

Entwicklung intelligenter Multi-Multiagentensysteme – Werkzeugunterstützung, Lösungen und offene Fragen

Karl-Heinz Krempels¹, Jens Nimis², Lars Braubach³, Alexander Pokahr³,
Rainer Herrler⁵, Thorsten Scholz⁶

¹ Lehrstuhl für Informatik IV, RWTH Aachen Aachen, D-52074 Aachen
krempels@informatik.rwth-aachen.de

² IPD, Universität Karlsruhe (TH), D-76128 Karlsruhe
nimis@ipd.uni-karlsruhe.de

³ Arbeitsbereich VSIS, Universität Hamburg, D-22527 Hamburg
{braubach|pokahr}@informatik.uni-hamburg.de

⁴ Lehrstuhl für Informatik VI, Universität Würzburg, D-97074 Würzburg
herrler@informatik.uni-wuerzburg.de

⁵ Technologie-Zentrum Informatik, Universität Bremen, 28359 Bremen
scholz@tzi.de

Abstract: In vielen Disziplinen werden für existierende Probleme neue Lösungsansätze verfolgt, die auf Agenten und Agentensystemen beruhen. Dies führte zu sehr unterschiedlichen Agenten, Agentensystemen und Agentenentwicklungsumgebungen. Der Entwicklungsprozess solcher Systeme wurde häufig jedoch nur unzureichend berücksichtigt. Erfolgreiche Standardisierungen im Bereich von Architekturen und Interoperabilität [Fi03] begünstigen die Erstellung kompatibler Systeme sowie von Multi-Multiagentensystemen (MMAS). Wir berichten hier die Erfahrungen aus dem Entwurfsprozess von zwei realen MMAS, wobei auch auf die Kriterien für die Design-Entscheidungen eingegangen wird. Weiter werden die Werkzeugunterstützung bei Modellierung und Implementierung in den verschiedenen Phasen des Entwurfsprozesses sowie mögliche Lösungsansätze für noch offene Fragen betrachtet. Während des Entwurfsprozesses verwendete Werkzeuge und hierfür erstellte Erweiterungen werden beschrieben und konzeptionell verglichen.

1 Einleitung

Das Agenten-Paradigma verspricht die Beherrschung komplexer Anwendungsgebiete, in denen herkömmliche Software-Systeme an ihre Grenzen stoßen. Dies wird belegt durch zahlreiche Simulationen und Fallstudien. Dennoch ist die Zahl der industriell eingesetzten Multiagentensysteme (MAS) überschaubar, ihr betrieblicher Einsatz bleibt die Ausnahme [Je01]. Das SPP 1083 „Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien“ hat sich deshalb die Aufgabe gestellt, die Eignung der Agententechnologie für den industriellen Einsatz zu überprüfen und weiter zu entwickeln.

Notwendige Voraussetzung für den erfolgreichen praktischen Einsatz einer in der Theorie ausreichend diskutierten und untersuchten Technologie ist die Verfügbarkeit weitrei-

chend akzeptierter Unterstützung auf technischer Ebene durch Entwicklungsmethoden und -werkzeuge. Durch den FIPA-Standard für MAS [Fi03], an dessen Entwicklung das SPP 1083 aktiv mitwirkt, und die zugehörigen konformen Plattformen ist eine erste technische Grundlage geschaffen. Zahlreiche Arbeiten auf dem Gebiet der Methodik und vereinzelt Entwicklungswerkzeuge schaffen weitere Voraussetzungen für die Realisierung von Systemen mit überschaubarer Größe. Weitgehend offen sind jedoch Fragen zur Einsatzbereitschaft der Agententechnologie. Besonders sobald Aussagen über den Einsatz in Anwendungen von realer Größe und Komplexität getroffen werden sollen. Folglich sind genau solche Systeme für das SPP 1083 von besonderem Interesse.

Mit *Agent.Hospital* [Ki03] und *Agent.Enterprise* [Fr03] wurden im SPP zwei große und komplexe Multi-Multiagentensysteme (MMAS) erstellt, die für eine Bewertung in Zukunft herangezogen werden können. Besonders positiv wirkte sich dabei der Aufbau des SPP 1083 aus so genannten Tandems mit je einem Domänen- und einem Technikexperten aus. Diese Zusammensetzung ermöglicht einerseits den gewünschten Bezug zu realen Gegebenheiten und führt andererseits zu technisch anspruchsvollen Lösungen.

Bei der Erstellung der beiden MMAS wurde in jedem Tandem zunächst ein prototypisches MAS mit dem Hauptaugenmerk auf den projektspezifischen Fragestellungen entwickelt¹. In einer zweiten Phase konnten durch die Mitarbeit aller Tandems an der Arbeitsgruppe „Agententechnologie“ des SPP diese zu zwei MMAS integriert werden. Die dabei erzielten Fortschritte, Erfahrungen und technischen Lösungen sowie die aufgeworfenen offenen Fragestellungen sind der Gegenstand dieses Berichts. In Abschnitt 2 wird der Stand der Technik der technischen Unterstützung, der Methodik und der Werkzeugunterstützung bei der MAS-Entwicklung dargestellt. Abschnitt 3 beschreibt den Entwurf der beiden MMAS besonders im Hinblick auf ausgewählte getroffene Entwurfsentscheidungen und Vorgehensweisen. Abschnitt 4 geht detailliert auf ausgewählte Probleme und Lösungsansätze ein. Abschnitt 5 enthält Zusammenfassung und Ausblick.

2 Stand der Technik

Die Entwicklung von intelligenten Agenten und Agentensystemen ist ein Prozess, der alle Schritte von der Analyse einer Domäne über das Design spezifischer Ontologien und Problemlösungsmethoden bis zur Implementierung abdeckt. Dieser Entwicklungsprozess wird teilweise von wenig aufeinander abgestimmten Werkzeugen unterstützt. Da die Werkzeuge auch keine standardisierten Schnittstellen haben, wird mitunter die mehrfache Analyse, Modellierung und Implementierung von Teilprozessen erzwungen. Im Folgenden wird auf existierende Infrastrukturen für Agentensysteme, ihre Typen sowie auf etablierte Entwurfs- und Entwicklungsmethoden eingegangen. Des Weiteren wird der aktuelle Stand des Entwicklungsprozesses für agentenbasierte Lösungen beschrieben. Seine Schwachstellen werden aufgezeigt und Ansätze für Verbesserungen aufgeführt.

¹ Eine projektübergreifende Sammlung der zahlreichen Veröffentlichungen zu Einzelergebnissen aus dem SPP, u.a. im Bereich der Agententechnologie, ist auf der RealAgentS Plattform verfügbar. <http://www.realagents.org>

2.1 Infrastrukturen für Agentensysteme

Die Implementierung von MAS stützt sich in hohem Maße auf die Verwendung von MAS Plattformen und Entwicklungswerkzeugen. MAS Plattformen bieten grundlegende Dienste für Agenten an. Dazu gehören das Versenden von Nachrichten und die Suche nach anderen Agenten beziehungsweise deren Diensten. Die Werkzeuge sind einerseits dafür gedacht, den Übergang von der Entwurfs- in die Implementierungsphase zu erleichtern, andererseits unterstützen sie den Anwendungsentwickler bei der Fehlersuche. Für die Realisierung eines MMAS auf der Basis heterogener kooperierender MAS-Diensterebringern entstehen Anforderungen, die über das hinausgehen, was für die Implementierung von einzelnen MAS erforderlich ist. Essentiell sind insbesondere die Interoperabilität zwischen verschiedenen Multiagentenplattformen und die Nutzbarmachung einer gemeinsamen Netzwerkinfrastruktur. Im Folgenden wird der aktuelle Stand der Technik für Plattformen, Interoperabilität und Infrastruktur näher betrachtet.

