Agentenorientiertes Software Engineering

*Gerhard Weiß**

Agentenorientiertes Software Engineering (AOSE) bezeichnet einen Bereich, welcher an der Schnittstelle zwischen Software Engineering einerseits und Agenten- und Multiagentensystemen andererseits im Entstehen begriffen ist. Gegenstand von AOSE sind Vorgehensweisen, Methoden, Techniken und Tools für die Erstellung und Handhabung von agentenorientierter Software.

Darunter versteht man Software, deren Struktur und Funktionalität unter dem Blickwinkel einer Menge von Agenten betrachtet wird. Ein Agent ist dabei charakterisiert als eine abgrenzbare Softwareeinheit, die zur Erreichung der von ihrem Benutzer oder Entwickler vorgegebenen Ziele und unter Wahrung seiner Interessen flexibel und autonom mit ihrer Umgebung und anderen Agenten interagiert. Ausgehend von diesem Agentenkonzept zielt AOSE vor allem auf Anwendungen ab, die mindestens eines der folgenden Merkmale aufweisen: es sind viele, dynamisch interagierende Komponenten involviert,

- die nicht unbedingt alle a priori bekannt sind (vgl. offene Systeme wie das Internet),
- die eine Fremdkontrolle nicht oder nur beschränkt gestatten (z.B. aufgrund individueller und konfliktträchtiger Ziele, Interessen, Rechte und Pflichten) und
- zwischen denen elaborierte Kommunikations- und Koordinationsprozesse erforderlich sind.

Solche Anwendungen finden sich mit zunehmender Rechnervernetzung und Plattform-Interoperabilität in verschiedensten Bereichen, von E- und M-Commerce über Supply Chain und Business Process Management bis hin zu Telekommunikation und Logistik. Nachfolgend werden die Grundlagen und der Entwicklungsstand von AOSE im Überblick dargestellt.

Agenten und Objekte

Es gibt es drei herausragende Merkmale eines Agenten: Flexibilität, Autonomie und Interaktivität. Dabei bedeutet Flexibilität, dass ein Agent sowohl zu reaktivem (d.h. Umweltveränderungen in angemessener Zeit berücksichtigendem) als auch zu proaktivem (d.h. in Hinblick auf Zwischen- und Endziele vorausschauendem) Verhalten fähig ist. Autonomie besagt, dass ein Agent bei Bedarf selbständig und unabhängig Entscheidungen über von ihm auszuführende zielrelevante Handlungen treffen kann und in diesem Sinn Kontrolle über seinen internen Zustand und sein Verhalten besitzt. Schließlich impliziert Interaktivität insbesondere, dass Agenten auf hohem Niveau in Wechselwirkung treten können, also etwa im Rahmen von automatisierten Verhandlungen, Vertragsabschließungen und verteilten Planungs- und Problemlösungsprozessen. Mit dieser Charakterisierung werden auch die beiden als entscheidend betrachtenden Unterschiede zwischen dem agentenorientierten Paradigma und dem objektorientierten Paradigma deutlich (z.B. [8]). Zum einen kapseln Objekte nur ihre Identität („wer“), ihren Zustand („was“) und ihr Verhalten beziehungsweise dessen Umsetzung („wie“), während Agenten zudem Freiheitsgrade in ihrer Aktionswahl und Interaktion („wann“, „warum“, „mit wem“ und „ob überhaupt“) kapseln. Zum anderen greift bloße Objektorientierung generell zu kurz, wenn es um adäquate, intuitive und natürliche Modellierung und programmiertechnische Umsetzung von komplexen Interaktionen (Kommunikation und Koordination, Kooperation und Konkurrenz) und Beziehungs-

*Gerhard Weiß,
Institut für Informatik, Technische Universität München,
80290 München
E-MAIL: weissg@in.tum.de

