Typ eines Matches, der erreicht werden kann. Dieser liegt dann vor, wenn zwei Beschreibungen äquivalent sind, wobei dieses einerseits Wortgleichheit aber auch andererseits logische Äquivalenz, die durch Inferenz ermittelt wurde, bedeuten kann.

Plug-in match Dieser Match ist die nächstschlechtere Stufe, die vorliegen kann. Er ergibt sich dann, wenn ein bestimmtes Angebot die Anforderungen einer Anfrage erfüllt, aber die beiden Beschreibungen dabei nicht identisch sind und sich in Ein- und Ausgabewerten und Beschränkungen unterscheiden.

Relaxed match Durch diesen Match wird keine Aussage bezüglich eines semantischen Matches zweier Beschreibungen getroffen, sondern lediglich die Entfernung der Beschreibungen durch Rückgabe eines Entfernungswerts festgestellt. Der Match liegt vor, falls der Entfernungswert kleiner als ein definierter Schwellwert ist.

Die nachfolgend definierten Filter können unabhängig voneinander ausgeführt und kombiniert werden und besitzen ansteigenden Berechnungsaufwand.

1. Context matching
2. Profile matching
3. Similarity matching
4. Signature matching
5. Constraint matching

Der Matchmaker stellt unterschiedliche Betriebsarten zur Verfügung, die die Match-Mechanismen auf mehrere Arten kombinieren und damit die Auswahl eines Modus ermöglichen, der der jeweiligen Abwägung zwischen Berechnungsaufwand und gewünschter Qualität des Matches entgegenkommt. Dabei kann entweder nur ein Kontext-Match in Kombination mit einem syntaktischen Match oder zusätzlich ein semantischer Match durchgeführt werden.

Die folgende Aufstellung gibt einen kurzen Überblick über die verschiedenen Arten der zur Verfügung stehenden Filter.

Context matching Zur Bestimmung dieses Matches wird auf den in den Spezifikationen angegeben Kontexten die Wortdistanz der dort angegebenen Wörter als Vergleichskriterium herangezogen. Ausgehend von definierten Schwellwerten wird solange von einem Match ausgegangen, bis diese Schwellwerte überschritten werden. Als Verfahren zur Bestimmung der Wortdistanz kommt das so genannte Trigger-Pair-Model [Rosenfeld 1994] zum Einsatz.

Centroid:	Candidate:
job_profile	Person
max_age	Age
max_wage	Expected_wage
job_description	Soft_skills
	Hard_skills

Abbildung 3.5.: Vereinfachtes Beispiel für Profile im Grappa-Ansatz nach [Veit u. a. 2002]

Profile matching Die Berechnung dieses Matches basiert auf einem Verfahren des Information Retrieval, dem so genannten Term Frequency-Inverse Document Frequency Weighting (TF-IDF) (siehe [Salton 1989]). Mit Hilfe dieses Verfahrens wird ausgehend von der gewichteten Worthäufigkeit in Anfragen und Beschreibungen und einem bestimmten Schwellwert das Vorliegen eines Matches geprüft.

Similarity matching Dieser Schritt prüft das Vorliegen eines Matches anhand der Distanz der Konzeptbeschreibungen in ihrer Definition.

Signature matching / Constraint matching In diesem Schritt wird unter Zuhilfenahme der logischen Beschreibungen der einzelnen Konzepte und Constraints durch den Aufbau einer Konzepthierarchie logische Inferenz durchgeführt. Es wird dabei überprüft, ob bestimmte Anfragen durch Angebote subsumiert werden.

3.2.3. Grappa

Grappa ist ein weiterer Ansatz zum Matchmaking auf elektronischen Marktplätzen [Veit u. a. 2002]. Der Ansatz schlägt eine XML-basierte Repräsentation von Angebots- und Anfrageprofilen vor. Angebote werden in diesem Ansatz als *Candidate Profiles*, Anfragen als *Centroid Profiles* bezeichnet. Eine Matchmaking-Funktion ist eine Funktion, die eine Menge von Candidate Profiles und ein Centroid Profile als Eingabe erwartet und eine bewertete Liste von k besten Angeboten passend zur Anfrage ausgibt. Daraus ergibt sich, dass das Matchmaking als *k-nearest-neighbors-Problem* aus dem Information Retrieval verstanden werden kann (siehe [Salton 1989]). Der Grappa-Ansatz unterscheidet sich insoweit von anderen Ansätzen, dass er die Gleichartigkeit zweier Profile nicht mit verschiedenen Verfahren nacheinander in mehreren Schritten prüft, sondern die Attribute der Profile aufteilt und parallel deren Gleichartigkeit überprüft. Abschließend werden dann die Ähnlichkeitsmaße der einzelnen Attribute zu einem globalen Maß der Ähnlichkeit kombiniert. Für verschiedene Arten der Attribute sind zu diesem Zweck verschiedene auf Grunddatentypen operierende Distanzfunktionen vorgesehen. Die separaten Einzelsummen werden dann durch ein geeignetes Verfahren zur Bildung einer Gesamtdistanz der Profile kombiniert. Ein Beispiel für ein Candidate und ein

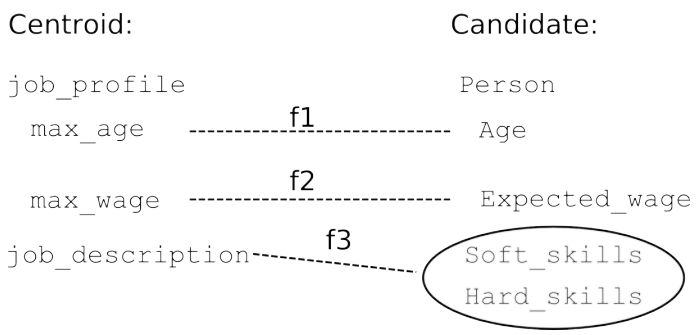


Abbildung 3.6.: Vereinfachtes Beispiel für Zusammenfassung von Attributen und der Zuweisung von Distanzfunktionen im Grappa-Ansatz nach [Veit u. a. 2002]

Centroid Profile ist in Abbildung 3.5 dargestellt. Abbildung 3.6 zeigt die Aufteilung der einzelnen Attribute des Profils und die Zuweisung von Distanzfunktionen. f_1 und f_2 können dabei beispielsweise die Distanz der Attribute auf natürlichen Zahlen berechnen und f_3 ist eine komplexere Funktion zur Berechnung der Distanz anhand des Inhalts der Texte.

Die Konfiguration des Systems erfolgt mit Hilfe von Document Type Definitions (DTD). Diese legen die Struktur der Profile fest, denen im Verlauf des Matchmaking-Prozesses die als XML-Dokumente definierten Angebote und Anfragen entsprechen müssen. In einem weiteren XML-Dokument wird die Konfiguration des Matchmakingsystems vorgenommen. Dieses umfasst unter anderem die Zusammenfassung von Attributen und die Zuweisung der Distanzfunktionen. Zusätzlich muss zu jedem zusammengefassten Attributpaar eine Bedingung angegeben werden, die eine Gewichtung der Attributpaare untereinander ermöglicht. Sollte sich beispielsweise für ein Attribut der Wert 0 für die Distanzfunktion in Verbindung mit der Bedingung *strong* ergeben, so ist die Gesamtdistanz ebenfalls als 0 anzunehmen.

3.2.4. Extended Matchmaking Component

In [Ströbel und Stolze 2001] wird eine so genannte *Extended Matchmaking Component* (EMC) definiert. Diese führt das Matchmaking Problem auf eine Darstellung als *Constraint Satisfaction Problem* (CSP) zurück. Der Ansatz geht davon aus, dass sich Transaktionen von Objekten auf einem elektronischen Marktplatz als Menge von *Properties* beschreiben lassen. In einem Beispiel zum Computerkauf ist dies beispielsweise ein Objekt *Notebook*, das unter anderem die Variable $price = \$2000$ besitzt. Dieses Angebot auch Offer-to-sell (O2S) genannt, deckt sich mit einer Anfrage (Offer-to-buy (O2B)) falls keine Konflikte der Constraints auf den Properties vorliegen. Für ein O2B mit der Variable $price \leq \$2000$ ist dies der Fall. Liegt dagegen das Constraint $price > \$2000$ vor, ergibt sich ein Konflikt. Die Aufgabe einer zentralen Matchmaking-Komponente ist durch das Lösen des CSP festzustellen, zwischen welchen Angeboten und welchen Anfragen ein Match vorliegt.

Die Constraints können zum Einen unär sein (z. B. *price* $\leq$ \$2000) oder binär (*payment* $<$ *delivery*), um anzugeben, dass die Bezahlung vor der Lieferung eines Objektes stattfinden soll. Die Constraints können außerdem durch die Operatoren AND oder OR kombiniert werden. Constraints können optional als *verhandelbar* gekennzeichnet werden. Denkbar ist hierbei, dass sich auch komplexere Constraint-Netze aufbauen lassen, die beispielsweise das Auftreten in mehreren Constraints zulassen.

Der Ablauf eines Matchmaking-Prozesses für ein Paar aus O2S und O2B ist folgender:

- Zunächst werden die Domänen der Variablen in O2S und O2B bestimmt.
- Die Variablen, die in beiden Wertedomänen enthalten sind bilden dann die *Compatible Property Value Domain* (CPVD).
- Falls die CPVD leer ist, schlägt der Match fehl.
- Wenn die CPVD mindestens einen Wert enthält, liegt für die Variable ein erfolgreicher Match vor.
- Gilt dies für alle Properties liegt auch zwischen dem O2S und dem O2B ein Match vor.

Zu beachten ist hierbei noch, dass sich *Agreement Value Ranges* ergeben können. Diese liegen dann vor, wenn zwar kein Konflikt der Constraints vorliegt, aber zunächst nur ein Wertebereich vorliegt aus dem heraus ein genauer Wert noch verhandelt werden muss. Ein Beispiel hierfür ist ein O2B mit *price* $<$ \$2000 und ein O2S mit *price* $>$ \$1800. Der sich ergebende Wertebereich ist dann \$1800 $<$ *price* $<$ \$2000.

Aufbauend auf dieser Basis erweitert die EMC den Ablauf insoweit, dass auch die Angaben zur Verhandelbarkeit der Constraints beachtet werden und die Vereinbarungen (Agreements) werden in verschiedene Klassen eingeteilt:

Matching agreements (MAS) In diese Klasse fallen die Matches, bei denen durch die Constraints auf den Variablen kein Wertebereich offenbleibt.

Distribution agreements (DAS) Für diese Constraints muss noch über einen endgültigen Wert einer Variablen verhandelt werden. Dieser Fall tritt dann ein, wenn für die Variablen noch ein Wertebereich offenbleibt.

Negotiable agreements (NAS) Diese Klasse eines Matches liegt dann vor, wenn Constraints Konflikte generiert haben. Durch die EMC wird dann überprüft, ob diese verhandelt werden können. Wenn beide Seiten ein Constraint als verhandelbar ausweisen, wird dieses als Double-sided Negotiable Constraint (DSNC) markiert, andernfalls wird es als Single-sided Negotiable (SSNC) Constraint markiert.

Unsuccessful matches Diese Klasse umfasst die Matches, die Konflikte hervorgerufen haben und bei denen keine Verhandlung möglich ist.

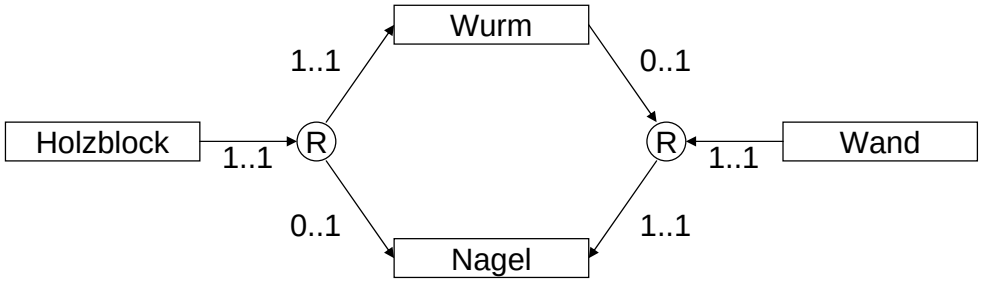


Abbildung 3.7.: Der Netzgraph einer partonomischen Relation nach [Werres 2002]

3.2.5. Effizientes und flexibles Pattern-Matching für komplexe Objektstrukturen

In [Werres 2002] wird ein flexibles Matchingverfahren, das auf Pattern-Matching basiert, definiert. Es ist konzipiert als Komponente für das Konfigurationssystem EngCon (Informationen siehe [Ranze u. a. 2002]). Das System besteht aus fünf Komponenten, die im Folgenden erläutert werden:

WorkingInstanceSet Diese Komponente enthält Wissen über Instanzen des Konfigurationssystems. Dieses liegt beim System EngCon als XML-Repräsentation vor.

ConceptualKnowledge Durch diese Komponente wird Wissen über Konzepte der Anwendungsdomäne repräsentiert. Dieses schließt Wissen über partonomische und taxonomische Relationen zwischen Konzepten ein. Die Komponente ist als Erweiterung des WorkingInstanceSets zu verstehen.

ConditionSet Diese Komponente repräsentiert Mengen von Bedingungen von Suchanfragen im Instanzraum des WorkingInstanceSet. Bedingungen können sowohl einzelne Instanzen betreffen, als auch Merkmale von Instanzen und deren Beziehungen untereinander abfragen. Eine Menge von Bedingungen wird als Muster bezeichnet.

ConflictSet Hierdurch wird die Suchergebnismenge, die durch das nachfolgend erläuterte *Pattern-Matcher-Modul* ermittelt wird, repräsentiert.

Pattern-Matcher-Modul Die Aufgabe dieses Moduls ist es, die oben erwähnten vier Module im Laufe des Konfigurationsprozesses konsistent zu halten. Dazu werden sich ergebende Änderungen zwischen den dynamischen Komponenten *WorkingInstanceSet* und *ConflictSet* sowie den statischen Komponenten *ConceptualKnowledge* und *ConditionSet* propagiert.

Das *ConceptualKnowledge* ergibt sich aus den modellierten Konzepten der Anwendungsdomäne. Ein Konzept ist als Abstraktion eines Objektes mit Attributen und Relationen zu verstehen. In diesem Zusammenhang werden die Attribute der Konzepte *Parameter* genannt, die durch *Deskriptoren* festgelegte Wertebereiche besitzen. Referenzen bzw. Relationen zwischen

Konzepten werden durch *Assoziationen* beschrieben. Eine speziellere Form der Assoziation ist die Aggregation, die eine Zugehörigkeitsbeziehung zwischen zwei Konzepten modelliert. Eine weitere Form der Relation ist die *Komposition*. Durch sie kann definiert werden, dass zwei Konzepte nicht einzeln existieren können und von außerhalb dieser Komposition nicht mehr referenziert werden können. Eine komplexere Klasse von Assoziationen bilden die so genannten *partonomischen Relationen*. Diese bestehen aus einer Relations-Definition und zugehörigen Relationen. Die Relationen selbst bestehen jeweils aus Verweisen auf Integriere- bzw. Zerlege-Konzepte. Den Verweisen auf die Relationen ist jeweils eine Kardinalität zugeordnet. Partonomische Relationen lassen sich wie in Abbildung 3.7 dargestellt durch Netzgraphen visualisieren. Bei Pfeilen, die auf die Relation *R* zeigen, handelt es sich um das Integrations-Element, das die Kombination aus Integriere-Konzept und einer zugehörigen Kardinalität ist. Analog dazu handelt es sich bei den von der Relation wegzeigenden Pfeilen um Dekompositions-Elemente, die die Kombination aus Zerlege-Konzept und der dafür geltenden Kardinalität sind. Zur genauen Definition der Darstellung als Netzgraph siehe [Werres 2002, S. 67]. Eine weitere Form der Relation ist die taxonomische Relation. Diese bildet die Hierarchie aller definierten Konzepte des Konzeptwissens. Unterkonzepte erben jeweils Parameter und Relationen ihrer Oberkonzepte.

Ausgehend von diesen Möglichkeiten der Repräsentation sind nun Abfragen nach bestimmten Pattern möglich. Interessant sind in diesem Zusammenhang Pattern, die Bedingungen an bestimmte Konzepte definieren. Dies ist beispielsweise das *patternConcept* oder der *patternParameter*. Diese Pattern definieren Anforderungen an eine Instanz, die z. B. Instanz eines bestimmten Konzepts sein muss oder bestimmte Parameter enthalten soll.

Das eigentliche Pattern-Matching findet im Pattern-Matcher-Modul statt. In [Werres 2002] wird dafür der so genannte *EngCon-V0-Netmatcher* konzipiert. Dieser besteht zunächst aus einem Algorithmus, der ein Testnetz generiert, auf dem später das Pattern-Matching operiert. Die Erstellung des Testnetzes gliedert sich in vier Phasen:

1. Zunächst wird das Testnetz initialisiert. Dazu wird über eine Liste von Variablen-Pattern-Paaren iteriert und entsprechende Knoten im Netz erzeugt.
2. Im diesem Schritt für Parameter und Relationsdefinitionen[4] entsprechende Knoten im Testnetz erzeugt.
3. Nun werden auf für Variablen-Pattern-Paare, die Beziehungen zu anderen Instanzen betreffen[5], Knoten im Netz erzeugt, denen eine Verbindung zur anderen Instanz mitgegeben wird.
4. Abschließend wird das Netz finalisiert. Dazu noch nicht eingebundene Knoten gefunden und entsprechend im Netz verbunden.