Gegenwärtig existieren über 40 verschiedene Multiagentenplattformen [Ma02], vom Ein-Mann Open Source Projekt bis zu kommerziellen Plattformen mit IDE Unterstützung. Wir unterscheiden zwei Kategorien: Die erste fokussiert die Kommunikations- und Plattformstandards nach FIPA [Fi03], z.B. JADE [BPR01], FIPA-OS [Em01] und ADK [Tr02]. Die zweite unterstützt agenteninterne Strukturen durch mentalistische Konzepte, z.B. JACK [Bu99], JAM [Hu99], AgentBuilder [Re00] und 3APL Interpreter [Hi99]. Letztere beruhen auf unterschiedlichen Agentensprachen, da kein Konsens besteht, welche mentalistischen Begrifflichkeiten am besten für die Beschreibung dieser Strukturen geeignet sind. Die bekanntesten Vertreter sind AgentSpeak(L) [Ra96], welches auf dem BDI-Konzept [Br87] basiert, Agent-0 [Sh93] und 3APL [Hi99]. Für die Kommunikation setzen diese Systeme in den meisten Fällen effiziente aber proprietäre Mechanismen ein. Vertreter der ersten Kategorie sind wegen ihrer FIPA-Konformität und ihrer besseren Unterstützung der Inter-Plattform-Kommunikation für die Implementierung von MMAS von besonderer Bedeutung. So steht mit der FIPA-ACL (FIPA-Agent Communication Language) eine der KQML vergleichbare Agentenkommunikationssprache mit vollständiger formaler Spezifikation zur Verfügung [LFP99]. Sie basiert auf der Sprechakttheorie [Se69] und setzt eine Drei-Schichten Architektur um, die Inhalt, Bedeutung und Sprechakt einer Nachricht trennt. Andere FIPA-Spezifikationen dienen der Standardisierung von Plattformdiensten, wie z.B. dem Agent Management und dem Directory Facilitator [Fi02a]. Da Standardisierung für das SPP 1083 von zentraler Bedeutung ist, wirkt das SPP aktiv an der Standardisierungsarbeit der FIPA mit.

Eine globale FIPA-konforme Infrastruktur für MMAS ist durch die frei zugänglichen Agentcities-Dienste² vorhanden. Das Agentcities-Projekt stellt eine verteilte Testumgebung für Agententechnologie und Dienstkomposition bereit. Vision ist es, eine dynamische und intelligente Komposition auf der Basis von semantischer Interoperabilität zu ermöglichen. Grundlage bildet ein weltweit verteiltes Netzwerk aus verschiedenen FIPA-konformen Plattformen, die über spezielle Agentcities Dienste angesprochen werden. Die registrierten Plattformen, Agenten und Services sind über einen Suchdienst auffindbar und werden in regelmäßigen Abständen auf ihre Verfügbarkeit überprüft.

² Agentcities Namens- und Überwachungsdienste sind verfügbar unter <http://www.agentcities.net/>.

2.2 MAS-Modellierungsmethoden

Neben der technischen Unterstützung durch Plattformen und Infrastrukturdienste sind anerkannte Entwicklungsmethoden eine zwingende Voraussetzung für die industrielle Akzeptanz von MAS. Vor diesem Hintergrund wurde in den vergangenen Jahren eine Reihe von Vorschlägen für solche Entwicklungsmethoden (z.B. Gaia [WJK00], MASE[WD01], Tropos [GMP02], PASSI [CP02], MASSIVE [Li01]) und Darstellungsformate (AUML [OPB01]) veröffentlicht. Sie unterstützen jeweils eine oder mehrere der Entwicklungsphasen (Analyse, Entwurf, Implementierung und Einführung) und variieren im Formalisierungsgrad ihrer Darstellung und in den semantischen Grundlagen ([HG02] und [IGG99] geben einen Überblick über (teilweise ältere) Methoden).

Gaia [WJK00] ist einer der bekanntesten Ansätze für agentenorientierte Analyse und Entwurf. Der Entwicklungsprozess beginnt mit der Anforderungsanalyse und liefert als Ergebnis einen über eine Reihe verschiedener Modelle verfeinerten Entwurf als Basis für die Implementierung. Der *Process for Agent Societies Specification and Implementation (PASSI)* [CP02] erstreckt sich von der frühen Anforderungsanalyse bis zur Implementierung. Dabei wird meist UML verwendet, es werden Entwurfstechniken aus der Softwaretechnik und der Künstlichen Intelligenz zusammengeführt und durch ein Zusatzmodul für das Modellierungswerkzeug *Rational Rose* unterstützt. Das *Multi-Agent SystemS Iterative View Engineering (MASSIVE)* [Li01] dagegen präferiert eine Modellbasierte Vorgehensweise. In Form von Sichten werden verschiedene, iterativ zu verfeinernde Modelle für das MAS angelegt. Ein Schwerpunkt liegt dabei auf der Wiederverwendung von Entwurfsfragmenten durch den Aufbau einer Sammlung von Entwurfsmustern.

	Gaia	PASSI	MASSIVE
Funktionalität (Use Cases)	- (als Eingabe)	Domain Description	Task View
Rollen / Agenten	Roles Model / Agent Model	Agent and Roles Identification / Role Description	Role View
Aufgaben	Roles Model (activities) / Services Model	Task Specification	Task View
Interaktion	Interactions Model / Services Model	Protocols Description	Interaction View
Domänenwissen	-	Ontology Description	-
Macro-Ebene	(noch nicht)	(Role Description)	Society View
Umgebung	Roles Model (rights, permissions and responsibilities)	-	Environment View
Architektur / Implementierung	-	Agent Implementation and Code Model	Architectural View
Betrieb	Acquaintance Model	Deployment Model	System View

Tabelle 1: Eigenschaften ausgewählter MAS-Entwicklungsmethoden

Tabelle 1 spiegelt die zentralen Konzepte für MAS wider: Agenten erbringen eine bestimmte Funktionalität, indem sie Aufgaben und die zugehörigen Rollen übernehmen. Durch Interaktion bilden sie eine Gesellschaft (MAS), in der sie sich mit Termen aus ihrem Domänenwissen austauschen. Bei den beschriebenen und den meisten anderen Methoden liegt der Focus auf der Erstellung von einzelnen, meist geschlossenen MAS. Keine der Methoden unterstützt dagegen explizit die Entwicklung von MMAS (vgl. 3.2)

2.3 Werkzeugunterstützung im übergreifenden Entwicklungszyklus

Die unterschiedlichen Anforderungen an Modellierungswerkzeuge bezüglich visueller Modellrepräsentation und Einsatzbereiche haben dazu geführt, dass in den meisten Domänen unterschiedliche Modellierungswerkzeuge mit meist auch unterschiedlichen Repräsentationssprachen eingesetzt werden. Der Export der Modelle zur Verwendung in anderen Werkzeugen (wie zum Beispiel zum Design von Ontologien, zur Entwicklung von Lösungsmethoden oder in Entwicklungsumgebungen für Programmierer) wird nur unzureichend unterstützt. Im Folgenden wird auf die Unterstützung des Modellierungsprozesses im Bereich der Domäne, im Ontologie- und Problemlösungs-Design sowie in der Entwicklung von Agenten und Agentensystemen eingegangen.