Vorschläge an: Prof. Dr. Frank Puppe, Institut für Informatik,
Universität Würzburg, Am Hubland, 97074 Würzburg
und Dieter Steinbauer, GEZ, Freimersdorfer Weg 6, 50829 Köln.
Eine Liste der „Aktuellen Schlagwörter“ seit 1988 gibt es unter
http://ki.informatik.uni-wuerzburg.de/~puppe/as.htm

strukturen (dynamische Organisation und verteilte Kontrolle) zwischen Agenten geht. Aufgrund dieser Unterschiede ermöglichen das Agenten- und das Objektkonzept auf qualitativ verschiedenen Abstraktionsebenen gleichermaßen relevante System- und Entwicklungssichten, die sich sinnvoll ergänzen statt gegenseitig ausschließen (z.B. [12]). Die Agentensicht impliziert damit aber auch quer über alle Phasen der Softwareentwicklung zahlreiche neue, von der Objektsicht nicht abgedeckte Anforderungen und Fragestellungen.

Analyse und Design
Die verfügbaren Analyse- und Designmethoden für agentenorientierte Software lassen sich in drei Gruppen unterteilen: Erweiterungen von objektorientierten Methoden (z.B. KGR [9]), Erweiterungen von Methoden aus dem Knowledge Engineering (z.B. MAS-CommonKADS [7]), und Methoden, die auf eine rein agentenspezifische Sicht ausgelegt sind (z.B. Gaia [15]). Trotz ihrer verschiedenen disziplinären Wurzeln liegt das gemeinsame Hauptaugenmerk dieser Methoden auf der Modellierung von Prozessen und Ereignissen, die *zwischen* Agenten stattfinden. Dies soll anhand von Gaia illustriert werden.

Gaia sieht u.a. die Verwendung der folgenden drei Modelle vor.

- Ein Rollenmodell zur Identifikation der verschiedenen Rollen, die von Agenten eingenommen werden können. Dabei sind Rollen definiert über Verantwortlichkeiten (zur Erhaltung erwünschter und Vermeidung unerwünschter Zustände), Zulässigkeiten (Rechte hinsichtlich Resourcen), Aktivitäten (die keine Interaktion erfordern) und Protokolle (zur Beschreibung der Art der Interaktion mit anderen Rollen).
- Ein Interaktionsmodell bestehend aus Interaktionsmustern zur Beschreibung von Abhängigkeiten und Beziehungen zwischen den verschiedenen Rollen in einem Multiagentensystem.
- Ein Bekanntschaftsmodell zur Definition der möglichen Kommunikationspfade zwischen Agententypen, wobei ein Agententyp definiert ist über Rollenmengen.

Mit Hilfe der verschiedenen Modelle will Gaia agentenspezifische Konzepte „aufbrechen“ und für objektorientierte und funktionale Techniken zugänglich machen.

Formale Spezifikation und Verifikation
Die formale Spezifikation und Verifikation von agentenorientierter Software gewinnt zunehmend an Bedeutung. Es lassen sich grob „traditionelle Formalismen“ und „logikbasierte Formalismen“ unterscheiden. Zu den traditionellen Formalismen zählen Repräsentationsformen und Techniken, die im Software und Knowledge Engineering zum Standard gehören, wie z.B. Z, VDM, UML, Petrinetze und funktionale Systemkomposition. Um agentenspezifische Konstrukte mit solchen Formalismen geeignet erfassen zu können, wurden entsprechende Erweiterungen und Anpassungen vorgeschlagen. Verwiesen sei hier aus Platzgründen lediglich auf den kompositionalen Modellierungsansatz DESIRE [2] für Multiagentensysteme und auf Agent-UML (z.B. [13]).

Inzwischen gibt es auch prototypische logikbasierte Ansätze, die speziell auf die Spezifikation und Verifikation von agentenorientierter Software ausgerichtet sind. Dabei handelt es sich häufig um erweiterte temporale Logiken bzw. um Sprachen, die auf solchen Logiken aufsetzen. Ein Beispiel hierfür ist die Logik TBL bzw. die Sprache Concurrent METATEM [5].

Implementierungssprachen
Es gibt drei Sprachtypen, die für die Realisierung von agentenorientierter Software und damit für AOSE zunehmend von Bedeutung sind. Diese Typen unterscheiden sich ebenfalls in ihrer fachlichen Verwurzelung und sie eröffnen damit unterschiedliche Zugänge zur programmiertechnischen Umsetzung von Agentensoftware.

