Information Technology

Agenten: Technologie für den Mainstream?

Agents: Technology for the Mainstream?

Alexander Pokahr: Verteilte Systeme und Informationssysteme, Fachbereich Informatik, Universtät Hamburg, Vogt-Kölln-Straße, 30, 22527 , Germany
Tel: +49 40 42883 2091, Fax: +49 40 42883 2328, E-Mail: pokahr@informatik.uni-hamburg.de
Alexander Pokahr studierte Informatik mit dem Schwerpunkt Verteilte Systeme und dem Nebenfach Philosophie an der Universität Hamburg. Seit drei Jahren befasst er sich als wissenschaftlicher Mitarbeiter der Universität Hamburg mit Software-Architekturen und Entwicklungswerkzeugen für Intelligente Agenten und verteilte Multiagentensysteme. Außerdem entwickelt er gemeinsam mit Lars Braubach das Open-Source BDI Agentensystem Jadex.

Lars Braubach: Verteilte Systeme und Informationssysteme, Fachbereich Informatik, Universtät Hamburg, Vogt-Kölln-Straße, 30, 22527 , Germany
Tel: +49 40 42883 2091, Fax: +49 40 42883 2328, E-Mail: braubach@informatik.uni-hamburg.de
Lars Braubach studierte Informatik mit dem Schwerpunkt Verteilte Systeme und dem Nebenfach Psychologie an der Universität Hamburg. Seit drei Jahren ist er im Rahmen des DFG-Schwerpunktprogramms Ïntelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarienßowohl mit der Grundlagenforschung im Bereich von Multiagentensystemen als auch mit der praktischen Erprobung dieser Technologie befasst.

Winfried Lamersdorf: Verteilte Systeme und Informationssysteme, Fachbereich Informatik, Universtät Hamburg, Vogt-Kölln-Straße, 30, 22527 , Germany
Tel: +49 40 42883 2421, Fax: +49 40 42883 2328, E-Mail: lamersd@informatik.uni-hamburg.de
Prof. Dr. Winfried Lamersdorf ist Leiter des AB Verteilte Systeme und Informationssysteme (VSIS) am FB Informatik der Universität Hamburg. Er studierte Informatik an der TU München und in Hamburg und promovierte im Bereich semantischer Datenmodelle. Danach arbeitete er am Europäischen Zentrum für Netzwerkforschung (ENC) der IBM in Heidelberg. In den letzten 15 Jahren hat er eine Vielzahl sowohl universitärer als auch drittmittelfinanzierter Projekte betreut.

MS-ID:	pokahr@informatik.uni-hamburg.de	15. April 2005
Heft: / ()		

Abstract

Agent technology is seen as a promising approach to address the problems of engineering today's complex software systems such as heterogeneous system environments and decentralised structures. This article presents a discerning overview of the current state of the art regarding the support that agent technology provides for the development of complex software systems. Agent technology is a highly diversified field, therefore it is tried to highlight the main influences, and their interrelations. The suitability of the presented technologies for certain application domains is shown by providing some real-world examples.

Zusammenfassung

Agententechnologie wird als ein vielversprechender Ansatz gesehen, die Probleme der Konstruktion komplexer Software-Systeme wie heterogene Systemumgebungen und dezentrale Strukturen besser beherrschbar zu machen. Dieser Beitrag wirft einen kritischen Blick auf den erreichten Stand der softwaretechnischen Unterstützung, den die Agententechnologie bei der Systementwicklung bietet. Da die Agententechnologie ein stark diversifiziertes Feld ist, wird versucht die unterschiedlichen Einflussfaktoren und ihre Wechselwirkungen aufzuzeigen. Zudem wird durch Beispiele aus der Praxis belegt, für welche Anwendungsfelder die vorgestellten Technologien besonders gut geeignet sind.

1 Einleitung

Agententechnologie verspricht, den Transfer von komplexen Problemstellungen in Softwarelösungen zu verbessern, da abstrakte naturanaloge Konzepte als Grundlage der Problemlösung eingeführt werden. Aufgrund des höheren Abstraktionsgrades sollen mittels der Agententechnologie auch derart komplexe Problemstellungen adressiert werden können, die mit herkömmlichen Techniken wie z.B. der Objektorientierung nicht oder nur schwer handhabbar sind. In der Agententechnologie wird ein Multiagentensystem (MAS) dabei als eine Gesellschaft eigenständig agierender Akteure gesehen, die in Konkurrenz oder Kooperation Aufgaben bewältigen. Schon aus dieser Beschreibung ergeben sich zwei unterschiedliche Konzepte, die integrale Bestandteile dieser neuen Technologie sind: *Agenten* und *Agentengesellschaften*.