Die Aufgabe des Testnetzes ist es nun im Verlauf des Pattern-Matchings von einer partiellen Belegung der Bindungen zu einer vollständigen Belegung zu kommen. Dabei wird versucht

[4] Die so genannten Intra-Bedingungen.

[5] Bedingungen, die Beziehungen zu anderen Instanzen haben, werden Inter-Bedingugnen genannt.

die Knoten des Testnetzes zu vermischen und Bindungskombinationen durch das Netz propagiert. Eine Bindungskombination ist dabei partielle Belegung einer Bedingung. Dies gilt sowohl für Knoten, die Intra-Bedingungen repräsentieren, als auch für Inter-Bedingungen.

3.3. Bewertung der vorgestellten Ansätze zum Matchmaking

In diesem Abschnitt werden die vorgestellten Match-Making-Ansätze bewertet und eine Auswahl eines Verfahrens zur Integration in den eigenen Ansatz getroffen.

3.3.1. Bewertung der vorgestellen Ansätze

Matchmaking Based on Semantic Web Technology

Dieser Ansatz definiert das Matchmaking zwischen Web Services durch den Einsatz der Service-Ontologie DAML-S. Durch die Formulierung von Anfragen (Requests) und Angeboten (Advertisements) als Konzepte einer OWL-Ontologie ermöglicht es den Einsatz einer Reihe von vorhandenen Reasonern, die die Inferenz über DAML- bzw. OWL-Ontologien beherrschen. Einige wurden in Abschnitt 2.2.3 vorgestellt. Der Ansatz lässt sich insoweit als flexibel in Bezug auf den im Rahmen einer konkreten Umsetzung einzusetzenden Reasoner beurteilen. Die eingesetzte Ontologiesprache DAML bzw. auch deren Nachfolger OWL ist ausdrucksmächtig genug zu bewerten, um die Anforderung an die Modellierung komplexer technischer Domänen zu ermöglichen. Die Möglichkeit zur Modellierung konkreter Domänen ist gegeben. OWL besitzt außerdem eine hohe Verbreitung in verschiedenen Anwendungsbereichen und unterliegt einer fortlaufenden Weiterentwicklung und Standardisierung. Die Entwicklung und Verarbeitung von OWL-Ontologien wird weiterhin durch eine Reihe von Software-Tools und -Frameworks unterstützt. Dieses erleichtert eine Integration in konkrete Umsetzungen eines Ansatzes zum Matchmaking. Über die Formulierung von OWL-Ontologien ist außerdem der Aufbau von Verallgemeinerungs- und Spezialisierungshierarchien möglich. Die schon erfolgte Implementierung dieses Ansatzes im Rahmen eines Agentensystems zur Zuordnung von Web Services zeigt die grundsätzliche Eignung dieses Ansatzes für Agentenumgebungen. Dieser Aspekt ist ebenfalls als positiv zu bewerten.

LARKS-Ansatz

Der LARKS-Ansatz definiert eine sehr umfangreiche Lösung zum Matchmaking-Problem, indem verschiedene Zuordnungsarten kombiniert werden. Er bietet neben einfachen textbasierten Vergleichen von Anfragen und Anboten auch komplexere Verfahren bis hin zur logischen

Inferenz und der Berechnung von Subsumtionsbeziehungen zwischen den Beschreibungen der Angebote und Anfragen. Außerdem wird ein abgestufter Prozess des Matchmakings definiert. In Bezug auf den Einsatz im Prozess der Verhandlungen in einem Agentensystem ist der Ansatz aber insgesamt als zu komplex zu bewerten. Dies widerspricht der Forderung nach einer ausreichenden Skalierbarkeit des zu entwickelnden Verfahrens, um auch in Agentenumgebungen mit einer höheren Anzahl anfragender und anbietender Agenten Matchmaking ohne größere Performanzeinbußen durchzuführen. Zusätzlich ist der Ansatz aufgrund seiner Komplexität als nicht geeignet anzusehen, entsprechend der Forderung nach einer Integration des zu entwickelnden Ansatzes in die Agentenplanung und in den Diskursagentenansatz berücksichtigt zu werden. Weiterhin erscheint der Aspekt der Bestimmung von Wortähnlichkeitsmaßen, den dieser Ansatz beinhaltet, wenn der Fokus auf technische Domänen gesetzt wird nicht zwingend notwendig. Standardisierungsbestrebungen ermöglichen hier, dass zumindest die grundlegende Begrifflichkeit von Problemen und Problemlösungen übereinstimmt.

GRAPPA-Ansatz

Der Grappa-Ansatz formuliert eine Matchmaking-Lösung, die auf einer XML-basierten Darstellung der Anfragen und Angebote basiert. Die Form der Repräsentation über Profile wird über DTD-Spezifikationen definiert. Die Ausdrucksmächtigkeit dieser Art der Spezifikation ist insoweit eingeschränkt, als sich keine Beziehungen zwischen den Profilen definieren lassen. Dies erscheint in Bezug auf die Forderung nach der Definition dieser Beziehungen als negativ. Die Möglichkeiten zur Modellierung komplexerer Domänen sind im Rahmen dieses Ansatzes als sehr beschränkt zu bewerten. Die Profildefinitionen besitzen außerdem keine formal definierte Semantik

EMC-Ansatz

Im Rahmen des Ansatzes zur Modellierung einer Extended Matchmaking Component erfolgt die Definition einer Lösung über die Formulierung von Constraint Satisfaction Problems. Positiv zu beurteilen ist bei diesem Ansatz die explizite Berücksichtigung konkreter Werte in der Formulierung der Constraints auf den Properties der Angebote und Anfragen. Dieses erfüllt die Forderung nach Berücksichtigung in der Entwicklung eines eigenen Ansatzes. Negativ fällt aber wie beim GRAPPA-Ansatz die Beurteilung hinsichtlich der Möglichkeit zur Definition von Beziehungen zwischen einzelnen Angeboten und Anfragen. Dadurch ist es fraglich, ob der Ansatz genug Ausdrucksmächtigkeit zur Modellierung komplexerer Domänen bietet, in denen diese Fähigkeit nötig ist, um beispielsweise bestimmte Abhängigkeiten abzubilden. Es fehlt zusätzlich die Definition einer formalen Semantik der Problemrepräsentationen. Außerdem lässt sich die Forderung nach einer flexiblen Matching über Spezialisierungs- und Verallgemeinerungshierarchien mit diesem Ansatz nicht umsetzen.

Pattern-Matching für komplexe Objektstrukturen

Dieser Ansatz definiert die Lösung des Matching-Problems über den Aufbau eines Netzes, das das vorhandene Wissen modelliert. Auf diesem operiert dann das eigentliche Pattern-Matching, indem es zu einem bestimmten Pattern passende Objekte findet. Positiv zu beurteilen ist bei diesem Verfahren die exakte Definition von Objektrelationen insbesondere von taxonomischen Relationen und die Berücksichtigung von Wertebereichen. Da der Ansatz aber ursprünglich als Teil eines Konfigurationssystems konzipiert wurde, an das generell andere Anforderungen gestellt werden, als an Systeme zur Zuordnung von Problemen zu Problemlösungen in Agentensystemen wird Rahmen dieser Arbeit von einer Verwendung dieses Ansatz Abstand genommen. Einem Einsatz dieses Ansatzes stehen somit die Forderungen nach einer Integration des Ansatzes in die Agentenplanung und in den Diskursagentenansatz entgegen.

3.3.2. Auswahl eines Ansatzes

Aufgrund der oben vorgestellten Vorteile des Ansatzes zum Matchmaking, das auf Semantic Web Technology beruht, wird dieser als Ausgangspunkt zur Entwicklung eines eigenen Ansatzes gewählt. Er stellt einen guten Kompromiss zwischen Ausdrucksmächtigkeit der Repräsentation und möglicher Flexibilität der Inferenz über diese Repräsentation einerseits, aber auch steigender Komplexität des Ansatzes andererseits. Eine nochmalige kurze Vorstellung der Grundidee des Ansatzes erfolgt im Rahmen der Darstellung der grundlegenden Konzeption des eigenen Ansatzes zu Beginn des folgenden Kapitels.

Kapitel 4.

Eigener Ansatz

In diesem Abschnitt soll der eigene Ansatz zur Umsetzung eines flexiblen Agentenmatchmakings vorgestellt werden. Dazu wird zunächst die grundlegende Konzeption vorgestellt und daran anschließend das Konzept formalisiert. Schließlich wird die Umsetzung in Form mehrerer Algorithmen spezifiziert. Ein abschließendes Beispiel erläutert einen Matchmakingdurchlauf für ein einfaches Beispiel.

4.1. Grundlegende Konzeption

In diesem Abschnitt wird zunächst dargestellt, auf welchem der im vorherigen Kapitel vorgestellten Ansätze mit dem eigenen Konzept aufgesetzt wird. Daran anschließend wird die Einbettung des Ansatzes in den Aufbau einer bestehenden Agentenarchitektur beschrieben. Bei dieser handelt es sich um die Architektur des *Diskursagenten*. Im dritten Teil des Abschnitts wird die Einbettung in den Aufbau eines Multiagentensystems (MAS) beschrieben.

4.1.1. Reflexion bestehender Ansätze

Im Kapitel 3 wurden verschiedene Ansätze zur Realisierung eines Match-Makings zwischen Problemen und Problemlösung vorgestellt. Aufbauend auf der abschließenden Bewertung dieser Ansätze wird sich der zu entwickelnde Ansatz an die Vorschläge des Ansatzes von Li und Horrocks in [Li und Horrocks 2004] anlehnen. Dieser erfüllt die Forderung nach einer Flexibilisierung des Match-Making, in dem er die Zuweisung von Problemen zu Problemlösern über die Eigenschaften dieser Einheiten ermöglicht. Weiterhin erscheint er im Gegensatz zu den weiteren vorgestellten Ansätzen als guter Kompromiss zwischen den Möglichkeiten, die das Verfahren bietet einerseits und der Skalierbarkeit in größeren Agentenumgebungen andererseits. Durch den Einsatz der Beschreibungslogik DAML+Oil ist zudem eine hohe Ausdrucksmächtigkeit zur Repräsentation komplexer Problemdomänen gegeben und

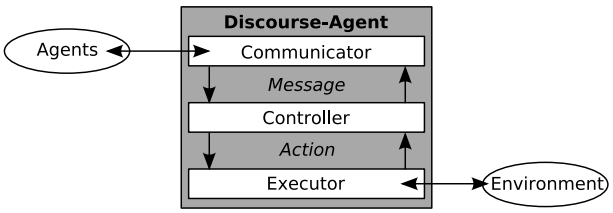


Abbildung 4.1.: Interaktionsebenen des Diskursagentenansatzes

es sind außerdem effiziente Inferenzverfahren vorhanden, mit denen Berechnungen auf diesen Problembeschreibungen durchgeführt werden können. Zur Umsetzung der Forderung nach einer expliziten Spezifikation der Agentencapabilities erfolgt weiterhin eine Repräsentation der Advertisements/Capabilities in der erwähnten Ontologiesprache DAML+Oil. In dieser Ontologiesprache werden dann zusätzlich die angefragten Requests/Features erzeugt, um mit Hilfe bestehender Inferenzverfahren Beziehungen zwischen Capabilities und Features zu berechnen. Aus diesen Berechnungen werden dann Rückschlüsse auf einen möglichen Match zwischen Capabilities und Features gezogen. An Li und Horrocks Ansatz wird außerdem die Definition der Features und Capabilities über ein allgemeines Superkonzept *Profil* angelehnt. Dies ermöglicht wie beispielsweise in der OWL-Ontologie DAML-S bzw. OWL-S für Webservices die vorherige Definition allgemeiner Eigenschaften der Problemdomäne. Bei DAML-S/OWL-S sind dies z. B. die Properties *inputs/outputs*, von denen speziellere Properties abgeleitet werden können.

Modifiziert werden muss für den zu definieren Ansatz der Ort der Einbettung des Capability-Matchings. Ist dieser nach [Li und Horrocks 2004] in eine zentrale Komponente, den sog. *Host* eingebettet, wird durch diesen Ansatz eine Einbettung in die jeweiligen Agenten selbst vorgenommen. Die nähere Begründung dieses Vorgehens erfolgt in Abschnitt 4.1.3. Außerdem wird die Ontologiesprache OWL-DL als Nachfolger von DAML+Oil zur Wissensrepräsentation eingesetzt werden. Im nächsten Abschnitt wird die Architektur des Ansatzes auf Agentenebene, d. h. seine Einbettung in den schon bestehenden Ansatz des Diskursagenten beschrieben werden.

4.1.2. Architektur auf Agentenebene

Der in Abschnitt 2.1.1 vorgestellte BDI-Ansatz zur Definition eines Ziel-basierten Agenten besitzt die Einschränkungen, dass einerseits die Modellierung des Zustands eines Multiagentensystems nicht möglich ist und andererseits die $\mathcal{VSK}$-Logik die Repräsentation und Abwägung von Überzeugungen, Zielen und Absichten nicht erlaubt. Zur Überwindung dieser Einschränkungen wurde durch [Timm 2004] die Architektur des *Diskursagenten* eingeführt. Diese Arbeit versucht die beiden genannten Ansätze zu vereinen und darauf aufbauend eine Agentenarchitektur zu entwickeln, die intern eine *konfliktbasierte Agentensteuerung* und ex-

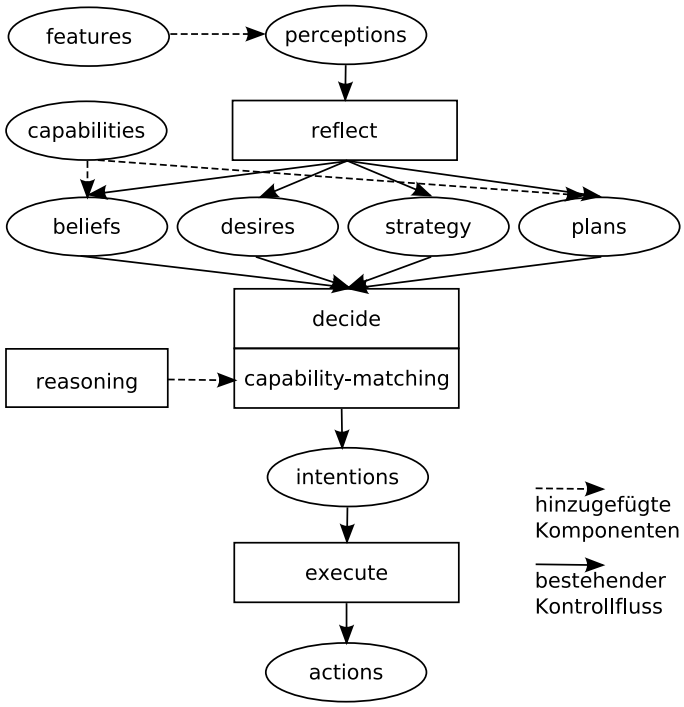


Abbildung 4.2.: Einbettung des Ansatzes in die Diskursagentenarchitektur

tern die sog. offene adaptive Kommunikation verwendet. Der Ansatz gliedert den Agenten grundsätzlich in drei Schichten, die jeweils unterschiedliche Aufgaben erfüllen:

Communicator Führt die Kommunikation mit dem Agentensystem sowie anderen Agenten aus. Dazu müssen die Nachrichten aus bzw. in ein definiertes Austauschformat umwandeln.

Controller Stellt die Steuerung und Kontrolle des Agenten zu Verfügung. Hier findet die Repräsentation von Wissen und das Fällen von Entscheidungen statt.

Executor Führt Aktionen des Agenten aus, die aufgrund von Entscheidungen in der Kontrollschicht zur Ausführung kommen sollen.

Dieser Aufbau ist in Abbildung 4.1 verdeutlicht. Die Kommunikationsschicht (Communicator) interagiert mit anderen Agenten des MAS. Eingaben und Ausgaben werden durch die Kontrollschicht (Controller) verarbeitet bzw. ausgelöst. Das Gleiche gilt für die Aktionen der Ausführungsschicht: Sie werden einerseits durch die Kontrollschicht ausgelöst und bewirken gegebenenfalls Veränderungen in der Umgebung, in die der Diskursagent eingebettet ist. Andererseits werden Ergebnisse der Aktionsausführung zurück an die Kontrollschicht geleitet.

