Mobile Software-Agenten für neuartige Funktionen und Nutzeffekte in intelligenten Gebäudesystemen

Marco Ahler
Innovationszentrum Intelligentes Haus Duisburg
marco.ahler@ims.fhg.de

1. Motivation

Zur Zeit gibt es einen großen Anstieg an anwendungsbezogener Software in intelligenten Gebäudesystemen. Für Klima- und Heizungssysteme wie auch Produkte aus dem Bereich Komfort und Sicherheit werden immer mehr Funktionen in Software realisiert. Dabei übersteigt die Lebensdauer eines Gebäude die Entwicklungszyklen der Software um ein Vielfaches. Aus diesem Grund werden heute bereits viele dieser Geräte mit einer Ethernet-Schnittstelle ausgestattet, um diese im laufenden Betrieb mit neuer Software (Updates), z.B. einer effizienteren Regelungssoftware, auszustatten.

Momentan bieten Internet-Plattformen eine gute Möglichkeit für Endkunden herstellerspezifische Updates zu bekommen. Diese aus der Informationstechnik bekannte Vorgehensweise kann auch für Haussysteme (z.B. Waschmaschine, Heizung) genutzt werden. Hierbei ist es denkbar, dass sich die Subsysteme selbst in die Firmen - Plattform einwählen und das passende Update herunterladen. Bei der Vielzahl der in Zukunft verfügbaren intelligenten Hausgeräten muß ein automatisches Updaten von Software und die Integration neuer Softwareprogramme unterstützt werden. Über Internetplattformen können dann auch neue, kostenpflichtige Dienste von Serviceanbietern zu den einzelnen Haushalten gelangen und dort, je nach Abonnement, auch wieder vom Serviceprovider entfernt werden.
Diese Dienste kommen vor allem aus dem Bereich der Sicherheit, des Komforts und des energieschonenden Wohnens, wo sie eine erstmalige Realisierung oder eine Optimierung der vorhandenen Technik implementieren. Hierzu gehört z.B. die Überwachung sensibler Software (allzeit verfügbare Systeme in lebenssichernden Anlagen) oder aber die Optimierung von Regelungen (Heizungsregelungen) unter Einbeziehung aller verfügbarer Parameter im Haus.
Ein anderer Bereich ist die Fernwartung. Hier sollen spezifische Dienste in die Plattform integriert werden, um dann selbständig die fehlerhaften Daten zu protokollieren und dem Wartungstechniker zu zusenden. Zur Zeit existieren erste Ansätze für diese Techniken, erlauben aber kaum durchgreifende Verfahren, um eine Fehlererkennung effizient realisieren zu können.
Im weiteren Text soll eine Möglichkeit für die Realisierung vorgestellt werden. Dabei wird auf bereits bekannte Internettechnologien, wie den mobilen Softwareagenten und dem OSGi-Standard, gesetzt.

2. Stand der Technik

Im folgenden sollen kurz zwei Technologien beschrieben werden, die zur Zeit entweder als Standard spezifiziert oder schon in Produkten integriert sind. Diese werden nach Meinung des Autors eine entscheidende Rolle bei der Vernetzung verschiedener Geräte innerhalb des Hauses spielen.
Für die o.a. dynamischen Updates von Software und die Integration neuer Dienste (Softwareprogramme) während des Betriebs wird der OSGi-Standard eingesetzt (siehe 2.1 OSGi-Framework). Dieser Standard erlaubt auch geräteübergreifende Applikationen, wie sie bei Optimierungen im Haus nötig sind. Für hochdynamische Regelungen und Optimierungen im Bereich Sicherheit, Komfort und energieschonendes Wohnen können mobile Softwareagenten als Kommunikationsmechanismus eingesetzt werden, um die im Gebäude vorhandenen heterogenen Netze optimal zu nutzen (siehe 2.2 Software-Agenten). Im weiteren werden diese Techniken detaillierter vorgestellt.

2.1 OSGi-Framework

Der OSGi–Standard unterstützt, als zentrale Intelligenz im Haus, die Anbindung der verschiedenen Bussysteme miteinander. Um eine hohe Flexibilität nutzbarer Hardware zu erreichen, wurde Java als plattformunabhängige Programmiersprache für die Implementierung busspezifischer Protokolle gewählt.

2.1.1 Personal Java

Java ist eine weitgehend betriebssystem- und hardwareunabhängige Sprache, bzw. eine komplette Entwicklungsplattform, deren Fokus zum einen auf dem Internet/Intranet, zum anderen in dem Bereich der datenbasierenden Konsumerelektronik liegt. Des Weiteren gilt Java als netzwerksicher und dadurch geeignet für Anwendungen, die in einer heterogenen Netzwerkumgebung ablaufen sollen. Entwickelt wurde Java von Sun Microsystems. Die offizielle Definition von Sun für Java lautet:

> Java: eine einfache, objektorientierte, dezentrale, interpretierte,
> stabil laufende, sichere, architekturneutrale, portierbare und
> dynamische Sprache, die Hochgeschwindigkeitsanwendungen und
> Multithreading unterstützt.

Java-Quellcode wird durch den Java-Compiler in ein plattformunabhängiges Bytecode-Format übersetzt. Dieser wird auf der Zielmaschine durch die so genannte Java-Virtal-Machine (JVM) interpretiert, einem virtuellen Computer, welcher nur im Speicher eines Rechners oder Kundengerätes zur Laufzeit resistent vorhanden ist. Die JVM ist der zentrale Kern von Java und muß auf jedem Gerät vorhanden sein, welches Java-Anwendungen ausführen möchte. Die JVM stellt eine Abstraktionsschicht zwischen dem kompilierten Bytecode und der zugrunde liegenden Hardwareplattform und dem Betriebssystem dar und realisiert die Umsetzung des Bytecode in ausführbare Prozessorbefehle.
Bedingt durch die JVM ist ein hoher Speicherbedarf der Hardwareplattform nötig. Auf eingebetteten Systemen sind allein für die Personal-JVM (siehe Abbildung 1:„Übersicht und Einteilung der verschiedenen Java-Versionen“) – eine abgespeckte JVM – mehr als 2 MByte RAM nötig. Die JVM auf einem PC benötigt ein Vielfaches an Arbeitsspeicher. Außerdem tragen die Programme zur Erhöhung des nötigen Arbeitsspeichers noch erheblich bei, so daß

auf einem eingebetteten System bei der Unterstützung von Standard-Klassen (awt, util, lang) über 6MByte RAM benötigt werden.