Das Konzept des Agenten ist dabei nicht durch eine exakte Definition festgelegt, sondern wird vielmehr durch die Angabe charakteristischer Eigenschaften näher bestimmt. In [19] werden die vier Merkmale *Autonomie*, *Reaktivität*, *Zielgerichtetheit* und *Interaktion* als Mindestvoraussetzungen für Agenten genannt (weak notion of agency). In einer starken Definition (strong notion of agency) wird zusätzlich die Verwendung von *mentalistischen Konzepten*, d.h. Konzepten, die dem menschlichen Denken nachempfunden sind, verlangt. Darüber hinaus können auch weitere Eigenschaften wie z.B. Mobilität oder Lernfähigkeit mit Agenten assoziiert werden.

Für Agentengesellschaften gestaltet sich die Definition noch schwieriger, da es unterschiedliche Auffassungen über deren Kerneigenschaften gibt. So müssen Multiagentensysteme nicht unbedingt explizite Strukturierungsmechanismen zu Grunde liegen, sondern die Struktur kann sich emergent durch das Verhalten der beteiligten Agenten ergeben. Alternativ können die Beziehungen zwischen Agenten z.B. aus den für sie individuell notwendigen und angebotenen Diensten resultieren. Organisationstheoretische Ansätze [5, 13] beschreiben Agentengesellschaften hingegen als eigenständiges rekursives Konzept, das mit Hilfe von *Rollen* und *Gruppen* die Handlungsmöglichkeiten der als Rollenträger fungierenden Agenten beeinflusst.

Exakte Definitionen der verwendeten Begriffe sind auch aufgrund der Breite des Themengebiets schwierig, da Forschergruppen aus unterschiedlichsten Disziplinen wie z.B. Informatik, Psychologie, Soziologie und Philosophie ihre eigene individuelle Sicht beitragen. Trotz oder gerade wegen der geringen Konzeptschärfe ist die Modellierung nach dem Agentenparadigma weit verbreitet. Dies wird z.B. im Bereich des Software Engineering durch eine große Anzahl an neu entstandenen agentenorientierten Methoden [16], andererseits aber auch durch die Einbeziehung der Agentenmetapher in etablierte Modellierungstechniken wie Use-Case Diagramme und aktive Objekte in UML belegt. Die Modellierung mit diesem neuen Paradigma ist besonders eingängig, da sich viele Probleme mit den oben genannten Konzepten auf eine natürliche Weise beschreiben lassen.

Nachdem dieser Abschnitt eine kurze Einführung in die grundlegenden Begriffe *Agent* und *Agentengesellschaft* gegeben hat, wird im Folgenden zunächst ein Überblick über die Agententechnologie präsentiert, der existierende Agentenplattformen nach ihren Ursprüngen und daraus resultierenden Eigenschaften einordnet. In Abschnitt 3 wird die Praxisrelevanz dieser Plattformen durch eine Betrachtung der Anwendungsfelder und konkrete Anwendungsbeispiele belegt. Darauf aufbauend bewertet Abschnitt 4 den erreichten Stand in Bezug auf Faktoren, die für die Etablierung der Agententechnologie im Mainstream wichtig sind. Abschließend werden in Abschnitt 5 die gewonnenen Erkenntnisse zusammengefasst.

2 Agententechnologie Überblick

Ziel der folgenden Abschnitte ist, einen Überblick über verfügbare agentenorientierte Plattformen zu geben und dabei wichtige Einflüsse von Seiten der Theorien, Architekturen und Sprachen deutlich zu machen.

Andere wichtige Aspekte des Agentenorientierten Software-Engineerings (AOSE), wie insbesondere Methoden und Vorgehensweisen zur Softwareentwicklung, sind bereits an anderer Stelle ausführlich betrachtet worden (siehe z.B. [18] für einen allgemeinen Überblick).

Die Ursprünge agentenorientierter Plattformen sind in Abb. 1 veranschaulicht, in dem Theorien als abstrakteste Einflussfaktoren innen dargestellt werden, und nach aussen in weiteren Stufen über Architekturen und Sprachen bis hin zu den Plattformen konkretisiert werden.[1] Diese Betrachtung motiviert eine Einteilung der Plattformen in verschiedene Kategorien, deren wichtigste Vertreter am Ende dieses Abschnittes kurz vorgestellt werden.

2.1 Theorien

Der Überblick beginnt ausgehend von den abstrakten, zu Grunde liegenden *Theorien*. Diese haben das Ziel, allgemeingültige Modelle zur Beschreibung von Individual- und Gruppenverhalten zur Verfügung zu stellen. Inspiriert wurde die Konzeption vieler Theorien durch den Wunsch, bestimmte beobachtbare Realweltphänomene adäquat erklären und nachbilden zu können. Es ist daher nicht verwunderlich, dass die Entstehung der Theorien im Kontext unterschiedlicher *Disziplinen* stattfand und dass ihre Anwendbarkeit auf einen bestimmten Aspekt begrenzt ist.