Aufgrund der ausführlichen Formalisierung des Ansatzes und der Verbindung der Vorteile des BDI-Ansatzes einerseits und der VSK-Logik andererseits, wird auf diesem Ansatz im

Weiteren aufgesetzt. Wichtige Aspekte des Diskursagentenansatzes sind die Definition des *reflektiven*, *deliberativen* und *exekutiven* Verhaltens des Diskursagenten. Das reflektive Verhalten ist in der Kommunikationsschicht des Agenten angesiedelt. Es ist für die Verarbeitung von Wahrnehmungen zuständig, die durch die Kommunikation mit dem Agentensystem entstehen. Hierzu ist die Abbildung *reflect* nach [Timm 2004] durch vier Funktionen definiert, die sich um die Anpassung der aktuellen Überzeugungen (*Beliefs*), der Ziele (*Desires*) und der Pläne (*Plans*) des Agenten kümmern. Außerdem erfolgt innerhalb des reflektiven Verhaltens eine Gewichtungsanpassung der Ziele aufgrund der aktuellen Wahrnehmungen und einer definierten globalen Strategie. Aus dieser Anpassung leitet sich weiterhin eine mittelfristige Strategie (Strategy) ab, die langfristig eine Erreichung der Ziele ermöglichen soll. Die formale Definition der Abbildung *reflect* findet sich in [Timm 2004, S. 81]. Innerhalb des deliberativen Verhaltens erfolgt dann die Umsetzung der gewählten Strategie durch die Wahl bzw. Generierung geeigneter Optionen, die zur Erfüllung dieser Strategie geeignet erscheinen. Dies wird in [Timm 2004, S. 81] durch die Definition einer Abbildung *decide* formalisiert. Im exekutiven Verhalten des Agenten erfolgt schließlich die Umsetzung der generierten Optionen durch die Ausführung der mit ihnen verbundenen Aktionspläne. Die Auswahl dieser Optionen erfolgt wiederum aufgrund der im reflektiven Verhalten erstellten Bewertungen. Die formale Definition des exekutiven Verhaltens erfolgt durch [Timm 2004, S. 83].

Die Einführung des Capability-Matchings hat auf jede Schicht des Diskursagentenansatzes Einfluss, da die Beliefs des Agenten in den Entscheidungszyklen aller drei Kontrollschichten Beachtung finden und die Capabilities des Agenten durch den vorgeschlagenen Ansatz als Teil der Beliefs modelliert werden. D. h. der Agent besitzt die Überzeugung, dass er durch seine Fähigkeiten zur Lösung bestimmter Probleme beitragen kann. Nimmt er nun die Nachfrage nach einer bestimmten Problemlösung wahr, wird diese durch Verarbeitung im Rahmen der Reflexionszyklus Teil seiner Überzeugungen. Der Agent ist sich sozusagen bewusst, dass innerhalb des Agentensystems, dessen Teil er ist, die Nachfrage nach einer Problemlösung besteht. Im Rahmen seines Deliberationszyklus kommt dann das einzuführende Capability-Matching zum Einsatz. Durch das Capability-Matching wird in diesem Schritt versucht einen Match zwischen dem angefragten Problem und den vorhandenen Capabilities nachzuweisen. Auf die Ausführungsschicht des Diskursagenten hat der Ansatz insoweit Auswirkungen, dass die zu modellieren Capabilities auch als mögliche Aktionen Teil der Pläne des Diskursagenten sind. Diese Pläne kommen durch die Ausführungsschicht, wiederum unter Einbeziehung der aktuellen Beliefs, des Agenten zum Einsatz.

Da sowohl Features als auch Capabilities innerhalb des Diskursagentenansatzes als Prädikate der dort definierten Modallogik repräsentiert werden, ist eine Übersetzungskomponente zwischen beiden nötig. Mit der durch diese Übersetzungkomponente ausgegebene ontologische Repräsentation wird dann mit Hilfe eines Reasoners zunächst die Subsumptionshierarchie zwischen den eingefügten Konzepten erstellt und darauf anschließend Aussagen über die Beziehungen zwischen den angefragten Features und Capabilitties ermittelt. Hieraus ergibt sich dann anhand der noch aufzustellenden Definition die Art des Matches. Diese Grundlegende Architektur ist in Abbildung 4.2 dargestellt. Nach dem Erhalt einer Anfrage für ein be-

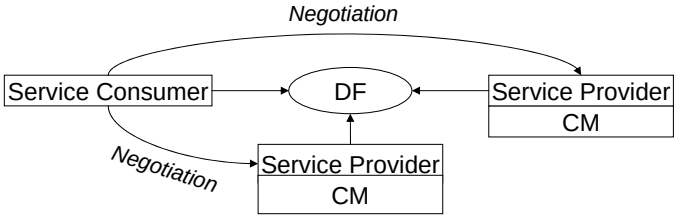


Abbildung 4.3.: Integration des Ansatzes in das Multiagentensystem

stimmtes Features, wird dieses Teil der Beliefs des Agenten. Die Capabilities sind sowohl Teil der Beliefs als auch der möglichen Pläne des Agenten. Die Entscheidung, ob eine Problemlösung möglich ist, fällt im Deliberationsschritt. Hierzu müssen die vorhandenen Prädikate zunächst transformiert werden. In der Capability-Matching-Komponente, die eine Anbindung an einen Reasoner besitzt, wird dann die Art des Matches zwischen Feature und Capabilities bestimmt. Diese Entscheidung hat schließlich Einfluss auf das exekutive Verhalten.

4.1.3. Architektur auf der Ebene des Multiagentensystems (MAS)

Anders als die in Abschnitt 3 untersuchten Ansätze wird das neu zu definierende Capability-Matching nicht in eine zentrale Instanz[1] integriert, sondern als Teil der Diskursagenten modelliert. Die Entscheidung über einen Match zwischen einem bestimmten Problem und dessen Lösung wird also nicht durch eine zentrale Anfrage beim Verzeichnisdienst des Agentensystem getroffen. Dies begründet sich einerseits in der von Agenten im Allgemeinen geforderten Autonomie[2] und ermöglicht andererseits Ergebnisse des Matchmakings direkt in die Planung des Agenten einfließen zu lassen. Dieser Ansatz ist in Abbildung 4.3 dargestellt. Jeder problemlösende Agent trifft autonom im Rahmen seines Entscheidungszyklus und mit Hilfe des Capability-Matchings (CM) die Entscheidung, ob er zur Lösung eines Problems beitragen kann und die Anfrage eines Agenten, der nach einer bestimmten Problemlösung sucht, positiv beantworten kann. Unabhängig davon wird aber eine Anmeldung der Agenten, die eine bestimmte Capability zur Verfügung stellen wollen, als allgemeine Problemlöser beim DF durchgeführt, um deren Auffinden zu ermöglichen. Der Ablauf der Suche eines Agenten nach einer Problemlösung gliedert sich dann in folgende Schritte:

1. Zunächst fragt der Agent, der nach einer bestimmten Problemlösung sucht, beim zentralen Verzeichnisdienst des Agentensystems nach Agenten, die dort ihre Dienste anbieten (Pfeil des Service Requesters zum DF). Er erhält dann im allgemeinen Fall eine Liste mit mehreren Agenten, die sich dort registriert haben (Pfeile der Service Provider zum DF).

[1] In der FIPA-Architektur wäre dies der DF-Service.
[2] Zum Begriff Autonomie sei auf Abschnitt 2.1 verwiesen.

2. Die eigentliche Feststellung, ob einer dieser Agenten die Lösung eines Problems ermöglicht, findet dann in einer separaten Verhandlung (Direkte Pfeile vom Service Requester zu den Service Providern) statt. In dieser fragt der Service Requester bei den Service Providern mit seinem Problem an und diese ermitteln, ob sie aufgrund ihrer Capabilities zu dessen Lösung fähig sind. Ein Agent erhält schließlich den Zuschlag und führt seine Problemlösungsstrategie durch.

Ein Beispiel für einen solchen Ablauf in der Domäne der Fertigungslogistik ist folgendes Szenario:

- Auf der einen Seite befindet sich ein Agent, der einen Auftrag repräsentiert und beispielsweise nach der Fertigung eines Loches mit einem bestimmten Durchmesser verlangt. Dieses ist das Feature, das im Rahmen der stattfinden Verhandlung nachgefragt wird.
- Zur Lösung dieses Fertigungsproblems stehen dem Auftrag mehrere Ressourcenagenten gegenüber, die jeweils Maschinen repräsentieren, die bestimmte Fertigungsverfahren zur Verfügung stellen. Einer dieser Agenten besitzt als Capability (Fertigungsverfahren) das benötigte Bohren. Die anderen Agenten besitzen andere Capabilities, die die Fertigung des Loches nicht ermöglichen. Alle haben sich aber wie gefordert beim zentralen Verzeichnisdienst des Agentensystems registriert, um gefunden werden zu können.

Der Ablauf ist wie oben dargestellt, dass zunächst der Auftragsagent beim Verzeichnisdienst nach einer Liste der Ressourcenagenten nachfragt und diese dann jeweils in einer separaten Verhandlung mit seiner Fertigungsanfrage kontaktiert. Die Verhandlung des Auftragsagenten mit den Ressourcenagenten kommt dann zu dem Ergebnis, dass nur der erwähnte Agent mit der Capability Bohren zur Fertigung des Loches in der Lage ist, woraufhin dieser den Zuschlag zur Auftragsausführung erhält.

Der in diesem Abschnitt kurz vorgestellte Ansatz zur Definition eines Capability-Matching und dessen Integration in einen bestehenden Agentenarchitekturansatz wird im folgenden Abschnitt näher erläutert und formalisiert.

4.2. Formale Darstellung des Ansatzes

Zunächst erfolgt in diesem Abschnitt eine Formalisierung der einzelnen Aspekte des zu entwickelnden Matchmakings. Daran schließt sich die Darstellung der Integration in den Diskursagentenansatz und die Schnittstelle zu dessen Planung an. Abschließend werden die zur Umsetzung des Konzepts entwickelten Algorithmen spezifiziert und erläutert.

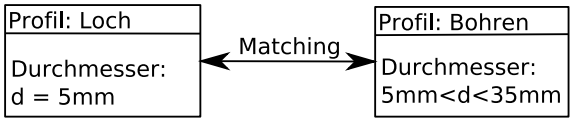


Abbildung 4.4.: Beispiel für das Matching eines Profils

4.2.1. Grundlagen des Matchmakings

Da in dieser Arbeit ein auf Beschreibungslogik (nähere Erläuterung in Abschnitt 2.2.1) basierender Ansatz gewählt wird, erfolgt die Definition der für das Matchmaking verwendeten Repräsentationseinheiten jeweils als Menge von beschreibungslogischen Konzeptdefinitionen. Die Definition des Matchmakings lehnt sich an [Li und Horrocks 2004] an. Zunächst die Definition eines *Profils* als abstraktes Konzept zur Definition von Features und Capabilities über die Vereinigung dieses Konzeptes mit bestimmten Beschränkungen von Wertezuweisungen. Diese Art der Repräsentation wurde gewählt, um im Weiteren die Möglichkeit zu schaffen, den Match zwischen benötigten Features und Agenten-Capabilities über die Bestimmung von Subsumptionsbeziehungen zwischen beiden festzustellen. Das Konzept des Profils ist an das *ServiceProfile* der in Abschnitt 3.2.1 vorgestellten Web-Service-Ontologie OWL-S angelehnt. Über dieses Konzept lassen sich beispielsweise bestimmte grundlegende Arten von Eigenschaften von Features und Capabilities vordefinieren und dann im Prozess des Matchmakings verwenden bzw. verfeinern. Ein Beispiel für dieses Konzept ist in Abbildung 4.4 dargestellt.

Definition 1 (Profil)
Für ein beschreibungslogisches Konzept p gelte $p \sqsubseteq \top$. Dieses Konzept p heiße Profil.

Hierauf aufbauend sind die Mengen von Features und Capabilities in einem Multiagentensystem wie folgt bestimmt. Für alle Elemente dieser Mengen soll gelten, dass sie Unterkonzepte des zuvor definierten Konzepts p sind.

Definition 2 (Konzeptmengen)
$\mathcal{C}$ und $\mathcal{F}$ seien die Mengen der möglichen Capabilities bzw. Features in einem Multiagentensystem, wenn gilt

- $\forall c \in \mathcal{C} : c \sqsubseteq p$ *bzw.*
- $\forall f \in \mathcal{F} : f \sqsubseteq p$.

Für die Definition einer solchen Capability c, der im Beispiel der Name *Bohren* gegeben wurde, soll also beispielsweise folgende Definition gelten, die festlegt, dass für die Property *hatDurchmesser* Werte zwischen 5 und 35 zulässig sind.

$$Bohren \equiv p \sqcap \geqslant_5 hatDurchmesser \sqcap \leqslant_{35} hatDurchmesser$$

Für ein mögliches Feature *Loch* ist dann folgende Definition denkbar.

$$Loch \equiv p \sqcap =_5 hatDurchmesser$$

Hieraus ergibt sich, dass das Konzept *Loch* offensichtlich unter das Konzept *Bohren* subsumiert werden kann.

Jeder Agent in einem Agentensystem soll nun eine bestimmte Menge von Capabilities besitzen, die Elemente der Menge $\mathcal{C}$ sind. Eine Capabilitiy-Menge eines Agenten ergibt sich demnach durch folgende Definition:

Definition 3 (Capabilities)
Wenn $\mathcal{C}$ die Menge der möglichen Capabilities in einem Multiagentensystem ist, dann sei

$$\mathcal{C}^A = \{\mathcal{C}_1^A, \ldots, \mathcal{C}_n^A\}$$

die Menge der Capabilities *eines Diskursagenten mit $\mathcal{C}_i^A \in \mathcal{C}$.*

Analog gilt für Features die nachfolgende Definition. Agenten, die zu lösende Probleme repräsentieren, besitzen jeweils eine Menge von Features, die zur Lösung des Problems umgesetzt werden müssen.

Definition 4 (Features)
Wenn $\mathcal{F}$ die Menge der möglichen Features *in einem Multiagentensystem ist, dann sei $\mathcal{F}_i^A$ ein mögliches* Feature *mit $\mathcal{F}_i^A \in \mathcal{F}$. Durch*

$$\mathcal{F}^A = \{\mathcal{F}_1^A, \ldots, \mathcal{F}_n^A\}$$

sei die Feature-Menge $\mathcal{F}^A$ eines Agenten gegeben, wenn gilt $\mathcal{F}_i^A \in \mathcal{F}$.

Aus diesen Mengen der Feature- und Capability-Konzepte $\mathcal{F}^A$ und $\mathcal{C}^A$ lässt sich nun eine ontologische Beschreibung zusammenstellen, die ein konkretes Matchmaking-Problem repräsentiert und über die inferiert werden kann. Diese ontologische Beschreibung eines Matchmaking-Problems ist formal wie folgt definiert:

Definition 5 (Ontologische Beschreibung eines Matchmaking-Problems)
Die Menge

$$M = \{\alpha_1, \ldots, \alpha_n\}$$

sei eine Ontologische Beschreibung eines Match-Making-Problems *wenn gilt* $\forall \alpha_i \in \mathcal{C}^A \cup \mathcal{F}^A$.

4.2.2. Definition des Matchmakings

Das Matchmaking erfolgt angelehnt an [Li und Horrocks 2004] durch die Bestimmung einer Menge $matches(f)$, die zu einem angefragten Feature f passende Capabilities c enthält. Formal ist diese Menge definiert als:

Definition 6 (Capability-Match-Menge nach [Li und Horrocks 2004])
Die Menge $matches(f)$ *der zu einem Feature* f *passenden Capabilities* $c_1, \ldots, c_n$ *sei gegeben durch*

$$matches(f) = \{c \in \mathcal{C}^A | \neg(c \sqcap f \sqsubseteq \bot)\}.$$

Ein Feature passt also zu einer Capability, wenn die Vereinigung der beiden Konzeptdefinitionen erfüllbar ist. Um die einfache Erfüllbarkeit der Vereinigung von Featurekonzept und Capabilitykonzept zu erweitern, sollen die in [Li und Horrocks 2004] definierten Grade eines Matches in den Ansatz integriert werden. Die Grade eines Matches sollen die Abwägung der Güte eines Matches ermöglichen.

Definition 7 (Grade eines Matches nach [Li und Horrocks 2004])
d *sei der* Grad *eines Matches zwischen einem Feature* f *und einer Capability* c. *Er heiße*

- Exact, *falls gilt* $c \equiv f$,
- PlugIn, *falls gilt* $f \subseteq c$,

- Subsume, *falls gilt* $c \subseteq f$,
- Intersection, *falls gilt* $\neg(c \sqcap f \sqsubseteq \bot)$ *sowie*
- Disjoint *und damit nicht erfolgreich, falls gilt* $c \sqcap f \sqsubseteq \bot$.

