

Open Services Gateway Initiative (OSGi) – Standardisierung einer offenen Infotainment-Plattform

Open Services Gateway Initiative (OSGi) – Standardization of an Open Infotainment-Plattform

Hans-Ulrich Michel

BMW Group, Fahrzeugforschung, 80788 München

Telefon: +89-382-30136, Fax: +89-382-70-30136, hans.michel@bmw.de

Zusammenfassung

Dieser Artikel beschäftigt sich mit der Entwicklung einer neuen Generation von Infotainment-Systemen im Fahrzeug. Er beleuchtet die strukturellen Veränderungen in der Fahrzeugindustrie und die hieraus resultierenden Herausforderungen und Chancen. Außerdem gibt er einen Einblick in die Standardisierungsaktivitäten, die von der OSGi Alliance (Open Services Gateway Initiative) durchgeführt werden.

Summary

This paper describes the challenges and possibilities that are linked with a new generation of infotainment systems in the vehicle industry. It further takes a look at the standardization activity based on the OSGi Alliance (Open Services Gateway Initiative) as an enabling technology for the development and the deployment of applications to the vehicle infotainment platform.

1. Einleitung

Im einem Zeitalter, in dem das Internet, mobile Endgeräte und Dienste die Welt verändern, verändern sich natürlich auch die Anforderungen an einen Fahrzeughersteller. Um Mobilität zu erhalten und zu verbessern, ist es notwendig sich mit den Bedürfnissen der Informationsgesellschaft von heute und morgen auseinander zusetzen.

Im Laufe der Jahre hat sich die Fahrzeugindustrie von einer rein durch Mechanik bestimmten Industrie zu einer Industrie entwickelt, die primär durch den Bereich Elektrik/Elektronik getrieben wird. 90% aller Innovationen sind heute verursacht durch Elektronik und Software. Der Anteil der Kosten ist permanent gestiegen und erreicht mittlerweile bei Fahrzeugen mit hochwertiger Ausstattung Werte bis zu 35%, bei einem Wachstum von 10-15% per anno, obwohl der Preisverfall bei bestimmten Komponenten wie z.B. Motor- oder Airbagsteuergeräte sehr hoch ist. Die Bedeutung

der Elektronik im Fahrzeug ist also noch viel stärker gestiegen als der dargestellte Herstellkostenanteil.

Kategorisiert man und betrachtet die Wachstumsraten in den einzelnen Anwendungssegmenten stellt man u.a. ein verstärktes Wachstum im Bereich Assistenzsysteme, Informationssysteme und Bussysteme fest. BMW's ConnectedDrive Konzept trägt der steigenden Bedeutung dieser Segmente Rechnung (Abbildung 1).

