Mobile Benutzerschnittstellen für Multiagentensysteme: Klassifizierung und Umsetzung am Beispiel des Mobile Supply Chain Event Managements

Anna Maria Jankowska, Kamil Nowakowski

Lehrstuhl für Wirtschaftsinformatik
Europa-Universität Viadrina
August-Bebel-Strasse 12
15234 Frankfurt/Oder
{jankowska, knowakowski}@uni-ffo.de

Abstract: Mobile Technologien in Verbindung mit Multiagentensystemen (MAS), die über weltweite Netzwerke interagieren und kooperativ Problemlösungen erarbeiten, können zur Effizienzsteigerung im Supply Chain Event Management (SCEM) beitragen. Dazu müssen entweder Multiagentenplattformen speziell für mobile Geräte entwickelt werden, so dass Agenten von diesen Geräten aus initialisiert werden können, oder bestehende Plattformen müssen um mobile Benutzerschnittstellen erweitert werden, damit sie auch von Thin Clients aus zugänglich sind. Der Beitrag diskutiert drei Ansätze für den mobilen Zugang zu Multiagentensystemen (Portal-Plattform, Stellvertreter-Plattform, Eingebettete Plattform) sowie Implementierungsbeispiele. Die jeweiligen Vorteile und Nachteile werden aufgezeigt. Am Beispiel eines agentenbasierten mobilen SCEM-Systems wird anschließend dargestellt, wie mobile Benutzerschnittstellen für MAS gestaltet und nutzenbringend eingesetzt werden können.

1 Einführung

Mit wachsender Komplexität der Logistiknetzwerke messen Unternehmen der Transparenz der Prozesse sowie der automatisierten Reaktionen auf Störungen immer größere Bedeutung bei. Die Reaktionsgeschwindigkeit auf Abweichungen im operativen Bereich kann erheblich erhöht werden, wenn man ungeplante Ereignisse schnell aufdeckt und behandelt [Ni02]. Diese Aufgabe übernehmen Supply Chain Event Management (SCEM)-Systeme.

Durch Nutzung von Multiagentensystemen, diensteorientierten Architekturen und mobilen Technologien können die Möglichkeiten der SCEM-Systeme weiter verbessert werden. Der Einsatz von intelligenten Agenten erweist sich bei den SCEM-Systemen als besonders vorteilhaft. Dank ihrer Eigenschaften wie z.B. Autonomie, Reaktionsfähigkeit, Selbstständigkeit und Lernfähigkeit, sind intelligente Agenten im Stande, aus vergangenen Problemlösungen Schlussfolgerungen zu ziehen und zukünftige

Störungen automatisch zu beheben. Sie können mit der Umgebung interagieren, Güterströme anpassen oder Produktionspläne modifizieren.

Wenn alle Teilnehmer eines Logistiknetzwerkes Zugang zu allen relevanten Systemen haben, können z.B. intelligente Agenten bei einem Lieferantenausfall automatisch nach neuen Teilnehmern in der Lieferkette suchen und mit ihnen Verhandlungen über eine Lösung führen. Der Prozess kann mit Hilfe eines mobilen Geräts von einem Manager überwacht und beim Auftreten von Problemen entsprechend gesteuert werden.

Dieser Beitrag untersucht die konzeptionellen Ansätze für den mobilen Zugang zu Multiagentensystemen (MAS) am Beispiel eines SCEM-Systems. In Kapitel 2 werden mobile agentenbasierte Benutzerschnittstellen kurz vorgestellt und kategorisiert. Das Hauptaugenmerk wird auf die Vor- und Nachteile der verschiedenen Typen gelegt. In Kapitel 3 wird die Architektur eines prototypischen SCEM-Systems dargestellt, wobei der Implementierung der mobilen Benutzerschnittstellen besondere Aufmerksamkeit zukommt. Kapitel 4 gibt Schlussfolgerungen mit Blick auf weitere Forschungsarbeiten wieder.