Die möglichen Grade d des Matches sind wie folgt zu gliedern:

- Ein *Exact-Match* stellt den bestmöglichen Grad eines Matches dar. Da Feature- und Capability-Konzept äquivalent sind, besitzt der problemlösende Agent genau die gewünschte Funktionalität zur Lösung der Aufgabe und kann diese durchführen.
- Ein *PlugIn-Match* ist der nächstschlechtere Grad eines Matches. Ein Beispiel hierfür wäre die Nachfrage nach einem Feature *Edge Round*, das unter ein Konzept *Fräsen* subsumiert wird. Bei diesem Typ des Matches kann davon ausgegangen werden, dass der Agent dadurch, dass das angefragte Feature unter seine allgemeinere Fähigkeit subsumiert wurde, alle nötigen Fähigkeiten besitzt und in der Bearbeitung des Features erfolgreich ist.
- Der nächstschlechtere Grad des Matches ist ein *Subsume-Match*, bei dem das angefragte Feature eine Agenten-Capability subsumiert. Daraus lässt sich ableiten, dass der Agent durch die Anwendung seiner spezielleren Agenten-Capability nur einen Teil des allgemeineren Features umsetzen kann. Zur Bewältigung des Gesamtproblems, d. h. zur Umsetzung weiterer Eigenschaften des Features müssen im Gegensatz zum PlugIn-Match noch weitere Capabilities dieses oder anderer Agenten zum Einsatz kommen.
- Der *Intersection-Match* bestätigt nur, dass Feature und Capability nicht grundsätzlich inkompatibel sind.
- Bei Vorliegen eines *Disjoint-Matches* sind angefragtes Feature und Capability nicht kompatibel, d. h. ein Agent kann mit dieser Capability nicht zur Problemlösung beitragen.

Im Fall des oben erwähnten Beispiels wäre ein PlugIn-Match feststellbar, da sich aus den gegebenen Konzeptdefinitionen

$$Loch \subseteq Bohren$$

ergibt. Bei Vorliegen dieses Falls wäre eine Lösung des „Loch-Problems“ mit Hilfe der Bohren-Capability möglich und führt zu einer erfolgreichen Bearbeitung.

4.2.3. Integration in den *Diskursagenten*-Ansatz

Formal definiert [Timm 2004] einen Diskursagenten als 7-Tupel, der unter anderem aus lokalem Zustand und verschiedenen Funktionen zum Reflektieren über den aktuellen Zustand und zum Auswählen und Ausführen von Aktionen besteht.

Definition 8 (Diskursagent nach [Timm 2004])
Der 7-Tupel

$$AG = \langle \mathcal{L},\ Act,\ see,\ reflect,\ decide,\ execute,\ L_0 \rangle$$

sei ein Diskursagent*, der Capabilities besitzt, mit:*

- $\mathcal{L}$ *als der Menge der lokalen Zustände,*
- Act *als Menge der mögliche Agentenaktionen,*
- *einer Funktion* $reflect$*, die über den lokalen Zustand des Agenten reflektiert und ihn gegebenfalls anpaßt,*
- *einer Funktion* $decide$ *zur Auswahl eines zu verfolgenden Aktionsplans,*
- *einer Funktion* $execute$ *zur Auswahl und Ausführung einer Aktion,*
- *dem initialen Zustand* $L_0 \in \mathcal{L}$*,*

Ein möglicher lokaler Zustand L des Diskursagenten ist dabei wie folgt definiert:

Definition 9 (Lokaler Zustand nach [Timm 2004])
Ein Quintupel

$$L = \langle B,\ D,\ I,\ Pln,\ weight, \rangle$$

sei ein lokaler Zustand*, wobei:*

- B *die Menge der Überzeugungen,*
- D *die Menge der Diskursagentenziele,*
- I *die Menge der Absichten des Agenten,*
- Pln *die Menge der Aktionspläne,*
- *und* $weight$ *eine Gewichtung ist.*

Die weitere Definition der Zustandsmenge und der Funktionen erfolgt in [Timm 2004, Kapitel 3]. Aufbauend darauf erfolgt die Definition der Mengen von Features und Capabilities im Diskursagentenansatz. Diese werden im Rahmen des Diskursagentenansatzes durch verschiedene Prädikate definiert, die sich wiederum aus Termen zusammensetzen. Eine Aufstellung der im Diskursagentenansatz erlaubten Terme findet sich in [Timm 2004, S. 172 f]. Die Definition eines Featureprädikats erfordert zunächst die Definition der Mengen der Problem-

und Featuretypen und der Menge der Identifikationen, aus denen das Featureprädikat zusammenstellt wird und die außerdem im Rahmen der Logik des Diskursagenten verarbeitet werden können. Diese Mengen sind wie folgt definiert:

Definition 10 (Problemtypen)
Die Menge

$$\Pi = \{\pi_1, \ldots, \pi_n\}$$

sei die Menge der Problemtypen, wobei π_i ein Term ist.

Definition 11 (Featuretypen)
Die Menge

$$\Theta = \{\theta_1, \ldots, \theta_n\}$$

sei die Menge der Featuretypen, wobei θ_i ein Term ist.

Definition 12 (Identifikationen)
Die Menge

$$\Upsilon = \{v_1, \ldots, v_n\}$$

sei die Menge der Identifikationen, wobei v_i ein Term ist.

Ein Beispiel für einen Problemtyp π_i ist $Term('Grundplatte')$. Ein Featuretyp θ_i ist beispielsweise ein $Term(„CountersunkHole')$. Die Menge der Identifikationen Υ enthält Schlüssel, die zur eindeutigen Kennzeichnung eines Prädikats benutzt werden. Auf diesen Mengen aufbauend ist ein Featureprädikat folgendermaßen definiert:

Definition 13 (Featureprädikat)
Ein dreistellige Prädikat

$$Feature(\pi,\ \theta,\ v)$$

sei ein Featureprädikat, wenn gilt

- $\pi \in \Pi$ *ist der Typ des Produkts, das repräsentiert wird,*

- $\theta \in \Theta$ *ist der Typ des Features, das repräsentiert wird*
- *und* $\upsilon \in \Upsilon$ *ist ein eindeutiger Schlüssel des Prädikats.*

Aus der Definition ergibt sich, dass innerhalb eines Featureprädikats jeweils der Problem- und Featuretyp spezifiziert wird. Beides sind Terme aus den zuvor angegebenen Termmengen der Problem- bzw. Featuretypen. Jedem Featureprädikat ist zusätzlich ein Term mit einem eindeutigen Schlüssel zugeordnet, durch die die nachfolgend zu definierenden Featureparameter dem Feature zugeordnet werden können. Ein Beispiel für ein Featureprädikat ist folgendes Prädikat:

$$Feature(Term('Grundplatte'), Term('CountersunkHole'), Term('GS4R')))$$

Sämtliche Featureprädikate sind Elemente der Menge *Features*. Diese ist formal folgendermaßen definiert:

Definition 14 (Featureprädikatmenge)
Die Menge

$$\Delta = \{\delta_1, \ldots, \delta_n\}$$

sei die Menge der Featureprädikate eines Diskursagenten, wenn gilt δ_i *ist ein Featureprädikat.*

Die das Feature definierenden Parameter werden über Featureparameter-Prädikate repräsentiert. Die Parameter können auch als die Eigenschaften eines Features angesehen werden. Zur ihrer näheren Definition müssen zunächst die Mengen der Namen dieser Parameter, der Typen und der Werte eingeführt werden. Diese definieren sich wie folgt:

Definition 15 (Parameternamen)
Die Menge

$$\Omega = \{\omega_1, \ldots, \omega_n\}$$

sei die Menge der Parameternamen eines Diskursagenten.

Definition 16 (Parametertypen)
Die Menge

$$\Xi = \{Integer, Double, Liste, Range, \ldots\}$$

sei die Menge der Parametertypen eines Diskursagenten.

Die Elemente dieser Menge entsprechen den Namen der erlaubten Datentypen einer zur Umsetzung des Konzepts gewählten Sprache. Denkbar sind hier wie in der Definition angegeben Ganz- und Fließkommazahlen, Daten-Container sowie Wertebereiche usw.

Definition 17 (Featureparameter-Prädikat)
Ein vierstellige Prädikat

$$FeatureParameter(\upsilon, \omega, \xi, \zeta)$$

sei ein Featureparameter-Prädikat, wenn gilt

- $\upsilon \in \Upsilon$ *ist ein Term mit der Identifikation des Featureprädikats, zu dem dieser Parameter gehört,*
- $\omega \in \Omega$ *ist ein Term mit dem Namen des Parameters,*
- $\xi \in \Xi$ *ist ein Term mit dem Typen des Parameters,*
- *und* ζ *ist ein Term mit einem String, der den Wert des Featureparameter-Prädikats repräsentiert.*

Zu beachten ist bei dieser Definition, dass der Term ξ die Angabe enthält, von welchem Typ der Inhalt des Terms ζ ist, d. h. wie die dort angegebene Zeichenkette zu interpretieren ist. Durch den Term mit der Identifikation des Featureprädikats wird die erwähnte Zuordnung zwischen Featureprädikaten und ihren Parametern vorgenommen. Ein Beispiel hierfür ist folgendes Prädikat:

$$FeatureParameter(Term('GS4R'), Term('Durchmesser'), Term('Integer'), Term('24')))$$

Die Featureparameter-Prädikate sind Elemente der folgenden Menge:

Definition 18 (Featureparameter-Prädikat-Menge)
Die Menge

$$\Lambda = \{\lambda_1, \ldots, \lambda_n\}$$

sei die Menge der Featureparameter-Prädikate eines Diskursagenten, wenn gilt λ_i ist ein Featureparameter-Prädikat.

Analog zu den Featuren werden die Capabilities innerhalb des Diskursagenten ebenfalls als Prädikate definiert. Hierzu muss zunächst einmal die Menge der Capabilitytypen definiert werden:

Definition 19 (Capabilitytypen)
Die Menge

$$\Gamma = \{\gamma_1, \ldots, \gamma_n\}$$

sei die Menge der Capabilitytypen, wobei γ_i ein Aktionsterm ist.

Hierbei ist zu beachten, dass es sich bei den einzelnen Capabilities um Aktionsterme handelt, die im Rahmen der Planung des Diskursagenten zum Einsatz kommen können. Ein Beispiel für einen Capabilitytypen ist folgender Aktionsterm:

$$Aktionsterm('Bohren')$$

Ein Capabilityprädikat und die Menge die diese Prädikate enthält definiert sich darauf aufsetzend folgendermaßen:

Definition 20 (Capabilityprädikat)
Ein zweistelliges Prädikat

$$Capability(v,\ \gamma)$$

sei ein Capabilityprädikat, wenn gilt

- $v \in \Upsilon$ *ist ein Term mit der eindeutigen Identifikation des Capabilityprädikats und*
- $\gamma \in \Gamma$ *ist die Capability, die das Prädiakt repräsentiert.*

Für die Capabilityprädikate existiert ebenfalls ein Term mit einem eindeutigen Schlüssel, über den eine Zuordnung der Parameter vorgenommen werden kann. Außerdem existiert ein Aktionsterm mit dem Capabilitytypen, den sie repräsentieren. Möglich ist beispielsweise folgendes Prädikat:

$$Capabililty(Term('UZR5'), Aktionsterm('Bohren'))$$

Die Capabilityprädikate sind Elemente der folgenden Menge:

Definition 21 (Capabilityprädikatmenge)
Die Menge

$$\Phi = \{\phi_1, \ldots, \phi_n\}$$

sei die Menge der Capabilityprädikate eines Diskursagenten, wenn gilt ϕ_i ist ein Capabilityprädikat.

Die Parameter der Capabilities werden analog zu den Parametern der Features als Prädikate definiert. Die Definition greift auf die bereits eingeführten Mengen der Prädikatnamen und Prädikattypen zurück. Ein Capabilityparameter-Prädikat ist anhand folgender Definition festgelegt:

Definition 22 (Capabilityparameter-Prädikat)
Ein vierstellige Prädikat

$$CapabilityParameter(\upsilon,\ \omega,\ \xi,\ \zeta)$$

sei ein Capabilityparameter-Prädikat, wenn gilt

- $\upsilon \in \Upsilon$ *ist ein Term mit der Identifikation des Capabilityprädikats, zu dem dieser Parameter gehört,*
- $\omega \in \Omega$ *ist ein Term mit dem Namen des Parameters,*
- $\xi \in \Xi$ *ist ein Term mit dem Typen des Parameters,*
- *und ζ ist ein Term mit einem String, der den Wert des Capabilityparameter-Prädikats repräsentiert.*

Auch bei dieser Definition gibt der Term ξ die Interpretationsvorschrift für den Wert des Terms ζ. Über den eindeutigen Schlüssel υ kann der Parameter einem Capabilityprädikat zugeordnet werden. Folgendes Prädikat ist ein der Definitionsvorschrift genügendes Parameterprädikat:

$$CapabililtyParameter(Term('UZR5'), Term('Durchmesser',$$
$$Term('Range'), Term('[5,67]')))$$

Dieses Beispiel enthält die Definition eines Intervalls, das in der späteren Verarbeitung besondere Beachtung findet. Die Capabilityparameter-Prädikate sind Elemente der folgenden Menge:

Definition 23 (Capabilityparameter-Prädikat-Menge)
Die Menge

$$\Psi = \{\psi_1, \ldots, \psi_n\}$$

sei die Menge der Capabilityparameter-Prädikate eines Diskursagenten, wenn gilt ψ_i ist ein Capabilityparameter-Prädikat.

Für die Mengen der Feature- und Capability-Prädikate Δ bzw Φ gelte weiterhin, dass sie Teilmengen der Menge der Beliefs B des Diskursagenten sind: $\Delta \subset B$ sowie $\Phi \subset B$. Im Deliberationszyklus des Diskursagenten muss nun eine Umwandlung der Feature- und Capabilityprädikate in OWL-Konzepte vorgenommen werden, damit im Rahmen der Entscheidungsfindung des Diskursagenten die Ausführung logischer Inferenz möglich ist. Diese Inferenz wird durchgeführt aufgrund der jeweils aktuellen Beliefs des Agenten, weswegen eine Übersetzung zwischen der Repräsentation des aktuellen Zustands auf der Diskursagentenseite und der ontologischen Repräsentation auf der anderen Seite unumgänglich ist. Aus den Prädikaten der Logik des Diskursagenten müssen dabei Konzepte einer OWL-Ontologie werden. Die jeweiligen Parameter der Prädikate werden zu Properties dieser Konzepte umgesetzt. Die Umwandlung wird durch die Funktion $transform$ vorgenommen, die formal folgendermaßen definiert ist:

Definition 24 (Umwandlungsfunktion)
Die Funktion

$$transform : \Phi \times \Psi \times \Delta \times \Lambda \mapsto \mathcal{C}^A \times \mathcal{F}^A$$

sei die Umwandlungsfunktion eines Diskursagenten. Dabei gelte

- $\forall \phi \in \Phi$ *wird ein entsprechendes* $c \in \mathcal{C}^A$ *erzeugt,*
- $\forall \delta \in \Delta$ *wird ein entsprechendes* $f \in \mathcal{F}^A$ *erzeugt,*
- *und eine Funktion* $createRestriction : \Psi \times \Lambda \mapsto \mathcal{R}$ *ordne dabei dem erzeugten Konzept entsprechende Wertebeschränkungen auf Konzepteigenschaften zu.* $\mathcal{R}$ *ist dabei die Menge der Rollen in einem Ontologie-Modell* M*.*

Die Funktionalität dieser Abbildung wird im Folgenden in Form einer algorithmischen Spezifikation näher verfeinert.

Im Rahmen des deliberativen Verhaltens des Diskursagenten, d. h. im Entscheidungszyklus wird dann die entsprechende Feststellung durch Bildung der erwähnten Menge *matches* getroffen, ob es dem Agenten möglich ist eine zum angefragten Feature passende Capability zur Verfügung zu stellen. Dieses Vorgehen wird über die *decide*-Abbildung[3] des Agenten durchgeführt, die die Entscheidungsfindung aufgrund seines aktuellen Zustands steuert und dazu unter anderem die Feature- und Capabilityprädikate heranzieht.

4.2.4. Schnittstelle zur Agentenplanung

Die Schnittstelle zur Planung des Diskursagenten ist dadurch gegeben, dass der Agent nach Feststellung eines Matches in seinen Beliefs die Überzeugung besitzt, dass er in einer laufenden Verhandlung ein Angebot zur Lösung eines Problems abgeben kann. Im Rahmen der Aktionsselektion des Diskursagenten wird dann entsprechend ein Angebot auf die Anfrage abgegeben, sofern der Agent im Rahmen seiner Deliberation zu dieser Überzeugung gekommen ist. In dem Fall, dass das Angebot des Agenten angenommen wurde, findet zusätzlich eine Übertragung der Capability, die zur Lösung des Problems, das angefragt wurde nötig ist, in die Planung des Agenten statt. Dies ist einfach möglich, da die Capabilities als Aktionsterme spezifiziert wurden, die als exekutive Aktionen in Aktionssequenzen der Diskursagentenplanung eingesetzt werden können. Formal definiert sich dieses als

$$Capability \subset Act^e.$$

Ein Plan ist innerhalb des Diskursagentenansatzes als

$$plan = \langle \varphi_p re,\ \varphi_p ost,\ Actions,\ status,\ select \rangle$$

definiert[4]. *Actions* ist dabei eine Teilmenge und Aktionssequenz $\alpha_1, \ldots, \alpha_n$ des Aktionspotenzials *Act* des Agenten. Zur näheren Definition der weiteren Elemente eines Plans und des Ablaufs der Planung innerhalb des Diskursagentenansatzes sei auf [Timm 2004] verwiesen.

Im nächsten Abschnitt wird die Spezifikation des Algorithmus vorgenommen, nach dem die Übersetzung der Feature- und Capability-Prädikate vorgenommen wird und außerdem der CapabilityMatch-Algorithmus spezifiziert.

