Autonome Dynamische Rekonfiguration im kooperativen Problemlösungsprozess

Markus Hannebauer

Fraunhofer FIRST
Kekuléstr. 7, 12489 Berlin
markus.hannebauer@first.fraunhofer.de

Moderne Produktzyklen werden immer kürzer und verlangen ständige Umstrukturierungen in der betrieblichen Organisation. In gleichem Maße werden zunehmend höhere Anforderungen an Software gestellt, die betriebliche Prozesse unterstützen soll. Im Rahmen des Forschungsvorhabens AUREON wurden die theoretischen, konzeptuellen und praktischen Grundlagen geschaffen, die es einer IT-Infrastruktur ermöglichen, Umstrukturierungsprozesse dynamisch zu unterstützen bzw. sogar vorwegzunehmen.

AUREON steht als Akronym für Autonome Dynamische Rekonfiguration. Diese neue Technologie implementiert einen Meta-Prozess, der einen kooperativen Arbeits- und Problemlösungsprozess zur Laufzeit begleitet und in dem intelligente Software-Agenten durch individuelle Verschmelzung und Zerteilung die Struktur ihrer eigenen Organisation selbstständig bestimmen. Anhand einer realistischen Fallstudie in der medizinischen Terminplanung konnte nachgewiesen werden, dass durch den Einsatz von AUREON der kooperative Prozess zwischen den Agenten deutlich effizienter abläuft und zu nachweislich besseren Ergebnissen führt.

1 Einleitung

Oft kann ein komplexes Problem nicht von einem Individuum allein gelöst werden, da ihm die Ressourcen oder Fähigkeiten dazu fehlen. Ein typischer und erfolgreicher Ausweg ist die Kooperation mit anderen, die sowohl hierarchisch als auch nichthierarchisch ausgeprägt sein kann. Kooperation basiert dabei zwangsläufig auf einer Verteilung von Problemlösungswissen, -zielen, -ressourcen und -fähigkeiten auf Individuen. Diese Verteilung nennen wir *Konfiguration*.

Im Rahmen unserer Forschung haben wir das Vorbild der menschlichen Kooperation aufgegriffen und auf künstliche Multiagentensysteme übertragen. Unsere Hauptthese ist, dass der Erfolg einer kooperativen Problemlösung maßgeblich von einer sinnvollen Konfiguration abhängt. Die Motivation dafür ist die Beobachtung, dass existierende Multiagentensysteme aufgrund falscher Konfiguration oft unzureichende Ergebnisse liefern. Unter unzureichenden Ergebnissen verstehen wir Lösungen mit inakzeptabel schlechter Qualität oder Lösungen, die erst nach unzumutbar langer Zeit oder unter hoher Ressourcenauslastung

gefunden werden. Wir haben diese Probleme adressiert und den Prozess der kooperativen Problemlösung verbessert, indem nicht der Prozess selbst verändert wird, sondern indem adaptiv die Konfiguration des Systems angepasst wird, also das System adaptiv rekonfiguriert wird. Dazu sind Mechanismen entwickelt worden, die diese Anpassung dynamisch und autonom vornehmen. Unser Konzept haben wir daher *Autonome Dynamische Rekonfiguration* (AUREKON, [Han02]) genannt.

Im Unterschied zu vielen anderen Ansätzen [Dur99, CG99] zielt unser Ansatz dabei eher auf die individuelle Ebene (Mikroebene) einzelner Agenten als auf die soziale Ebene (Makroebene), da die Entscheidung, einen Agenten mit unterschiedlichem Wissen, Zielen, Ressourcen und Fähigkeiten auszustatten, das Verhalten des Agenten direkt beeinflusst. Nichtsdestotrotz hat die Festlegung der Konfiguration auf der individuellen Ebene erhebliche Auswirkungen auf die Makroebene und damit auf das Kommunikationsverhalten der Agenten im Multiagentensystem.

Das Konzept der Autonomen Dynamischen Rekonfiguration adressiert zwei wesentliche Problemfelder:

- Zum einen wird versucht, natürliches organisatorisches Verhalten zu verstehen und nachzubilden. Das Multiagentensystem kann so den betrieblichen Prozess analysieren und sich entsprechend anpassen.
- Zum anderen wird es ermöglicht, äußerst flexible und dabei aber effiziente verteilte Systeme zu entwerfen. Es kann sowohl die Funktionalität des Systems zu seiner Laufzeit geändert werden, als auch auf Veränderungen in der Umwelt des Systems reagiert werden. Das System kann in beiden Fällen nach einer Änderung wieder eine effiziente Konfiguration erreichen.