•	• Java	• PersonalJava	• EmbeddedJava	• JavaCard
• ROM	• >4 Mbytes	• 1 Mbytes	• Mbytes	• MBytes
• RAM	• 4-8 Mbytes	• >2 Mbytes	• <512 Kbytes	• 512 bytes
• Prozessor	• 100 Mhz+	• 50 Mhz+	• 25 Mhz+	• 300 KIP

Abbildung 1:„Übersicht und Einteilung der verschiedenen Java-Versionen“

Die Stabilität der Programmiersprache Java beruht zum Teil auf den Garbage Collector, der nicht benutze Speicherbereiche zufällig oder in festen zeitlichen Abständen immer wieder frei gibt.
Zudem sorgt die JVM dafür, daß zur Laufzeit benötigte Informationen bereitstehen. Stehen diese nicht zur Verfügung können Programmbibliotheken dynamisch von einer beliebigen Maschine geladen werden. Diese Möglichkeit Programme noch zur Laufzeit nachzuladen wird über die Java-Klasse „Classloader“ realisiert. Diese Technik wird ebenso vom OSGi-Standard genutzt als auch von den später beschriebenen mobilen Softwareagenten, um neue Programme oder Updates nachladen zu können.

2.1.2 Technologie

Weiterhin setzt der OSGi-Standard noch eine weitere Abstraktionsebene – das OSGi-Framework – auf die Java-VM, um Schnittstellen zu den heruntergeladenen Programmen anbieten zu können (Lit. 1 [OSGi]). All diese Programme müssen auch selber bestimmte Schnittstellen zum Framework implementieren, über die sie gemanagt werden können. Sobald Programme diese Schnittstellen zur Verfügung stellen werden sie Bundles genannt. Die Abstraktionsebene „OSGi-Framework“ verwaltet die verschiedenen Bundles (siehe Abbildung 2:„Das OSGi-Service-Gateway“).

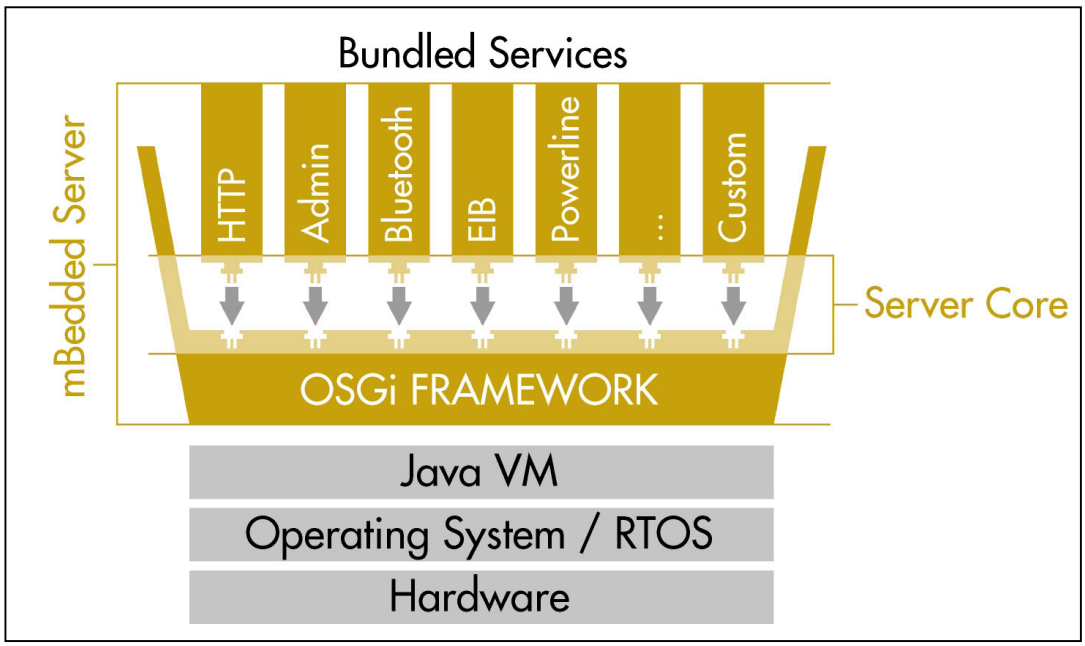

Abbildung 2:„Das OSGi-Service-Gateway“

Schnittstellen der unterschiedlichen Bundles werden nun dem OSGi-Framework bekannt gemacht und können von anderen Bundles genutzt werden. Die Schnittstellen zum Framework sind, wie o.a. bei allen anderen Bundles auch, nach OSGi-Spezifikation standardisiert. Das Framework verwaltet die vorhandenen, auch proprietären, Schnittstellen und ermöglicht *auch* applikationsübergreifende Programme und ein Management der Bundles. Dazu gehört zum Beispiel das Updaten von Bundles aus dem Internet (siehe Abbildung 3:„OSGi-Standard in einem Residential Gateway“)

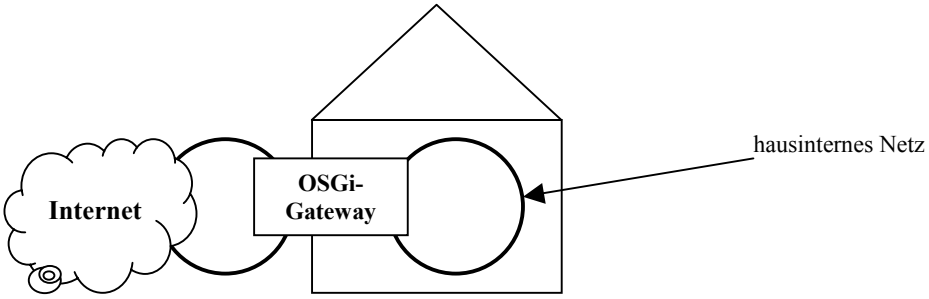

Abbildung 3:„OSGi-Standard in einem Residential Gateway“

2.1.3 Überblick - OSGi-Standard

Vorteile einer Anbindung an ein OSGi – Framework gegenüber herkömmlichen statischen Verfahren:

- Hohe Wiederverwertbarkeit von Softwaremodulen
- Entkopplung der Applikation von den verschiedenen Geräteschnittstellen
- Dynamische Verwaltung von Anwendungen und Diensten durch den Standard OSGi
- Anschlußmöglichkeiten für alle Arten von mobilen Endgeräten wie PDA, Handy und Notebooks - für die Nutzung von "außen" und von "innen".
- Auf Web-Services aufsetzende Kommunikationsschnittstellen zur plattformüber-greifenden Interoperabilität und Integration von Web-Services.

Die meisten Nachteile ergeben sich aus dem Nutzen der Programmiersprache Java:

- Im Vergleich zu proprietären Lösungen größere Anforderungen an Speicher und Prozessorleistung.
- Die OSGi-Spezifikation beinhaltet derzeit gute Definitionen für Systemdienste, aber nur wenige für Anwendungsdienste
- Noch keine Erfahrungen aus Markteinführungen vorhanden, da u.a. die Prozesskette mit vielen Teilnehmern und unterschiedlichen Interessen das Gesamtsystem sehr komplex macht.

Das OSGi-Framework dient u.a. zur Verwaltung von Programmteilen, die sogenannten Bundles, und deren Schnittstellen. Die Kommunikation geschieht auch weiterhin wahlweise über proprietäre oder standardisierte Buskommunikationsprotokolle. Diese sind in Java auf einer sehr hohen Abstraktionsebene implementiert und können sehr dynamisch angepaßt, geändert, erneuert und erweitert werden.

2.2 Software-Agenten

Software Agenten werden in den verschiedensten Gebiete der Informatik wie der künstlichen Intelligenz, den verteilten Systemen und den Betriebssystemen untersucht und entwickelt. Daher kursieren auch verschiedene Auffassungen über die Leistungsmerkmale und Eigenschaften dieser Agenten. Ein Software Agent kann als ein Stück Programm-Code beschrieben werden, das in der Lage ist durch genaues Einhalten der Vorgaben seines Inhabers in dessen Namen eine Aufgabe zu erfüllen (Lit. 2 [AgentFramework]).
Software Agenten lassen sich in verschiedene Agenten aufteilen:

- System Agenten
- User Agenten

Für die beschriebenen neuen Funktionen und Anwendungen in intelligenten Gebäuden werden mobile Softwareagenten eingesetzt und werden deswegen im weiteren näher erläutert.

2.2.1 Mobile Softwareagenten

Mobile Agenten gehören zu den User Agenten. Diese Aufteilung ist sehr verbreitet und wird bei vielen Agenten Systemen eingesetzt. Danny Lange, der das IBM-Entwicklungsteam für mobile Softwareagenten leitete beschreibt wie folgt, warum mobile Softwareagenten verwendet werden sollten (Lit. 3 [AgentenIntro], Lit. 4 [AgentMain]):

- Mit ihnen lässt sich die Arbeitsbelastung verringern, denn mobile Agenten wandern zu einem System und erledigen ihre Arbeit dort, anstatt das ganze Netzwerk für Nachrichtenübermittlungen zu beanspruchen.
- Sie verkürzen die Wartezeiten im Netzwerk, da sie sich auf dem lokalen Rechner aufhalten können und nicht von einem Remote-Computer aus arbeiten müssen.
- Sie können Protokolle verkapseln, während sie sich durch das Netz bewegen und mit anderen mobilen Agenten kommunizieren.
- Sie sind autonom tätig, so dass sie auch dann weiterarbeiten können, wenn die Netzwerkverbindung abbricht.
- Sie passen sich dynamisch an, wenn sich die Systembelastung verändert.
- Sie sind in heterogenen Netzen einsetztbar.
- Sie sind fehlertolerant, da sie sich aus einem System wegbewegen können, in dem Probleme auftreten oder das zu versagen droht.

Diese genannten Punkte machen mobile Softwareagenten zu einem hochflexiblen und sicheren Kommunikationsmechanismus für die heterogene und flexible Infrastruktur eines intelligenten Gebäudes.

2.2.2 Technologie

Mobile Agenten sind Softwareagenten, die autonom Aufträge abarbeiten können. Zusätzlich haben sie die Fähigkeit zu migrieren, d.h. sie können ihre Aufgaben in einem Netzwerk erledigen. Dabei suchen sie System Agenten auf, die genau den Service anbieten, den sie suchen. Um den Service nutzen zu können verbinden Sie sich mit einer Softwareschnittstelle auf einer anderen Plattform, die sogenannte Location. Dort können die spezifischen Dienste genutzt werden. Anhand des Ergebnisses kann der Softwareagent nun weitere Entscheidungen treffen. Möglich wäre der Versand des Ergebnisses an einem vorher erzeugten System Agenten, der dann das Ergebnis anzeigen oder weiter verarbeiten kann.

Der Mobile Agent würde sich dann nach erfolgreichem Versand selber beenden, um auf dem Host wieder Ressourcen freizugeben. Es wäre natürlich auch denkbar, dass der Mobile Agent weiter migriert um andere ihm aufgetragenen Aufgaben zu erledigen. Damit erreicht er folgende Zustände innerhalb seiner Lebensdauer.