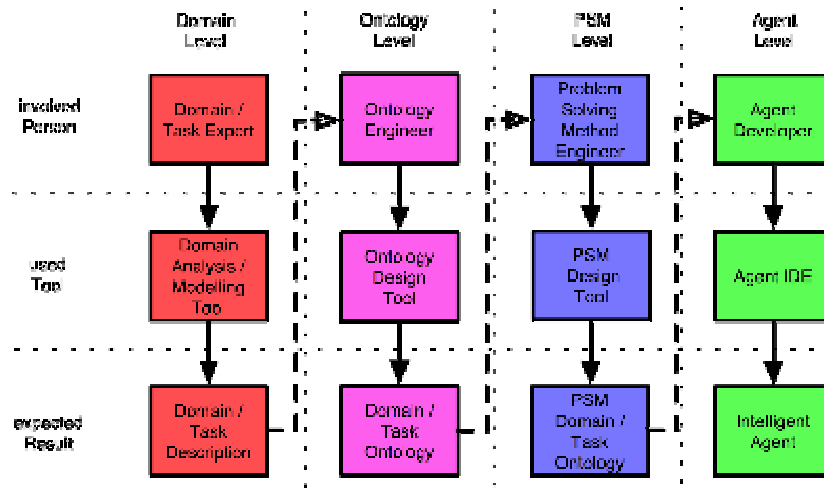


Abbildung 1: Vom Problem zum Agentensystem

Ebene der Domäne(n)

Auf Ebene der Domäne werden Modelle aufgrund von Experteninterviews, Analyse von Prozessabläufen oder statistischen Auswertungen erstellt. Breite Verwendung finden hier die Werkzeuge ARIS-Toolset³ und Visio⁴. ARIS-Toolset unterstützt unter anderem die Modellierung, Optimierung sowie Simulation von Prozessen und bietet zusätzlich die Unterstützung, einen optimierten Prozess in eine fertige Anwendung umzusetzen. Zur Prozessmodellierung werden Ereignisgesteuerte Prozessketten (EPK) verwendet, wobei der Import und Export nach UML⁵ unterstützt wird. Die Modellierung mit ARIS-Toolset wird von einem oder mehreren Domänenexperten vorgenommen. Das Ergebnis der Modellierung ist ein in ARIS-Toolset und in ausgewählten anderen Applikationen (z.B. eM-Plant zur Prozesssimulation) verwertbares Prozessmodell, das semantische Hintergrundinformationen enthält. Visio ist ein reines Modellierungswerkzeug und bietet keine Mög-

³ IDS Scheer GmbH, <http://www.ids-scheer.de/>.

⁴ Microsoft Inc., <http://www.microsoft.com/office/visio/>.

⁵ Unified Modelling Language, <http://www.uml.org/>.

lichkeit die Modelle zu simulieren. Es wird meistens für die schnelle Erstellung grafischer Repräsentationen gering komplexer Modellen verwendet. Zur Prozessmodellierung können EPK oder UML verwendet werden. Zusätzlich werden Import- und Exportformate angeboten, allerdings nur auf rein grafischer Ebene ohne Semantik. Bei der Veranschaulichung des aktuellen Entwicklungsprozesses für agentenbasierte Lösungen in Abbildung 1, wird die Domänenmodellierung durch die erste Spalte dargestellt.

Ebene der Ontologie(n)

Design und Implementierung von Ontologien basiert auf Prozessmodellen aus der Domäne. Es gibt verschiedene Ontologie-Modellierungswerkzeuge wie Protégé [Ge02], OilEd [Be01] oder WebOnto [Do98], von denen Protégé am besten geeignet für den Einsatz in der MAS-Entwicklung ist, da es Werkzeuge zur Überführung modellierter Ontologien in (Java basierte) Agentensysteme anbietet. Protégé unterstützt das Framebasierte Design von Ontologien und stellt hierfür Standardkonzepte und Slots zur Verfügung. Des Weiteren können Instanzen von definierten Konzepten erstellt und dadurch Domänen- beziehungsweise Expertenwissen akquiriert werden. Für diesen Zweck werden automatisch Eingabemasken erstellt, die flexibel an die Bedürfnisse der zu bedienenden Benutzergruppe angepasst werden können. Dies soll die Wissensakquisition erleichtern. Eine Auswahl von definierten Konzepten und Instanzen kann über Anfragen vorgenommen werden. Die Funktionalität von Protégé kann über dynamisch ladbare Plugins erweitert werden. Zurzeit existieren Plugins mit Schnittstellen zu Datenbanken, Visualisierungswerkzeugen, Expertensystemen sowie zum Export von Ontologien in verschiedene Repräsentationssprachen oder als Java Beans. Der Protégé Plugin Mechanismus ermöglicht die Erstellung weiterer Plugins als Schnittstelle zu anderen etablierten Systemen, wie z.B. zu Visio und ARIS Toolset. Die erstellten Ontologien aus Protégé können derzeit im RDF-Format, im XML-Format oder über JDBC in SQL oder CLIPS gespeichert werden. Die Erstellung von Ontologien wird von Ontologie-Designern in Alleinarbeit sowie in Zusammenarbeit mit Domänenexperten vorgenommen.

Ebene der Problemlösungsmethode(n)

Eine Bibliothek von Problemlösungsmethoden (PSM) beruht, unabhängig von der Anwendungsdomäne auf der Spezifikation mehrerer Lösungsmethoden für eine gegebene Problemklasse. Die Erstellung einer Bibliothek besteht im Wesentlichen in der Akquisition der Lösungsmethoden von Experten für Probleme der gegebenen Klasse und ihrer formalen Implementierung, wobei der Fokus auf dem Lösungsweg und nicht auf der Lösung selbst liegt. Beispiele für PSM, die für Probleme aus den Bereichen Planung, Scheduling, Design oder Data-Mining eingesetzt werden, sind unter anderem: Propose and Revise, Propose and Exchange, Hill-Climbing, Simulated Annealing, A* oder Depth/Breadth-Search. Als PSM können auch Dekompositionstechniken für bestimmte Probleme in kleinere Teilprobleme angesehen werden, die mit existierenden PSM gelöst werden können. Die Werkzeugunterstützung zur Erstellung von Bibliotheken für Problemlösungsmethoden hängt vom verwendeten Formalisierungsgrad und der gewählten Sprache für die Implementierung ab. Zurzeit existiert ein Plugin für Protégé, dass die Nutzung von Ontologien in der Entwicklung von PSM unterstützt. Als größte und am meisten verbreitete PSM Bibliothek kann CommonKADS [BV94] angesehen werden. Die Erstellung von PSM Bibliotheken wird von KI-Experten vorgenommen.

Ebene der Agentensystem(e)

Die Entwicklung von Agenten und Agentensystemen wird meistens von Programmierern vorgenommen. Die Unterstützung durch Werkzeuge reicht hier von Frameworks zur reinen Programmierung von Agenten bis hin zu graphischen Modellierungswerkzeugen für das Design von Agenten. Einige sehr bekannte Frameworks sind JADE [BPR01], JACK [Bu99], FIPA-OS [Em01], SeSAm [KI03]. Das Ergebnis der Programmierung kann je nach verwendetem Framework ein Programm sein, das während des Ablaufs einen Agenten repräsentiert oder es ist eine formale Beschreibung eines Agenten. Letztere wird von einem speziellen Interpreter des Agentensystems verarbeitet.