So wurde die Belief Desire Intention (BDI) Theorie [1] mit dem Ziel formuliert, rationales Handeln erklärbar zu machen. Diese Theorie ist maßgeblich philosophisch motiviert und erklärt Denken nicht nach biologischem Vorbild, sondern mit Hilfe von abstrakten Konzepten und deren Beziehungen. Aus einer ähnlichen Motivation heraus wurde auch die Theorie des Agent Oriented Programming (AOP) [15] aufgestellt, die andere grundlegende Bausteine der menschlichen Rationalität annimmt und bereits stärker von der Idee einer softwaretechnischen Umsetzung beeinflusst wurde. Neben der Philosophie war auch die Psychologie Quelle diverser, meist sehr spezialisierter Theorien für Erklärung einzelner Aspekte menschlichen Verhaltens wie z.B. des Problemlösens, der Entscheidungsfindung und des Lernens. Mit den Unified Theories of Cognition (UTC) von Newell wurde der erste Versuch unternommen, die Mikrotheorien in einem einheitlichen Modell zusammenzuführen [11]. Auch innerhalb der Biologie wurden zahlreiche Erkenntnisse aus Beobachtungen gewonnen, die sich in Theorien zur Beschreibung von Verhalten niedergeschlagen haben. So verwendet die Subsumption Theorie das Verhalten einfacher Organismen wie z.B. Insekten als Grundlage für die Beschreibung intelligenten Verhaltens in einer komplexen Umwelt [2].

Zusätzlich zu den Theorien für die Beschreibung von Individualverhalten existieren ebenfalls Theorien zur Erklärung von Gruppenverhalten, die zumeist soziologisch oder organisationstheoretisch motiviert sind. Die Fragestellungen in diesem Zusammenhang betreffen sowohl die Struktur- als auch die Verhaltensdimension. Hinsichtlich struktureller Aspekte ist von Interesse, mit welchen Konzepten eine Gruppe beschrieben wird und welche Beziehungen zwischen Gruppenmitgliedern und unterschiedlichen Gruppen als ganzes bestehen. Innerhalb der Verhaltensdimension stehen die Art und Weise der Gruppenbildung bzw. Auflösung und das Verhalten der Gruppe als Ganzes unter Einbeziehung der Wechselwirkungen zwischen Gruppenmitgliedern im Mittelpunkt der Betrachtung.

Für einzelne dieser Subthematiken existieren unterschiedliche Ansätze. So führt die JointIntentions Theorie [3] Konzepte ein, um das zielgerichtete Verhalten einer Gruppe formal mit Hilfe mentalistischer Begriffe zu beschreiben, wohingegen das Agent-Group-Role (AGR) Modell [5] ein sehr einfaches Modell zur Abbildung von Organisationsstrukturen darstellt und damit eine rekursive Spezifikation von Agentengesellschaften ermöglicht. Sehr ähnliche Konzep-

[1] Verweise zu den Webseiten der abgebildeten Werkzeuge sind unter http://vsis-www.informatik.uni-hamburg.de/projects/jadex/links.php zu finden.