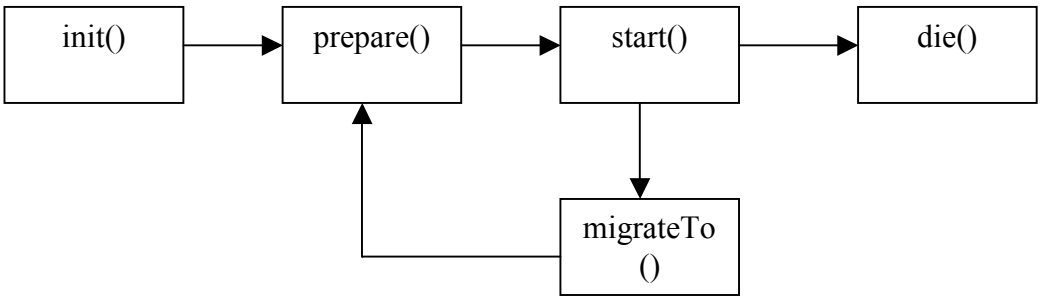

Abbildung 4:„Lebenszyklus eines Mobilen Agenten“

Die Migration ist ein Mechanismus um die aktuelle Ausführung eines Agenten auf einer anderen Location fortzuführen (siehe Abbildung 5:„Migration eines Mobilen Agenten“). In einem Agenten System wird die Migration durch den Agenten und nicht durch das System initiiert. Wobei die Durchführung, also die Migration, die Location vornimmt. Nach dem der Aufruf zum Migrieren erfolgt ist, wird die Ausführung des Agenten gestoppt, von der aktuellen Location entfernt und zu der Ziel-Location transportiert. Danach wird die Ausführung des Agenten fortgesetzt. Ein Agent besitzt folgende Zustände:

- Der Programm-Code (Programmzustand)
- Der Inhalt der Instanzvariablen (Datenzustand)
- Der Ausführungszustand

Für eine ausgereifte Migration müssen alle drei Zustände zur Ziel-Location übermittelt werden. Deswegen muß festgelegt werden, welche Objekte zum Agenten gehören und welcher Code für diese Objekte gebraucht wird. Dann kann das Programm, die Daten und der Ausführungszustand in eine Transport-Form transformiert werden.
Darüber hinaus dürfen nur die Agenten komplett von der Location entfernt werden, die keine andere physikalische Referenz zu einem Objekt haben, daß der migrierende Agent mit sich trägt.

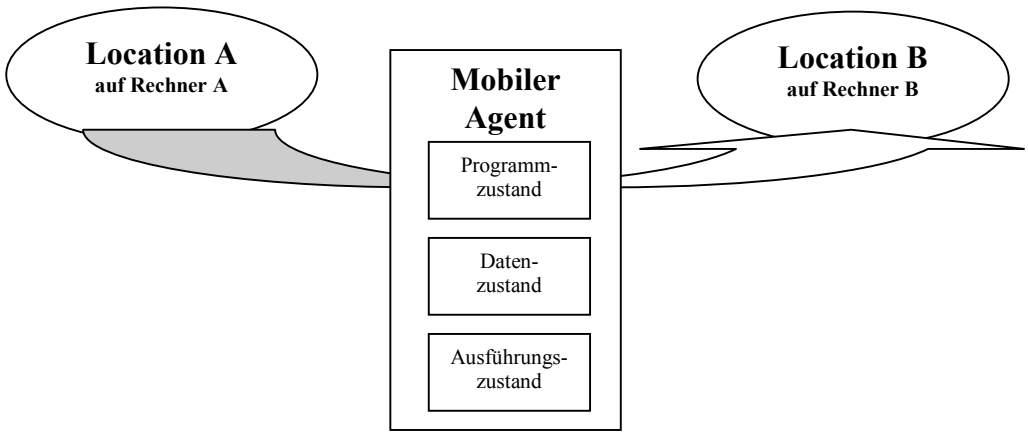

Abbildung 5:„Migration eines Mobilen Agenten“

Diese ausgereifte Migration ist eine sehr komplexe Durchführung, weil der Ausführungszustand mit dem Programm-Code und den Datenzustand transportiert werden muss. Der Programm-Code stellt im Fall von Java die .class-Dateien dar. Datenzustände sind während der Laufzeit eines Agenten auftretende Daten, die in Variablen abgespeichert werden.

2.2.3 Anwendungen

Mobile Softwareagenten werden im Internet für Suchdienste mehrfach eingesetzt. Als eines der bekanntesten und größten Suchdienste im „World Wide Web“, setzt "Google" mobile Softwareagenten innerhalb ihres Clusters ein, um eine schnelle und effektive Suche zu gewährleisten. Dabei werden bei einer Suchanfrage mehrfach mobile Softwareagenten gestartet die das Rechner-Cluster durchsuchen und die gefundenen Ergebnisse dem Benutzer anzeigen.
Auch große Kreditinstitute setzten für eine schnelle Suche mobile Softwareagenten ein. Doch nicht nur bei der Suche sondern auch in heterogenen Produktionsnetzen von Firmen werden sie zur Kontrolle von Anlageparametern als auch zur Überwachung eigener Software eingesetzt, da sie selbständig Entscheidungen treffen und lokal reagieren können. Außerdem sind sie sehr sicher (safty), denn durch die Möglichkeit migrieren zu können, können instabile Plattformen umgangen werden.

3. Mobile Softwareagenten in intelligenten Gebäuden

Die Integration der Agenten - Technologie in bestehende Netzinfrastrukturen wird erleichtert, wenn es innerhalb des Gebäudes eine zentrale Verwaltungseinheit gibt, die den Zugang zu externen Netzen und Geräten im Haus erlaubt. Der OSGi-Standard bietet hierfür gute Vorraussetzungen, da neue Dienste und Services in Form von Bundles auf dem OSGi-Framework (s.o. 2.1 OSGi-Framework) während des Betriebs integriert werden können.
Damit mobile Softwareagenten sich in bestehende Strukturen bewegen können benötigen sie Locations, an die sie sich andocken können. Diese Locations können in Form von Bundles jederzeit auf das OSGi-Framework installiert werden und bieten so den Agenten die Möglichkeit zur Datenabgabe und -aufnahme.