Die Hauptidee zur Umsetzung Autonomer Dynamischer Rekonfiguration sind die beiden Operatoren *Agentenverschmelzung* (engl. *agent melting*) und *Agentenzerteilung* (engl. *agent splitting*). Agentenverschmelzung bezeichnet den Prozess, bei dem ein Agent von einem anderen Problemlösungswissen, -ziele, -ressourcen und -fähigkeiten übernimmt. Agentenzerteilung hingegen bezeichnet den Prozess, in dem ein einzelner Agent Teile seines Problemlösungswissens, seiner -ziele, -ressourcen und -fähigkeiten an einen neu erzeugten oder schon existierenden anderen Agenten übergibt. Abbildung 1 stellt drei mögliche Konfigurationen des gleichen Problems dar, die durch konsekutive Anwendung von Agentenverschmelzung auseinander entstanden sind.

Für unser Kernkonzept lässt sich folgende griffige Charakterisierung angeben.

> Autonome Dynamische Rekonfiguration ist ein Meta-Prozess, der einen kooperativen Problemlösungsprozess begleitet und in dem intelligente Software-Agenten die Struktur ihrer eigenen Organisation selbständig und auf der individuellen Ebene bestimmen. Grundlage dafür ist die Verschmelzung und Zerteilung von Agenten und somit die Neuverteilung von Problemlösungswissen, -zielen, -ressourcen und -fähigkeiten.

Die Umsetzung Autonomer Dynamischer Rekonfiguration verlangte nach einem Rahmenwerk, das das dynamische Zuweisen und Entziehen von Wissen, (Teil-)Aufgaben und Fä-

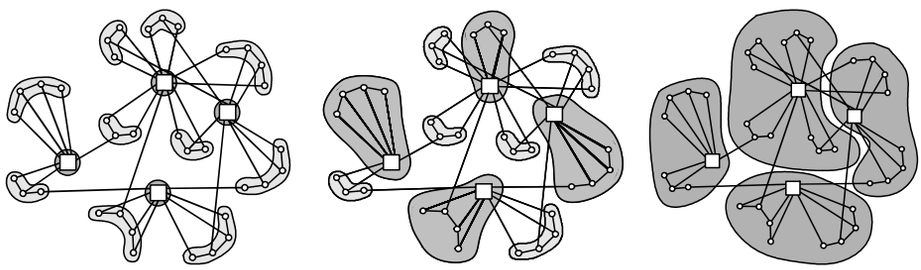

Abbildung 1: Rekonfiguration durch Agentenverschmelzung

higkeiten auf Agenten in einem Multiagentensystem gestattet. Dies hat Auswirkungen auf die Architektur sowohl auf der Mikroebene, in der festgelegt wird, wie ein einzelner Agent aufgebaut ist, als auch auf der Makroebene des Multiagentensystems, die das Zusammenspiel der Agenten und das Agentenmanagement beschreibt. Schließlich mussten Überlegungen zur Verknüpfung dieser beiden Ebenen in einer Rekonfigurationssteuerung und zu Werkzeugen für die Evaluierung angestellt werden.

2 Mikroebene: Ein Agent und seine Entscheidungen

Wir haben die einzelnen Agenten unseres Multiagentensystems nach dem *Belief-Desire-Intention-Ansatz* (BDI-Ansatz, [RG95]) aufgebaut. Durch die Trennung von Glauben (Belief), Wünschen (Desires) und Absichten (Intentions) erlaubt der BDI-Ansatz das Auflösen von Wunschkonflikten, sowie das Zusammenfassen mehrerer untereinander konsistenter Wünsche in einer Absicht. Absichten stehen für die machbaren und zum Ziel erhobenen Wünsche des Agenten einschließlich eines groben Umsetzungsschemas. Agenten besitzen zusätzlich Fähigkeiten, um diese Umsetzungsschemata schrittweise zu verfeinern und so auszufüllen.

