Middleware für Mobile Java Anwendungen*

Dr. Silvano Maffeis, CTO, Softwired AG
silvano.maffeis@softwired-inc.com

Die Vermählung mobiler "Communicator" Geräte mit Datendiensten wie GPRS und UMTS ermöglicht die Entwicklung einer neuen Generation von mobilen Anwendungen. Dieser Artikel durchleuchtet Anforderungen solcher Anwendungen und stellt einen auf dem Java Message Service Standard basierenden Lösungsansatz vor.

1 Ein Milliardenvermögen!

Gerätehersteller wie Intel, Compaq, Nokia, Ericsson und Motorola investieren ein Milliardenvermögen um eine neue Generation mobiler Endgeräte noch dieses Jahr auf den Markt zu bringen. Dabei handelt es sich um sogenannte *Communicator* Geräte: die Vermählung von Mobiltelefon und PDA.

Abb. 1: Communicator Geräte von Nokia, Symbian und Ericsson.

Gleichzeitig findet bei Telekom Unternehmen ein Wettlauf statt, die mobile Bevölkerung mit immer mehr Übertragungskapazität zu beglücken. Die heutigen, verbindungsorientierten Datendienste wie GSM (Europa) und DAMPS (USA), werden durch immer schnellere paketorientierte Datendienste abgelöst.

Zu erwähnen ist dabei GPRS (General Packet Radio Service), welches seit Ende 2000 flächendeckend in der Schweiz und in anderen Ländern zur Verfügung steht. GPRS bietet eine Datenübermittlungsrate von bis zu 114 kbps. Zu erwähnen ist auch das etwas kontroverse UMTS (Universal Mobile Telecommunications System) [WHATIS], welches, so wohl die Technologie will, Übertragungsraten im Megabit Bereich bieten wird. Klar ist auf jeden Fall, dass auch hier wieder Milliarden investiert werden.

Worüber hingegen weniger Klarheit herrscht, ist wie diese Rieseninvestition profitabel gemacht werden soll, oder wie man eine genügend grosse, und zahlungswillige, Benutzerbasis aufbaut. Die Antwort ist klar: es braucht "Killeranwendungen" welche durch die Kombination von Communicators und paketorientierten Datendiensten erst möglich werden. Welches sind also die Anwendungen, welche Sie und mich, sowie meine Eltern und einst meine beiden Töchter dazu bringen werden, einen teuren Communicator zu erwerben (oder sich zu wünschen!) und 50 Euro im Monat für einen Datendienst zu bezahlen (bzw. "bezahlen zu lassen")?

Der Ideen gibt es viele. Gemäss Analysten [TIMELABS, NOMURA] werden die mobilen Killeranwendungen der Zukunft vor allem im Unterhaltungsbereich zu finden sein: Spielen, Chatten,

* *Dieser Artikel erscheint in SIGS OBJEKTspektrum 2/2001.*

Flirten, Restaurants und Kinos auffinden, sowie Benachrichtigungen absenden (Instant Messaging). Im geschäftlichen Bereich könnten Finanzdienste (Aktienkurse, Benachrichtigungen bei grösseren Schwankungen auf den Finanzmärkten, usw.) etliche Benutzer anziehen. Ob mCommerce (Mobile Commerce) genügend attraktiv sein wird, darüber scheiden sich die Analysten.

Was für die Software-Industrie sicherlich feststeht, ist, dass die Entwicklung solcher Dienste wesentlich weniger gut verstanden ist als die Entwicklung skalierbarer Web-Portale für nicht-mobile Endgeräte. Dieser Artikel durchleuchtet einige der wichtigen Anforderungen, welche mobile Dienste an Softwarewerkzeuge stellen. Unsere Schlussfolgerung ist, dass die Kombination von Java und Nachrichtenorientierter Middleware (MOM) einen robusten und skalierbaren Lösungsansatz bietet.