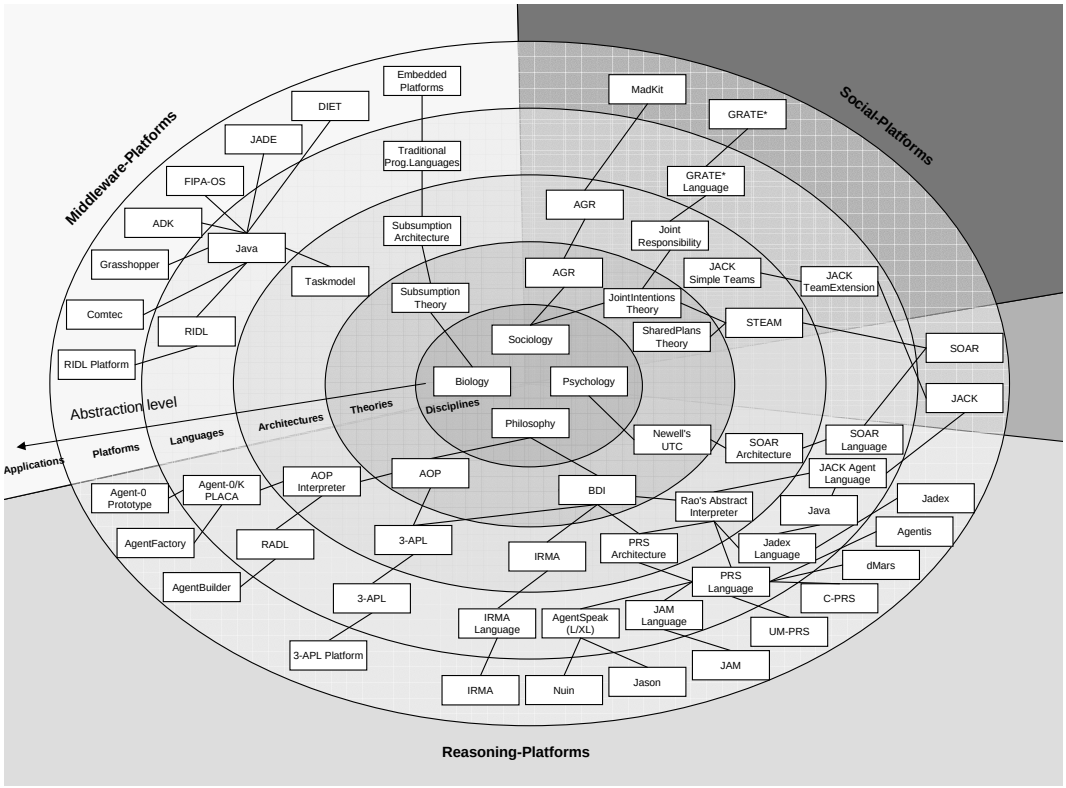

Abbildung 1: Agententechnologie Landschaft

te finden sich auch in den Ansätzen zur Organisationsmodellierung der AUML [13]. Trotz der vielen Spezialtheorien fehlt in diesem Bereich zur Zeit eine umfassende integrative Theorie zur Beschreibung aller relevanten Aspekte auf geeignetem Abstraktionsniveau.

2.2 Architekturen

Auf dem Weg, diese Theorien in die softwaretechnische Ebene zu übertragen, werden *Architekturen* entwickelt. Diese definieren die notwendigen Strukturen und Abläufe für die Konstruktion theoriekonformer Plattformen. Beim Übergang von der Theorie zu einer Architektur müssen Entscheidungen zur Konkretisierung gefällt werden, die von der Interpretation der jeweiligen Theorie abhängig sind. Je nach Eindeutigkeit der zu Grunde liegenden Theorie können zum Teil sehr unterschiedliche Architekturen für dieselbe Theorie entstehen, was z.B. sehr deutlich im Bereich der BDI-Architekturen erkennbar ist. Kerneigenschaft aller Architekturen für Agentenverhalten ist es, einen Verarbeitungszyklus (oft interpreter cycle genannt) festzulegen, der die Schritte vom Wahrnehmen der Umwelt bis zum Ausführen einer Aktion beschreibt. Für alle einflussreichen Architekturen wie dem abstrakten BDI Interpreter [14], dem AOP Interpreter [15], 3APL [6] und SOAR [10] wurden derartige abstrakte Verarbeitungsfolgen definiert.

Neben diesen theoriegetriebenen Architekturen existieren in der Informatik noch weitere Architekturen, die ohne Bezug zu Theorien aus anderen Disziplinen als softwaretechni-

sche Grundlage konkreter Plattformen dienen. Am häufigsten sind Ansätze, die durch Komposition von einzelnen Verhaltensbausteinen (meist als tasks oder behaviours bezeichnet) die Funktionalität eines Agenten beschreiben. Diese werden in der Abbildung unter dem Begriff Taskmodell zusammengefasst.

2.3 Sprachen

Um zu einsetzbaren Plattformen zu gelangen ist es notwendig, *Programmiersprachen* passend zur Architektur und Theorie zu konzipieren. Die Sprachen sind von besonderer Wichtigkeit, da sie den Zugangspunkt für Softwareentwickler auf Implementationsebene darstellen. Aus der Grafik lässt sich ersehen, dass einerseits eine Vielzahl unterschiedlicher Agentensprachen existiert und dass diese meist mit einer speziellen Architektur und einem konkreten System eng verwoben sind. Andererseits gibt es eine Reihe von Ansätzen, die existierende objektorientierte Sprachen wie Java erweitern oder direkt zur Realisierung von Agenten einzusetzen.

Beide Ansätze bieten Vor- und Nachteile. So erlauben spezielle Agentensprachen, die in Analyse und Design verwendeten Abstraktionskonzepte direkt als Teil der Programmiersprache zu verwenden. Dies leitet zudem den Programmierer dazu an, in agentenorientierten Konzepten zu denken. Ein Nachteil von Agentensprachen ist der oftmals hohe Einarbeitungsaufwand, der aufgrund der starken Divergenz agentenorientierter Konzepte für jedes System erneut notwendig wird. Traditionelle Programmiersprachen hingegen beruhen auf einem Konsens von Konzepten, wie z.B. Polymorphie, Vererbung und Zustandskapselung in objektorientierten Sprachen. Sie sind außerdem weit verbreitet und daher den meisten Entwicklern bereits bekannt. Zudem gibt es zu existierenden Programmiersprachen ausführliche Dokumentation und eine Vielzahl von Entwicklungswerkzeugen wie Integrated Development Environments (IDE), die im Rahmen der agentenorientierten Softwareentwicklung weiter eingesetzt werden können.

Teilt man die Sprachen in die unterschiedlichen Kategorien für Programmiersprachen (funktional, objektorientiert und logikbasiert) ein, stellt man fest, dass nahezu alle Sprachen mit Theoriebezug logikbasiert sind, wohingegen diejenigen ohne Theoriefundierung auf Basis des objektorientierten Paradigmas entstanden sind. Nur wenige Sprachen versuchen, eine objektorientierte Sprache um theoretisch fundierte agentenspezifische Konzepte zu erweitern.