Der BDI-Ansatz eignet sich auf natürliche Weise für Rekonfiguration. Die Konfiguration eines Multiagentensystems entspricht der Verteilung von Glauben, Wünschen, Absichten und Fähigkeiten auf Agenten. Rekonfiguration bedeutet also, dass Glauben, Wünsche, Absichten oder Fähigkeiten zwischen Agenten ausgetauscht werden; sie sind dem Agenten nicht fest zugeordnet, sondern können sich zur Laufzeit ändern. Um dies zu ermöglichen, haben wir bekannte Techniken aus der komponentenorientierten Programmierung angewandt: Wir haben Schnittstellen entworfen, die die Syntax und Semantik festlegen, über die Glauben, Ziele, Absichten und Fähigkeiten (sogenannte *mentale Komponenten*) eines Agenten verändert werden können. Einen so strukturierten Agenten haben wir *komponierbaren BDI-Agenten* genannt [MÖ1].

Unser komponierbarer BDI-Agent besteht aus einem generischen Kern, der die mentalen Komponenten verwaltet. Dieser Agentenkern ist so entworfen, dass ihm zur Laufzeit beliebige mentale Komponenten hinzugefügt oder entzogen werden können. Der komponierbare BDI-Agent bildet so die Basis für Autonome Dynamische Rekonfiguration. Diese

Architektur ist weitgehend frei von domänenabhängigem Anwendungswissen, eignet sich also für viele Anwendungsbereiche, in denen in einer dynamischen, unsicheren Umwelt ein rekonfigurierbares Multiagentensystem realisiert werden soll.

Neben dem BDI-Reasoning steht es jeder Implementierung eines BDI-Agenten frei, weitere Reasoning-Aufgaben direkt in mentalen Komponenten umzusetzen. Ein Beispiel dafür ist unsere *Distribution-Aware-Constraint-Specification-Architecture* (DACSA, [Han00f, Han00e]). DACSA wandelt die objektorientierte Darstellung des von einem Agenten zu lösenden Problems in eine contraint-logische Darstellung, die dann direkt an einen contraint-logischen Löser, wie SICStus Prolog [Swe01], übergeben werden kann. Mit DACSA kann ein Agent sowohl sein eigenes Wissen als auch externes Wissen über Problemelemente, wie Variablen, Bedingungen, Optimierungskriterien und Lösungsheuristiken, in eine deklarative Spezifikation umwandeln.

Wichtig für den Rekonfigurationsprozess ist außerdem, dass DACSA selbst aus einer Menge getypter Komponenten – *Fabriken* und *Objekte* – besteht, von denen eine Agent beliebig viele besitzen kann. Rekonfiguration im Sinne des Constraint-Lösens kann daher einfach durch Zusammenführen bzw. Auseinandernehmen der Menge von Fabriken und Objekten geschehen, die ein Agent in seiner DACSA-Instanz besitzt.

3 Makroebene: Mehrere Agenten und ihre Kooperation

Der komponierbare BDI-Agent bietet auf der Mikroebene die geeignete Struktur für Autonome Dynamische Rekonfiguration. Um Rekonfiguration tatsächlich in einem Multiagentensystem durchführen zu können, mussten auf der Makroebene Protokolle entworfen werden, die die Kooperation verschiedener komponierbarer BDI-Agenten erlauben und sicherstellen, dass trotz ständiger Rekonfiguration des Systems eine gute Lösung für die übertragenen Probleme gefunden wird.

Jeder unserer Agenten erzeugt zur Lösung seiner Probleme über DACSA constraintlogische Programme. Da ein Agent aber immer nur einen Teil eines gemeinsamen Problems bearbeitet, müssen die Lösungen der verschiedenen Agenten untereinander abgestimmt werden, um eine global konsistente Lösung zu erhalten. Dazu ist es zunächst nötig, dass ein Agent weiß, mit welchen anderen Agenten er sich abstimmen muss. Dieses Wissen erhält ein Agent in Form von Referenzen zusammen mit den ihm übertragenen Aufgaben. Da sich das System rekonfigurieren kann, zeigen diese Referenzen nicht direkt auf einen Agenten, mit dem Abstimmungsbedarf besteht, sondern müssen erst über einen Verzeichnisdienst aufgelöst werden.