2 Plattformen für Mobile Informationsdienste

Unter einem "mobilen Informationsdienst" verstehen wir einen Dienst (Service) oder eine Anwendung, welche für mobile Endgeräte nutzbar ist (Abb. 2). Zum Beispiel ein Chat System oder die Übermittlung von Finanzdaten. Mobile Informationsdienste bestehen aus einer mobilen Applikation, welche auf den Endgeräten installiert wird. Diese Applikation kann ein Browser sein oder eine für den Dienst zugeschnittene Anwendung (Client). Soviel sei vorweggenommen, dass unser Lösungsansatz erst recht dann interessant ist, wenn für den Dienst spezifische Software auf den mobilen Engeräten vorgesehen ist.

Mobile Informationsdienste bestehen aber auch aus einer skalierbaren Server-Anwendung. Die Geräte müssen sicher und zuverlässig mit der Server-Anwendungen kommunizieren, und dies oftmals über diverse Drahtlos-Protokolle (GPRS, UMTS, SMS, WAP, etc.). Eine komplexe Gleichung mit mehreren Parametern.

Die Übermittlung von komplexen Daten "in Echtzeit" ist eine weitere wichtige Anforderung mobiler Informationssysteme. GPRS und UMTS machen es möglich, dass ein mobiles Gerät grundsätzlich immer am Netzwerk "angeschlossen" ist und Echtzeit-Daten, wie z.Bsp. Alarme, jederzeit empfangen kann. Der Benutzer bezahlt für die übertragene Datenmenge, und nicht für die Verbindungszeit.

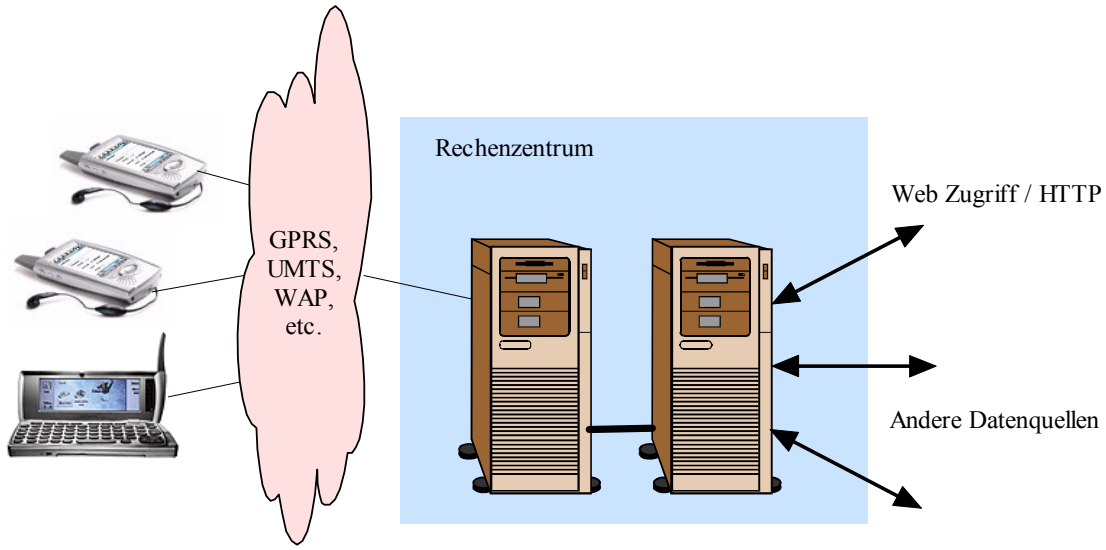

Abb. 2: Generelle Architektur mobiler Informationsdienste.

Wird auf dem Endgerät eine für den Dienst zugeschnittene Anwendung benötigt, so ist ferner zu berücksichtigen, dass es heute noch nicht klar ist, welches mobile Betriebssystem eine dominante Rolle spielen wird im Communicator Markt. Die Wettstreiter sind Palm Computing mit dem Palm OS, Symbian mit der EPOC Plattform und Microsoft mit Windows CE bzw. Pocket PC. Red Hat versucht sich mit Embedded Linux [REDHAT] ebenfalls in diesem Markt zu etablieren.