2.4 Plattformen

Die verfügbaren *Plattformen* lassen sich grob in die drei Kategorien Middleware, Intelligente und Soziale Plattformen einteilen. Middleware Plattformen haben gemeinsam, dass sie eine robuste, skalierbare Basis für die Ausführung von Agenten (hier im Sinne autonomer Prozesse) bereitstellen und auf theoriefundierte Beschreibungen von Agenten zugunsten einer einfachen aufgabenorientierten Sichtweise (Taskmodell) verzichten. Aus diesem Grund werden Agenten innerhalb dieser Plattformen meist mit Hilfe von objektorientierten Sprachen realisiert und kapseln damit in vielen Fällen relativ einfaches Verhalten. Die Open-Source Plattform JADE von der TILAB[2] und das kommerzielle ADK (Agent Development Kit) von Tryllian sind typische Middleware-Plattformen. Beide Systeme setzen im Sinne interoperabler MAS die FIPA-Standards[3] für Agentenkommunikation um und ergänzen diese um weitere Middleware-Funktionalitäten wie z.B. Mobilitäts-, Sicherheits- und Persistenzdienste.

Unter dem Schlagwort Intelligente-Plattformen sind all diejenigen Systeme subsumiert, die auf einer Theorie zur Beschreibung von Individualverhalten beruhen (BDI, AOP oder UTC). Ziel der meisten Vertreter dieser Kategorie ist es, die Bausteine einer solchen Theorie

[2] Telecom Italia Laboratory
[3] http://www.fipa.org

in eine Agentensprache abzubilden und damit die Basis für die Beschreibung von intelligenten Agenten mit Hilfe mentalistischer Konzepte zu schaffen. Die große Mehrzahl dieser Agentensprachen wurde von Grund auf neu definiert. Typische Systeme dieser Kategorie sind im Bereich der BDI-Systeme JACK und Agentis, im Bereich der AOP-Systeme AgentFactory und im Bereich der UTC-Systeme SOAR. Jadex ist ein hybrider Ansatz mit dem Ziel BDI Reasoning für existierende Middleware Plattformen zu unterstützen.

In der Kategorie Soziale-Plattformen sind die Systeme eingeordnet, die auf Basis von Theorien zur Beschreibung von Agentengesellschaften beruhen. Je nachdem ob die zugrundeliegende Theorie mehr die strukturellen (AGR, AUML) oder die verhaltensbezogenen Faktoren (JointIntentions) in den Vordergrund stellt, unterstützt das System mehr die Abbildung vorhandener Strukturen der Realwelt oder konzentriert sich auf die Realisierung von Teamwork über gemeinsame Zielvorgaben. Verfügbare Systeme wie z.B. MadKit, das das AGR-Modell umsetzt und JACK, welches den sog. SimpleTeams Ansatz zur Unterstützung von BDI-Teams verwirklicht, bieten partielle Unterstützung unter bewusster Ausblendung bestimmter Charakteristika wie z.B. komplexen Mechanismen der Gruppenbildung.

Eine Übersicht der hier kurz präsentierten Systeme ist in Tab. 1 dargestellt. Es finden sich in der Auswahl an Plattformen sowohl kommerzielle Systeme, die z.T. als Trial-Versionen verfügbar sind, als auch eine Reihe von Open-Source Lösungen, die direkt aus dem Internet heruntergeladen und getestet werden können. Neben der Systemplattform selbst, spielen auch die vorhandenen Entwicklungswerkzeuge und der Dokumentationsgrad eine wichtige Rolle für ihre Einsetzbarkeit. In der Tabelle ist aufgeschlüsselt, welche Art von Werkezeugen für die jeweiligen Plattformen zur Verfügung stehen. Dabei wird hier grob zwischen IDEs (I) zur Programmierung von Source-Code, Administrationstools (A) zur Laufzeitverwaltung von Agenten und Debugging Werkzeugen (D) zur Fehlersuche unterschieden. Zumindest die beiden letztgenannten Werkzeugarten werden von den meisten Systeme bereits unterstützt. Für alle genannten Systeme gibt es umfangreiche Anleitungen und zum Teil sogar Tutorials, die eine Schritt-für-Schritt Einarbeitung ermöglichen.

3 Anwendungsfelder

Dieser Abschnitt beschäftigt sich mit der Frage, welche Eigenschaften Anwendungsfelder für Agententechnologie interessant machen und auf welchen Gebieten in welchen Applikationen typische Vertreter der drei Kategorien Middleware, Reasoning und Soziale Plattformen eingesetzt werden. Nach Weiß [18] ist Agententechnologie im Besonderen für diejenigen Anwendungsfelder von großem Interesse, die aus vielen interagierenden Komponenten bestehen und mindestens eins der folgenden Merkmale aufweisen:

- Die Komponentenmenge ist nicht im vorhinein bekannt und Komponenten können zur Laufzeit hinzukommen oder wieder verschwinden (offenes System).
- Eine Fremdkontrolle der Komponenten ist nicht möglich oder nicht erwünscht (autonome Einheiten).
- Koordination findet im System durch komplexe Kommunikationsbeziehungen statt (Verhandlungen).