Während in den aus der Literatur bekannten verteilten Problemlösungssystemen Agenten konkrete Lösungen und Zustimmungen bzw. Ablehnungen zu diesen Lösungen austauschen [YDIK98, SSHF00], tauschen komponierbare BDI-Agenten Constraint-Objekte aus, die für eine Menge von Einschränkungen aus der Lösungsmenge stehen. Der Agent, der für die Lösung eines Teilproblems verantwortlich ist, beginnt die Abstimmung für dieses Teilproblem durch den Versand einer Menge möglicher Lösungen. Alle empfangenden Agenten antworten darauf mit jeweils einem Constraint-Objekt, das die Menge der Lösun-

gen beschreibt, die bei ihnen zu keinem Konflikt führt. Alle beteiligten Agenten einigen sich dann in einem dritten Schritt auf eine konkrete Lösung aus dieser Menge. Wir verzeichnen durch diese Methode eine deutliche Reduzierung des Abstimmungsaufwands. Senden die Agenten jeweils noch Optimierungsobjekte mit den Constraint-Objekten mit, wird es den empfangenden Agenten möglich, Lösungen unter Berücksichtigung der Prioritäten des Kooperationspartners zu finden und so schneller eine gesamtoptimale Lösung zu finden.

Dieses mehrstufige Verfahren zur Abstimmung zwischen Agenten haben wir *Multiphase-Agreement-Finding-Protokoll* (MPAF, [Han00c, Han00b, Han00d, HM01]) genannt. Es baut auf DACSA auf, da es die von DACSA als Zwischenstufe erstellten Constraint- und Optimierungsobjekte als Einschränkungen bzw. Priorisierungen an andere Agenten zur Abstimmung senden kann und deren Antwort – ebenfalls Constraint-Objekte – wie eigene in das von DACSA erzeugte Programm einbauen kann. MPAF ist zudem konfigurationsinvariant, da es keine Rolle spielt, ob viele einfache Agenten über MPAF kommunizieren oder nur einige wenige hochkomplexe.

4 Steuerung der Rekonfiguration

Bevor die Rekonfigurationssteuerung entworfen werden konnte, musste zunächst ergründet werden, welchen Kriterien eine gute Konfiguration genügt. Wir haben zunächst mit den beiden Forderungen gearbeitet, dass eine gute Konfiguration zu möglichst wenig Kommunikations- und Koordinationsaufwand zwischen den Agenten führt und zudem jeder einzelne Agent seine Aufgaben mit dem ihm zugewiesenen Fähigkeiten in einer vom Nutzer angegebenen Zeit bearbeiten kann. Diese Forderungen wurden formalisiert und die Konsequenzen analytisch untersucht. Die Problemstellung, eine gute Konfiguration für ein gegebenes Problem zu finden, weist eine interessante mathematische Struktur auf und ist nachweislich schwer zu lösen. Durch die Charakterisierung von Agentenverschmelzung und Agentenzerteilung als Operationen auf Äquivalenzrelationen konnten wichtige Eigenschaften dieser Operationen nachgewiesen werden, wie Strukturerhalt, Wirkung, Hinlänglichkeit und nebenläufige Einsetzbarkeit [Han00a, Han02].

Da die von uns betrachteten Probleme inhärent verteilt sind und sich auch über Kompetenzgrenzen erstrecken können, können Rekonfigurationsentscheidungen nur jeweils lokal und autonom durch die jeweiligen Agenten getroffen werden. Es musste demnach analysiert werden, welches Wissen für gute Rekonfigurationsentscheidungen benötigt wird und wie es prozessbegleitend aus rein lokaler Sicht gesammelt werden kann. Aufbauend darauf wurden geeignete Rekonfigurationsstrategien entwickelt. Dazu gehören das statistische Erfassen des Nachrichtenaufwands, den das oben beschriebene MPAF-Protokoll zum Finden einer Lösung benötigt, und die Zeit, die ein Agent benötigt, um das durch DACSA erstellte contraint-logische Programm zu lösen.