3.1 Integration dieser Technologien in das intelligente Haus (inHaus)

Locations können als Bundles auf einem OSGi-Gateway gestartet werden, so dass mobile Software-Agenten lokal verfügbare Services nutzen können. Außerdem können die Locations-Bundles mobile Software-Agenten zur internen Kommunikation (z.B. Bedienung über PDA, lang andauernde Berechnungsprogramme zur Effizienzüberwachung, Redundanz für sicherheitsrelevante Applikationen) zwischen hochdynamischen Netzteilnehmern einsetzen. Damit kann die Technik der mobilen Software - Agenten konsequent in den o.a. Standard integriert werden und vergrößert die Funktionalität und Flexibilität der einzelnen Systeme.
Verglichen mit herkömmlichen Methoden, lassen sich daraus folgende Vorteile ableiten:

- Anwendungen in heterogenen Netzen
- Geringe Netzlasten
- Applikationen für hochdynamische Netzteilnehmer

3.1.1 Mobile Softwareagenten-Kommunikation auf einem OSGi-Framework

In Abbildung 6:„Kommunikation zwischen verschiedenen Plattformen im und mit dem Haus“ sind innerhalb des hausinternen Netzes zwei Locations dargestellt. Die Location vom spezifischen Gerät benötigt kein OSGi-Framework und kann deswegen sehr klein und schlank gehalten werden. Es muß lediglich eine Schnittstelle zu den mobilen Softwareagenten und den internen Funktionen des physikalischen Gerätes zur Verfügung stellen.

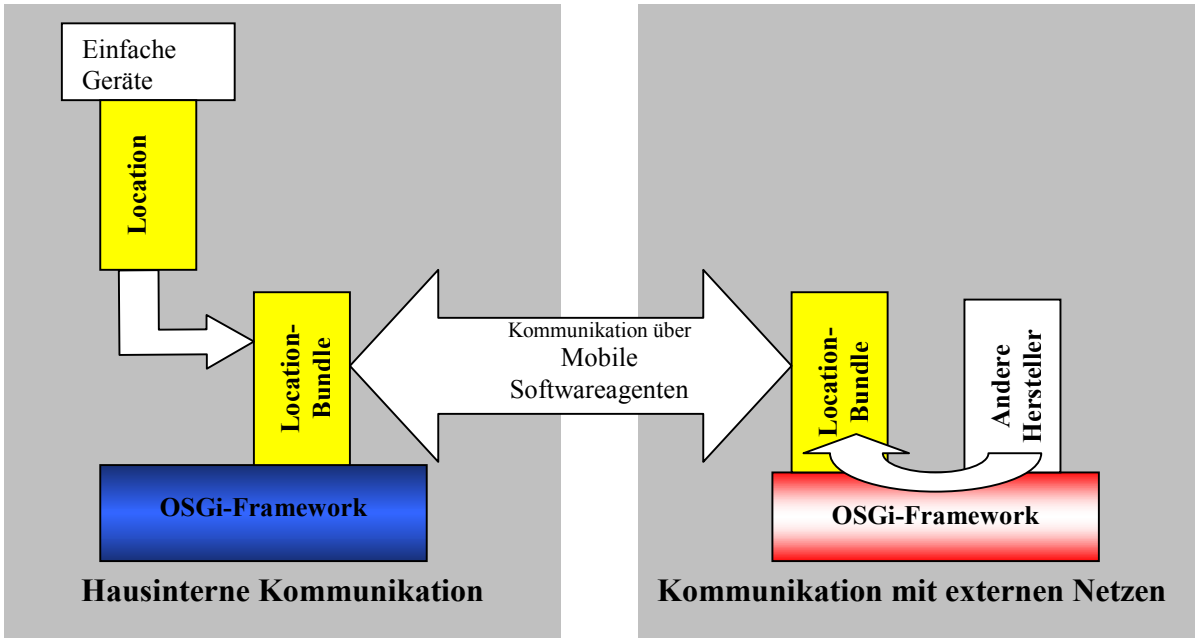

Abbildung 6:„Kommunikation zwischen verschiedenen Plattformen im und mit dem Haus“

Die mobilen Agenten sind in diesem Fall herstellerabhängig, da nur der Hersteller weiß welche Funktionen sein Gerät unterstützt. Somit müßte auch der Hersteller ein spezifisches Location-Bundle zur Verfügung stellen, damit sich seine mobilen Agenten sich hier anbinden können.
Zwischen den OSGi-Frameworks können die mobilen Agenten über die Locations-Bundles kommunizieren und hierüber nicht nur die aktuellen Parameter sondern auch Protokolle zu den verschiedenen Plattformen übertragen.
Innerhalb eines anderen OSGi-Frameworks stehen diese Funktionen wieder allen Teilnehmern offen (siehe Abbildung 6:„Kommunikation zwischen verschiedenen Plattformen im und mit dem Haus“ - externe Netze). Über diese standardisierten Schnittstellen zum Framework können nun auch andere Hersteller auf die Funktionalität der mobilen Softwareagenten zugreifen.

3.1.2 Einführung in die inHaus-Integration

Das Agenten System wurde auf ein OSGi-Framework portiert und mit speziellen Applikationen getestet, um Vergleiche mit herkömmlichen Verfahren machen zu können. Weiterhin wurden einige "Andockstellen" geschaffen, die eine Abfrage der gerätespezifischen Parameter erlauben und andererseits die Bindung von mobilen Softwareagenten unterstützen.

Für die Integration temporärer Netzteilnehmer, die einen dynamischen Netzwechsel erlauben wurde beispielhaft auf einem Personal Digital Assistent (PDA) ein mobiler Agent gestartet,