Beispiele für den industriellen Einsatz von Agententechnologie sind mittlerweile zahlreich zu finden. Leider beschreiben die meisten wissenschaftlichen Veröffentlichungen experimentelle Prototypen und Demonstratoren, die (bisher) nicht in der Praxis eingesetzt werden. Ausnahmen finden sich u.A. in [8] und in [4]. Vielfach werden über erfolgreich umgesetzte Projekte und angebotene Lösungen jedoch Case-Studies und White Papers veröffentlicht, z.B. auf den Web-Seiten der Software Hersteller

Name	Kategorie	Informationen im Web	Verfügbarkeit	Werkzeuge	Dokumentation
JADE	Middleware	http://jade.tilab.com	Open Source	A, D	Tutorials, Guides
Tryllian ADK	Middleware	http://www.tryllian.com	Kommerziell	A	Userguide
JACK	Reasoning	http://www.agent-software.com	Trial-Version	I, D	Tutorial, Guides
Agentis	Reasoning	http://www.agentissoftware.com	Kommerziell	I, A, D	Nicht bekannt
Jadex	Reasoning	http://jadex.sourceforge.net	Open Source	A, D	Tutorial, Guides
AgentFactory	Reasoning	http://agentfactory.sourceforge.net	Open Source	I, A, D	Guides
SOAR	Reasoning	http://sitemaker.umich.edu/soar	Open Source	I, D	Tutorial, Guides
	Reasoning	http://www.soartech.com	Kommerziell	I, D	Nicht bekannt
MadKit	Sozial	http://www.madkit.org	Open Source	A, D	Guides

I: IDE, A: Administration, D: Debugging

Tabelle 1: Ausgewählte Agentenplattfomen und Frameworks

wie Agentis[5], Soar-Technology,[4] Tryllian[3], oder der Plattformen selbst (z.B. JADE).[5]

In den letzten Abschnitten wurde die Heterogenität der unterschiedlichen agentenbasierten Ansätze und schließlich auch der daraus abgeleiteten Plattformen herausgestellt. Die Konsequenz aus der Vielfalt existierender Plattformen ist, dass trotz der oftmals behaupteten allgemeinen Anwendbarkeit der Systeme, der Anwendungskontext ein entscheidendes Kriterium für die Wahl einer geeigneten Plattform darstellt. Dies wird insbesondere durch die unterschiedlichen Wurzeln der Systeme verursacht, die verschiedene Aspekte fokussieren. Im Folgenden werden daher für die drei Plattformkategorien typische Anwendungen herausgegriffen und kurz vorgestellt.

[4]http://www.soartech.com
[5]http://jade.tilab.com

3.1 Anwendungen von Middleware Plattformen

Middleware Plattformen werden in offenen, heterogenen Umgebungen eingesetzt. Der Rückgriff auf objektorientierte Ansätze vereinfacht die Integration mit Altsystemen. Neben der Robustheit und Skalierbarkeit sind gerade in offenen Umgebungen Sicherheitsfragen von Interesse, die daher auch von den meisten Plattformen adressiert werden. Weitere Dienste wie die Mobilität (d.h. Migration) von Agenten oder das Ausführen von Agenten auf mobilen (ressourcenschwachen) Geräten erweitern den Einsatzkontext in den Consumer Bereich, da Persönliche Agenten auf den Desktop des Anwenders migrieren oder diesen auf einem Mobilgerät begleiten können.

Im Bereich der Integration von Geschäftsprozessen wird zur Zeit großes Potential für Agententechnologie gesehen. Viele Firmen wie Whitestein, Tryllian, Calico Jack und Acklin sind intensiv auf diesem Gebiet tätig. Das Virtual Harbor Projekt von Tryllian ist ein Beispiel dafür, wie Agententechnologie eingesetzt werden kann, um die Kommunikation zwischen beteiligten Prozesspartern z.B. im Bereich Containerlogistik zu verbessern und zu automatisieren. Als typisches Beispiel für eine Anwendung im mobilen Kontext kann das mPower System [9] dienen, das von BT Exact zur Unterstützung und Koordination von Arbeitern im Außendienst auf Basis der JADE/LEAP Plattform entwickelt wird.

3.2 Anwendungen von Reasoning Plattformen

Typische Anwendungen auf Basis von Reasoning Plattformen sind Systeme zur Entscheidungsunterstützung und -automatisierung. Durch die ausgefeilten Konzepte zur Beschreibung der agenteninternen Strukturen und Entscheidungsprozesse wird die Konzeption und Realisierung derartiger Systeme erleichtert. Die Ähnlichkeit des Agentenaufbaus zu Aspekten des menschlichen Denkens und Handelns kann dabei direkt ausgenutzt werden und ist u.a. im der Bereich der Abbildung von menschenähnlichem Verhalten auf Softwarestellvertreter von Vorteil.

Diese einfache Abbildung findet z.B. vielfältigen Einsatz im Bereich der Erstellung von agentenbasierten Simulationsszenarien. So werden insbesondere SOAR- und BDI-basierte Agentenplattformen häufig für derartige Anwendungen eingesetzt (siehe z.B. [17]). Ebenso Anwendung findet diese Eigenschaft im Bereich der Steuerung autonomer Systeme. So wurde im Avatar/JACK Projekt [7] gezeigt, dass mit der heute verfügbaren Technologie bereits der vollständig autonome Betrieb unbemannter Fluggeräte möglich ist.