Mit dem komponierbaren BDI-Agenten lässt sich die Rekonfigurationssteuerung elegant umsetzen. In einer Glaubenskomponente werden die prozessbegleitenden Daten gesammelt, indem eingehende und ausgehende Nachrichten protokolliert und Aufrufe des

constraint-logischen Lösers überwacht werden. Aus diesen Daten leitet der Agent einen Rekonfigurationswunsch ab, der durch eine entsprechende Wunschkomponente repräsentiert wird. Die Entscheidung über die Zerteilung eines Agenten wird optimal mithilfe von DACSA gefällt, die Entscheidung über die Verschmelzung zweier Agenten wird mithilfe Fallbasierten Schließens auf Grundlage des Erfahrungswissens des Agenten getroffen [Sch02]. Der aus der Entscheidung resultierende Rekonfigurationswunsch wird dann in einer Rekonfigurationsabsicht umgesetzt, die die entsprechenden Rekonfigurationsschritte durchführt. Diese basieren hauptsächlich auf der Möglichkeit, mentale Komponenten zu serialisieren und so an andere Agenten zu übertragen.

Da Rekonfigurationsentscheidungen von jedem Agenten autonom getroffen werden können, müssen die Agenten vor einer Verschmelzung ihre Entscheidungen untereinander abstimmen. So vermeiden sie die durch diese Asynchronität möglichen Konflikte, Deadlocks oder Racing Conditions. Wir haben geeignete, aus einer Anfragephase und einer Aktionsphase bestehende Protokolle entworfen, in denen Konflikte, wie „Crosstalk“, schon in der Anfragephase erkannt und behoben werden können, und jedem komponierbaren BDI-Agenten die Fähigkeit gegeben, diese Protokolle einzusetzen. Damit können die Agenten kritische Situationen erkennen und beheben.

5 Umsetzung und Evaluierung

Nach Erarbeitung der grundlegenden Mechanismen der Autonomen Dynamischen Rekonfiguration hatten wir uns vorgenommen, die Funktionalität, Effektivität und Effizienz unseres Ansatzes anhand eines realistischen Szenarios aus der Krankenhauslogistik nachzuweisen. Als Szenario haben wir die Terminplanung innerhalb einer Klinik unter Einbeziehung von Stations-, Ambulanz- und Privatpatienten gewählt.

Da unsere in diesem Stadium prototypische Software nicht in einem realen Umfeld im Krankenhaus eingesetzt wird, mussten wir unser System mit simulierten, auf realen Bedingungen an der Charité Berlin beruhenden Daten testen. Dafür wurde der *Hospital Scenario Generator* (HSG, [RH01]) entwickelt, der es erlaubt, prototypische diagnostische Infrastrukturen zu erzeugen, dynamische Patientenströme zu simulieren und so Terminplanungsprobleme zu spezifizieren.

Neben dem HSG haben wir weitere Werkzeuge für die Evaluierung von AUREON geschaffen, darunter einen Online-Monitor zur Beobachtung der gegenwärtigen Konfiguration, Logdateibetrachter, Terminplanbetrachter und eine 3D-VRML-Repräsentation beliebiger Konfigurationen. Abbildung 2 zeigt die Werkzeuge zur Online-Evaluierung von AUREON, insbesondere den Online-Monitor, der Agenten durch blaue Graphknoten, sowie Patienten durch gelbe und diagnostische Einheiten durch rote Knoten darstellt. Kanten zwischen Agenten und Patienten/Diagnoseeinheiten stehen für Verantwortungsbeziehungen, Kanten zwischen Agenten stellen dynamisch durch ihre Stärke den gegenwärtigen Kommunkationsaufwand dar.

Da man mit unseren Werkzeugen auch manuell Konfigurationen vorgeben kann, konnten wir die durch die Verwendung der Autonomen Dynamischen Rekonfiguration erzeugten