der dann zum Location-Bundle auf einem OSGi-Gateway gewandert ist. Danach kann der PDA das Netz verlassen oder ausgeschaltet werden. In der Zwischenzeit absolviert der Agent seine zu erledigende Arbeit und überträgt die Parameter bei Wiedereintritt der dynamischen Netzteilnehmer. Die Abbildung 7:„Die Bedienoberfläche des Agenten-Systems für den PDA“ zeigt die Oberfläche des PDAs, welche zum Starten der Agenten benutzt werden kann.
Zu den Arbeiten, die der mobile Softwareagent erledigt, gehören auch Testprogramme von Herstellern spezifischer Produkte. So brauchen die Analyseprogramme keine leistungsstarken Rechner, sondern können nach dem Start die Plattform wechseln und die Infrastruktur innerhalb des Hauses nutzen. Parallel sollen Überwachungsprogramme die restliche Software anderer Systeme überwachen. Hierzu protokollieren sie z.B. den Speicherverbrauch und können abhängig davon oder anderer Parameter eine Migration veranlassen, um das System vor einem möglichen Crash zu schützen (Safty-Aspekt von Software in sicherheitsrelevanten Umgebungen). Um ganz sicher zu gehen werden die Programme schon beim Start geklont um diese Instanzen redundant im System zu haben. Auch das Klonen gehört zu den Grundfunktionalitäten eines mobilen Softwareagenten und erleichtert das automatische Kopieren gleicher Software. Dabei wird aber nur ein System gestartet. Das ruhende System wird, wie es bei mobilen Softwareagenten üblich ist, vorinitialisiert und kann dann direkt gestartet werden (siehe Abbildung 4:„Lebenszyklus eines Mobilen Agenten“). Während der Laufzeit kann ein regelmäßiger Austausch der Instanzvariablen und der Ausführungszustände erfolgen. Um die Busbelastung gering zu halten wird im vorliegenden Fall auf diese Aktualisierung verzichtet und der zweite Softwareagent mit seinem vorinitialisierten Werten gestartet, wenn der erste mobile Agent durch einen Crash gelöscht wurde.

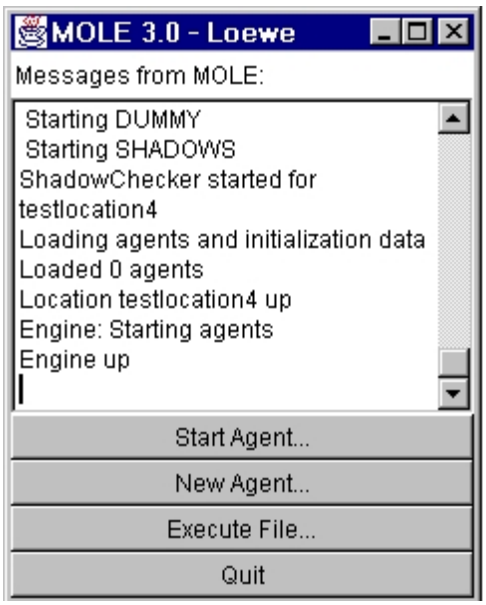

Abbildung 7:„Die Bedienoberfläche des Agenten-Systems für den PDA“

3.2 Anwendungsbeispiele

Im weiteren sollen einige Anwendungsbeispiele folgen, wie sie im Fraunhofer Projekt "inHaus" (Lit. 6 [inHaus]) in Duisburg oder in Stand-alone-Demonstrationen bereits realisiert sind. Dazu gehört vor allem ein Energieoptimierung-Programm, welches mit Hilfe von mobilen Agenten zwischen verschieden Geräten, wie Solarkollektoren, Heizungskörpern, Temperaturthermostaten usw., eine Optimierung vornimmt, indem verschiedenste Parameter ausprobiert werden. Ein Ausprobieren ist hier nötig, da verschiedene Gebäude unterschiedlichste Wärmekoeffizienten besitzen und damit kaum analytische Berechnungen zulassen. Gleichzeitig wurden dynamische Netzteilnehmer (SmartPhone, PDA, Fahrzeug) integriert, um von dort aus die Kommunikation zu testen. Als Agenten System wurde hier das Mole-System der Universität Stuttgart verwendet. Es basiert auf der Programmiersprache Java und ist frei verfügbar. Hiermit werden die ersten Versuche und Test gefahren, um die Effizienz und den Nutzen dieser neuen Technologie bewerten zu können.

3.2.1 Mobile Softwareagenten als Kommunikationsmechnismus innerhalb eines intelligenten Hauses

Um die wandernden Agenten verfolgen zu können bestehen Logging-Files, die den aktuellen Standort bekanntgeben und protokollieren. Außerdem kann auf jeder Plattform angezeigt werden, ob und wieviele Agenten sich gerade auf diesem Rechner befinden. Beispielhaft sei hier der View-Manager des Mobilen Softwareagenten-Systems "Mole" dargestellt, der durch kleine Icons auf dem PDA die Agenten symbolisiert (siehe Abbildung 8: "Der Moleview-Manager angepasst für den PDA"). Da der Zustand und der Standort jedes mobilen Softwareagenten bekannt ist, können verschiedene Parameter von der Zentrale noch nach der Migration übermittelt werden. Hierüber können auch Nachrichten vom mobilen Softwareagenten zur Zentrale geschickt werden. Dieses Vorgehen ermöglicht den Einsatz mehrere Agenten, die parallel arbeiten und über eine Zentrale Nachrichten austauschen können und um somit ihre Parameter abgleichen können.

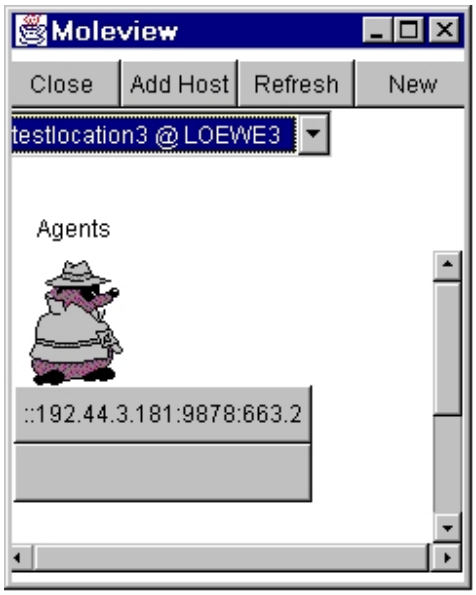

Abbildung 8: "Der Moleview-Manager angepasst für den PDA"

Die Zentrale ist im OSGi-Gateway integriert und ermöglicht hier den Zugriff auch auf andere Geräte und externe Netze. Gleichzeitig können aus dem Internet kommende Agenten untersucht und auf Sicherheitskriterien überwacht werden. So entsteht eine "Firewall" für Agenten aus dem Internet. Die Authentifizierung kann über standardisierte Sicherheits-mechanismen geschehen; soll aber im weiteren Text nicht Bestandteil der Diskussion sein.