Problemstellung

Der in Abbildung 1 aufgezeigte Entwicklungsprozess zeigt deutlich die fehlende Integration der Werkzeuge. Die Ergebnisse einer Modellierungsphase müssen in den aufeinander folgenden Arbeitsschritten in vielen Fällen erneut umgesetzt werden, ein direkter Modelltransfer zwischen den Werkzeugen wird nicht unterstützt. Nicht selten geschieht der so notwendige manuelle Modelltransfer in Verbindung mit einem Personenwechsel. Wie auch im Rahmen des SPP's ansatzweise beobachtet, kann die unzureichende Integration der einzelnen Werkzeuge zu Ineffizienzen entlang des gesamten Entwicklungsprozesses führen. Modellierungsfehler erzwingen nicht selten einen kompletten und zeitaufwendigen Neutransfer der Modelle, menschliche Fehlinterpretationen von Modellen führen zu Folgefehlern, die oftmals erst am Ende des Entwicklungsprozesses auffallen. Eine verbesserte, insbesondere Medienbruch freie Verknüpfung der einzelnen Werkzeuge könnte den Entwicklungsprozess wesentlich effizienter gestalten.

3 Der Entwurf von Agent.Enterprise und Agent.Hospital

Die folgenden Abschnitte beschreiben die Entwicklungsprozesse der beiden Agententechnologie-Testbeds *Agent.Hospital* [Ki03] und *Agent.Enterprise* [Fr03].

3.1 Technische Entwurfsentscheidungen

Die Integration komplexer Systeme, wie sie die Verknüpfung der Teilsysteme von *Agent.Enterprise* und *Agent.Hospital* darstellt, verlangt eine Einigung über die Verwendung bestimmter technischer Schnittstellen. Da diese technischen Entwurfsentscheidungen signifikanten Einfluss auf die resultierenden Systeme und deren Entstehungsweg haben, werden sie vor einer Diskussion des Entwurfs und der Entwicklung hier vorgestellt. Zusammen bilden sie das so genannte *Gateway-Agent Concept*.

Die erste technische Entwurfsentscheidung ist die verbindliche Verwendung des FIPA-Standards. Durch Verwendung eines kompatiblen Frameworks wie JADE oder FIPA-OS bedürfen Kommunikations- und Infrastrukturfragen keiner weiteren Beachtung und eine Konzentration auf anwendungsorientierte Fragestellungen wird ermöglicht.

Die zweite weitreichende Entwurfsentscheidung betrifft die Systemarchitektur des MMAS (siehe Abbildung 2). Jedes der zu integrierenden MAS wird im resultierenden

MMAS durch einen ausgezeichneten Agenten repräsentiert. Aus Sicht der Software-Technik setzt man also die Entwurfsmuster Fassade und Singleton [GHJV97] ein, um die komplexen Subsysteme zu kapseln. Die Gateway-Agenten bilden ein virtuelles MAS im MMAS, in dem die projektübergreifenden Synergien erzielt werden.

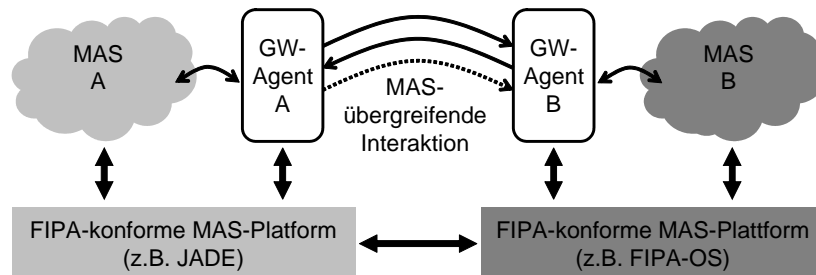


Abbildung 2: Das *Gateway-Agent Concept*

Die Kopplung verschiedener MAS mittels der Gateway-Agenten bringt eine Reihe von Vorteilen mit sich. Die Entwickler können sich bei der Integration auf einen einzelnen Agenten konzentrieren, der alle Funktionalitäten seines MAS im virtuellen MAS vertritt und im Betrieb müssen nur die Adressen der verschiedenen Gateway-Agenten bekannt gemacht werden. Bei Betrachtung des Entwurfsprozesses in Abschnitt 3.2 zeigt sich, dass dadurch auch einige Entwurfsschritte vereinfacht werden.

3.2 Eigene (pragmatische) Vorgehensweisen beim Entwurf

Aus dem Abschnitt 2.2 zum Stand der Technik der MAS-Modellierung und der in Abschnitt 3.1 beschriebenen Entwurfsentscheidung der Gateway-Agenten lassen sich nun Anforderungen an einen Entwicklungsprozess für MMAS ableiten.

- Der Funktionsumfang eines MMAS leitet sich aus den Anforderungen des Geschäftsprozesses ab, zu dessen Unterstützung das MMAS eingesetzt werden soll. Funktionsbedarf, der sich aus Teil-Prozessen ergibt, wird innerhalb der MAS umgesetzt. Eine funktionale Erweiterung im Rahmen des MMAS ist daher nur insoweit nötig, wie sie der Integration der Teilsysteme dient.
- Die Rollendefinition verschiebt sich auf die Ebene der MAS, denen im Rahmen des Gesamtsystems eine oder mehrere Rollen zugewiesen werden. Das *Gateway-Agent Concept* erübrigt die Zuweisung von Rollen zu bestimmten Agenten, da die Gateway-Agenten alle Rollen ihres jeweiligen MAS einnehmen.
- Bei der Integration mehrerer MAS mit eigenen Weltmodellen zu einem MMAS spielt die Erstellung einer gemeinsamen Ontologie eine überaus zentrale Rolle. Ein wichtiges Problem stellt dabei die semantische Interoperabilität dar.
- Neben der Einigung über die statische Schnittstelle durch die Verwendung von FIPA-konformen Plattformen muss schließlich auch die Dynamik im MMAS in Form von Interaktionsprotokollen modelliert werden.

Um diesen Anforderungen an eine Vorgehensweise bei der Modellierung gerecht zu werden, wurde ein pragmatischer Entwurfsansatz konzipiert, der die Verteiltheit der Entwicklung eines MMAS entsprechend unterstützt. Hierzu wurden drei Schritte identifiziert, die im Folgenden am Beispiel von *Agent.Enterprise* erläutert werden.

Rollendefinition und -zuweisung

Im *Agent.Enterprise* Szenario wird eine vollständige Lieferkette ausgehend von der Bestellung des Kunden bis hin zur Auslieferung des Auftrages modelliert. Die beteiligten Projekte nehmen unterschiedliche Rollen entlang der Lieferkette ein, die anhand der jeweiligen Forschungsschwerpunkte verteilt wurden. Nach der Rollenverteilung bestand der nächste Schritt darin, diesen konkrete Aufgaben zuzuteilen.