2 Klassifizierung der mobilen agentenbasierten Benutzerschnittstellen

Mobile Benutzerschnittstellen für agentenbasierte Systeme können allgemein auf drei Ansätzen beruhen: Portal-Plattform, Stellvertreter-Plattform und Eingebettete Plattform [CB03]. Es können jeweils unterschiedliche Technologien sowie Agentenplattformen benutzt werden. Das häufig verwendete Java Agent Development Framework (JADE) [Be05] stellt dem Entwickler alle drei Plattformtypen zur Verfügung (vgl. Abbildung 1).

Die Wahl hängt primär von der Speicherkapazität der mobilen Endgeräte ab: Bei leistungsfähigeren Geräten wie Smartphones oder PDAs kann eine Stellvertreter- oder Eingebettete Plattform eingesetzt werden, während bei einfacheren Geräten nur eine Portal-Plattform in Betracht kommt.

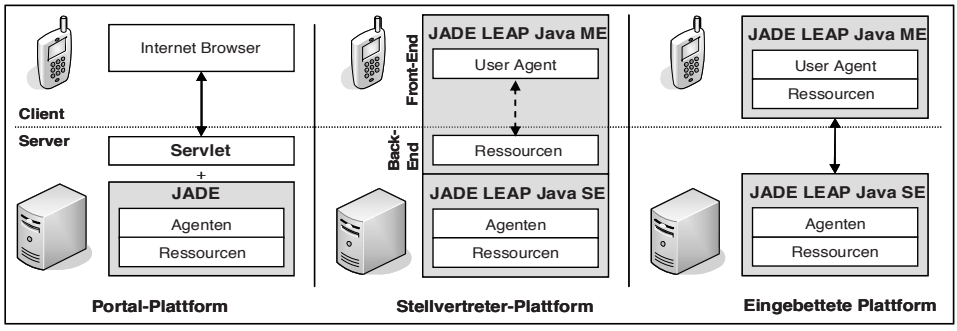

Abbildung 1: Optionen für den mobilen Zugang zu MAS in JADE

Bei einer *Portal-Plattform* befinden sich die Agenten und Ressourcen auf dem Server. Das Endgerät beherbergt keine Agentenlaufzeitumgebung. Für die Kommunikation zwischen Agenten und Endgerät wird eine mobile Benutzerschnittstelle benötigt. Diese kann entweder als Java ME-Anwendung oder als eine webbasierte Lösung realisiert werden. Der Client bezieht die vorgefertigten Daten von einem Server, auf dem die Agentenplattform läuft, und ist hauptsächlich für die Präsentationslogik und die Darstellung der Ergebnisse zuständig. Die Präsentationslogik, d.h. die mit der Interpretation, Konvertierung und Aufbereitung der Daten verbundenen Aufgaben, wird von der reinen Präsentation getrennt. Somit entspricht die Portal-Plattform dem traditionellen Client-Server-Modell.

Das größte Problem sind die fehlenden Möglichkeiten einer standardisierten graphischen Darstellung der Ergebnisse auf dem mobilen Gerät. Die meisten Agentenplattformen (z.B. JADE, FIPA-OS) unterstützen die Integration einer traditionellen Benutzerschnittstelle, implementiert in AWT oder Swing. Für mobile Geräte müssen individuelle Lösungen erarbeitet werden, die entweder eine serverseitige Technologie wie Servlets, JavaServerPages (JSP) [Ha00] oder Active Server Pages (ASP) [Kr00] als Vermittlungsschicht benutzen oder die Ergebnisse via HTTP-Protokoll übertragen (wie bei MobiAgent [Ma01]) und als ein MIDlet darstellen.

Bei der Portal-Plattform braucht keine besondere Ressourcenbeschränkung beachtet zu werden, weil sich die Geschäftslogikschicht, die durch die Implementierung von Geschäftsobjekten und –prozessen das eigentliche Geschäftsmodell realisiert, und teilweise auch die Präsentationslogikschicht auf dem Server befinden. Die Portal-Plattform bietet jedoch nur eine sehr eingeschränkte Kommunikation zwischen dem Benutzer und dem Multiagentensystem. Der Server kann auf Benutzeranfragen reagieren, verhält sich aber nicht proaktiv. Um die Proaktivität zu ermöglichen, müssen der Lösung spezielle Push-Techniken wie z.B. das Senden von E-Mails und SMS hinzugefügt werden.