2.1 Palm OS

PalmOS ist der Marktführer im PDA Bereich mit einem Marktanteil von 65% in den USA. Das bedeutet aber nicht, dass Palm Computing automatisch auch im Communicator Bereich Marktführen sein wird. Im letzten Jahr hat PalmOS ca. 5% an Marktanteil verloren, was an sich aber nicht beängstigend ist, da der PDA Markt stark gewachsen ist und dadurch Mitstreiter wie Handspring angelockt wurden.

Was Palm Computing wohl mehr zu schaffen macht ist die Tatsache, dass das einfache Palm OS Betriebsystem eher für Taschenagendas und "Todo" Listen konzipiert wurde, und weniger für kommunikationsfreudige Communicator-Anwendungen, welche komplexe Daten in Echtzeit über GPRS und Bluetooth [WHATIS] austauschen. Wer mit der KVM oder mit einer anderer Java Umgebung auf PalmOS entwickelt hat, kennt die Einschränkungen bezüglich Speicherverwaltung, Kommunikationsprotokollen und Multithreading.

Der Communicator Markt stellt wesentlich höhere Anforderungen ans mobile Betriebssystem: Echtes Multithreading, virtuelle Speicherverwaltung, eingebaute Verschlüsselung, Zertifikate-Management, APIs für Telephonie, Interprozesskommunikation, Spracherkennung, und so fort. Palm Computing hat folglich zwei Möglichkeiten um den Communicator Markt zu erobern: Das Palm OS neu konzipieren, was erheblich Zeit kostet, oder eine Plattform wie Symbian zu lizenzieren. Es kursieren Gerüchte, wonach Palm und Symbian in entsprechenden Verhandlungen stehen. Fest steht zumindest, dass Nokia die PalmOS Benutzerschnittstelle auf Symbian portiert.

2.2 Windows CE

Windows CE ist das 32-Bit Betriebssystem für mobile Geräte von Microsoft. In technischer Hinsicht bietet Windows CE einige ideale Voraussetzungen für den Communicator Markt, wie zum Beispiel Multithreading, Unterstützung von hochauflösende Farbdisplays, und eine breite Auswahl an Kommunikationsprotokollen (TCP/IP, SSL, Serial, IrDA, Telephony API (TAPI), SNMP).

Trotz massiven Marketing-Ausgaben ist es Microsoft jedoch (noch) nicht gelungen, einen grösseren Marktanteil im PDA Markt zu erzielen. Ein oft kritisiertes Merkmal ist die fehlende Ergonomie der Windows CE Benutzungsschnittstelle. Die "Desktop" Metapher mag zwar gut sein für PC Systeme, aber viele Windows CE Benutzer erfahren diese als unnatürlich und "mühsam".

2.3 Symbian

Die britische Firma Symbian [SYMBIAN] wurde von Ericsson, Motorola, Nokia und Psion im Dezember 1998 gegründet. Matsushita (Panasonic) hat sich im May 1999 ebenfalls an Symbian beteiligt. Symbian hat ein Betriebssystem für Communicators und Mobiltelefone entwickelt. Als Grundlage dazu diente das EPOC Betriebsystem von Psion. Technisch gesehen, bietet diese 32-Bit Plattform eine ideale Basis für Communicators und Mobiltelefone der nächsten Generation. Auch ist Symbian der einzige Wettstreiter, welcher Java offiziell unterstützt. Weder Palm Computing noch Microsoft erachten zum jetzigen Zeitpunkt Java als strategisch, oder haben sich zumindest nie offiziell zu entsprechenden Aussagen bewegen lassen.

Auch hat es Symbian verstanden, Firmen wie Ericsson, Motorola und Nokia einzuspannen und für die Symbian Plattform zu verpflichten. Das Ericsson R380s Mobiltelefon, welches seit Herbst 2000

erhältlich ist, basiert auf dem Symbian Betriebssystem. Das von Nokia angekündigte 9210 Modell ebenfalls. Ericsson hat einen auf Symbian beruhenden GPRS Communicator angekündigt. Ebenfalls interessant sind die XScale Communicators von Intel, welche auf der Symbian Plattform beruhen.

Meiner Einschätzung nach hat Symbian die besten Aussichten, einen grossen Anteil des Communicator Marktes zu erobern. Dafür sprechen technische und strategische Gründe.