3.2.2 Kommunikation mit Plattformen von Serviceanbietern im Internet

Für die Gerätehersteller eröffnet sich durch die Vernetzung der Geräte im Haus ein enormes Innovations- aber auch Einsparungspotential. Nicht nur die Bedienung ihrer Geräte läßt sich über Standardmechanismen erreichen, sondern auch die Software kann zu einem späteren Zeitpunkt aktualisiert und erweitert werden. Zu den Erweiterungen gehören zum Beispiel Dienste zur Optimierung von Vorgängen im Haus (siehe Abbildung 9: "Kommunikation zwischen Internetplattformen und internen Netzen).

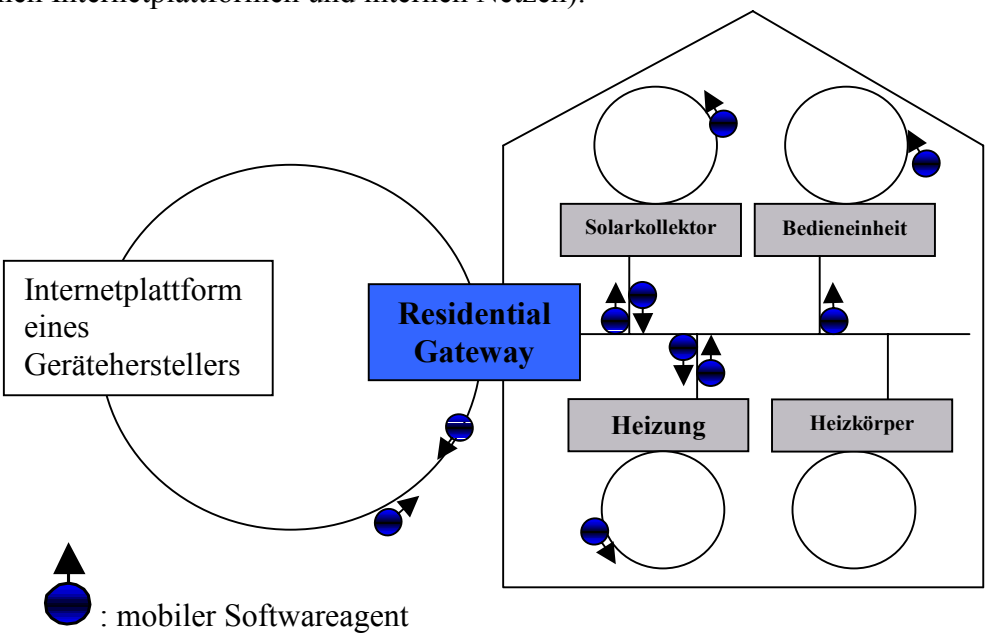

Abbildung 9: "Kommunikation zwischen Internetplattformen und internen Netzen

Kostenlose Dienste können entweder zur Kundenbindung dienen oder bei hohem Einsparpotential durch ressourcenschonendes Wohnen dem Bewohner in Rechnung gestellt werden. Weiterhin können Testapplikationen im System nach Fehler zu suchen und so aus der Ferne auch behoben werden.
Da die Systeme immer komplexer werden und durch die Vernetzung weitere Abhängigkeiten untereinander entstehen, werden auch die Optimierungsschritte und Fehleranalyseprogramme immer komplexer. Gerade des wegen lohnen sich hier mobile Softwareagenten, die selbständig im Subnetz der abhängigen Geräte hin und her wandert. So können sowohl eine Reihe von Parametern zur Optimierung wie auch zur Fehleruntersuchung eingestellt werden.
Über die zentrale Intelligenz (Residential Gateway) besteht auch die Anbindung an externe Netze. Hierüber können Hersteller ihre gekapselten Programme auf die Geräte herunterladen. Bei großflächigen Änderungen können "Klon"-Funktionen der Agenten genutzt werden, die sich dann selbständig vervielfachen und in die entsprechenden Geräte einbringen. Somit muß die Internetplattform des Geräteherstellers keine Verbindung zu jedem einzelnen Gerät aufbauen, sondern im Optimalfall nur einen mobilen Softwareagenten kreieren, der dann selbst durch Klonen zu allen Geräten im Haus bzw. erst einmal zu den Residential Gateways wandert.

4. Zusammenfassung und Ausblick

Am Ende dieses Texte soll eine kurze Zusammenfassung der vorgestellten Techniken und Ergebnisse gegeben werden, die innerhalb des Fraunhofer Projektes "Innovationszentrum Intelligentes Haus Duisburg" ermittelt wurden. Weiterhin folgt ein kurzer Ausblick bzw. eine Vision zukünftiger Kommunikationsmechanismen im und um eines intelligenten Hauses.