Eine erste Annäherung an eine fertige Modellierung wurde durch ein Rollenspiel bewirkt, in dem je ein Teilnehmer aus jedem Projekt die Rolle des entsprechenden MAS einnahm und eine informelle Spezifikation der von seinem MAS benötigten Informationen lieferte. Hiernach wurden Karten verteilt, die je einen Kommunikationsakt repräsentierten, auf denen der Versender und Empfänger der Nachricht sowie deren Inhalt notiert wurden. Beginnend mit dem Initiator eines Auftrages – dem Kunden – wurde die vollständige Lieferkette durchgespielt. Alle Kommunikationsakte wurden auf den Karten notiert, bis zum Schluss die letzte Karte die Auslieferung des geordneten Produktes vom OEM zum Kunden vermeldete. Aus diesem Rollenspiel resultierte eine informelle Spezifikation der benötigten Kommunikationsakte und Informationen, um die Lieferkette mit dem MMAS abzubilden, die im nächsten Entwurfsschritt formalisiert werden konnte.

Ontologie Design

Nach der Definition der Rollen jedes teilnehmenden MAS stellte der nächste Schritt sicher, dass eine Kommunikation zwischen den Systemen in Bezug auf die Interpretation von Nachrichteninhalten möglich ist. Die Heterogenität der Wissensrepräsentationen sowie der Semantik der Kommunikationsinhalte der einzelnen Systeme stellte ein Hindernis für die Verhandlungen zwischen den MAS dar.

Das Gateway-Agent Concept definiert ein virtuelles MAS, in dem die Agenten über verschiedene Plattformen verteilt sind, so dass eine Ontologie für die Kommunikation zwischen den Gateway-Agenten die Semantik der ausgetauschten Nachrichten spezifizieren kann. Bei der Verwendung von Ontologien zur Kommunikation existieren grundsätzlich zwei Methoden um sicherzustellen, dass sich Partner verstehen: eine gemeinsame Ontologie oder semantische Mediation.

In einem ersten Ansatz wurde eine gemeinsame Ontologie entworfen, da die Zahl der teilnehmenden Projekte überschaubar war. Um in Zukunft offene Schnittstellen anbieten zu können, sollen zukünftige Arbeiten semantische Mediation in der Kommunikation, auf Basis einer allgemeinen Terminologie zwischen Gateway-Agenten, integrieren.

Die Modellierung der *Agent.Enterprise*-Ontologie erfolgte in Anlehnung an [NM01] und wurde durch Protégé unterstützt. Das Protégé-Plugin Beangenerator [Aa02] bietet die Möglichkeit, die Ontologie direkt in Java-Code zu exportieren, wie er für die Agentenplattform JADE [BPR01] verwendet wird. Bei der Ontologie-Modellierung wurden

zunächst die Aktionen von Agenten identifiziert, die von Kommunikationspartnern ausgelöst werden können. Diese konnten direkt im Rollenspiel abgeleitet werden und spezifizieren, welche Aufgaben ein Agentensystem durchführen kann. Anschließend wurden die Konzepte spezifiziert, die die innerhalb der Agentenaktionen behandelten Informationen (wie z.B. Stücklisten, Preisangebote, usw.) repräsentierten.

Interaktions-Protokoll Design

Im nächsten Schritt der Integration verschiedener MAS zu einem MMAS wurde der dynamische Teil der Kommunikation definiert: die Auswahl der zu Interaktionsprotokolle. Die informelle Beschreibung von Interaktionen aus dem Rollenspiel wurde auf entsprechende FIPA-Protokolle abgebildet. Daraus resultierte die Spezifikation des Verhaltens jedes Gateway-Agenten in einer Kommunikation mit anderen Gateways. Im letzten Schritt musste die Interaktion zwischen den Gateway-Agenten und dem jeweils darunter liegenden Agentensystemen spezifiziert werden. Da dieses jedoch keine weitere Koordination benötigte, wurden diese Arbeiten direkt von den teilnehmenden Projekten durchgeführt. Das Ergebnis dieses Prozesses bildet das MMAS *Agent.Enterprise* [Fr03].

4 Die Entwicklung von *Agent.Enterprise* und *Agent.Hospital* - Technische Lösungen zu ausgewählten Problemen

Die folgenden Abschnitte beschreiben ausgewählte Probleme, die bei der Erstellung von MMAS auftreten, und welche Lösungsansätze hierzu im SPP verfolgt werden.

4.1 Technische Unterstützung der Prozesskopplung und Leistungsbewertung

Die in *Agent.Hospital* und *Agent.Enterprise* entwickelten interoperierenden Agentendienste stellen eine mögliche informationstechnische Infrastruktur für ein Krankenhaus bzw. einen Fertigungsbetrieb dar. Ein solches System zeigt seine Fähigkeiten und Eigenschaften in einer dynamischen Umwelt, bestehend aus anderen klassischen Informationssystemen und realen Akteuren. Die zu entwickelnde Anwendung kann deshalb nicht ohne weiteres für sich allein, sondern nur in einer realen oder simulierten Umwelt getestet werden. Im Hinblick auf die Leistungsbewertung von MASen und MMASen wurden deshalb zwei Simulationsansätze angewandt, die im Folgenden beschrieben werden.

SeSAM als dynamisches Kopplungselement in *Agent.Hospital*

Innerhalb der *Agent.Hospital* Gruppe wurde zum Testen der interoperierenden MAS der Teilprojekte ein integriertes Simulations-Szenario erstellt. Da in diesem nicht nur die Anwendungssysteme selbst, sondern auch die Agentenumgebung - das Krankenhaus an sich - simuliert werden sollte, wurde die Simulations-Entwicklungsumgebung SeSAM (Shell für Simulierte Agentensysteme) zur Modellierung gewählt. Diese erfüllt zudem die weitere Anforderung, dass das Szenario auch von Domänen-Experten im Simulationswerkzeug erstellt werden kann. Ein ausgewählter Patienten-Prozess war Ausgangspunkt für die Erstellung eines Simulationsmodells, das ein „virtuelles Krankenhaus“ darstellt. In dieses konnten die Agentensysteme der Teilprojekte eingebunden werden.

SeSAM ist eine generische Entwicklungs- und Ablauf- sowie Experimentierumgebung für agentenbasierte Simulationen. Es ist hauptsächlich ausgerichtet auf die einfache Konstruktion komplexer Modelle und soll auch von Wissenschaftlern ohne Programmierkenntnisse bedient werden können [K101]. SeSAM gestattet die Agentenmodelle durchgängig visuell zu modellieren (siehe Abbildung 3).