3 Die Rolle von Java

Anwendungen wie Chat, Spiele, Wireless-Napster, Echtzeit-Finanzdaten, und so weiter benötigen eine spezielle Anwendung auf dem mobilen Endgerät. Es ist also nicht genug, einen WAP oder Web Browser auf dem Gerät vorzusehen. Folglich muss bei mobilen Informationsdiensten oft spezielle Software für die Engeräte entwickelt werden. Bloss für welche Plattform ist die Frage?

Dank Java brauchen Sie sich nicht bereits heute exklusiv auf eine Betriebssystem-Plattform festzulegen. Das Risiko wäre zu gross, da der "Gewinner" noch nicht abzusehen ist. Möglich ist auch, dass der Communicator Markt von zwei oder mehr Plattformen dominiert werden wird. Folglich ist der Betreiber eines Informationsdienstes sicherlich im Vorteil, wenn seine Dienste für diverse mobile Plattformen zugänglich sind. Durch die Verwendung von Java Technologie auf mobilen Engeräten ist es möglich, Anwendungen zu entwickeln, welche auf unterschiedlichen Plattformen lauffähig sind. Dadurch kann eine grössere Benutzerbasis für den Informationsdienst aufgebaut werden.

3.1 Die Java-2 Micro Edition

Bei der Java-2 Micro Edition (J2ME) handelt es sich um eine Standard Java Plattform für mobile Geräte [J2ME]. Die J2ME besteht aus einer Java Virtual Machine (JVM) sowie aus APIs, welche den Bedürfnissen und Einschränkungen mobiler Geräte gerecht werden.

Die J2ME besteht aus diversen *Konfigurationen*. Jede Konfiguration kann zudem eins oder mehrere sogenannte *Profile* beherbergen. Abb. 3 stellt die J2ME Konfiguration für Java-Mobiltelefone dar. Diese besteht aus einer J2ME JVM, aus den CLDC (Connected Limited Device Configuration) APIs, sowie aus dem MIDP Profil (Mobile Information Device Profile).

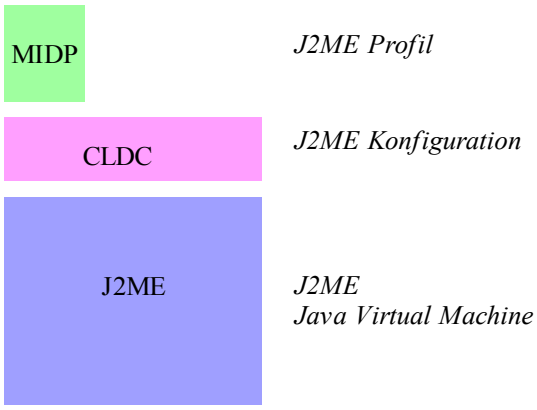

Abb. 3: J2ME Version für Java-Mobiletelefone

CLDC enthält zum Beispiel Java Klassen, um Netzwerkverbindungen aufzubauen (TCP/IP, HTTP, Serielle Schnittstelle).
MIDP enthält APIs, um Daten auf einem mobilen Gerät persistent zu speichern, oder um grafische Benutzungsschnittstellen auf kleinen Displays realisieren zu können.

J2ME Entwicklungsumgebungen werden von verschiedenen Herstellern angeboten, zum Beispiel von Sun Microsystems (KVM), von IBM (J9) und von Esmertec (Jbed).

4 Die Rolle von Middleware

Bei der Entwicklung von mobilen Informationssystemen spielt mobile Middleware mehr und mehr eine zentrale Rolle. Mobile Middleware ermöglicht dem Endgerät, eine Verbindung mit dem Server (Abb. 2) aufrecht zu erhalten, selbst dann, wenn die Netzabdeckung zeitweise nicht gewährleistet ist. Mobile Middleware kann zudem eine Verschlüsselung der Daten bewerkstelligen, Transaktionen steuern, oder mobilen Endgeräten einen Zugriff auf einen JNDI Namendienst im Rechenzentrum ermöglichen.