3.3 Anwendungen von Sozialen Plattformen

Soziale Plattformen werden in Umgebungen verwendet, in denen die Strukturen der Realwelt aus Gruppengefügen und Machtstrukturen bestehen und diese eine wichtige Rolle für das zu realisierende System spielen, wie z.B. Entscheidungsprozesse in Unternehmen oder militärische Kommandostrukturen. Weiterhin sind vielfältige Szenarien interessant, in denen es nicht ausreicht, wenn jeder Teilnehmer seine eigenen Interessen durchzusetzen sucht, sondern globale Bedingungen entscheidenden Einfluss haben. Durch diese globalen Vorgaben wird eine Koordination zwischen den Akteuren notwendig, die mit Hilfe von sozialen Strukturen explizit beschrieben und durchgeführt werden kann.

Im Bereich der militärischen Anwendungen werden von der DARPA (Defense Advanced Research Projects Agency) viele Forschungsprojekte gefördert, die das Ziel haben, bestehende militärische Strukturen zu flexibilisieren. Die Agententechnologie liefert das Rüstzeug zur Abbildung von individuellen Einheiten wie auch daraus rekursiv zusammengesetzten Verbänden (siehe z.B. SOFSoar[6]). Neben dem direkten Einsatz der Technologie, z.B. zur Steuerung von Einheiten, werden auch aufwendige Simulationssysteme zur Analyse von Szenarien und Kampfhandlungen eingesetzt [12].

4 Bewertung

In Nachfolgenden soll betrachtet werden, welche Faktoren für die Etablierung einer neuen Technologie auf breiter Basis notwendig sind. Ausgangspunkt und größter Motivationsfaktor ist die Situation der Anwender einer Technologie und der Leidensdruck, der durch Unzulänglichkeiten der momentan adaptierten Mechanismen entsteht und den Wunsch nährt, kostengünstiger und effektiver Software zu erstellen.

Damit sich ein neues Paradigma verbreiten kann, müssen sowohl Faktoren auf der konzeptuellen als auch auf der Realisierungsebene erfüllt sein. Die Konzepte einer neuen Technologie sollten allgemein anerkannt sein und einen Vorteil in der Modellierung erbringen (etwa durch größere Ausdrucksmächtigkeit, einfachere Beschreibung oder Nähe zur Anwendungsdomäne). Im Sinne der Kontinuität sollten bewährte Konzepte bestehender Paradigmen so weit wie möglich bewahrt werden. Auf der Realisierungsebene sind etablierte Standards sowie ausgereifte Entwicklungsprozesse und Modellierungsmethoden notwendig. Diese sollten durch konkrete Werkzeuge für den Entwurf (CASE-Tools), die Programmierung (Programmiersprachen und IDEs) und für den täglichen Betrieb (Applikationsumgebungen) unterstützt werden.

Die Agententechnologie bietet durch die höhere Abstraktion gegenüber bestehenden Paradigmen den Vorteil, Probleme nä-

her an der Anwendungsdomäne zu beschreiben. Sie wird heutzutage weniger als Gegenentwurf zur Objektorientierung gesehen, sondern vielmehr als notwendige Ergänzung und Erweiterung betrachtet [18]. Zur Zeit mangelt es jedoch an Einigkeit über die grundlegenden Konzepte. Dies kann als größtes Manko angesehen werden und spiegelt sich auf der Realisierungsebene in den verschiedenen in Abschnitt 2 vorgestellten Strömungen wieder.

Dessen ungeachtet sind im Bereich der Plattformen und Sprachen bereits ausgereifte Produkte verfügbar und für Middleware Plattformen gibt es mit FIPA bereits anerkannte Standards. Zunehmend rücken damit auch die sekundären Aspekte der Anwendungsrealisierung wie graphische Entwurfswerkzeuge (etwa bei JACK und Agentis) und Ausführungsumgebungen (z.B. auf Basis von J2EE-konformen Application Servern) in den Fokus der Systemhersteller. Bezüglich agentenorientierter Entwicklungsprozesse zeichnet sich allerdings bisher kein Konsens ab. Kritisch zu sehen ist insbesondere, dass bei Bewertung und Vergleich von agentenorientierten Vorgehensweisen die plattformspezifischen Aspekte bisher zu wenig berücksichtigt werden [16]. Lediglich bei den Modellierungsmethoden gibt es mit der AUML Arbeitsgruppe[6] Bestrebungen zur Vereinheitlichung bestehender Ansätze.

[6] http://www.auml.org

5 Zusammenfassung

Ziel dieses Artikels ist es, dem Leser einen Überblick über die aktuelle Landschaft der Agententechnologie zu geben und die Querbezüge zwischen verfügbaren Plattformen, Sprachen, Architekturen und den zugrunde liegenden Theorien zu verdeutlichen. Durch mit Beispielen belegte Anwendungsfelder wird das Einsatzpotential der Agententechnologie verdeutlicht. Zudem wird die Frage diskutiert, was Agententechnologie noch vom Mainstream des Software Engineerings trennt.