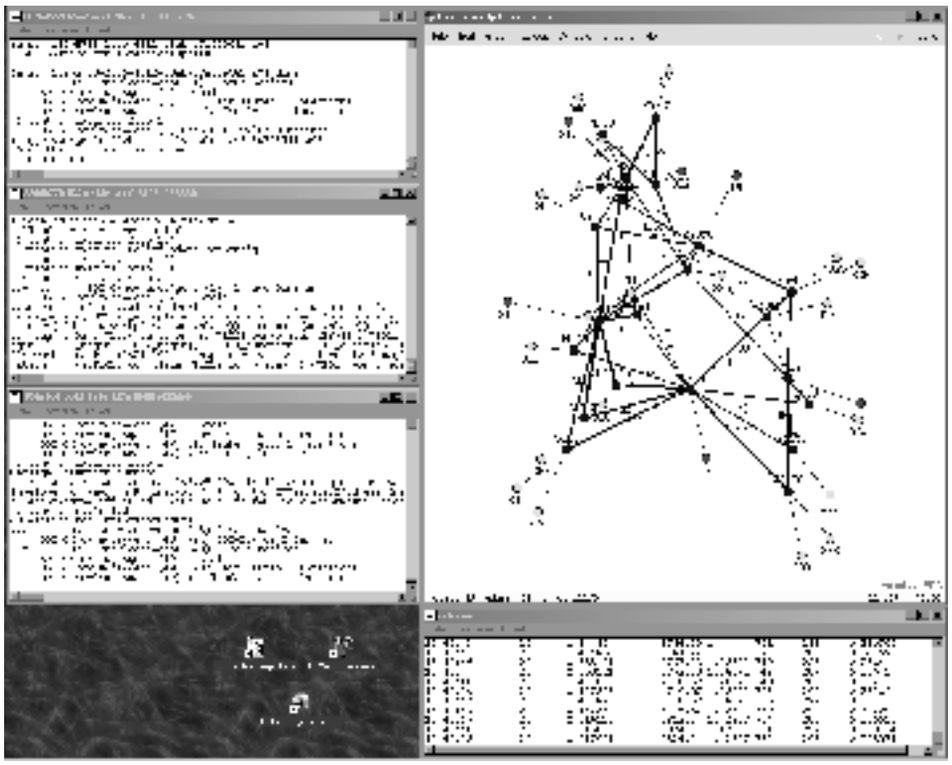

Abbildung 2: Werkzeuge für die Online-Evaluierung von AURECON

Konfigurationen direkt mit einer zentralen Konfiguration und einer hochverteilten Konfiguration vergleichen. Abbildung 3 zeigt die beiden extremen Konfiguration – zentral mit nur einem Agenten sowie hochverteilt mit einem Agenten pro aktiver Problemeinheit – sowie eine durch AURECON erzeugte Konfiguration, die deutliche Cluster-Bildung erkennen lässt. Abbildung 4 zeigt, wie AURECON Schritt für Schritt die Konfiguration anpasst. Agenten mit starkem Kommunikationsaufwand werden verschmolzen, während relativ unabhängige Agenten erhalten bleiben.

Dass die durch AURECON erzeugten Konfigurationen wirklich Vorteile gegenüber zentralistischem und hochverteiltem Lösen haben, wurde durch die Evaluierung acht verschiedener Qualitätsmaße nachgewiesen. Abbildung 5 zeigt exemplarisch den Vergleich in Bezug auf die Maße Patientenkalenderdichte und Nachrichtenaustausch. Messbares Resultat dieser Untersuchungen ist, dass ein mit AURECON automatisch konfiguriertes System bis zu 20% bessere Optimierungsresultate als ein hochverteiltes System und damit ähnlich gute Resultate wie ein zentraler Löser liefert. Gleichzeitig wird der Kommunikationsbedarf im Vergleich zur hochverteilten Konfiguration um zwei Drittel verringert.

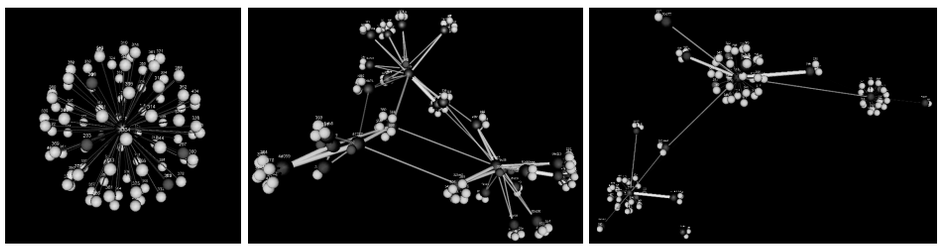

Abbildung 3: Extreme manuelle Konfigurationen und die durch AURECON erzeugte Konfiguration