Zu bemerken ist, dass die Kommunikations-APIs, welche von der J2ME Plattform angeboten werden, relativ primitiv sind und darum keinen Ersatz für mobile Middleware bieten können. Mobile Middleware baut hingegen auf diesen Primitiven auf, z.Bsp. auf dem CLDC "Connection" Framework [CLDC].

Speziell berücksichtigt werden muss auch das Kommunikationsumfeld von mobilen Anwendungen. Die Übertragungsraten schwanken oft stark, und mobile Geräte verlieren oft die Netzwerkabdeckung (d.h., die "Verbindung" zum Rechenzentrum) um sie wenige Sekunden oder Minuten später zurückzugewinnen. Zudem gibt es in der mobilen Welt etliche Standards zur mobilen Datenübertragung. Das bedeutet, dass Middleware eine Auswahl an Übertragungsprotokollen (GPRS, UMTS, SMS, etc.) und Dienstqualitäten (zuverlässig, transaktional, unzuverlässig, verschlüsselt) unterstützen muss.

Meiner Einschätzung nach bietet nachrichtenorientierte Middleware (Message Oriented Middleware (MOM)) die besten Voraussetzungen für mobile Systeme. MOM basiert auf dem asynchronen "store-and-forward" Modell, welches die Zustellung wichtiger Nachrichten selbst dann garantiert, wenn ein Gerät zeitweise keine Netzwerkverbindung hat.

Im Gegensatz dazu stehen Middleware Systeme, welche auf dem "Anfrage/Antwort" Modell beruhen, wie z.Bsp. CORBA und RMI. Diese Art von Middleware ist in einem Kommunikationsumfeld, welches durch stark schwankende Übertragungsraten und häufige Netzwerkunterbrüche charakterisiert ist, weit weniger gut geeignet als MOM.

5 Eine Architektur für Mobile Informationsdienste

Im Java Umfeld gibt es einen etablierten Standard für MOM: der Java Message Service (JMS) von Sun Microsystems [JMS]. JMS ist Bestandteil der Java-2 Enterprise Edition (J2EE) und wird von jedem Applikationsserver unterstützt, welcher kompatibel ist zu J2EE 1.3 oder höher. Zudem gibt es auch optimierte JMS Implementierungen, welche ohne einen J2EE Applikationsserver auskommen.

Obschon JMS Teil der Server-seitigen Java Welt ist, kann JMS auch auf der J2ME Plattform vorgesehen werden. Beim iBus Produkt der Firma Softwired handelt es sich um das bislang einzige JMS Middleware-System, welches nahtlos auf Servern, Workstations, sowie mobilen Geräten eingesetzt werden kann [IBUS].

Das Resultat ist die in Abb. 4 dargestellte Systemarchitektur. Dabei handelt es sich um eine "end-to-end" JMS Plattform. Das heisst, auf Communicator Geräten werden Anwendungen installiert, welche auf dieselben JMS Kommunikationskanäle zugreifen können, wie die Anwendungen auf dem Applikations-Server. Der Entwickler der mobilen Anwendung benutzt immer dasselbe JMS API, egal ob

die Nachrichten über GPRS, TCP/IP, SMS, oder über ein anderes Protokoll übermittelt werden. Die Entwicklung von mobilen Informationssystemen wird dadurch enorm erleichtert.

Die Architektur unterstützt aber auch mobile Geräte auf die keine Java Anwendungen installiert werden können. z.Bsp. SMS und WAP Mobiltelefone, sowie Pager.