4.2.5. Spezifikation der algorithmischen Umsetzung

Zur Spezifikation der Umsetzung werden Algorithmen in Pseudo-Code-Form entwickelt. Durch die Angabe dieser Algorithmen werden die in Abschnitt 4.2 vorgenommenen Formalisierungen näher spezifiziert. Als erstes soll der Algorithmus *Transform* zur Übersetzung der

[3] Die Abbildung *decide* ist in [Timm 2004, S. 96] näher erläutert.
[4] Vergleiche Definition II.24 in [Timm 2004, S. 72]

Prädikate in Konzepte eines Ontologiemodells erläutert werden. Er ist in der Algorithmusdefinition 1 dargestellt.

Algorithmus 1 : *Transform* zur Übersetzung zwischen Prädikaten und Konzepten

Input : Δ, Λ, Φ, Ψ
Output : Ein Ontologiemodell M

forall $Feature \in \Delta$ **do**
  $\beta \leftarrow$ ein Vektor mit allen Parametern für *Feature*
  $M \leftarrow$ `GenerateOntClass` (*Feature*, β)

forall $Capability \in \Phi$ **do**
  $\beta \leftarrow$ ein Vektor mit allen Parametern für *Capability*
  $M \leftarrow$ `GenerateOntClass` (*Capability*, β)

Der Algorithmus erzeugt aus allen ihm übergebenen Prädikaten Elemente eines Ontologiemodells M. Übergeben werden ihm zum Einen die Feature-Prädikate und die zugehörigen FeatureParameter-Prädikate sowie andererseits die Capability-Prädikate und deren Parameterprädikate. Der Algorithmus sammelt nun zunächst für alle Feature-Prädikate die zugehörigen Parameter ein und übergibt das Prädikat und seine Parameter an einen weiteren Algorithmus zur Erzeugung eines Ontology-Konzepts (siehe Definition 2). Diese erzeugt zunächst eine Konzeptdefinition über die Vereinigung mit dem schon vorhandenen Konzept p. Diese Definition wird anschließend um Properties entsprechend der Parameterprädikate erweitert.

Algorithmus 2 : *GenerateOntClass* zur Erzeugung eines Ontologiekonzepts

Input : ein Prädikat $Prd \in \Delta \cup \Phi$ und seine Parameter $Paramtrs \in \Lambda \cup \Psi$
Output : Ein äquivalentes Ontologiekonzept c

$c \leftarrow$ Definition über Vereinigung mit Konzept p
forall $p \in Paramtrs$ **do**
  Property $\leftarrow p.\omega$
  Type $\leftarrow p.\xi$
  Value $\leftarrow p.\zeta$
  $c \leftarrow$ `CreateRestriction` (Property, Type, Value)

Hierzu kommt eine weiterer Algorithmus zum Einsatz, der entsprechend des Typs des Parameters entweder eine `HasValueRestriction` erzeugt wenn ein Basisdatentyp[5] vorliegt oder eine `AllValuesFromRestriction` mit einem neuen benutzerdefinierten Datentyp anlegt, wenn es sich beispielsweise um einen zu definierenden Wertebereich handelt. Dasselbe Verfahren wird für sämtliche Capabilityprädikate wiederholt, für die wiederum die je-

[5]Baisdatentypen sind beispielsweise Integer, Float, Bool usw.

weils zugehörigen Parameter eingesammelt werden und anschließend das Ontologiekonzept mit Properties und Restrictions erzeugt wird.

Algorithmus 3 : *CreateRestriction* zur Erzeugung der Wertbeschränkungen

Input : Property, Type, Value
Output : Eine Beschränkung r der angegebenen Property Property

if Type = *Basisdatentyp* **then**
  r = HasValueRestriction mit dem Wert Value
else
  r = AllValuesFromRestriction mit dem Wert Value

Der Algorithmus 4 zeigt dann das Vorgehen zur Feststellung eines Matches zwischen Feature und Capabilites. Dieser bedient sich zunächst des oben vorgestellten Algorithmus zum Umwandeln der Prädikate, um ein Ontologiemodell zu erzeugen, über das inferiert werden kann. Nun wird für jedes Capabilitykonzept und das Featurekonzept überprüft, welcher Typ des Matches vorliegt. Die Reaktion auf bestimmte Typen des Matches ist abhängig vom Einsatz in einer bestimmten Domäne. Eine entsprechende Festlegung der Reaktion auf eine Art des Matches ist in der jeweiligen domänenabhängigen Umsetzung vorzunehmen.

Algorithmus 4 : *CapabilityManagementMatch* zur Feststellung eines Matches

Input : $Feature \in \Delta, \Lambda, \Phi, \Psi$
Output : Grad des Matches d zwischen Feature und Capabilities

$M \leftarrow$ `Transform(`$Feature, \Lambda, \Phi, \Psi$`)`
forall $c \in M$ **do**
  $d \leftarrow$ bestimmen durch Inferenz über f und c unter Einsatz eines Reasoners
  switch d **do**
    Domänenabhängig Reaktion auf einen bestimmten Typ auslösen

4.2.6. Beispiel für den Ablauf des Matchmakings

Abschließend wird in diesem Abschnitt ein kompletter Ablauf des Machtmaking-Prozesses anhand des in diesem Kapitel schon verwendeten simplen Beispiels des Matchings zwischen dem Feature *Loch* und einer Agenten-Capability *Bohren* vorgestellt.

Zu Beginn des Matchmakings liegen diese als Prädikate der Logik des Diskursagenten vor. Diese definieren sich über die Terme der Identifikation-, Problem-, Feature- und Capability-

typen. Die Elemente dieser Mengen stellen sich in diesem Beispiel wie folgt dar:

$$\Upsilon = \{Term('M4SR'),\ Term('HFD4')\}$$

$$\Pi = \{Term('Grundplatte')\}$$
$$\Theta = \{Term('Loch')\}$$
$$\Gamma = \{Aktionsterm('Bohren')\}$$

Aufbauend auf diesen Termen sind das Feature und die Capability folgendermaßen definiert:

$$\Delta = \{Feature(Term('Grundplatte'),\ Term('Loch'),\ Term('M4SR'))\}$$

$$\Phi = \{Capability(Term('HFD4'),\ Aktionsterm('Bohren'))\}$$

Die Parameter des Features und der Capability sind ebenfalls über die entsprechenden Parameterprädikate repräsentiert:

$$\Omega = \{Term('Durchmesser')\}$$

$$\Lambda = \{FeatureParameter(Term('M4SR'),\ Term('Durchmesser'),\ Term('Integer'),\ Term('5'))\}$$
$$\Psi = \{CapabilityParameter(Term('HFD4'),\ Term('Durchmesser'),\ Term('Range'),\ Term('[5,35]'))\}$$

Diese Prädikate werden im Rahmen der Transform-Funktion in beschreibungslogische Entsprechung umgewandelt, so dass sich im Anschluss folgendes Ontologiemodell ergibt, über das ein Reasoner die angegebene Subsumptionsrelation folgern kann:

$$M = \{Loch \equiv p \sqcap =_5 hatDurchmesser,\ Bohren \equiv p \sqcap \geqslant_5 hatDurchmesser \sqcap \leqslant_{35} hatDurchmesser\}$$
$$(p \sqcap =_5 hatDurchmesser) \subseteq (p \sqcap \geqslant_5 hatDurchmesser \sqcap \leqslant_{35} hatDurchmesser)$$

Aus der festgestellten Subsumptionsbeziehung kann auf das Vorliegen eines PlugIn-Matches geschlossen werden, aus dem sich wiederum der Fakt ableiten lässt, dass das Feature mit der angegebenen Capability herzustellen ist.

Kapitel 5.

Evaluation

Die Evaluation des vorgestellten Ansatzes erfolgt mit Hilfe einer prototypischen Implementierung innerhalb des im Rahmen des DFG-Projektes IntaPS entwickelten Softwareprototypen. Durch diesen Prototypen sollen anschließend unter Einsatz eines an das Sterlingmotor-Szenario angelehnten Evaluationsszenarios verschiedene Versuchsdurchläufe durchgeführt werden. Die erzielten Ergebnisse werden anschließend dahingehend untersucht, inwieweit der entwickelte Ansatz Vor- bzw. Nachteile gegenüber anderen Matchingverfahren besitzt. Im folgenden Kapitel wird zunächst die Implementierung des Ansatzes erläutert und anschließend das verwendete Evaluationsszenario und die erzielten Ergebnisse vorgestellt. Abschließend erfolgt eine Interpretation der erzielten Ergebnisse.

5.1. Prototypische Implementierung

Die Implementierung des eigenen Ansatzes erfolgt innerhalb des bestehenden IntaPS-Prototypen. Dieser stellt bereits zwei andere Matchingverfahren zur Verfügung, gegen die ein Vergleich erfolgen kann. Bei der Implementierung des Ansatzes kamen mehrere vorhandene Softwarebibliotheken zum Einsatz. Hier ist zunächst die Reasoner-Implementierung *Pellet*[1] [Sirin und Parsia 2004] zu nennen. Pellet ist ein in Java implementierter OWL-DL-Reasoner, dessen Einsatz aus mehreren Gründen günstig erschien. Zunächst lässt er sich als JAVA-Bibliothek direkt in andere JAVA-Programme integrieren und muss nicht per HTTP über die DIG-Schnittstelle[2] angesprochen werden. Dieses erschien einerseits deswegen sinnvoll, um den Protokoll-Overhead einer in jedem Deliberationszyklus notwendigen Kommunikation zwischen Diskursagent und Reasoner über eine HTTP-Verbindung zu vermeiden. Andererseits wurde eine Implementierung des Diskursagenten ermöglicht, in der jeder Agent sein eigenes Ontologiemodell seiner Capabilities aufbauen kann und darüber inferiert. Zu diesem Zweck stellt Pellet zwei Schnittstellen zu Ontologie-Frameworks zur Verfügung. Hier