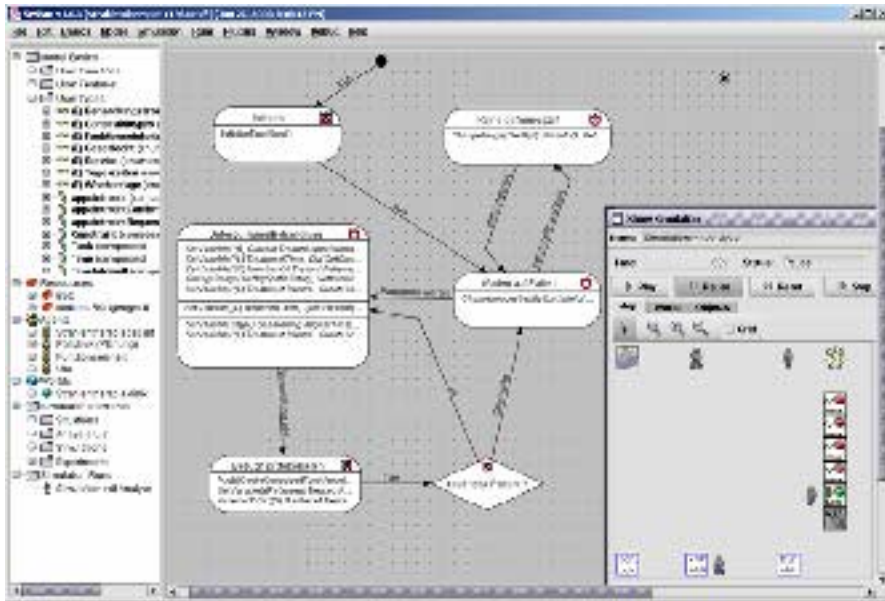


Abbildung 3: Modell einer Strahlentherapie-Klinik in SeSAM

Da SeSAM ursprünglich zur Simulation in sich geschlossener und vollständig in der Entwicklungsumgebung repräsentierter Agentensysteme konzipiert war, fehlten anfangs Mechanismen zur Interaktion mit externen Agentensystemen. Durch die neuen Anforderungen wurden Erweiterungen vorgenommen, die ein neues Einsatzgebiet von SeSAM als Simulationsumgebung für FIPA-konforme Agentensysteme eröffneten [K103]. Diese stehen jeweils als Plugin zur Verfügung:

- *SeSAM Ontologie-Import*. Gegenseitiges Verständnis von Agenten erfordert den Bezug auf gleiche Ontologien. In den etablierten FIPA-Plattformen ist der Einsatz von Ontologien, die mit Modellierungstools erzeugt wurden, bereits Standard und deren Einbindung weitgehend automatisiert. Ähnlich können in SeSAM Protégé-Ontologien importiert und bei der Erstellung von Nachrichten verwendet werden. Zudem ist es möglich, in der Ontologie beschriebene Konzepte, die Agenten oder Ressourcen darstellen, direkt in Objekte mit entsprechenden Eigenschaften umzuwandeln. Dies erleichtert die Konstruktion der Simulationsmodelle, wenn umfangreiche Ontologien zur Verfügung stehen.
- *SeSAM FIPA Communication Support*. Ein weiteres Plugin bietet FIPA-konformes Agent Management und standardisierte Kommunikationsfähigkeiten für SeSAM-Agenten. Es stellt Verhaltensprimitive zur Registrierung und Deregistrierung an ei-

nem Directory Facilitator sowie Primitive zum Senden und Empfangen von Nachrichten zur Verfügung.

- *SeSAM FIPA SL Language Support*. Zur Unterstützung der Content-Sprache FIPA SL ermöglicht diese Erweiterung die graphisch gestützte Eingabe von SL-konformen Ausdrücken. Dabei wird der Benutzer in der Eingabe so eingeschränkt, dass er nur syntaktisch korrekte SL-Nachrichten formulieren kann. Instanzen ontologischer Begriffe können, soweit es die Syntax erlaubt, direkt referenziert und zur Nachrichtenübertragung automatisch serialisiert werden.

Alle drei Erweiterungen zusammengenommen ermöglichen eine einfache Kopplung der Simulationsumgebung an agentenbasierte Anwendungen, sofern sie mit FIPA-konformen Plattformen realisiert wurden. In zwei mehrtägigen Implementierungswerkshops konnten auf dieser Basis die Agentensysteme der Krankenhausprojekte des SPP's miteinander gekoppelt werden. Das Simulationsmodell stellt, ausgehend von einem exemplarischen Patientenprozess, Anfragen (z.B. Termin- oder Behandlungsentscheidungsanfragen) an die einzelnen Agentensysteme. Die Ergebnisse der Anfragen fließen in den Fortgang der Simulation ein. Der Vorteil der Verwendung von SeSAM zeigte sich insbesondere darin, dass es mit nur wenigen Modellierungsschritten möglich ist, ein funktionsfähiges und anschauliches Beispiel-Szenario für Agentendienste zu entwickeln.

FIPA Time Service Agent

Als ein weiterer Ansatz zur Simulationssteuerung wurde der Agentendienst des Time Service entwickelt. Durch die von den Gateway-Agenten realisierten, jeweilig übergeordneten Prozessmodelle von *Agent.Enterprise* und *Agent.Hospital* ist eine grobe Ablaufsteuerung der MMAS vorgegeben. Die Notwendigkeit einer feineren Synchronisation ergibt sich allerdings daraus, dass über alle Systeme hinweg die korrekte Reihenfolge der in den Projekten intern modellierten zeitverbrauchenden Aktivitäten (Bearbeitung von Werkstücken bzw. Behandlung von Patienten) sichergestellt werden muss.

Die als Alternative zur Ablaufsteuerung mit SeSAM entwickelte Time Service Komponente für MMAS hat die Aufgabe die zeitliche Reihenfolge der in einem Prozess spezifizierten Aktivitäten zu gewährleisten [Br03]. Im Gegensatz zur Ablaufsteuerung in SeSAM muss dabei nicht ein zentrales Gesamtmodell aller Projekt übergreifenden Prozesse vorliegen. Die Synchronisation der verteilten Abläufe erfolgt direkt in den jeweiligen Systemen. Ein zentraler Time-Service Agent (TSA) sorgt dafür, dass die einzelnen Aktivitäten in der korrekten Reihenfolge ausgeführt werden. Dabei berechnen die einzelnen MAS intern ihre nächsten Aktivierungszeitpunkte, die vom TSA in einer globalen Liste verwaltet werden und warten bis der Aktivierungszeitpunkt eintritt. Der TSA wählt den jeweils nächsten Aktivierungszeitpunkt, versendet an den entsprechenden Agenten eine Aktivierungsnachricht und wartet daraufhin selbst bis die Liste der Aktivierungszeitpunkte aktualisiert ist. Der TSA wurde als FIPA-konformer Dienst in JADE realisiert.

Vergleich der beiden Alternativen SeSAM - Time Service Agent

Simulation ist ein wichtiges Werkzeug zum Testen der entwickelten Agentenanwendungen. Im Rahmen des SPP 1083 wurden dabei zwei Ansätze verfolgt, die prinzipiell beide zur Simulationssteuerung geeignet sind, jedoch aus unterschiedlichen Anforderungen

heraus entwickelt bzw. eingesetzt wurden. Der Fokus und die Stärke von SeSAM liegen dabei in der Modellierung eines Prozessmodells und der Visualisierung der Simulation. Es dient innerhalb von *Agent.Hospital* vor allem der Erstellung des übergreifenden Szenarios und Integration mehrerer Agentensysteme. Der Fokus des Time Service Agenten ist die (evtl. nachträgliche) Integration einer zeitlichen Koordination in die MAS. Dessen Dienst ermöglicht die einzelnen verteilten Aktivitäten der Agenten und Interaktionen zwischen den Agenten eines MMAS global zu synchronisieren. Dabei muss für die Synchronisation nicht wie bei SeSAM ein zentrales Gesamtmodell vorgehalten werden, sondern sie erfolgt intern in den Agenten bzw. den Interaktionsprotokollen, wobei der zentrale Time Service Agent lediglich die korrekte Reihenfolge von Ereignissen sicherstellt. Ein weiterer Unterschied ist die Simulationsart: Der TSA unterstützt die Ereignis gesteuerte Simulation während SeSAM Takt gesteuert arbeitet. Beide Werkzeuge besitzen deshalb die jeweiligen Vor- und Nachteile ihrer grundlegenden Simulationsart.