Der zweite Ansatz, die *Stellvertreter-Plattform,* ist dadurch charakterisiert, dass ein Teil der Agentenplattform auf dem mobilen Gerät läuft. Das Gerät beinhaltet eine Laufzeitumgebung für die Agenten sowie die Agenten selbst und kann sich proaktiv verhalten. Die Präsentationslogik- und Präsentationsschicht befinden sich auf dem Client. Das Client-Server-Modell wurde hier durch ein Peer-to-Peer-Modell ersetzt, in welchem jede Softwareeinheit eine Aktion starten und auf Nachrichten reagieren kann. Durch Anwendung der "Push Registry"-Klassenbibliothek in Java ME MIDP 2.0 kann ein bestimmtes MIDlet auf eine MAS-Anfrage automatisch gestartet werden [Be03a].

Die Aufspaltung der Laufzeitumgebung in ein Frontend und Backend ermöglicht die Entlastung des mobilen Geräts von ressourcenintensiven Operationen. Verbindungsprobleme, die zwischen dem mobilen Gerät und dem Server auftreten können, müssen allerdings behandelt werden. Der Hauptnachteil der Stellvertreter-Plattform ist die Belastung der beschränkten Ressourcen des mobilen Gerätes. Ein weiteres Problem stellt die Notwendigkeit der Synchronisation zwischen Frontend und Backend dar.

Bei einer *Eingebetteten Plattform* befindet sich die komplette Agentenplattform auf dem mobilen Gerät. Das führt zwar zu einer hohen Belastung des Endgeräts, ermöglicht jedoch eine Kommunikation zwischen den Agenten, welche sich auf dem Gerät befinden, und solchen, die in einem MAS interagieren. Auf dieser Grundlage kann ein mobiles Endgerät Informationen von Agenten beziehen und auch selbst auf ihre Anfragen reagieren. Einige Implementierungsbeispiele dieses Ansatzes sind JADE-LEAP [Be03b], MicroFIPA-OS [TL02] und AgentLight [KM02].

3 Implementierung einer Benutzerschnittstelle für mobiles SCEM

Im Supply Chain Event Management können mobile Endgeräte nutzenbringend eingesetzt werden, weil sie den Zugang zu Echtzeitinformationen von einem beliebigen Benutzerstandort aus erlauben. Im Rahmen eines Forschungsprojekts, das die SCEM-Unterstützung durch Agenten zum Gegenstand hat, wurde eine Architektur implementiert, die den Zugang zu unterschiedlichen Enterprise Resource Planning (ERP)-Systemen durch Agenten und die Kommunikation zwischen Agenten auf Basis von JADE gewährleistet (vgl. Abbildung 2).

Ein beliebiges ERP-System (z.B. Compiere) kann über eine Web Services Façade [KJ06] erreichbar sein. Die Kommunikation der Agenten mit den Webdiensten erfolgt mit Hilfe des Web Service Integration Gateway (WSIG) [GC04]. Von mobilen Endgeräten aus kann in dieser Architektur über zwei Wege auf das MAS zugegriffen werden, mit Hilfe einer Stellvertreter- und einer Portal-Plattform.