Einer dieser Sprachtypen ist durch *agentenorientierte Programmiersprachen* gegeben, die als Weiterführung üblicher Programmiersprachen betrachtet werden können. Verschiedene Prototypen solcher Sprachen sind verfügbar, wie z.B. Derivate des „Klassikers“ AGENT-0 [14] und die neuere Sprache 3APL [6]. Solche Sprachen unterstützen z.B. die Umsetzung von Verpflichtungen zwischen Agenten und planbasiertes Verhalten.

Einen zweiten relevanten Sprachtyp bilden *Kommunikationssprachen*, die ihre Wurzeln im Bereich der wissensbasierten Systeme haben. Bekannte Prototypen sind KQML, ARCOL und FIPA-ACL [10], die dem sprechaktbasierten Austausch von Wissen zwischen Agenten dienen. Andere „kommunikationsrelevante Sprachen“ sind die Sprache KIF zur Beschreibung

von Wissensinhalten und Sprachen wie ONTOLINGUA zur Konstruktion von Ontologien.

Der dritter relevanter Sprachtyp sind *Koordinationssprachen* wie etwa COOL [1], die eine gewisse Nähe zu herkömmlichen Sprachen für parallele und verteilte Programmierung aufweisen. Konzeptionell ist dieser Sprachtyp „oberhalb" der beiden anderen angesiedelt. Koordinierungssprachen dienen der expliziten Handhabung von Abhängigkeiten zwischen Aktivitäten verschiedener Agenten.

Entwicklungstools

Es sind verschiedene Entwicklungstools für agentenorientierte Software verfügbar (z.B. JACK [3] und ZEUS [11]). Neben einigen kommerziellen Produkten handelt es sich dabei zumeist um Forschungsprototypen. Diese Tools betonen typischerweise Interaktionsaspekte (Kommunikation, Koordination und Organisation), wobei sie sich zum Teil deutlich in ihrer Funktionalität und Abdeckung des Software-Lebenszyklus unterscheiden. Um einen Eindruck von diesen Tools zu vermitteln, wird ZEUS kurz charakterisiert.

ZEUS wird am British Telecom Intelligent Systems Research Laboratory entwickelt und ist vollständig in Java 2 implementiert. Das Tool umfasst drei Komplexe:

- eine Bibliothek mit funktionalen Agentenkomponenten wie z.B. TCP/IP-basierte Übermittlung von sprechaktbasierten Kommunikationsprimitiven, vordefinierte Koordinationsstrategien und Organisationsbeziehungen, einen allgemeinen Planungs- und Schedulingmechanismus und eine Schnittstelle zu existierender Software;
- mehrere Editoren zur Beschreibung und Spezifikation verschiedener agentenspezifischer Aspekte wie z.B. Ontologien und organisationale Strukturen;
- verschiedene Tools zur Visualisierung z.B. von Agent-Agent Beziehungen, Aufgaben- und Ausführungsverteilung und statistischen Auswertungen.

ZEUS ist ein sehr umfangreiches Tool, da es auf die Abdeckung aller Phasen abzielt und ein recht allgemeines Agentenmodell verwendet.

Fazit

AOSE ist ein junger und sich schnell entwickelnder Bereich. Für das wachsende Interesse an AOSE und agentenorientierter Software gibt es verschiedene Gründe:

Anwendungspotential: AOSE stellt sich der ständig wachsenden Herausforderung, Softwaresysteme für immer komplexere – offene, verteilte, dynamische, heterogene und interaktive – Anwendungsfelder zu realisieren.

Industrielle und kommerzielle Resonanz: Die Industrie zeigt zunehmend Interesse an agentenorientierten Softwareprodukten. Dies spiegelt sich z.B. auch wider in den verschiedenen industrieseitig forcierten Standardisierungsbestrebungen im Umfeld von Agentensoftware [17].

Verträglichkeit: Agentenorientierung ist mit anderen Entwicklungen auf dem Gebiet des Software Engineering konsistent (z.B. Design Patterns) oder schließt an existierende Methodologien beinahe nahtlos an (z.B. aufgabenorientierte Softwareentwicklung). Darüber hinaus sind agenten- und objektorientierte Softwaresysteme prinzipiell integrierbar.