4.2 Implementierungsunterstützung für die BDI-Agentenarchitektur

Wie in Abschnitt 2.1 aufgezeigt wurde, sind auf der Implementierungsebene sowohl Agentenplattformen verfügbar, die FIPA-Konformität unterstützen, als auch Systeme, die eine interne Agentenarchitektur (z.B. das BDI-Modell) umsetzen. Es existiert jedoch bisher keine Lösung, die beide Aspekte gleichermaßen unterstützt. Da FIPA-Konformität im Rahmen der *Agent.Enterprise* und *Agent.Hospital* Initiativen eine essentielle Eigenschaft der einzelnen Prototypen darstellt, setzen die meisten Systeme auf bestehenden Agentenplattformen wie z.B. JADE auf. Dies erschwert jedoch die sich aus den einzelnen Projektinhalten ergebenden Anforderungen umzusetzen, da den Entwurfsentscheidungen bezüglich der Agentenfunktionalität keine entsprechenden Konzepte auf der Implementierungsebene gegenüberstehen. Auch fehlt den Plattformen eine Unterstützung für die von der FIPA angestrebte semantische Interoperabilität, nach der Agenten nicht nur auf vorgefertigte Nachrichten reagieren, sondern in der Lage sind, eine Nachricht zu verstehen und selbst entscheiden zu können, wie damit zu verfahren ist.

Um die o.g. Nachteile auszugleichen, wird zurzeit das Jadex System [PBL03] entwickelt, das ausgehend von JADE, welches einen hohen Reifegrad und ein sehr einfaches und daher leicht zu erweiterndes internes Agentenmodell besitzt, eine Agentenarchitektur umsetzt und als Add-on für bestehende Agentenplattformen eingesetzt werden kann. Die Agentenarchitektur folgt dem BDI-Modell. Dieses erlaubt eine abstrakte, zielorientierte Beschreibung von Agenten, die (entsprechende Ausführungsumgebung vorausgesetzt) direkt auf die Implementationsebene übertragen werden kann (vgl. PRS, JACK). Da der zielorientierte Entwurf in flexibleren Systemen als eine starre Abbildung von Anwendungsfällen auf Programmcode resultiert, bietet das BDI-Modell auch einen Ansatz für semantische Interoperabilität (auch die FIPA-ACL besitzt eine BDI-Semantik, [Fi02b]). Jadex ermöglicht somit den Transfer von Designentscheidungen in die Laufzeitebene ohne dabei die Unterstützung der FIPA-Konformität zu vernachlässigen.

Entsprechend dem BDI-Modell besitzt ein Jadex Agent Wissen über sich und seine Umgebung, verfolgt Ziele, und hat Pläne um seine Ziele zu erreichen. Die Ausführung der Pläne erfolgt in einer geschützten Umgebung, damit der Agent das Fehlschlagen eines

Planes feststellen und entsprechend reagieren kann. Er kann z.B. neue Pläne auswählen, einen fehlgeschlagenen Plan versuchen erneut umzusetzen oder das Ziel für unerreichbar erklären. Ziele und Pläne werden auf Grund interner und externer Ereignisse ausgelöst.

Die Ausführungsumgebung Jadex ist in einer ersten Version realisiert und frei verfügbar (<http://sourceforge.net/projects/jadex/>). Im Rahmen des MedPAge Projektes wird sie bereits zur Umsetzung der Anwendungsfunktionalität eingesetzt. Im nächsten Schritt wird die Verbesserung der Benutzbarkeit des Systems durch die Bereitstellung von Werkzeugen fokussiert. Hierfür wurde begonnen, ein Werkzeug zu entwickeln, mit dem sich der interne Zustand eines BDI-Agenten zur Laufzeit überwachen und manipulieren lässt. Des Weiteren ist geplant, ein Modellierungswerkzeug zu erstellen, mit dem sich die Agenteninterna entwerfen und in einer Spezifikationsdatei speichern lassen. Im Rahmen dieser und anderer Arbeiten wird auch das zugrunde liegende Agentenmodell verfeinert und gemäß weiteren Anforderungen aus dem SPP erweitert. Ein zweiter wichtiger Schritt in der Fortentwicklung des Jadex Systems betrifft die verbesserte Einbindung der BDI-Architektur in die Realisierung von MAS. Es wird angestrebt, die BDI-Semantik auch auf die Interaktionen zwischen Agenten zu übertragen (FIPA-Forderung).

5 Zusammenfassung und Ausblick

Der vorstehend beschriebene Entwicklungsprozess von MMAS sowie der diesen unterstützenden Werkzeuge zeigen weitere Herausforderungen für zukünftige Forschungsarbeiten im Bereich der Agententechnologie. Fragestellungen bezüglich der möglichen Integrationstiefe existierender Modellierungs- und Entwicklungswerkzeuge in neue oder existierende offene Frameworks, wie zum Beispiel Protégé [Ge02] und Eclipse (<http://www.eclipse.org/>) werden erst nach weiteren Forschungsarbeiten beantwortet werden können. Wichtige technische Herausforderungen sind hier die Verwendung semantischer Mediation in MMAS und die weitere Integration von Expertensystemen. Hierdurch könnte die Abbildung ontologischer Terme aus Sprechakten für Agenten auf Regeln oder Funktionen in Expertensystemen vorgenommen werden. Die weitere Integration von BDI Ansätzen in die beiden MMAS *Agent.Enterprise* und *Agent.Hospital* wird Aussagen bezüglich der praktischen Anwendbarkeit dieser Ansätze ermöglichen.

Im Bereich der Architekturen stehen Untersuchungen zur dynamischen Dienstkomposition, Zuverlässigkeit und Verfügbarkeit von MAS sowie zu Mobilien Agenten an. Weitere Fragen betreffen das Benchmarking von MAS und MMAS, um ihre Leistungsfähigkeit mit bereits existierenden Lösungen vergleichen zu können.

Die enge Zusammenarbeit der Mitglieder der Technologie AG mit internationalen Forschern aus dem industriellen und universitären Bereich und mit der Standardisierungsorganisation FIPA ermöglichen eine Überführung aktueller Forschungsergebnisse in den Standardisierungsprozess von Agentensystemen. Geplant ist hier eine stärkere Öffnung der einzelnen Projekte nach Außen und eine intensive Verfolgung des Open Source Gedanken. Zudem sollen die beiden MMAS *Agent.Enterprise* und *Agent.Hospital* für die Integration weiterer MAS anderer Forschungseinrichtungen geöffnet werden.