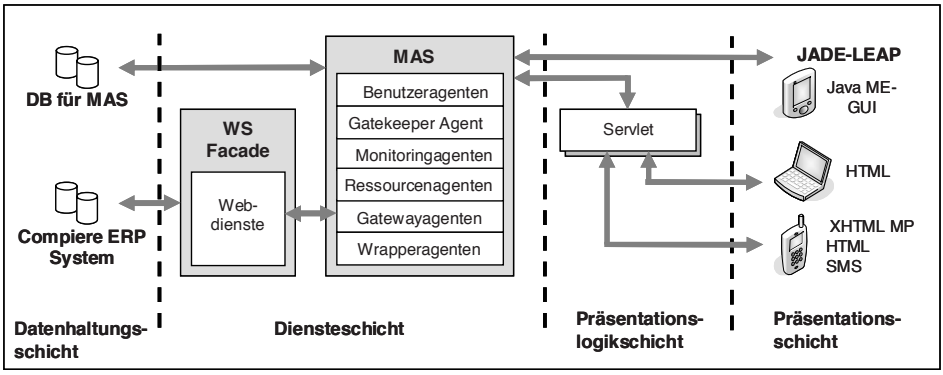

Abbildung 2: Architektur eines mobilen, agentenbasierten SCEM-Systems

Für die *Portal-Plattform* wurde ein komplexes Servlet entwickelt, das die von JADE unterstützte Objektkommunikation als Basis für die Kommunikation mit Agenten nutzt.[1] Beim Starten eines Agenten wird ein einfaches Synchronisationsobjekt erzeugt, das ein boolesches Attribut enthält. Die entsprechende Objektreferenz wird dem startenden

[1] Ausgehend von der Idee von Fabien Gandon, beschrieben im Beitrag "Linking a servlet to a JADE agent". Servlet Add-on Dokumentation: http://jade.tilab.com/community-3rdpartysw.htm#servlet.

Agenten als Parameter übergeben. Das Objekt enthält auch eine Methode, die solange den laufenden Thread anhält (mittels `java.Lang.Object.wait()`), bis ein Agent initialisiert und der Wert des Attributs auf `true` gesetzt wird. Dadurch erreicht man eine Synchronisation zwischen dem Servlet und dem Agenten.

Ein ähnlicher Mechanismus wird bei der Kommunikation mit Agenten eingesetzt: Das Servlet wartet auf Benachrichtigung der Agenten, dass ihre Aufgaben erfüllt sind. Das erstellte Kommunikationsobjekt enthält die Referenzen auf `HttpServletResponse` und `HttpServletRequest`, die Anweisungen (Strings) für den Agenten und das Ergebnis der Agentenaktionen.

Die Ausgabe der Ergebnisse kann grundsätzlich auf zwei Wegen erfolgen. Der Agent kann die Ergebnisse selbst formatieren und sie in einem Browser ausgeben. Hierfür benutzt er die ihm im Kommunikationsobjekt übergebene Referenz auf `HttpServlet-Response`. Alternativ dazu können die Ergebnisse der Agentenaktionen durch die Abfrage eines entsprechenden Attributs der gemeinsam zugänglichen Objektinstanz über das Servlet in Erfahrung gebracht werden (vgl. Abbildung 3). Danach kann das Servlet dem Benutzer die entsprechenden Resultate präsentieren. In beiden Fällen werden die Informationen in Form einer (X)HTML-Seite mittels eines PrintWriter-Objekts dargestellt.