4.1 Zusammenfassung der dargestellten Ergebnisse

Innerhalb des Projektes "Innovationszentrum Intelligentes Haus Duisburg" wurden u.a. mobile Softwareagenten als Kommunikationstechnologien zwischen vernetzen Heimgeräten und externen Netzen eingesetzt. Entscheidend für die Integration dieser neuen Technologie waren vor allem die Stärken der mobilen Softwareagenten gegenüber herkömmlichen statischen Verfahren:
Es müssen nur die Softwareschnittstellen für die Agenten von den Geräten bereitgestellt werden, um mit allen anderen Netzteilnehmern kommunizieren zu können. Daraus ergeben sich Anwendungen in heterogenen Netzen.
Für Test und Logging-Funktionen treten bei einer zentralen Haltung der Daten enorme Netzbelastungen auf, obwohl oder gerade weil es oft nur Bit-Werte sind, die übermittelt werden müssen. Gerade bei externen Anfragen ist es effizienter, wenn bestimmte *gewisse* Applikationen direkt vor Ort ausführt werden und die gesammelten bzw. ausgewerteten Daten dann übertragen werden. Daraus ergibt sich eine geringere Netzbelastung
Die Mobilität der Menschen spielt eine wichtige Rolle. Dieses spiegelt sich natürlich auch im Haus der Zukunft wieder. Viele dynamische Teilnehmer (SmartPhone, PDA, Fahrzeug) nutzen die Infrastruktur des Hauses und können so die Steuerung verschiedenste Geräte im Haus ermöglichen. Somit entstehen Applikationen für dynamische Netzteilnehmer
Die verschiedenen Studien haben gezeigt, dass mobile Softwareagenten durchaus in bestehende Infrastrukturen intelligenter Häuser zu integrieren sind. Dabei konnten die o.a. Vorteile in weiten Teilen bestätigt werden. Gerade Testprogrammen, die vom Hersteller geliefert werden, können effiziente Lösungen ohne viel Netzverkehr erbringen, wenn sie direkt vor Ort ausgeführt werden und eine Vorauswahl der zu übermittelnden Parameter tätigen.
Bei der Anbindung von dynamischen Netzteilnehmern traten im Anfang des Projektes Dateninkonsistenzen auf, da bestimmte Anfragen an das Haus schon veraltet waren, als der Teilnehmer das Netz wieder betrat. Hier wurden im Test verschiedene Verfahren entwickelt um dieses zu unterdrücken. Dabei werden verschiedene Prioritäten vergeben, die abhängig von den jeweiligen geschätzten Ergebniserwartungszeiten sind. So wird eine lang andauernde Testapplikation sofort gestartet und behält das Ergebnis bis zum Wiedereintritt des dynamischen Netzteilnehmers vor, während bei schnellen Anfragen erst bei Wiedereintritt der Wert abgefragt und übermittelt wird. Die Auswahl, ob es sich um eine rechenintensive, aufwendig Auswertung oder um eine einfache Abfrage handelt wird zur Zeit vom Benutzer festgelegt, soll aber in Zukunft über eine lernende Applikation automatisch erfolgen.

4.2 Ausblick und Vision

In der Zukunft werden die verschiedenen Netze aus dem Wohnhaus, dem Auto und Büro über verschiedenste Standards (UMTS, WLAN, Bluetooth, GPRS, usw.) immer mehr zusammen wachsen. Damit sind die Services jedes Teilnehmers aus jedem Netz erreichbar. Aufgrund des hohen Trends zur Mobilität und der Informationsversorgung muß auch die Software die hohe Flexibilität unterstützen. Schon jetzt sind Fern-Updates und Erweiterungen der Software im Betrieb möglich. Dieses soll zukünftig noch erweitert werden. Mobile Softwareagenten bieten hier Möglichkeiten Software in Kapseln zwischen den Netzknoten zu verschicken, wobei sie selber auf bestimmte äußere Einflüsse reagieren können. So wurden schon Kommunikationsprotokolle in mobilen Agenten gekapselt, um diese mit einer aufzubauenden Kommunikation mitwandern zu lassen.

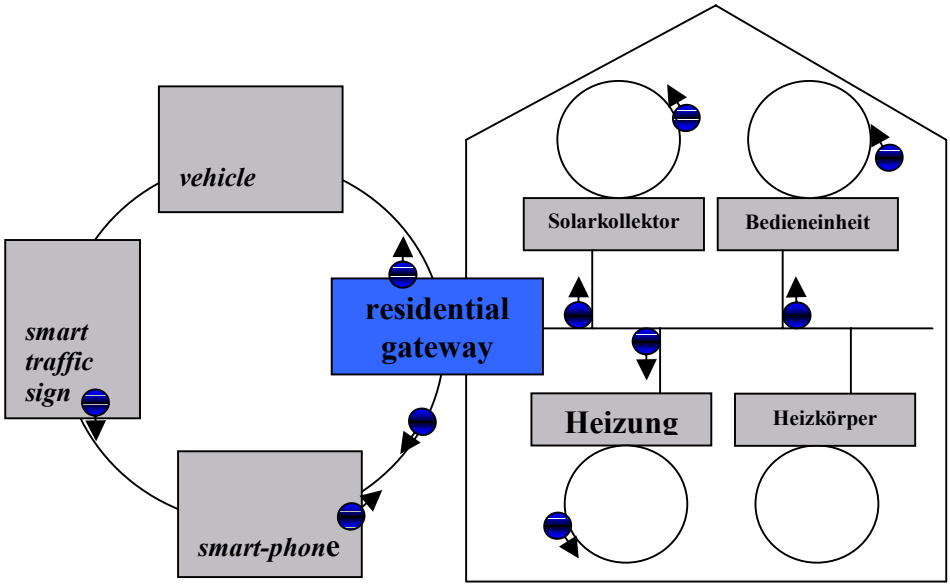

Abbildung 10: "Kommunikation mit externen Netzen"

Ein anderes Beispiel (siehe Abbildung 10: "Kommunikation mit externen Netzen") ist die Kommunikation des Fahrzeugs über eine Andockstation (z.B. Verkehrszeichenanlage) und dem Haus. Der mobile Softwareagent kann an der Ampel in das Fahrzeug wandern und sich so auch physikalisch transportieren lassen und Informationen und Softwareprogramme weitertragen.
Diese neuen Technologien werden einen großen Beitrag liefern, die Akzeptanz für intelligente Haussysteme beim Endverbraucher zu erhöhen.

5 Literaturverzeichnis

Lit. 1 [OSGi]
Open Service Gateway Initiative (OSGi) http://www.osgi.org

Lit. 2 [AgentFramework]
Ichiro Satoh," A Mobile Agent-Based Framework for Active Networks", IEEE Systems, Man, and Cybernetics Conference, October 1999, http://islab.is.ocha.ac.jp/agent/index.html.

Lit. 3 [AgentenIntro]
http://www.minet.uni-jena.de/fakultaet/ips/braunpet/agents/maintro.html

Lit. 4 [AgentMain]
http://jens.nedon.de/studium/agenten98/

Lit. 5 [OSGiIntegration]
D. Frommeyer, M. Ahler "Mobile Softwareagenten als Kommunikationsmechanismus und Middelware für eingebettete Systeme", Oktober 2002, Diplomarbeit Fraunhofer IMS

Lit. 6 [inHaus]
http://www.inhaus-duisburg.de