[1] Bezug unter `http://www.mindswap.org/2003/pellet`

[2] siehe Abschnitt 2.2.3

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        targetNamespace="file:schema.xsd">
<simpleType name="between5and20">
    <restriction base="integer">
        <minInclusive value="5"/>
        <maxInclusive value="20"/>
    </restriction></simpleType>
</schema>
```

Abbildung 5.1.: Ein Beispiel für eine XML-Schema-Definition eines benutzerdefinerten Datentypen.

fiel die Wahl auf das *Jena Sematic Web Framework*[3] [McBride 2002], das gegenüber dem *OWL API*[4] [Bechhofer u. a. 2003] als das besser gepflegte Framework erschien, das darüber hinaus weitere Möglichkeiten bietet, die bei zukünftigen Anwendungen im Agentenbereich gegebenenfalls von Nutzen sein können.

Pellet bietet neben seiner Eignung zur Verarbeitung des vollen Sprachumfangs von OWL-DL, die beispielsweise auch die Verarbeitung von Nominals einschließt, die Möglichkeit zum Reasoning über Datentypen. Diese Eigenschaft ist für die Verarbeitung von Domänen wie z. B. des Sterlingmotorszenarios wichtig, um die hier gegebenen Beschränkungen der Bearbeitungsgrößen von Maschinen als auch die konkreten Werte der zu fertigenden Aufträge zu berücksichtigen. Eine grundlegende Unterstützung ist durch verschiedene unterstützte Basisdatentypen, wie beispielsweise numerische Werte, gegeben. Es können aber auch zusätzlich benutzerdefinierte Datentypen verarbeitet werden. Durch dieses Merkmal können unter anderem auf den Basisdatentypen aufbauende Wertebereiche definiert werden, die z. B. als Datentyp `between5and20` einen Wertebereich $[5, 20]$ definieren, durch den die untere und obere Grenze für eine Bearbeitungsgröße einer bestimmten Maschine festgelegt wird. Diese benutzerdefinierten Datentypen können in OWL-DL durch den Einsatz von XML-Schema [Walmsley 2004] definiert werden. Hierzu muss zum Einen der Basisdatentyp angegeben werden, auf dem der Datentyp aufbaut und zum anderen die jeweiligen Beschränkungen des Datentyps spezifiziert werden. Abgelegt werden diese Definitionen in separaten Dateien, die aus einer OWL-Ontologie heraus referenziert werden können. Eine beispielhafte XML-Schema-Datei für den erwähnten Datentyp `between5and20` ist in Abbildung 5.1 dargestellt.

In der durchgeführten Implementierung kommt das XML-Binding-Framework *Castor*[5] zum Einsatz, das die dynamische Erzeugung der XML-Schema-Dateien aus den aktuellen Ca-

[3] siehe `http://jena.sourceforge.net`
[4] siehe `http://owl.man.ac.uk/api.shtml`
[5] siehe `http://www.castor.org`

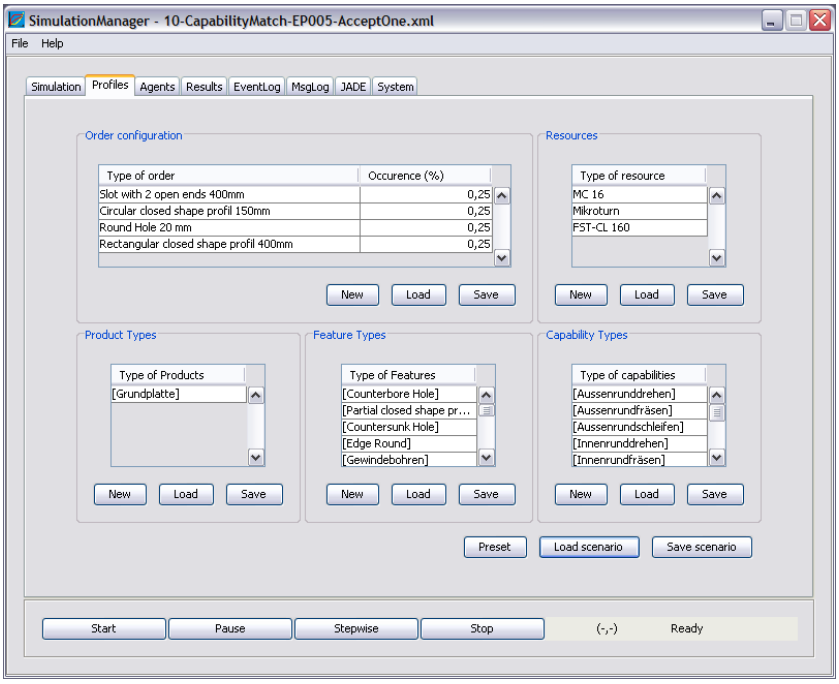


Abbildung 5.2.: Screenshot des IntaPS-Protoypen

pabilities des Agenten ermöglicht. Diese werden dann im Ontologie-Modell, das mit Hilfe des Jena-Frameworks erzeugt wird, eingelesen und die dynamisch erzeugten Datentypen als RDF-Schema-Datentypen in der Formulierung von Beschränkungen auf den Properties der Ontologie-Konzepte verwendet. Die schon vorhandene Funktionalität des IntaPS-Prototypen ermöglicht dann in den zu spezifizierenden Evaluationsszenarien die Angabe bei der Spezifikation der Ressourcen, mit Hilfe welchen Verfahrens die Feststellung eines Matches zwischen anfragendem und angefragten Agenten erfolgen soll. Die Eingabemaske für die Auftrags- und Ressourcenspezifikation des IntaPS-Prototypen ist in Abbildung 5.2 dargestellt. Die näheren Details der Szenarioerstellung werden in der Vorstellung des verwendeten Evaluationsszenarios im nächsten Abschnitt verdeutlicht.

5.2. Evaluationsszenario

Konkret erfolgt die Evaluation mit dem Ziel der Feststellung der Eignung des entwickelten Matching-Verfahrens für intelligente Softwareagenten in Domänen, in denen eine Zuweisung von Problemen zu Agenten, die diese Probleme lösen können, nötig ist. Das allgemeine Ziel dieser Zuweisung ist es den Problemlösern die Möglichkeit zu geben, ihre Eignung zur Lösung einer bestimmten Aufgabe zu erkennen. Außerdem ist eine hohes Maß an Flexibilität wünschenswert, damit beispielsweise in geeigneter Weise auf Störungen reagiert werden

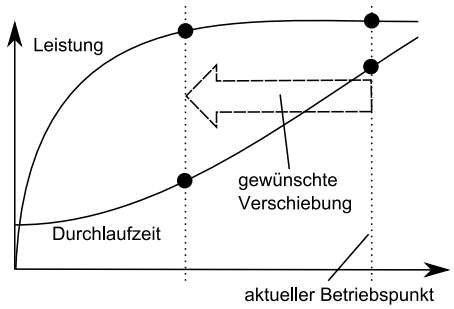


Abbildung 5.3.: Zusammenhang von Durchlaufzeit und Maschinenauslastung.

kann. Das in den Grundlagen vorgestellte Sterlingmotorszenario ist als ein Beispiel für ein Szenario der Fertigungsdomäne auswählt und das erstellte Szenario daran angelehnt worden. Die hier notwendige Zuordnung von Aufträgen zu Maschinen entspricht der oben erwähnten Eigenschaft der Aufgabenzuordnung. Eine besondere Eigenschaft des zu wählenden Szenarios ist, dass zum Einen die Bearbeitung der Aufträge nur durch bestimmte Maschinen möglich ist und zusätzlich durch die Formulierung eines Engpasses die Anforderungen an die Verteilung der Aufträge auf die Maschinen weiter erhöht wird. Die Kriterien nach denen die Auswertung erfolgen soll, sind einerseits die Durchlaufzeit von Aufträgen und andererseits die Auslastung der vorhandenen Maschinen. Diese sind neben anderen Kriterien wichtige Kenngrößen der Produktionslogistik. Die Durchlaufzeit ist die Zeit von der Ausschreibung eines Auftrags bis zu seiner Fertigstellung auf der Maschine. Die Auslastung bezeichnet, wie hoch der Anteil der Nutzung der Maschine in einem bestimmten Zeitraum gegenüber der Zeit ist, in der sie sich im Leerlauf befindet. Durchlaufzeit und Auslastung stehen nach [Nyhuis und Wiendal 1999] in dem in Abbildung 5.3 gezeigten Zusammenhang. Angenommen das Verhältnis dieser beiden Kenngrößen befindet sich momentan an der rechten gestrichelten Linie, dem aktuellen Betriebspunkt, so ist eine Verschiebung dieses Betriebspunkt in Richtung des Ursprungs gewünscht, um die Durchlaufzeit von Aufträgen zu verringern. Gegenläufiges Ziel ist aber gleichzeitig die Auslastung des Maschinenparks nicht übermäßig zu verringern. Das Szenario, das zur Feststellung einer gelungenen Verschiebung dienen soll, wird nun im Weiteren näher erläutert und vorgestellt werden.

Die Elemente des zur Evaluation verwendeten Szenarios sind zunächst einmal die Aufträge und die Maschinen[6] mit ihren jeweiligen Eigenschaften. Bestandteil des IntaPS-Ansatzes ist es, diese Aufträge und Resourcen durch Agenten zu repräsentieren. Die grundlegende Architektur dieses Ansatzes ist in Abbildung 5.4 dargestellt. Wichtige Elemente dieser Architektur sind einerseits die Agenten $OAg_1, \ldots OAg_n$, die zum Lösen ihrer Probleme die Agenten $RAg_1, \ldots, RAg_n$ kontaktieren. Erzeugt werden die Aufträge andererseits durch einen so genannten zentralen Order-Generator, der aus der jeweiligen Szenariospezifikation ableitet, welche Aufträge zu starten sind und diese anschließend in Form eines Agenten erzeugt. Die Aufträge repräsentieren jeweils ein zu fertigendes Feature, zu dem eine Ressource gesucht

[6] Im folgenden werden Maschinen auch Ressourcen genannt.

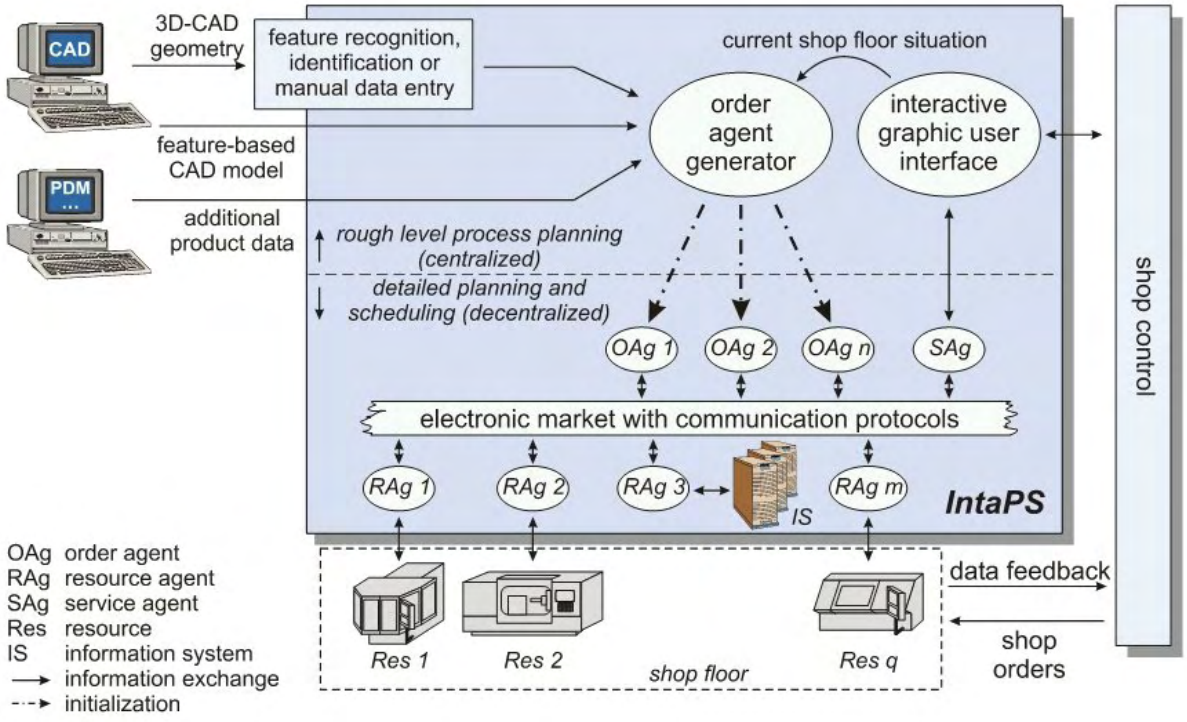


Abbildung 5.4.: Architektur des IntaPS-Ansatzes

Tabelle 5.1.: Aufträge des verwendeten Evaluationsszenarios

Auftragsfeature	**Eigenschaft**
Circular closed shape profile 150mm:	
Circular closed shape profile	Durchmesser = 150mm
Slot with 2 open ends 400mm:	
Slot with 2 open ends	Länge = 400mm
Round Hole 20mm:	
Round Hole	Durchmesser = 20mm
Rectangular closed shape profile 400mm:	
Rectangular closed shape profile	Länge = 400mm

Tabelle 5.2.: Ressourcen des verwendeten Evaluationsszenarios

Ressourcenname	**Eigenschaften**
MC16:	
Bohren	Durchmesser := [0mm, 25mm]
Profilfräsen	Länge := [0mm, 630mm]
Aussenrundfräsen	Durchmesser := [0mm, 1000mm]
Mikroturn:	
Aussenrunddrehen	Durchmesser := [0mm, 500mm]
Innenrunddrehen	Durchmesser := [0mm, 400mm
FST-CL 160:	
Aussenrundfräsen	Durchmesser := [0mm, 1000mm]
Planfräsen	Länge := [0mm, 630mm]

wird, die in Lage ist, dieses unter Beachtung der jeweiligen Parameter des Features herzustellen. Die vier verwendeten Aufträge sind in Tabelle 5.1 dargestellt. Ein Beispiel für einen Auftrag ist die Anfrage zur Fertigung eines runden Loches (Round Hole) mit dem Durchmesser von 20mm. Auf der Ressourcenseite kommen in diesem Szenario drei unterschiedliche Maschinen mit unterschiedlichen Fähigkeiten zum Einsatz. Diese sind in Tabelle 5.2 aufgeführt. Ein Beispiel für eine Resource ist die Maschine MC16. Diese verfügt in diesem Szenario über drei Capabilities mit zugehörigen Beschränkungen. Dies ist beispielsweise die Capability Bohren, die zur Fertigung von Features mit einem Durchmesser zwischen 0 und 25mm zum Einsatz kommen kann.

Zum Vergleich mit dem einwickelten Verfahren werden die oben erwähnten bereits im Prototypen implementierten Matchingverfahren herangezogen. Dies ist einerseits die direkte Maschinenzuordnung in der Spezifikation der Aufträge und andererseits die feste Spezifikation von zur Fertigung nötigen Capabilities in der Spezifikation der Aufträge. Das erste Verfahren wird im Folgenden *Machine-Matching* genannt und ist aus der Praxis motiviert, dass die

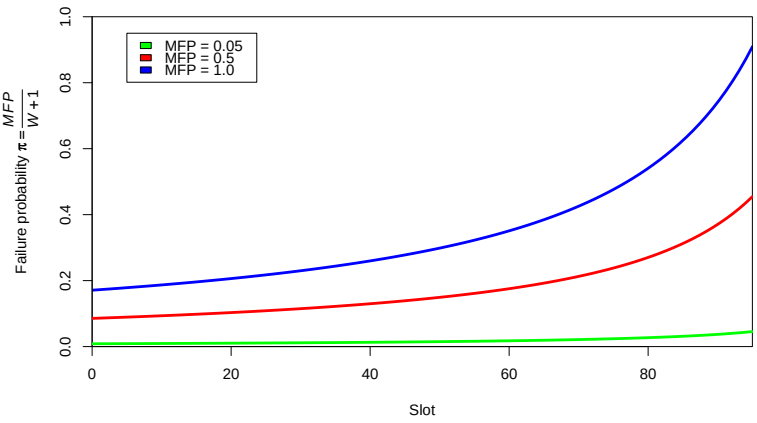


Abbildung 5.5.: Verlauf der Ausfallwahrscheinlichkeit π für unterschiedliche *MFP*-Werte.

Erstellung eines Arbeitsplans manuell durch einen Arbeitsplaner durchgeführt wird [Eversheim 2002]. Dieser ordnet für eine bestimmte Aufgabe eine einzelne Maschine zu und es besteht keine Möglichkeit aktuelle Ereignisse in der Fertigung in diese Arbeitsplanung einfließen zu lassen. Das zweite Verfahren wird im Folgenden *Skill-Matching* genannt und ist als erster Ansatz zu verstehen, die Zuordnung von Aufträgen zu Maschinen dadurch zu flexibilisieren, dass Aufträge über die benötigten Capabilities auf Maschinenseite spezifiziert werden. Ein weiteres Merkmal des verwendeten Szenarios ist die Einstellung unterschiedlicher Ausfallwahrscheinlichkeiten für die verwendeten Maschinen. Innerhalb des IntaPS-Prototypen errechnet sich diese Ausfallwahrscheinlichkeit π nach folgender Formel:

$$\pi = \frac{MFP}{W+1}$$

W ist dabei der so genannte *Machine Wear Out Factor*. Dies ist der Betrag, um den sich der Wartungszustand einer Machine pro Simulationsrunde verringert. Voreinstellt ist hier der Wert 0,05, der den initialen Wartungszustand 4,85 pro Runde verringert. Der Wert *MFP* ist die so genannte *Machine Failure Probability*. Diese gibt an, wie anfällig die Machine generell für Störungen ist. Hier wurden vier mögliche Fälle durch die Wahrscheinlichkeiten 0, 0,05, 0,5 und 1 abgedeckt. Diese unterschiedlichen Einstellungen wurden gewählt, um das Verhalten der eingesetzten Verfahren bei Störungen und des daraus resultierenden Flexibilitätsbedarfs auf Ressourcenseite zu überprüfen. In Abbildung 5.5 sind die Verläufe der Wahrscheinlichkeiten π für einen Maschinenausfall über den Lauf des Simulationssystems dargestellt. Je höher der Wert der *MFP* ist, desto anfälliger wird die Maschine zum Ende einer Simulationsrunde hin. Wenn im Folgenden von einem Wert für die Ausfallwahrscheinlichkeit gesprochen wird, bezieht sich dieser auf die *MFP*. Aufbauend auf diesen Voraussetzungen wurden die folgenden Szenariospezifikationen erstellt, die sich aus 3 unterschiedlichen Matchingverfahren und 4 unterschiedlichen Ausfallwahrscheinlichkeiten ergeben[7]:

[7]Für alle Testläufe wurde als Annahmeverhalten der Ressourcen *AcceptOne* gewählt, das den Res-

1. Machine-Matching, Ausfallwahrscheinlichkeit 0
2. Machine-Matching, Ausfallwahrscheinlichkeit 0,05
3. Machine-Matching, Ausfallwahrscheinlichkeit 0,5
4. Machine-Matching, Ausfallwahrscheinlichkeit 1
5. Skill-Matching, Ausfallwahrscheinlichkeit 0
6. Skill-Matching, Ausfallwahrscheinlichkeit 0,05
7. Skill-Matching, Ausfallwahrscheinlichkeit 0,5
8. Skill-Matching, Ausfallwahrscheinlichkeit 1
9. Capability-Matching, Ausfallwahrscheinlichkeit 0
10. Capability-Matching, Ausfallwahrscheinlichkeit 0,05
11. Capability-Matching, Ausfallwahrscheinlichkeit 0,5
12. Capability-Matching, Ausfallwahrscheinlichkeit 1

Für jedes Szenario wurden jeweils 30 Durchläufe durchgeführt, um möglichst signifikante Ergebnisse zu erzielen. Die Ergebnisse der Auftrags- und Ressourcenagenten wurden durch den IntaPS-Prototypen in einer MySQL-Datenbank[8] abgelegt und anschließend in *Comma-Separated-Values*-Dateien exportiert, die mit Hilfe eines erstellten Konverterprogramms in ein für das Statistikprogramm SYSTAT[9] einlesbares Format überführt wurden. Mit Hilfe dieses Programms wurden die Ergebnisse statistisch ausgewertet. Die erzielten Ergebnisse werden im nächsten Abschnitt vorgestellt und anschließend interpretiert.

5.3. Darstellung der Ergebnisse

Tabelle 5.3.: Statistik der erzeugten Auftragsagenten

	Capability-Matching	**Machine-Matching**	**Skill-Matching**	**Total**
Nicht erfolgreich	885	1018	866	2769
Erfolgreich	2293	1920	2161	6374
Total	3178	2938	3027	9143

Zunächst einmal kurz die Eckdaten der durchgeführten Evaluation: Insgesamt durchführt wurden 360 Simulationsdurchläufe mit jeweils 96 Einzelslots woraus sich eine Gesamtzahl

sourcen die größte Flexibilität beim Annehmen von Aufträgen bietet

[8] siehe `http://www.mysql.org`

[9] siehe `http://www.systat.com`

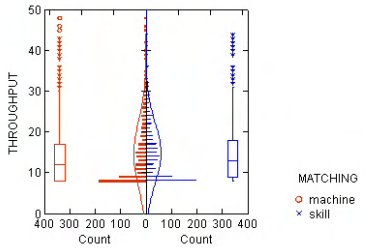