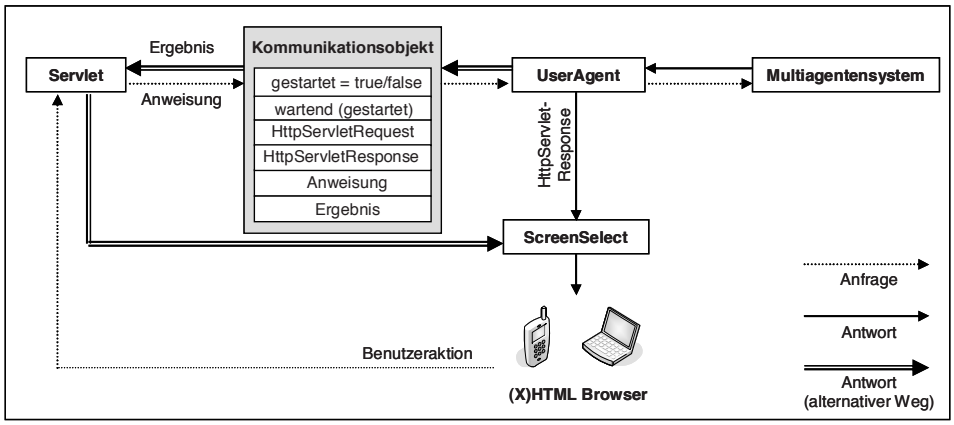

Abbildung 3: Architektur der Portal-Plattform

Als Erweiterung der Architektur wurde eine `ScreenSelect`-Klasse implementiert, die dem Aufbau der (X)HTML-Seiten dient. Diese Klasse transformiert die von den Agenten erhaltenen Ergebnisse in die Auszeichnungssprache des Endgeräts. Auf die Methoden dieser Klasse können sowohl Agenten als auch Servlets zugreifen. Anschließend werden die Ergebnisse in einem Browser dargestellt. Als Ausgleich für die fehlende Proaktivität der Portal-Plattform wurde ein Nachrichten-Push-Modell vorgesehen. Mobile Benutzer werden durch E-Mail oder SMS über kritische Ereignisse informiert.

Bei der zweiten Lösung – der *Stellvertreter-Plattform* – wurde die Benutzerschnittstelle in Form einer proaktiven Java-Anwendung implementiert, die die Ergebnisse der

Kommunikation zwischen Agenten und mobilen Geräten darstellen kann. Auf dem mobilen Gerät wird eine "leichtgewichtige" Version der JADE-Plattform installiert – die JADE LEAP (Lightweight Extensible Agent Plattform) [Be03b].

Das für die protoypische Implementierung benutzte Gerät ist ein Java-fähiges Nokia 6680 mit Unterstützung von Java ME MIDP 2.0. Als Benutzerschnittstelle wurde die MIDP-Version von JADE LEAP verwendet. Voraussetzung für die Aufspaltung in Frontend und Backend ist eine auf dem Server bereits laufende Umgebung – die Java SE-Version von JADE LEAP.

Die Anbindung der Benutzerschnittstelle an die Agentenplattform ist in JADE-LEAP sehr einfach. Die Hauptklasse (`MicroBoot`) überlädt die `startApp()`-Methode eines MIDlets und implementiert zusätzlich die Methoden für das Starten der Agenten. Somit stellt sie dem Programmierer alle notwendigen Methoden zur Verfügung, die für die Initialisierung von Agenten und zugleich für eine spätere Visualisierung ihrer Ergebnisse in Form eines MIDlets notwendig sind. Eine Referenz zu dem aktuellen MIDlet wird in der statischen Variablen `midlet` gespeichert.

Der Aufbau graphischer Oberflächen in Form von MIDlets kann dann direkt von einem existierenden Agenten übernommen werden, indem er die in der `midlet`-Variablen vorhandene Referenz nutzt. Mit Hilfe dieser Referenz kann der Agent direkt auf die Methoden der Klasse `Display` zugreifen (`Display.getDisplay(Agent.midlet)`), um dadurch z.B. Displayeigenschaften abzufragen und die zur Ausgabe auf dem Gerät bestimmten Objekte auszuwählen. Diese Objekte, die MIDlet-Formulare darstellen, werden in einer separaten Klasse definiert. Der Agent greift auf die Methoden dieser Klasse zu. Diese wandeln die Ergebnisse der Agentenkommunikation in die Formularobjekte um, damit sie dem Benutzer in einem MIDlet präsentiert werden können.

Bei der skizzierten Architektur ist der Benutzeragent fähig, Meldungen anderer Agenten entgegenzunehmen und auf diese zu reagieren. Hierfür wurde ein Mechanismus erarbeitet, der es erlaubt, Nachrichten aus dem MAS-System zu entnehmen und sie auf dem mobilen Gerät zu speichern.

4 Schlussfolgerungen und Ausblick

Mobile Benutzerschnittstellen sind derzeit noch nicht integrale Teile von Agentenplattformen, sondern werden meist nachträglich ergänzt. Mit steigender Leistungsfähigkeit mobiler Endgeräte können Lösungen realisiert werden, bei denen sich zumindest eine beschränkte Agentenlaufzeitumgebung auf dem mobilen Gerät befindet. Dadurch wird es möglich, das autonome und proaktive Verhalten von Agenten besser zu unterstützen und die Agenten beispielsweise in die Lage zu versetzen, Verhandlungen ohne menschliches Eingreifen zu führen.