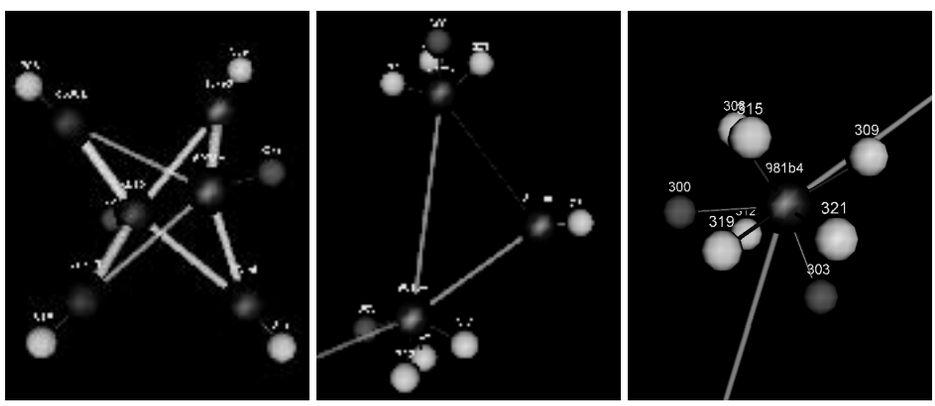

Abbildung 4: Beobachtung einer Abfolge von Agentenverschmelzungsschritten

6 Bilanz

Im Forschungsvorhaben AURECON wurde ein neuartiges Konzept für die Selbstorganisation von intelligenten Agenten von Grund auf erarbeitet, theoretisch fundiert, technisch realisiert und praktisch ausgewertet. In direktem Zusammenhang mit diesem Projekt sind innerhalb von zwei Jahren über 20 Veröffentlichungen erschienen, darunter drei Diplomarbeiten und eine Dissertation. Auf praktischer Seite ist ein System entstanden, das mit einem Umfang von mehr als 70.000 Programmzeilen in ein Reifestadium gelangt ist, praktisch erprobt werden zu können.

Wie jede innovative Forschungsarbeit wirft das AURECON-Projekt mehr neue Fragen auf als beantwortet werden. Weitere Arbeitsfelder sind dabei vorrangig durch die Frage bestimmt, wie sich die Steuerung der Rekonfiguration für die Bewältigung unsteter dynamischer Problementwicklung erweitern lässt, wie also Zyklen und sprunghaftes Rekonfigurationsverhalten vermieden und die Stabilität der Konfiguration gesichert werden können.

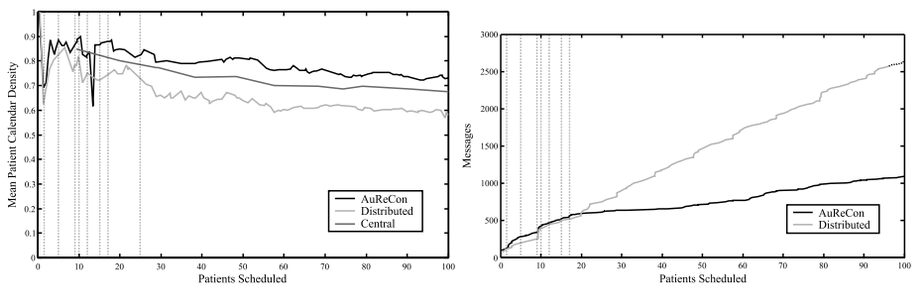

Abbildung 5: Nachweis der Qualitäts- und Effizienzsteigerungen durch AURECON

Literaturverzeichnis

[CG99] K. M. Carley and L. Gasser. Computational Organization Theory. In Gerhard Weiss, editor, *Multiagent Systems — A Modern Approach to Distributed Artificial Intelligence*, pages 165–199. MIT Press, 1999.

[Dur99] Edmund H. Durfee. Distributed Problem Solving and Planning. In Gerhard Weiss, editor, *Multiagent Systems — A Modern Approach to Distributed Artificial Intelligence*, pages 121–163. MIT Press, 1999.

[Han00a] Markus Hannebauer. A Formalization of Autonomous Dynamic Reconfiguration in Distributed Constraint Satisfaction. *Fundamenta Informaticae*, 43(1–4):129–151, 2000.

[Han00b] Markus Hannebauer. How to Model and Verify Concurrent Algorithms for Distributed CSPs. In Rina Dechter, editor, *Proceedings of the Sixth International Conference on Principles and Practice of Constraint Programming (CP-2000)*, volume 1894 of *LNCS*. Springer, 2000.