von 34560 simulierten Slots ergibt. Zusammengenommen wurden auf der Auftragsseite 9143 Ergebnisse erzeugt, was gleichzeitig der Anzahl der erzeugten Aufträge entspricht. Weitere Ergebnisse sind in Tabelle 5.3 aufgelistet. In den Simulationsdurchläufen für das hier vorgestellte Capability-Matching wurden beispielsweise 3178 Aufträge erzeugt, von denen 72% erfolgreich bearbeitet wurden. Nachfolgend nun eine nähere Betrachtung der erzielten Ergebnisse. Die Ergebnisse werden aufgeschlüsselt nach den unterschiedlichen Ausfallwahrscheinlichkeiten vorgestellt. Ziel ist es einen Hinweis auf eine gelungene Verschiebung des in Abschnitt 5.2 erläuterten Zusammenhangs zwischen Durchlaufzeit und Auslastung nachzuweisen. *N* ist in den Tabellen der Ergebnisaufstellung die Anzahl der vorliegenden Werte, d. h. der Auftragsagentenergebnisse. Zusätzlich errechnet wird der Mittelwert der erzielten Ergebnisse und die Standardabweichung *s*, die bei den erzielten Ergebnissen gegeben ist. Sämtliche Ergebnisse im Bereich der Auftragsagenten sind statistisch signifikant.

5.3.1. Experimente ohne Ressourcenausfall

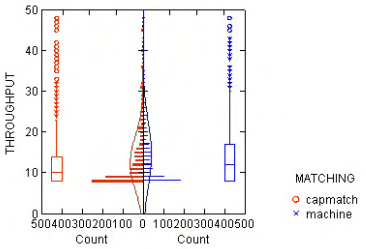
	N	**Mittelwert**	**s**
Machine-Matching	593	14,541	8,182
Skill-Matching	665	14,874	7,542

	N	**Mittelwert**	**s**
Capability-Matching	724	12,370	6,567
Machine-Matching	593	14,541	8,182

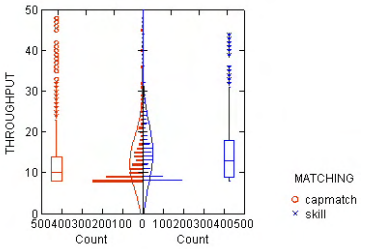
	N	**Mittelwert**	**s**
Capability-Matching	724	12,370	6,567
Skill-Matching	665	14,874	7,542



(a) Maschinenzuordnung gegen Skill-Zuordnung



(b) Maschinenzuordnung gegen Capability-Matching



(c) Skill-Zuordnung gegen Capability-Matching

Abbildung 5.6.: Durchlaufzeiten der Aufträge ohne Ressourcenausfall

Die in Abbildung 5.6 aufgeführten Ergebnisse sind durch die Untersuchung des Verhaltens der Durchlaufzeit der Aufträge ohne Maschinenausfall erzielt worden. Es zeigt sich hierbei eine größere Anzahl erfolgreicher Auftragsabarbeitungen von 724 erfolgreichen Aufträgen

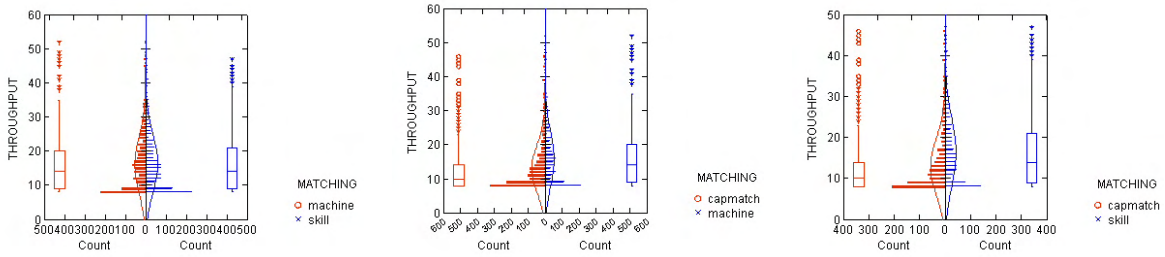
gegenüber 665 erfolgreichen Aufträgen beim Skill-Matching-Verfahren bzw. 593 erfolgreichen Aufträgen bei einer festen Maschinenzuordnung. Bei der Durchlaufzeit der Aufträge ergibt sich ein Vorteil für das Capability-Matching von durchschnittlich 12 Slots gegenüber 15 Slots bei Machine- und Skill-Matching. Außerdem ist eine geringere Standardabweichung der Ergebnisse bei Einsatz der Capability-basierten Ansatzes nachweisbar. Die Abweichung der Ergebnisse beträgt beim erstgenannten Verfahren 6,6. Die anderen Ansätze liegen hier bei 7,5 bzw. 8,2.

5.3.2. Experimente mit Ausfallwahrscheinlichkeit 0,05

	N	**Mittelwert**	**s**
Machine-Matching	546	15,914	8,180
Skill-Matching	639	16,423	8,477

	N	**Mittelwert**	**s**
Capability-Matching	684	12,904	6,893
Machine-Matching	546	15,914	8,180

	N	**Mittelwert**	**s**
Capability-Matching	684	12,904	6,893
Skill-Matching	639	16,423	8,477



(a) Maschinenzuordnung gegen Skill-Zuordnung

(b) Maschinenzuordnung gegen Capability-Matching

(c) Skill-Zuordnung gegen Capability-Matching

Abbildung 5.7.: Durchlaufzeiten der Aufträge bei einer Ausfallwahrscheinlichkeit von 0,05

In der Zusammenstellung der Abbildung 5.7 sind die Ergebnisse der Auftragsagenten bei einem Wert von 0,05 für die Ausfallwahrscheinlichkeit der Ressourcen dargestellt. Es zeigt sich wiederum, dass es mit dem Einsatz des Capability-Matching gelungen ist, eine höhere Anzahl von erfolgreichen Auftragsabarbeitungen durchzuführen. Die Werte sind hier 684 erfolgreiche Aufträge beim Capability-Match gegenüber 639 bzw. 546 beim Skill- und Machine-Matching. Für die durchschnittliche Durchlaufzeit der Aufträge ergibt sich in diesem Szenario 13 Slots beim Capability-Matching gegenüber ca. 16 Slots bei den übrigen Verfahren.

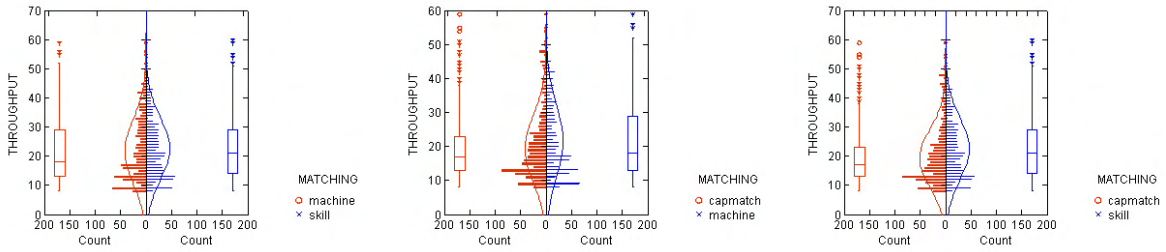
Die Standardabweichung liegt hier bei 6,9 für das Capability-Matching und einem mit 8,5 für das Skill-Matching höheren Wert als bei der direkten Maschinenzuordnung. Dort liegt sie bei 8.2.

5.3.3. Experimente mit Ausfallwahrscheinlichkeit 0,5

	N	**Mittelwert**	**s**
Machine-Matching	458	21,692	10,945
Skill-Matching	501	22,645	10,715

	N	**Mittelwert**	**s**
Capability-Matching	505	19,398	9,838
Machine-Matching	458	21,692	10,945

	N	**Mittelwert**	**s**
Capability-Matching	505	19,398	9,838
Skill-Matching	501	22,645	10,715



(a) Maschinenzuordnung gegen Skill-Zuordnung

(b) Maschinenzuordnung gegen Capability-Matching

(c) Skill-Zuordnung gegen Capability-Matching

Abbildung 5.8.: Durchlaufzeiten der Aufträge bei einer Ausfallwahrscheinlichkeit von 0,5

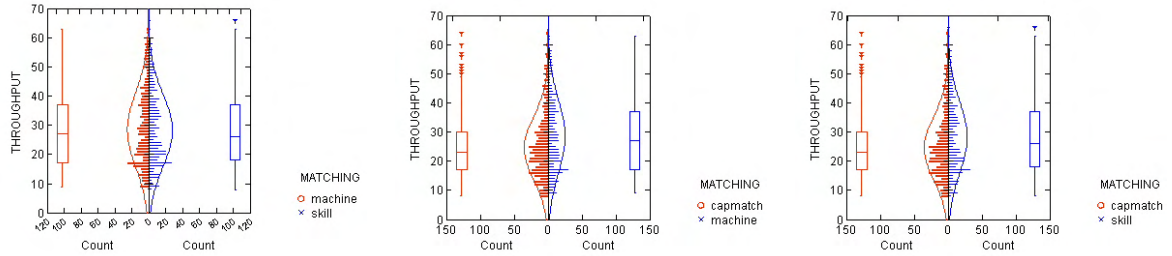
Die Ergebnisse der Abbildung 5.8 wurden durch die Versuchsdurchläufe mit der Ausfallwahrscheinlichkeit 0,5 der Ressourcenagenten ermittelt. Erfolgreich verlaufene Auftragsgenerierungen sind hier in 505 Fällen bei Einsatz des Capability-Matchings erzielt worden. Unter Einsatz des Skill-Matching verliefen 501 Aufträge erfolgreich. Durch das Machine-Matching wurden 458 erfolgreiche Aufträge ermöglicht. Die durchschnittliche Durchlaufzeit lag beim Capability-Matching bei 19 Slots. Bei den übrigen Verfahren lag sie bei 22 Slots. Die Standardabweichungen liegen zwischen 9,8 und 11,0. Insgesamt fällt auf, dass sich bei dieser Ausfallwahrscheinlichtkeit die Werte stärker ähneln, als bei den vorherigen Szenarien. Dies gilt insbesondere für das Machine- und Skill-Matching.

5.3.4. Experimente mit Ausfallwahrscheinlichkeit 1

	N	Mittelwert	s
Machine-Matching	323	28,526	12,800
Skill-Matching	356	28,292	12,539

	N	Mittelwert	s
Capability-Matching	380	24,516	10,780
Machine-Matching	323	28,526	12,800

	N	Mittelwert	s
Capability-Matching	380	24,516	10,780
Skill-Matching	356	28,292	12,539



(a) Maschinenzuordnung gegen Skill-Zuordnung

(b) Maschinenzuordnung gegen Capability-Matching

(c) Skill-Zuordnung gegen Capability-Matching

Abbildung 5.9.: Durchlaufzeiten der Aufträge bei einer Ausfallwahrscheinlichkeit von 1

Für die vierte Ausfallwahrscheinlichkeit mit einem Wert von 1 wurden die in Abbildung 5.9 zusammengestellten Ergebnisse erzielt. Erwartet ist der starke Rückgang der Anzahl der erfolgreichen Auftragsabarbeitungen. Hier beträgt der Wert für das Verfahren des Capability-Matches nur noch 380. Mit Hilfe des Skill-Matches konnten noch 356 Aufträge erfolgreich zugeordnet und abgearbeitet werden. Das Machine-Matching ermöglichte noch 323 erfolgreiche Aufträge. Durch die hohe Ausfallwahrscheinlichkeit steigt auch der Wert der durchschnittlichen Durchlaufzeit. Sie liegt bei einem Wert von 1 für die Ausfallwahrscheinlichkeit der Ressourcen beim Capability-Match bei 25 Slots. Für die anderen Verfahren liegt der Wert bei 28 bzw. 29 Slots. Die Standardabweichung liegt hier bei 10,8, 12,5 und 12,8 für das Capability-, das Skill- sowie das Machine-Matching.

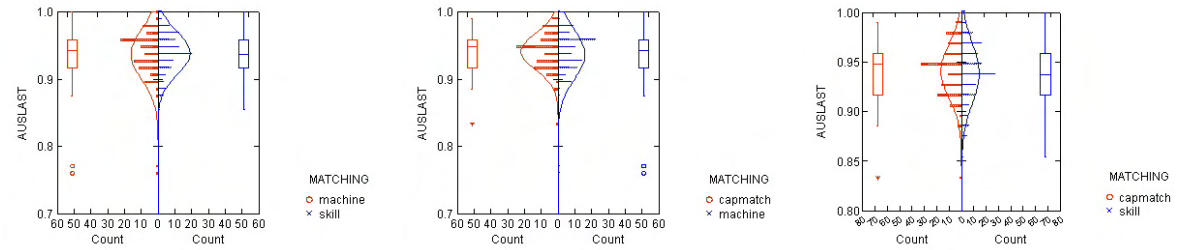
5.3.5. Auslastung des Maschinenparks

In Abbildung 5.10 ist die Auslastung des Maschinenparks bei einer Ressourcenausfallwahrscheinlichkeit von 0,05 aufgeführt. Für die verschiedenen Matching-Verfahren sind jeweils 90

	N	Mittelwert	s
Machine-Matching	90	0,937	0,037
Skill-Matching	90	0,939	0,031

	N	Mittelwert	s
Capability-Matching	90	0,941	0,027
Machine-Matching	90	0,937	0,037

	N	Mittelwert	s
Capability-Matching	90	0,941	0,027
Skill-Matching	90	0,939	0,031



(a) Maschinenzuordnung gegen Skill-Zuordnung

(b) Maschinenzuordnung gegen Capability-Matching

(c) Skill-Zuordnung gegen Capability-Matching

Abbildung 5.10.: Auslastung des Maschinenparks bei einer Ausfallwahrscheinlichkeit von 0,05

Ergebnisse vorhanden[10]. Bei allen Ergebnissen liegt der Wert der durchschnittlichen Auslastung bei ca. 94%. Die Standardabweichung liegt zwischen 0,027 und 0,037, wobei die höhere Abweichung beim Machine-Matching-Verfahren vorliegt. Die Ergebnisse der Ressourcenauswertung sind jedoch statistisch nicht signifikant. Die weitere Interpretation der Ergebnisse erfolgt im nächsten Abschnitt.

5.4. Interpretation der Ergebnisse

In diesem Abschnitt erfolgt die Interpretation der im vorherigen Abschnitt vorgestellten Ergebnisse. In den Ergebniszusammenstellungen der Abbildungen 5.6, 5.7, 5.8 und 5.9 sowie der Statistik über alle abgearbeiteten Aufträge in Tabelle 5.3 lässt sich grundsätzlich eine höhere Anzahl der erfolgreich verlaufenen Aufträge bei Einsatz des Capability-Matchings feststellen. Dies lässt darauf schließen, dass durch das Capability-Matching die Aufträge besser auf die vorhandenen Maschinen verteilt wurden und somit eine höhere Flexibilität in der

[10]Dies ergibt sich aus jeweils 30 Ergebnissen von drei Maschinenagenten.

Auftragszuordnung gegeben war. Dies zeigt sich beispielsweise in einem Auftragsbearbeitungswert von 684 gegenüber 639 bzw. 546 bei einer Ressourcenausfallwahrscheinlichkeit von 0,05. Aber auch das Verfahren des Skill-Matchings bietet hier schon deutliche Vorteile gegenüber einer simplen Maschinenzuordnung. Es zeigt sich weiterhin, dass mit Einsatz des Capability-Matchings die Ergebnisse grundsätzlich weniger Schwankungen zeigen. Dies begründet sich in der über alle Versuche geringeren Standardabweichung der Daten. Es ist also durch den Einsatz des Capability-Matchings möglich, eine gleichbleibend bessere Auftragsverteilung auf die Ressourcen zu erreichen. Die Durchlaufzeit der Aufträge wurde durch das Capability-Matching über alle Versuche gesehen verringert. Dies zeigt weiterhin eine bessere Verfügbarkeit einer zum Auftrag passenden Maschine, was ebenfalls auf die Erreichung einer höheren Flexibilität schließen lässt. Bei der Betrachtung der Ergebnisse für die Auslastung der Ressourcen kann, unter der Beachtung der Tatsache, dass diese Ergebnisse statistisch nicht signifikant sind, festgehalten werden, dass zumindest der Schluss nahe liegt, dass sich keine Verschlechterung der Ressourcenauslastung ergibt. Dies entspricht insoweit ebenfalls dem erwarteten Ergebnis.

Insgesamt gesehen lässt sich feststellen, dass offensichtlich eine Verschiebung des Verhältnisses zwischen Durchlaufzeit und Auslastung erreicht wurde. Bei dieser Verschiebung handelt es sich um den Effekt, der in der Erläuterung des Evaluationsszenarios und in Abbildung 5.3 erläutert und gewünscht wurde. Der Einsatz des Capability-Matchings ermöglichte eine Verschiebung dieses Verhältnisses in Richtung des Ursprungs durch eine Verringerung der Durchlaufzeit und einer gleichzeitig weiterhin hohen Auslastung des Maschinenparks. Durch die durchgeführte Evaluation wurde die Eignung für eine einfache technische Domäne gezeigt. Die Übertragbarkeit des Ansatzes ist durch die Ausdrucksmächtigkeit der eingesetzten Beschreibungslogik gegeben. Diese ermöglicht die Modellierung wesentlich komplexerer Domänen und somit den Einsatz des vorgestellten Verfahrens in weiteren Domänen, in denen eine Flexibilisierung des Agentenmatchmakings nötig und gewünscht ist.

Diese Ergebnisse sind im Rahmen des erstellten Szenarios erwartungsgemäß, da generell Kompensation von Engpässen und Störungen eine Flexibilisierung der Problemverteilung auf die Problemlöser erfordert. Die Grenzen des erstellten Ansatzes lassen sich demnach in Szenarien finden, in denen diese Eigenschaften nicht vorhanden sind. Hier sind einerseits Szenarien zu nennen, in denen keine Engpässe in der Zuweisung von Problemen zu Problemlösern auftreten und diese auch nicht gestört sein können. Andererseits ist natürlich auch eine dementsprechend große Anzahl von Problemlösern denkbar, die dadurch, dass sich immer ein anderer Problemlöser finden lässt, das Auftreten von Engpässen verhindert oder auftretende Störungen kompensieren kann. In beiden Fällen kann vermutet werden, dass beim Einsatz der untersuchten Matchingverfahren keine großen Unterschiede in Verteilung ergeben und das Capability-Matching hier nicht in der Lage wäre, eine Verbesserung der bestehenden Situation zu bewirken.

Kapitel 6.

Zusammenfassung und Ausblick

In dieser Arbeit erfolgte die Ausarbeitung eines Ansatzes zur flexiblen Zuordnung von Agenten, die der Lösung eines bestimmten Problems bedürfen zu Agenten die Problemlösungen anbieten. Die Arten der Problemlösungen, die diese problemlösenden Agenten zur Verfügung stellen, wurde als Menge ihrer *Capabilities* eingeführt. Im Rahmen der Untersuchung des aktuellen Stands der Forschung erfolgte die Untersuchung verschiedener Ansätze, die eine Lösung des Matchmaking-Problems versprechen. Die Wahl fiel nach einer Bewertung dieser Ansätze auf einen Ansatz aus dem Bereich der Semantic Web Services. Dieser Ansatz schlägt die ontologische Repräsentation von Problemen und Problemlösungen durch eine Beschreibungslogik vor. Dieser Vorschlag ermöglicht den Einsatz eines der vorhandenen Inferenzverfahren zur Bestimmung eines Matches über Inferenz auf dieser ontologischen Repräsentation. Der Ansatz schlägt weiterhin die Einordnung von Problemen und Problemlösungen unterhalb eines vorhandenen Oberkonzeptes vor, um dann die Feststellung eines Matches über die Berechnung von Subsumptionsbeziehungen zwischen diesen zu ermöglichen.

Aufbauend auf diesem Ansatz erfolgte dann die Ausarbeitung eines eigenen Ansatzes zur flexiblen Zuordnung von Problemen und Problemlösungen im Rahmen von Multiagentensystemen. Dazu erfolgte zunächst die Formalisierung der Konzepte *Feature* und *Capability* als Konzepte einer Beschreibungslogik sowie die formale Definition eines Matches. Anschließend wurde die Einbindung dieses Ansatzes in die bestehende Agentenarchitektur des *Diskursagenten* vorgenommen.