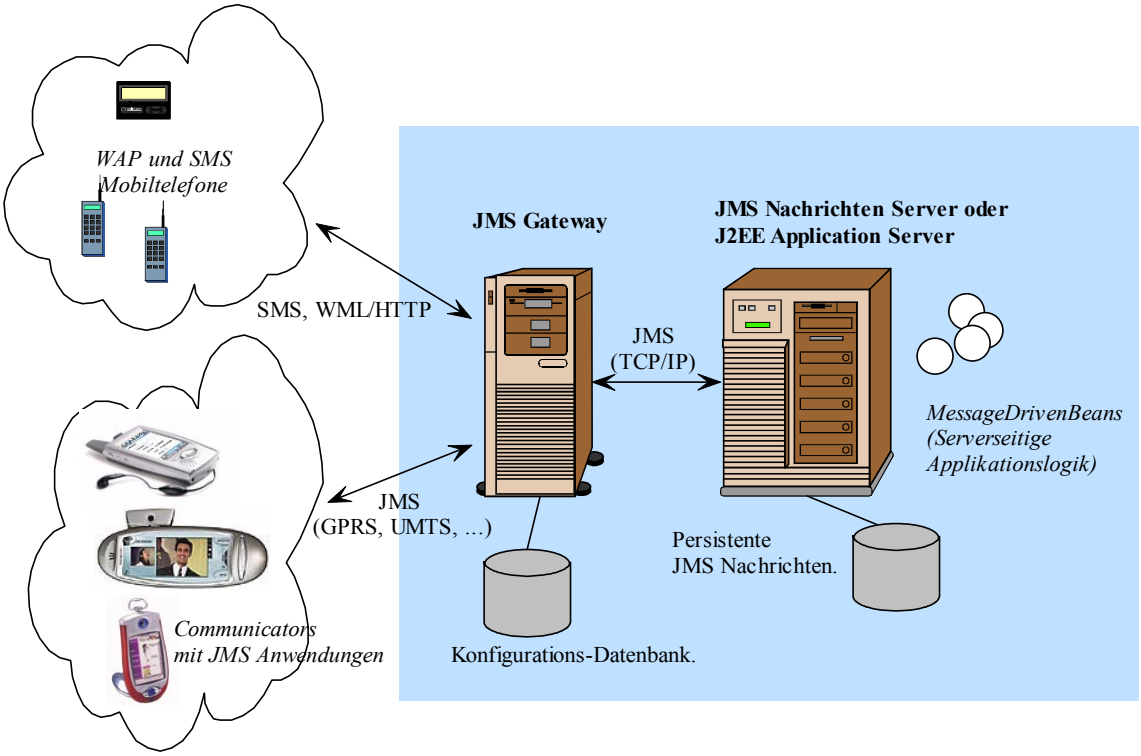

Abb. 4: JMS Architektur für Mobile Informationsdienste der nächsten Generation.

5.1 Interaktionen

Die "Applikationslogik", das heisst der Server-seitige Zustand des Informationsdienstes, wird in Form von JMS-EJBs (sogenannte MessageDrivenBeans) implementiert und unter die Kontrolle eines J2EE Applikationsservers gestellt. Die MessageDrivenBeans senden und empfangen JMS Nachrichten unter Verwendung von JMS Kommunikationskanälen (Topics und Queues). Typischerweise wird XML in die JMS Nachrichten eingebettet. Das XML wird vom JMS Gateway in ein für das Endgerät passendes Format umgesetzt, z.Bsp. WML oder SMS.

Bei den mobilen Geräten unterscheiden wir zwischen Geräten, welche nicht programmierbar sind (SMS und WAP Geräte **ohne** JMS), sowie Geräten, welche eine J2ME oder eine PersonalJava Laufzeitumgebung vorsehen (Communicators **mit** JMS).

Beispiel 1: Ein Mobiltelefon übermittelt eine SMS Nachricht an eine Dienstnummer. Der JMS Gateway empfängt die SMS Nachricht, konvertiert sie in eine JMS Nachricht, und legt sie in ein JMS Topic oder eine JMS Queue ab. Ein MessageDrivenBean kann diese Nachricht nun bequem empfangen und verarbeiten. Als Resultat der Verarbeitung kann das MessageDrivenBean eine JMS Antwort-Nachricht an den JMS Gateway senden. Der Gateway wird die Antwort automatische in ein SMS konvertieren, und ans Mobiltelefon übermitteln.

Beispiel 2: Ein WAP Mobiltelefon öffnet eine bestimmte WAP URL. Der WAP Gateway (nicht aufgeführt in Abb. 4) richtet eine HTTP Anfrage an den JMS Gateway. Der Gateway konvertiert die Anfrage in

eine JMS Nachricht und übermittelt diese via JMS an ein MessageDrivenBean. Das Bean übermittelt eine Antwort-JMS Nachricht, welche vom Gateway automatisch in WML umgewandelt und ans WAP Mobiltelefon übermittelt wird. Das heisst, der Gateway ermöglich den Zugriff auf MessageDrivenBeans ohne dafür WML-Servlets oder dergleichen entwickeln zu müssen.