[Han00c] Markus Hannebauer. Multi-phase consensus communication in collaborative problem solving. In Günter Hommel, editor, *Communication-Based Systems*, pages 131–146. Kluwer, 2000.

[Han00d] Markus Hannebauer. On Proving Properties of Concurrent Algorithms for Distributed CSPs. In *Proceedings of the CP-2000 Workshop on Distributed Constraint Satisfaction*, Singapore, 2000.

[Han00e] Markus Hannebauer. Their Problems are my Problems — The Transition between Internal and External Conflict. In C. Tessier, L. Chaudron, and H.-J. Müller, editors, *Conflicting Agents: Conflict Management in Multi-Agent Systems*, pages 63–109. Kluwer, 2000.

[Han00f] Markus Hannebauer. Transforming Object-Oriented Domain Models into Declarative CLP Expressions. In Francois Bry, Ulrich Geske, and Dietmar Seipel, editors, *Proceedings of the 14th Workshop on Logic Programming (WLP-99)*, pages 65–76, Würzburg, Germany, 2000.

[Han02] Markus Hannebauer. *Autonomous Dynamic Reconfiguration in Multi-Agent Systems – Improving Collaborative Problem Solving (to appear)*, volume 2427 of *LNAI*. Springer, 2002.

[HM01] Markus Hannebauer and Sebastian Müller. Distributed constraint optimization for medical appointment scheduling. In *Proceedings of the Fifth International Conference on Autonomous Agents (AGENTS-2001)*, Montréal, Canada, 2001.

[MÖ1] Sebastian Müller. Ein komponentenbasierter BDI-Agent in einem rekonfigurierenden verteilten System. Master's thesis, Freie Universität Berlin, Germany, 2001.

[RG95] A. S. Rao and M. P. Georgeff. BDI Agents: From theory to practice. In V. Lesser, editor, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 312–319. MIT Press, 1995.

[RH01] Frank Rehberger and Markus Hannebauer. HSG 1.1 — Ein Java-basierter Problemgenerator für das Online-Scheduling im medizinischen Bereich (in German). GMD Report 133, German National Research Center for Information Technology, 2001.

[Sch02] Gunnar Schrader. Adaptive Steuerung autonomer dynamischer Rekonfiguration. Master's thesis, Technische Universität Berlin, Germany, 2002.

[SSHF00] M. C. Silaghi, D. Sam-Haroud, and B. Faltings. Distributed Asynchronous Search with Private Constraints. In *Proceedings of the Fourth International Conference on Autonomous Agents (AGENTS-2000)*, pages 177, 178, Barcelona, Spain, 2000.

[Swe01] Swedish Institute of Computer Science. *SICStus Prolog*, 2001. `http://www.sics.se/sicstus/`

[YDIK98] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara. The Distributed Constraint Satisfaction Problem: Formalization and Algorithms. *IEEE Transactions on Knowledge and DATA Engineering*, 10(5), 1998.

Markus Hannebauer hat an der FU Berlin und der HU Berlin Informatik und BWL studiert und sein Diplom nach 8 Semestern 1998 mit Auszeichnung und als Jahrgangsbester abgeschlossen. Gefördert durch das DFG-Graduiertenkolleg „Kommunikationsbasierte Systeme“ hat er 2001 im Alter von 25 Jahren an der TU Berlin bei Prof. Dr. Jähnichen mit Auszeichnung promoviert.

In seiner Forschung hat sich Herr Hannebauer auf intelligente verteilte Systeme, Multiagentensysteme, konzentriert. Zu diesem Thema hat er in vier Jahren über 35 Publikationen verfasst, darunter fünf Zeitschriftenartikel. Einige dieser Artikel sind bereits in der Studienzeit entstanden, nachdem das deutsche Team 1997 unter Leitung von Prof. Dr. Burkhard die Weltmeisterschaft im simulierten Roboterfußball gewonnen hatte.

Während seines Studiums war Herr Hannebauer drei Jahre leitend in einer Software-Firma tätig. Bei Fraunhofer FIRST hat er dann das DFG-Projekt AURECON initiiert und geführt und sich stark in der nationalen und internationalen KI-Community engagiert. Zur Zeit ist Herr Hannebauer Managing Director von think-cell, einem Technologieunternehmen, das für sein innovatives Geschäftskonzept den Berliner StartUp-Preis 2002 erhalten hat.