Der erarbeitete Ansatz wurde außerdem in einen bestehenden Software-Prototypen integriert, um die Evaluation des Ansatzes zu ermöglichen. Dabei kam die Ontologiesprache *OWL* zum Einsatz, deren Anbindung bzw. Verarbeitung über das Softwareframework *Jena* ermöglicht wurde. Mit Hilfe des OWL-Reasoners *Pellet* erfolgte dann die Inferenz über die Konzepte der erzeugten Ontologie. Die Evaluation erfolgte über die Erarbeitung eines konkreten Szenarios im Rahmen der Domäne Fertigungslogistik. Als Ergebnis der Evaluation muss festgehalten werden, dass das Ziel der Flexibilisierung des Matchmakings zwischen Problemlösungen und Problemen durch den aufgestellten Ansatz erreicht wurde.

Als Ausblick ließe sich die weitere Integration der Grade des Matches mit der aktuellen Situation in der Planung des Agenten vornehmen. Bestimmte Zustände der Planung könnten als Einflüsse auf das Annahmeverhalten des Agenten im Bezug auf das Vorliegen eines bestimmten Typ des Matches definiert werden. Interessant könnte außerdem eine weitergehende Evaluation der Vor- bzw. Nachteile des Ansatzes im Rahmen eines wesentlich komplexeren und realen Szenarios sein.

Literaturverzeichnis

Baader und Nutt 2003
BAADER, F. ; NUTT, W.: Basic description logics. In: D. CALVANESE, F. B. amd (Hrsg.) ; MCGUINNESS, D. (Hrsg.) ; NARDI, D. (Hrsg.) ; PATEL-SCHNEIDER, P. F. (Hrsg.): *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003

Bechhofer 2003
BECHHOFER, S.: The DIG Description Logic Interface: DIG/1.1 / University of Manchester. 2003. – Forschungsbericht

Bechhofer u. a. 2003
BECHHOFER, S. ; LORD, P. ; VOLZ, R: Cooking the Semantic Web with the OWL API. In: *2nd International Semantic Web Conference, ISWC*. Sanibel Island, Florida, October 2003

Berners-Lee u. a. 2001
BERNERS-LEE, T. ; HENDLER, J. ; LASSILA, O.: The Semantic Web – A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. In: *Scientific American* 5 (2001), Nr. 1, S. 35–43

Blanchard 1992
BLANCHARD, B. S.: *Logistics Engineering and Management*. 4. Auflage. New Jersey : Prentice Hall, 1992

Brachman und Schmolze 1985
BRACHMAN, R. J. ; SCHMOLZE, J.: An Overview of the KL-ONE Knowledge Representation System. In: *Cognitive Science* 9 (1985), Nr. 2, S. 171–216

Bratman 1987
BRATMAN, M. E.: *Intentions, Plans, and Pratical Reason*. Cambrigde, Massachusetts : Harvard University Press, 1987

Bratman u. a. 1988
BRATMAN, M. E. ; ISRAEL, D. J. ; POLLAK, M. E.: Plans and resource-bounded practical reasoning. In: *Computational Intelligence* 4 (1988), S. 349–355

Burkhard 2003
BURKHARD, H.-D.: Software-Agenten. In: GÖRZ, G. (Hrsg.) ; ROLLINGER, C.-R. (Hrsg.) ; SCHNEEBERGER, J. (Hrsg.): *Handbuch der Künstlichen Intelligenz*. 4. Auflage. München : Oldenbourg Verlag, 2003, Kap. 24, S. 943–1020

Chinnici u. a. 2006
CHINNICI, R. ; MOREAU, J.-J. ; RYMAN, A. ; WEERAWARANA, S.: Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language / World Wide Web Consortium. März 2006. – Forschungsbericht

Connolly u. a. 2001
CONNOLLY, D. ; HARMELEN, F. van ; HORROCKS, I. ; MCGUINNESS, D. L. ; PATEL-SCHNEIDER, P. F. ; STEIN, L. A.: DAML+OIL (March 2001) Reference Description / World Wide Web Consortium. URL `http://www.w3.org/TR/daml+oil-reference`, März 2001. – Forschungsbericht

DIN 8580 2003
Norm, 2003. *Fertigungsverfahren - Begriffe, Einteilung*

DIN 8589-0 2003
Norm, 2003. *Fertigungsverfahren Spanen - Teil 0: Allgemeines; Einordnung, Unterteilung, Begriffe*

Edelkamp und Hoffmann 2004
EDELKAMP, S. ; HOFFMANN, J.: PDDL 2.2: The Language for the Classical Part of the 4th International Planning Competition / Universität Dortmund. 2004. – Forschungsbericht

Eversheim 2002
EVERSHEIM, W.: *Organisation in der Produktionstechnik 3. Arbeitsvorbereitung*. Berlin : Springer Verlag, 2002

Fensel u. a. 2000
FENSEL, D. ; HORROCKS, I. ; HARMELEN, F. V. ; DECKER, S. ; ERDMANN, M. ; KLEIN, M.: OIL in a nutshell. In: DIENG, R. (Hrsg.): *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling, and Management (EKAW 2000)*, Springer-Verlag, October 2000 (Lecture Notes in Artificial Intelligence, LNAI 1937), S. 1–16

Fikes u. a. 2003a
FIKES, R. ; HAYES, P. ; HORROCKS, I.: OWL-QL – A Language for Deductive Query Answering on the Semantic Web / Knowledge Systems Laboratory, Stanford University,. Stanford, CA, 2003. – Forschungsbericht. – URL `http://ksl.stanford.edu/KSL_Abstracts/KSL-03-14.html`

Fikes u. a. 2003b
FIKES, R. ; JENKINS, J. ; FRANK, G.: JTP: A System Architecture and Component Library for Hybrid Reasoning / Knowledge Systems Laboratory, Stanford University. Stanford, CA, 2003. – Forschungsbericht. – URL `http://ksl.stanford.edu/KSL_Abstracts/KSL-03-01.html`

Fikes und Nilsson 1971
FIKES, R. ; NILSSON, N.: STRIPS: a new approach to the application of theorem proving to problem solving. In: *Artificial Intelligence* 2 (1971), S. 189–208

FIPA 2002a
FIPA – FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS (Hrsg.): *FIPA Contract Net Interaction Protocol Specification*. 2002. – URL `http://fipa.org/specs/fipa00029/SC00029H.html`

FIPA 2002b
FIPA – FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS (Hrsg.): *FIPA Specification*. 2002. – URL `http://www.fipa.org/specifications/index.html`

Glaser u. a. 1992
GLASER, H. ; GEIGER, W. ; ROHDE, V.: *PPS: Produktionsplanung und -steuerung*. 2. Auflage. Wiesbaden : Gabler Verlag, 1992

Gómez-Pérez u. a. 2004
GÓMEZ-PÉREZ, A. ; FERNÁNDEZ-LÓPEZ, M. ; CORCHO, O.: *Ontological Engineering*. London : Springer Verlag, 2004

Gruber 1993
GRUBER, T. R.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In: *Knowledge Acquisition* 5 (1993), Nr. 2, S. 199–220

Gruninger und Lee 2002
GRUNINGER, M. ; LEE, J.: Ontology - applications and design. In: *Communications of the ACM* 45 (2002), Nr. 2, S. 39–41

Haarslev und Möller 2001
HAARSLEV, V. ; MÖLLER, R.: RACER system description. In: *Proceedings of the International Joint Conference on Automated Reasoning, IJCAR'2001, June 18-23, 2001, Siena, Italy, LNCS*. Berlin : Springer Verlag, Juni 2001

Horridge u. a. 2004
HORRIDGE, M. ; KNUBLAUCH, H. ; RECTOR, A. ; STEVENS, R. ; WROE, C.: A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools / University of Manchester. Manchester, August 2004. – Forschungsbericht. – URL `http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf`

Horrocks 1998
HORROCKS, I.: Using an Expressive Description Logic: FaCT or Fiction? In: COHN, A. G. (Hrsg.) ; SCHUBERT, L. (Hrsg.) ; SHAPIRO, S. C. (Hrsg.): *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR'98)*. San Francisco, CA : Morgan Kaufmann Publishers, Juni 1998, S. 636–647

ISO 14649-10 2004
Norm, 2004. *ISO 14649: Industrial automation systems and integration – Physical device control – Data model for computerized numerical controllers – Part 10: General process data*

Li und Horrocks 2004
LI, L. ; HORROCKS, I.: A Software Framework for Matchmaking Based on Semantic Web Technology. In: *Int. J. of Electronic Commerce* 8 (2004), Nr. 4, S. 39–60

Maedche und Motik 2003
MAEDCHE, A. ; MOTIK, B.: Repräsentations- und Anfragesprachen für Ontologien – eine Übersicht. In: *Datenbank-Spektrum* 6 (2003), S. 43–53

Mamdani 1998
MAMDANI, A.: The Social Impact of Software Agents. In: *Proceedings of the Workshop on The Impact of Agents on Communications and Ethics: What do and don't we know?* Dublin, July 1998

Manola und Miller 2004
MANOLA, F. ; MILLER, E.: RDF Primer / World Wide Web Consortium. URL `http://www.w3.org/TR/rdf-primer`, Februar 2004. – Forschungsbericht

Martin u. a. 2004
MARTIN, D. ; PAOLUCCI, M. ; MCILRAITH, S. ; BURSTEIN, M. ; MCDERMOTT, D. ; MCGUINNESS, D. ; PARSIA, B. ; PAYNE, T. ; SABOU, M. ; SOLANKI, M. ; SRINIVASAN, N. ; SYCARA, K.: Bringing Semantics to Web Services: The OWL-S Approach. In: *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*. San Diego, California, USA, July 2004

McBride 2002
MCBRIDE, B.: JENA: A Semantic Web toolkit. In: *IEEE Internet Computing* 6 (2002), Nov-Dec, S. 55–59

Nyhuis und Wiendal 1999
NYHUIS, P. ; WIENDAL, H.-P.: *Logistische Kennlinien*. Berlin : Springer Verlag, 1999

Odell u. a. 2001
ODELL, J. ; PARUNAK, H. Van D. ; BAUER, B.: Representing Agent Interaction Protocols in UML. In: CIANCARINI, P. (Hrsg.) ; WOOLDRIDGE, M. (Hrsg.): *Agent-Oriented Software Engineering*. Berlin : Springer, 2001, S. 121–140. – URL `http://www.fipa.org/docs/input/f-in-00077/`

Pednault 1986
PEDNAULT, E. P. D.: Formulating multiagent, dynamic-world problems in the classical planning framework. In: GEORGEFF, M. P. (Hrsg.) ; LANSKY, A. L. (Hrsg.): *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*. Timberlinge, Oregon : Morgan Kaufmann, 1986, S. 47–82

Prud'hommeaux und Seaborne 2006
PRUD'HOMMEAUX, E. ; SEABORNE, A.: SPARQL Query Language for RDF / World Wide Web Consortium. URL `http://www.w3.org/TR/2004/REC-owl-guide-20040210/`, Februar 2006. – Forschungsbericht

Ranze u. a. 2002
RANZE, K. C. ; SCHOLZ, T. ; WAGNER, T. ; GÜNTER, A. ; HERZOG, O. ; HOLLMANN, O. ; SCHLIEDER, C. ; ARLT, V.: A Structure Based Configuration Tool: Drive Solution Designer - DSD. In: *AAAI/IAAI*, 2002, S. 845–852

Rosenfeld 1994
ROSENFELD, R.: *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. Pittsburgh, PA, Carnegie Mellon University, Dissertation, 1994

Russel und Norvig 2003
RUSSEL, S. ; NORVIG, P.: *Artificial Intelligence - A Modern Approach*. 2nd. Prentice Hall, 2003

Salton 1989
SALTON, G.: *Automatic Text Processing*. Reading, MA : Addison-Wesley, 1989

Schmidt-Schauß und Smolka 1991
SCHMIDT-SCHAUSS, M. ; SMOLKA, G.: Attributive concept descriptions with complements. In: *Artificial Intelligence* 48 (1991), Nr. 1, S. 1–26

Schöning 2000
SCHÖNING, U.: *Logik für Informatiker*. 5. Heidelberg/Berlin : Spektrum Akademischer Verlag, 2000

Seaborne 2004
SEABORNE, A.: RDQL – A Query Language for RDF / World Wide Web Consortium. Januar 2004. – Forschungsbericht

Shvaiko und Euzenat 2005
SHVAIKO, P. ; EUZENAT, J.: A Survey of Schema-Based Matching Approaches. In: *Journal on Data Semantics* 4 (2005), S. 146–171

Sirin und Parsia 2004
SIRIN, E. ; PARSIA, B.: Pellet: An OWL DL Reasoner. In: *Description Logics*, 2004

Smith u. a. 2004
SMITH, M. K. ; WELTY, C. ; MCGUINNESS, D. L.: OWL Web Ontology Language Guide / World Wide Web Consortium. URL `http://www.w3.org/TR/owl-guide`, Februar 2004. – Forschungsbericht

Ströbel und Stolze 2001
STRÖBEL, M. ; STOLZE, M.: A matchmaking component for the discovery of agreement and negotiation spaces in electronic markets. In: *Proceedings of the Group Decision and ANegotiation Conference, La Rochelle, France*, 2001, S. 61–75

Sycara u. a. 1998
SYCARA, K. ; LU, J. ; KLUSCH, M.: Interoperability among Heterogeneous Software Agents on the Internet / Carnegie Mellon University, PA (USA). URL `citeseer.ist.psu.edu/article/sycara98interoperability.html`, 1998 (CMU-RI-TR-98-22). – Forschungsbericht

Sycara u. a. 2002
SYCARA, K. ; WIDOFF, S. ; KLUSCH, M. ; LU, J.: Larks: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace. In: *Autonomous Agents and Multi-Agent Systems* 5 (2002), Nr. 2, S. 173–203. – ISSN 1387-2532

Timm 2004
TIMM, I. J.: *Dynamisches Konfliktmanagement als Verhaltenssteuerung Intelligenter Agenten*. Berlin : Akademische Verlagsgesellschaft Aka GmbH, 2004

Timm und Woelk 2003
TIMM, I. J. ; WOELK, P.-O.: Ontology-based Capability Management for Distributed Problem Solving in the Manufacturing Domain. In: SCHILLO, M. (Hrsg.) u. a.: *Proceedings of the first German Conference on Multi-Agent System Technologies (MATES 2003), Erfurt. LNAI 2831*. Berlin : Springer Verlag, 2003, S. 168–179

Tönshoff u. a. 2002
TÖNSHOFF, H. K. ; HERZOG, O. ; TIMM, I. J. ; WOELK, P.-O.: Integrated Process Planning and Production Control Based on the Application of Intelligent Agents. In: TETI, R. (Hrsg.) u. a.: *Proceedings of the 3rd CIRP International Seminar on Intelligent Computation in Manufacturing Engineering - ICME 2002, 3-5 July, Ischia, Italy*, 2002, S. 135–140

Veit u. a. 2002
VEIT, D. ; MÜLLER, J. P. ; WEINHARDT, C.: Multidimensional Matchmaking for Electronic Markets. In: *Journal of Applied Artificial Intelligence* 16(9-10) (2002), S. 853–869

Walmsley 2004
WALMSLEY, P.: XML Schema Part 0: Primer Second Edition / World Wide Web Consortium. URL `http://www.w3.org/TR/xmlschema-0`, Oktober 2004. – Forschungsbericht

Weiß 1999
WEISS, G. (Hrsg.): *Multiagent Systems*. Cambridge, Massachusetts : The MIT Press, 1999

Weiß und Jakob 2005
WEISS, G. ; JAKOB, R.: *Agentenorientierte Softwareentwicklung*. Berlin : Springer Verlag, 2005

Werres 2002
WERRES, Y.: *Effizientes und flexibles Pattern-Matching für komplexe Objektstrukturen*, Universität Bremen, Diplomarbeit, 2002

Wooldridge und Lomuscio 2001
WOOLDRIDGE, M. ; LOMUSCIO, A.: A Computationally Grounded Logic of Visibility, Perception and Knowledge. In: *Logic Journal of the IGPL* 9 (2001), Nr. 2, S. 257–272

Wooldridge 2002
WOOLDRIDGE, M. J.: *An Introduction to MultiAgent Systems*. Chichester, England : John Wiley & Sons Ltd, 2002

Wooldridge und Jennings 1995
WOOLDRIDGE, M. J. ; JENNINGS, N. R.: Intelligent Agents: Theory and Practice. In: *The Knowledge Engineering Review* 10 (1995), Nr. 2, S. 115–152

Wooldrigde 2000
WOOLDRIGDE, M. J.: *Reasoning about Rational Agents*. Cambrigde, Massachusetts : The MIT Press, 2000