Beispiel 3: Eine mobile JMS Anwendung wird auf einem Communicator installiert. Auf der Server-Seite befindet sich ein MessageDrivenBean, welches Aktienkurse in eine JMS Topic einspeist. Die mobile Anwendung meldet sich einfach auf das entsprechende Topic an, um die Finanzdaten über GPRS oder UMTS bequem und effizient zu empfangen ("Push" Modell).

5.2 Komponenten

Der JMS Nachrichtenserver in Abb. 4 bildet das Herzstück der Architektur. Dieser implementiert das JMS Kommunikationsmodell, welches sowohl Publish/Subscribe Kanäle (Topics) als auch Nachrichtenwarteschlangen (Queues) vorsieht.

Die Applikationslogik des Dienstes wird in der Form von EJB MessageDrivenBeans oder von eigenständigen Java Server-Applikationen entwickelt. Der Einsatz eines J2EE Applikationsservers ist in unserer Architektur also nicht zwingend. Die Applikationslogik verwendet wiederum das JMS Kommunikationsmodell um mit mobilen Geräten, sowie mit anderen Server-Anwendungen, JMS Nachrichten auszutauschen.

Der JMS Gateway bildet die "Brücke" zwischen der JMS Server-seitigen Welt und der mobilen Welt. Der Gateway kann JMS Nachrichten in die verschiedensten Drahtlosformate (WML, SMS, usw.) umwandeln, oder Nachrichten an mobile JMS Anwendungen mittels GPRS und andere Protokolle weiterleiten. Formatumwandlungen, wie z.Bsp. JMS-zu-WML oder WML-zu-JMS, werden vom Gateway transparent mittels XML Style Sheets vollzogen.

6 Zusammenfassung

Das JMS (Java Message Service) API ist einfach zu erlernen und wird den speziellen Bedürfnissen von mobilen Anwendungen gerecht. Die zuverlässige und sichere Zustellung von Daten mittels verschiedenster drahtlosen Datendienste wird durch eine geeignete JMS Infrastruktur sichergestellt.

In diesem Artikel stellten wir fest, dass Informationsdienste der nächsten Generation aus einer skalierbaren Server-Seite, sowie aus anwendungsspezifischer Software für die mobilen Geräte bestehen. Wir haben eine Architektur bestehend aus JMS, Java Applikations-Servern, und XML/XSL vorgestellt, welche die Entwicklung von skalierbaren, mobilen Anwendungen ermöglicht.

7 Referenzen

[WHATIS] Information zu Akronymen wie GPRS, UMTS, Bluetooth, GSM: http://www.whatis.com

[NOMURA] "*The Wireless Internet*", Nomura Equity Research, September 2000. http://www.nomura.co.uk

[TIMELABS] "*Winning in Mobile eMarkets*", Market study by TIMElabs Research Center, 2000, http://www.timelabs.de

[WROX-JMS] "*Mobile JMS*", Silvano Maffeis, in: *Professional JMS Programming,* WROX Publishing, erscheint Frühjahr 2001.

[SYMBIAN] Symbian Ltd., http://www.symbian.com

[REDHAT] Red Hat Embedded Linux, http://www.redhat.com/embedded/

[J2ME] Java-2 Micro Edition, http://www.java.sun.com/products/j2me

[CLDC] J2ME Connected Limited Device Configuration, http://java.sun.com/products/cldc/

[JMS] Java Message Service Standard, http://www.java.sun.com/products/jms

[IBUS] iBus mobile JMS Middleware, http://www.JavaMessaging.com/

Silvano Maffeis (E-Mail: silvano.maffeis@softwired-inc.com) ist CTO bei Softwired AG in Zürich. Er ist Mitentwickler von iBus – eine auf Java Standards beruhende Middleware-Lösung für mobile Informationssysteme.