Literaturverzeichnis

- [Aa02] van Aart, C. J.; Pels, R.F.; Caire, G.; Bergenti, F.: Creating and Using Ontologies in Agent Communication. In: *Proceedings of the Workshop on Ontologies in Agent Systems* at the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS2002). Bologna, Italy, 2002.
- [Be01] Bechhofer, S.; Horrocks, I.; Goble, C.; Stevens, R.: OilEd: a Reason-able Ontology Editor for the Semantic Web. In: *Proceedings of Joint German/Austrian conference on Artificial Intelligence (KI2001)*. Springer-Verlag, Berlin, 2001.
- [BPR01] Bellifemine, F.; Poggi, A.; Rimassa, G.: JADE: a FIPA2000 Compliant Agent Development Environment. In *Proceedings of the Fifth International Conference on Autonomous Agents*. ACM Press, 2001.
- [Br87] Bratman, M.: *Intention, Plans, and Practical Reason*. Harvard University Press, 1987.
- [Br03] Braubach, L.; Pokahr, P.; Krempels, K.-H.; Woelk, P.-O.: Time Synchronization for Process Flows. Submitted to Second Seminar on Advanced Research in Electronic Business: Special Track on Agent-Oriented Technologies for E-Business. 2003.
- [Bu99] Busetta, P. et al.: JACK Intelligent Agents - Components for Intelligent Agents in Java. In: *AgentLink News Letter*, January 1999.
- [BV94] Bruker, J.; Van de Velde, W.: *CommonKADS Library for Expertise Modelling: Reusable problem solving components*. IOS Press, 1994.
- [CP02] Cosentino, M.; Potts, C.: A CASE Tool Supported Methodology for the Design of Multi-Agent Systems. In: *Proceedings of the International Conference on Software Engineering Research and Practice*. CSREA Press, 2002.
- [Do98] Domingue, J.: Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the Web. In: *Proceedings of the 11th Workshop on Knowledge Acquisition for Knowledge-based Systems*. Banff, Canada, 1998.
- [Em01] Emorphia Limited: FIPA-OS V2.1.0 Distribution Notes. <http://sourceforge.net/projects/fipa-os/>, 2001.
- [Fi02a] Foundation for Intelligent Physical Agents: FIPA Agent Management Specification. Document no. SC00023J, <http://www.fipa.org>, 2002.
- [Fi02b] Foundation for Intelligent Physical Agents: FIPA Communicative Act Library Specification. Document no. SC00037J, <http://www.fipa.org>, 2002.
- [Fi03] Foundation for Intelligent Physical Agents (FIPA). [Internet], Available from: <http://www.fipa.org> [Accessed 13.06.03]
- [Fr03] Frey, D.; Stockheim, T.; Woelk, P.-O.; Zimmermann, R.: Integrated Multi-agent-based Supply Chain Management. In: *Proc. of the 1st Intern. Workshop on Agent-based Computing for Enterprise Collaboration*. IEEE Computer Society Press, 2003.
- [Ge02] Gennari, J.; Musen, M. A.; Ferguson, R. W.; Grosso, W. E.; Crubézy, M.; Eriksson, H.; Noy, N. F.; Tu, S. W.: *The Evolution of Protégé: An Environment for Knowledge-Based Systems Development*. Stanford Medical Informatics Technical Report SMI-2002-0943. Stanford, 2002.
- [GHJV97] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: *Design Patterns – Elements of Reusable Object-Oriented Software*. Addison Wesley, 1997.
- [GMP02] Giunchiglia, F.; Mylopoulos, J.; Perini, A.: The Tropos Software Development Methodology: Processes, Models and Diagrams. In: *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems*. ACM Press, 2002.
- [HG02] Henderson-Sellers, B.; Gorton, I.: Agent-Based Software Development Methodologies. White Paper of the Workshop on Agent-Oriented Methodologies at OOPSLA2002. Seattle, 2002.
- [Hi99] Hindriks, K. et al.: Agent Programming in 3APL. In (Jennings, N.; Sycara, K.; Georgeff, M. Hrsg.): *Autonomous Agents and Multi-Agent Systems*, vol. 2, no. 4, pp. 357-401, Kluwer Academic publishers, 1999.

- [Hu99] Huber, M.: JAM: A BDI-Theoretic Mobile Agent Architecture. In (Etzioni, O.; Müller, J.P.; Bradshaw, J. Hrsg.): Proceedings of the Third Annual Conference on Autonomous Agents, pp. 236-243, ACM Press, 1999.
- [IGG99] Iglesias, C. A.; Garijo, M.; Gonzales, J. C.: A Survey of Agent-oriented Methodologies. In: Proceedings of the 5th International Workshop on Agent Theories, Architectures, and Languages (ATAL'98). Springer-Verlag, Berlin, 1999.
- [Je01] Jennings, N. R.: An agent-based approach for building complex software systems. In: Communications of the ACM, Vol. 44, No. 4, pp. 35-41. 2001.
- [Ki03] Kirn, S.; Heine, C.; Herrler, R.; Krempels, K.-H.: Agent.Hospital - Agent-based Open Framework for Clinical Applications. In: Proceedings of the 1st International Workshop on Agent-based Computing for Enterprise Collaboration. IEEE Computer Society Press, 2003.
- [KI01] Klügl, F.: Multiagentensimulation - Konzepte, Werkzeuge, Anwendung. Addison Wesley, April, 2001.
- [KI03] Klügl, F.; Herrler, R.; Oechslein, C.: From Simulated to Real Environments: How to use SeSAM for software development. Accepted at the First German Conference on Multiagent System Technologies (MATES2003), 2003.
- [LFP99] Labrou, Y.; Finin, T.; Peng, Y.: Agent Communication Languages: The Current Landscape. IEEE Intelligent Systems, vol. 14, no. 2, pp. 45-52, 1999.
- [Li01] Lind, J.: Iterative Software Engineering for Multiagent Systems: the MASSIVE Method. Lecture Notes in Computer Science and Lecture Notes in Artificial Intelligence. Vol. 1994. Springer-Verlag, Berlin, 2001.
- [Ma02] Mangina, E.: Review of Software Products for Multi-Agent Systems. In: AgentLink, software report, 2002.
- [NM01] Noy, N. F.; McGuinness, D.L.: Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880. 2001.
- [OPB01] Odell, J.; Parunak, H.V.D.; Bauer, B.: Representing Agent Interaction Protocols in UML. In: Proceedings of the First International Workshop on Agent-oriented Software Engineering (AOSE2000). Springer-Verlag, Berlin, 2001.
- [PBL03] Pokahr, A.; Braubach, L.; Lamersdorf, W.: Jadex: Implementing a BDI-Infrastructure for JADE Agents. In: EXP – in search of innovation, TILAB Journal, 2003. (to appear)
- [WD01] Wood, M.F.; DeLoach, S.A.: An Overview of the Multiagent System Engineering Methodology. In (Ciancarini, P.; Wooldridge, M. Hrsg.): Proceedings of the First International Workshop on Agent-oriented Software Engineering. Springer-Verlag, Berlin, 2001.
- [WJK00] Wooldridge, M.; Jennings, N. R.; Kinny, D.: The Gaia Methodology for Agent-Oriented Analysis and Design. In: Journal of Autonomous Agents and Multi-Agent Systems, 3(3):285–312. 2000.
- [Ra96] Rao, A.: AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In (van der Velde, W.; Perram, J. Hrsg): Agents Breaking Away, Springer-Verlag, 1996.
- [Re00] Reticular Systems, Inc.: AgentBuilder User's Guide (Version 1.3). <http://www.agentbuilder.com/Documentation/usermanual.html>, 2000.
- [Se69] Searle, J. R.: Speech Acts. Cambridge University Press, Cambridge, 1969.
- [Sh93] Shoham, Y.: Agent-oriented programming. Artificial Intelligence, vol. 60, pp. 51-92, Elsevier Amsterdam, 1993.
- [Tr02] Tryllian Solutions B.V.: The Developer's Guide (ADK 2.1 Edition). http://www.tryllian.com/development/DOCS2_1/DeveloperGuide.pdf, 2002.