Auf der Basis von geräteunabhängigen und kontextsensitiven Ansätzen können mobile Benutzerschnittstellen für agentenbasierte Systeme entwickelt werden, die ein breites Spektrum von Geräten abdecken und eine automatische Anpassung der Inhalte in Abhängigkeit von kontextsensitiven Informationen vornehmen. In unserem Projekt erfolgt die Anpassung an die Geräteeigenschaften anhand der Informationen in den HTTP-Headers. Einschränkungen ergeben sich allerdings daher, dass die Gerätehersteller diesen Ansatz nur unvollständig unterstützen. Er wird in der Zukunft voraussichtlich durch Verwendung von Composite Capability/Preference Profiles (CC/PP) [W3C04] und die DELI-Klassenbibliothek [Bu02] ersetzt. Des Weiteren werden separate, parametrisierte Klassen für alle (X)HTML- und MIDletobjekte definiert (z.B. Formularobjekte wie Textfelder, Schaltflächen etc.), aus denen der Entwickler beliebige Bildschirmausgaben zusammensetzen kann.

Literaturverzeichnis

[Be03a] Bellifemine, F. et. al.: JADE – A White Paper. In: The TILAB Journal „EXP – in Search of Innovation“ 3, Special Issue on JADE No. 3, 2003.

[Be03b] Berger, M. et. al.: Porting Agents to Small Mobile Devices – The Development of the Lightweight Extensible Agent Platform. In: The TILAB Journal „EXP – in Search of Innovation“ 3, Special Issue on JADE No. 3, 2003.

[Be05] Bellifemine, F.; Caire, G.; Trucco, T.; Rimassa, G.: JADE Programmer's Guide, 2005. http://jade.tilab.com/doc/index.html.

[Bu02] Butler, M.: DELI: A Delivery Context Library for CC/PP and UAProf, 2002. http://www.hpl. hp.com/personal/marbut/DeliUserGuideWEB.htm.

[CB03] Carabelea, C.; Boissier, O.: Multi-Agent Platforms on Smart Devices: Dream or Reality? In: Proceedings of Smart Object Conference sOc'2003, Grenoble, 2003.

[GC04] Greenwood, D.; Calisti, M.: Engineering Web Service – Agent Integration. In: Proceedings of the IEEE Systems, Cybernetics and Man Conference, Hague, 2004.

[Ha00] Hall, M.: Core Servlets and JavaServer Pages. Upper Saddle River: Sun Microsystems Press/Prentice Hall, 2002.

[KJ06] Kurbel, K.; Jankowska, A.M.: Diensteorientierte Architekturen und intelligente Agenten im Supply Chain Event Management; ERP Management 2 (2006) 1, 2006; S. 27-30.

[KM02] Koch, F.; Meyer, J-J. C.: Project AgentLight: Developing Logic-based Autonomous Agents for Small Devices. In: Proc. of the 1st Workshop of Thesis and Dissertations on Artificial Intelligence (WTDIA), Recife, 2002.

[Kr00] Krause, J.: Microsoft Activer Server Pages. Addison-Wesley Verlag, München, 2000.

[Ma01] Mahmoud, Q.H.: MobiAgent: An Agent-based Approach to Wireless Information Systems. In: Proceedings of the 3rd Int. Bi-Conference Workshop on Agent-Oriented Information Systems, Montreal, 2001.

[Ni02] Nissen, V.: Supply Chain Event Management. In: Wirtschaftsinformatik 44/5, 2002; S. 477-480.

[TL02] Tarkoma, S., Laukkanen, M.: Supporting Software Agents on Small Devices. In: Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2002), Bologna, 2002; S. 565-566.

[W3C04] W3C: CC/PP Structure and Vocabularies 1.0, 2004. http://www.w3.org/TR/CCPP-struct-vocab/