Natürliche Entwicklung: Allgemeiner kann Agentenorientierung als ein natürlicher nächster Schritt in der Programmierentwicklung betrachtet werden, denn ausgehend von maschinennaher bis hin zur objektorientierten Programmierung zeigt sich zunehmende Abstrahierung und Lokalisierung.

Diese Gründe sollten jedoch nicht darüber hinwegtäuschen, dass AOSE in verschiedenerlei Hinsicht noch in den Kinderschuhen steckt. Handlungsbedarf besteht in Hinblick auf eine weitere konzeptuelle und theoretische Fundierung von agentenspezifischen Konzepten ebenso wie die Weiterentwicklung von Analyse- und Designmethoden, Verifikations- und Testverfahren und Entwicklungstools. Die bislang existierenden agentenorientierten Hilfsmittel für die verschiedenen Softwareentwicklungsphasen bewegen sich größtenteils auf einer prototypischer Ebene. Hier ist weitere Grundlagenforschung und Anwendungserfahrung erforderlich.

Abschließend sei noch auf den erstmals ausgerichteten AOSE Workshop [4] im Rahmen der letztjährigen 22nd International Conference on Software Engineering verwiesen; der Nachfolge-Workshop findet Ende Mai statt [16].

Literatur

1. Barbuceanu, M., Fox, M.S.: Capturing and modeling coordination knowledge for multiagent systems. International Journal of Cooperative Information Systems 5(2-3), 275-314 (1996)
2. Brazier, F.M.T. et al.: DESIRE: Modelling multi-agent systems in a compositional formal framework. International Journal of Cooperative Information Systems 6(1), 67-94 (1997)
3. Busetta, P. et al.: JACK Intelligent Agents – Components for intelligent agents in Java. Agentlink News 2, 2-5 (1999)
4. Ciancarini, G., Wooldridge, M. (eds.): Agent-oriented software engineering. Springer-Verlag 2001
5. Fisher, M., Wooldridge, M.: On the formal specification and verification of multi-agent systems. International Journal of Cooperative Information Systems 6(1), 37-65 (1997)
6. Hindriks, K.V. et al.: Agent programming in 3APL. Autonomous Agents and Multiagent Systems 2(4), 357-401 (1999)
7. Iglesias, C.A. et al.: Analysis and design of multiagent systems using MAS-CommonKADS. Proceedings of the 4th International Workshop on Agent Theories, Architectures, and Languages (ATAL'97), 313-327, Springer-Verlag 1998
8. Jennings, N.R.: On agent-based software engineering. Artificial Intelligence 117, 277-296 (2000)
9. Kinny, D., Georgeff, M., Rao, A.: A methodology and modelling technique for systems of BDI agents. Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96), 56-71, Springer-Verlag 1996
10. Kone, M.T., Shimazu, A., Nakajima, T.: The state of the art in agent communication languages. Knowledge and Information Systems 2, 259-284 (2000)
11. Nwana, H.S. et al.: ZEUS: A toolkit for building distributed multiagent systems. Applied Artificial Intelligence 13(1), 129-186 (1999)
12. Odell, J.: Objects and agents: Is there room for both? Distributed Computing, 44-45 (November 1999)
13. Odell, J., Parunak, H.V.D., Bauer, B.: Extending UML for agents. 2nd International Workshop on Agent-Oriented Information Systems (AOIS'2000)
14. Shoham, Y.: An overview of agent-oriented programming. In: Bradshaw, J.M. (ed.): Software agents, 271-290, AAAI Press 1997
15. Wooldridge, M., Jennings, N.R., Kinny, D.: The Gaia methodology for agent-oriented analysis and design. Autonomous Agents and Multiagent Systems 3, 285-312 (2000)
16. 2nd International Workshop on Agent-Oriented Software Engineering (AOSE'01), http://www.csc.liv.ac.uk/~mjw/aose/
17. FIPA (http://www.fipa.org) und OMG Agents Working Group (http://www.objs.com/isig/agents.html)

erscheint in: Informatik Spektrum, 24, 2001 (Aktuelles Schlagwort)