In der Entwicklung von der maschinennahen, über die strukturierte hin zur objektorientierten Programmierung wurde bewusst immer weiter von der Datenverarbeitung im Rechnersystem abstrahiert und eine den Anwendungsdomänen nähere Beschreibungsform gewählt. Agenten könnten der nächste logische Schritt in dieser Kette sein, da sie sich gegenüber anderen Nachfolgern der Objektorientierung wie Komponenten oder Web-Services, dadurch auszeichnen, dass sie nicht nur eine technische Lösung darstellen, sondern auf allen Ebenen der Anwendungsentwicklung neue Abstraktionskonzepte bereitstellen.

Die Etablierung der Agententechnologie in den Mainstream der Softwareentwicklung hängt massgeblich davon ab, ob es gelingt, die bestehenden Probleme, vor allen Dingen den fehlenden Konsens bezüglich der Konzepte und den Mangel an ausgereiften Entwicklungsprozessen, zu lösen.

Literatur

[1] M. Bratman. *Intention, Plans, and Practical Reason*. Harvard University Press, Cambridge, Massachusetts, 1987.

[2] R. Brooks. A Robust Layered Control System For A Mobile Robot. *IEEE Journal of Robotics and Automation*, 2(1):24–30, March 1986.

[3] P. R. Cohen and H. J. Levesque. Teamwork. Technote 504, SRI International, Menlo Park, CA, March 1991.

[4] R. Evertsz, M. Fletcher, R. Jones, J. Jarvis, J. Brusey, and S. Dance. Implementing Industrial Multi-agent Systems Using JACK. In M. Dastani, J. Dix, and A. Seghrouchni, editors, *Programming Multi-Agent Systems (PROMAS 2003)*, pages 18–48. Springer, 2004.

[5] J. Ferber, O. Gutknecht, and F. Michel. From Agents to Organizations: an Organizational View of Multi-Agent Systems. In P. Giorgini, J. Müller, and

J. Odell, editors, *AOSE*, volume 2935 of *Lecture Notes in Computer Science*, pages 214–230. Springer, 2003.

[6] K. Hindriks, F. De Boer, W. Van der Hoek, and J.-J. Meyer. Agent Programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, November 1999.

[7] J. Hinz and G. Ferguson (eds.). Avatar JACKs up connectivity. *Australian Defence Magazine*, 12(8), 2004.

[8] N. R. Jennings and M. J. Wooldridge. *Agent Technology - Foundations, Applications and Markets*. Springer Verlag, 1998.

[9] H. Lee, P. Mihailescu, and J. Shepherdson. Empowering the mobile workforce. White paper 80081, Issue 1 (06/04/04), BT Exact, 2004.

[10] J. F. Lehman, J. E. Laird, and P. S. Rosenbloom. A gentle introduction to Soar, an architecture for human cognition. *Invitation to Cognitive Science*, 4, 1996.

[11] A. Newell. *Unified Theories of Cognition*. Harvard University Press, 1990.

[12] P. E. Nielsen, D. Smoot, R. Martinez, and J. D. Dennison. Participation of TacAir-Soar in RoadRunner and COYOTE Exercises of Air Force Research Lab, Meza AZ. In *Proceedings of the 9th Annual Conference on Computer Generated Forces and Behavioral Representation (CGF-BR-00)*, 2000.

[13] J. J. Odell, H. Van Dyke Parunak, and M. Fleischer. The role of roles in designing effective agent organizations. In *Software Eng. for Large-Scale MAS*, pages 27–38. Springer, 2003.

[14] A. Rao and M. Georgeff. BDI Agents: from theory to practice. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95)*, pages 312–319, San Francisco, CA, USA, 1995. The MIT Press: Cambridge, MA, USA.

[15] Y. Shoham. Agent-oriented programming. In D. G. Bobrow, editor, *Artificial Intelligence Volume 60*, pages 51–92, Elsevier Amsterdam, The Netherlands, 1993.

[16] J. Sudeikat, L. Braubach, A. Pokahr, and W. Lamersdorf. Evaluation of Agent-Oriented Software Methodologies – Examination of the Gap Between Modeling and Platform. In *Proceedings of the Fifth International Workshop on Agent-Oriented Software Engineering (AOSE-2004)*, 2004.

[17] G. Taylor, R. Frederiksen, R. Vane, and E. Waltz. Agent-based Simulation of Geo-Political Conflict. In D. McGuinness and G. Ferguson, editors, *Proc. of the 16th Conference on Innovative Applications of Artificial Intelligence*, pages 884–891. AAAI Press, 2004.

[18] G. Weiß. Agent Orientation in Software Engineering. *Knowledge Engineering Review*, 16(4):349–373, 2002.

[19] M. Wooldridge and N. Jennings. Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.