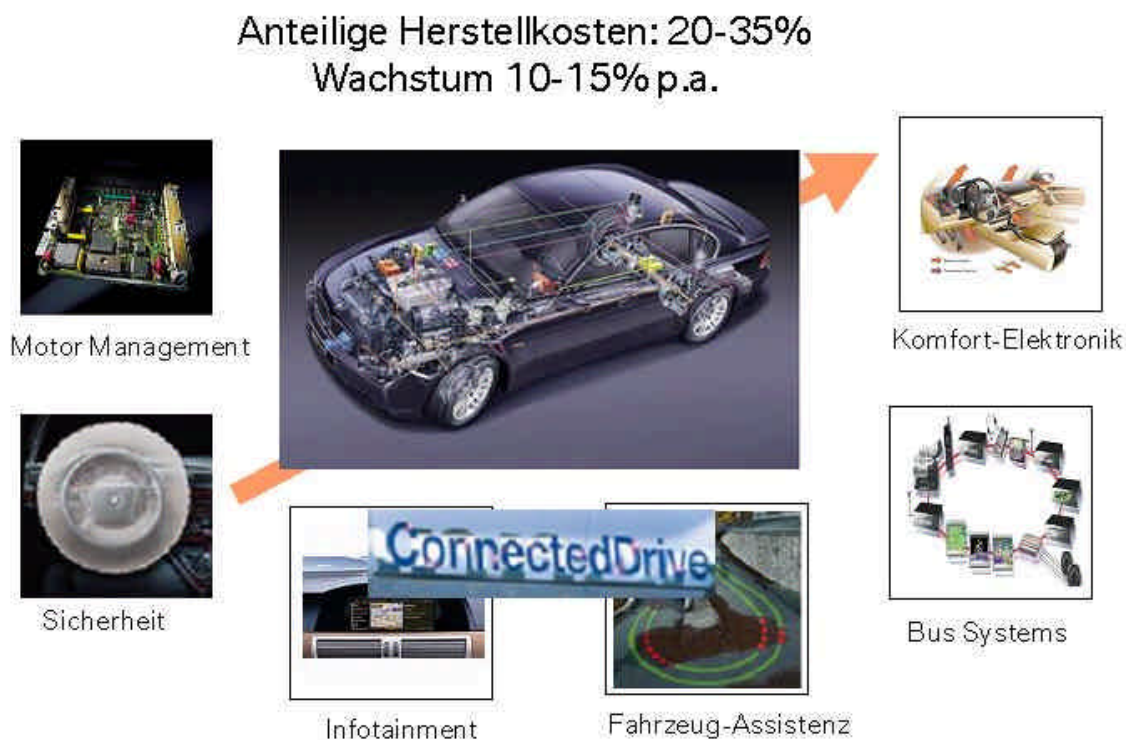


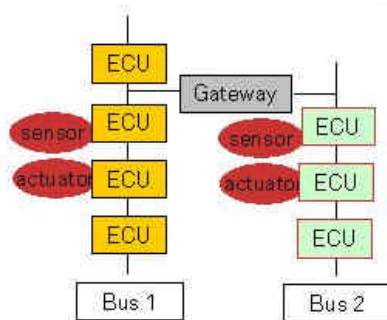
Abbildung 1: Kategorisierung der Fahrzeugelektronik
Figure 1: Categorization of Vehicle Electronics

2. Herausforderung an den Fahrzeughersteller

Eine nähere Betrachtung des Bereiches Bussysteme bzw. Vernetzung zeigt, dass die Anzahl der Steuergeräte aufgrund der steigenden Anzahl von Funktionen im Fahrzeug stetig gestiegen ist. Der zunehmende Vernetzungsgrad ist einer der Faktoren für einen Anstieg der Systemkomplexität im Fahrzeug (Abbildung 2).

SG-Architektur: BMW Luxussegment

Anzahl SG's (ECU)	16-24	33-70
Zunehmende Vernetzung	PT-CAN (500 kBd) I/K-BUS (9,6 kBd) P-BUS (9,6 kBd)	PT-CAN (500 kBd) K-CAN (100 kBd) K-CAN-P (100 kBd) byteflight (10 MBd) MOST (22 MBd)



Steigender Vernetzungsgrad weil:

- Zunehmende Zahl von SG's / Funktionen
- Mehrfachnutzung von Sensorsignalen
- Ressourcensharing
- Verteilte Funktionalität

Abbildung 2: Steigender Vernetzungsgrad im Fahrzeug

Figure 2: Increased networking of vehicle-components

Fokussieren wir im weiteren auf den Infotainmentbereich, dann ist ein weiterer Faktor, der diese Herausforderung unterstreicht, der Technologiepush, der sich in der Consumerindustrie abspielt und der zunehmenden Einfluss der Fahrzeugindustrie gewinnt.

Internettechnologie (webservices), die Entwicklung kabelloser Übertragungstrecken und neuer Endgeräte (Handy, PDA) führen zu einem Angebot an Funktionen bzw. Diensten, die auch im Zusammenspiel mit dem Fahrzeug erwartet werden. Die nahtlose Verfügbarkeit von personenbezogenen Diensten, im Büro oder Heimbereich auf dem PC, zu Fuß unterwegs auf dem Handy oder PDA und im Fahrzeug stellen neue Anforderungen an das Fahrzeug, das zunehmend eingebunden wird in ein komplexes, externes Netzwerk.

Die Integration von Funktionen aus der Consumerindustrie, die verglichen mit der Fahrzeugindustrie wesentlich kürzere Entwicklungs- und Produktlebenszyklen haben, ist somit ein weiterer Faktor für steigende Systemkomplexität (Abbildung 3).

Integration von Funktionen aus der Consumerindustrie mit anderen Lebenszyklen



Abbildung 3: Das Fahrzeug als Teil eines zukünftigen Dienste-Netzwerkes
Figure 3: The vehicle – part of a future service value chain

Die Möglichkeit über unterschiedliche, externe Netzwerke mit dem Fahrzeug zu kommunizieren verschärft die Problematik weiter (Abbildung 4). Welche Informationen bzw. Daten werden im Fahrzeug benötigt, wo sollen die Daten gehalten werden, wer benötigt sie, wie werden Daten synchronisiert, Sicherheitsaspekte bezüglich persönlicher Daten – dies sind alles Themen die betrachtet werden müssen.

Die steigende Systemkomplexität hat nun eine Vielzahl von Konsequenzen, eine wesentliche ist, dass sich die Entwicklungszeiten verlängern und diese schwerer abschätzbar werden.

Die Lösung für diese Problematik heißt Standardisierung. Das Ziel muss die Entwicklung eines offenen Standards auf Funktionsebene sein, der zum einen die unterschiedlichen Produktlebenszyklen zwischen Hardware und Software bzw.

zwischen Consumer- und Fahrzeugwelt entkoppelt und der zum anderen die Komplexität der Interaktion für den Anwendungsentwickler verringert.

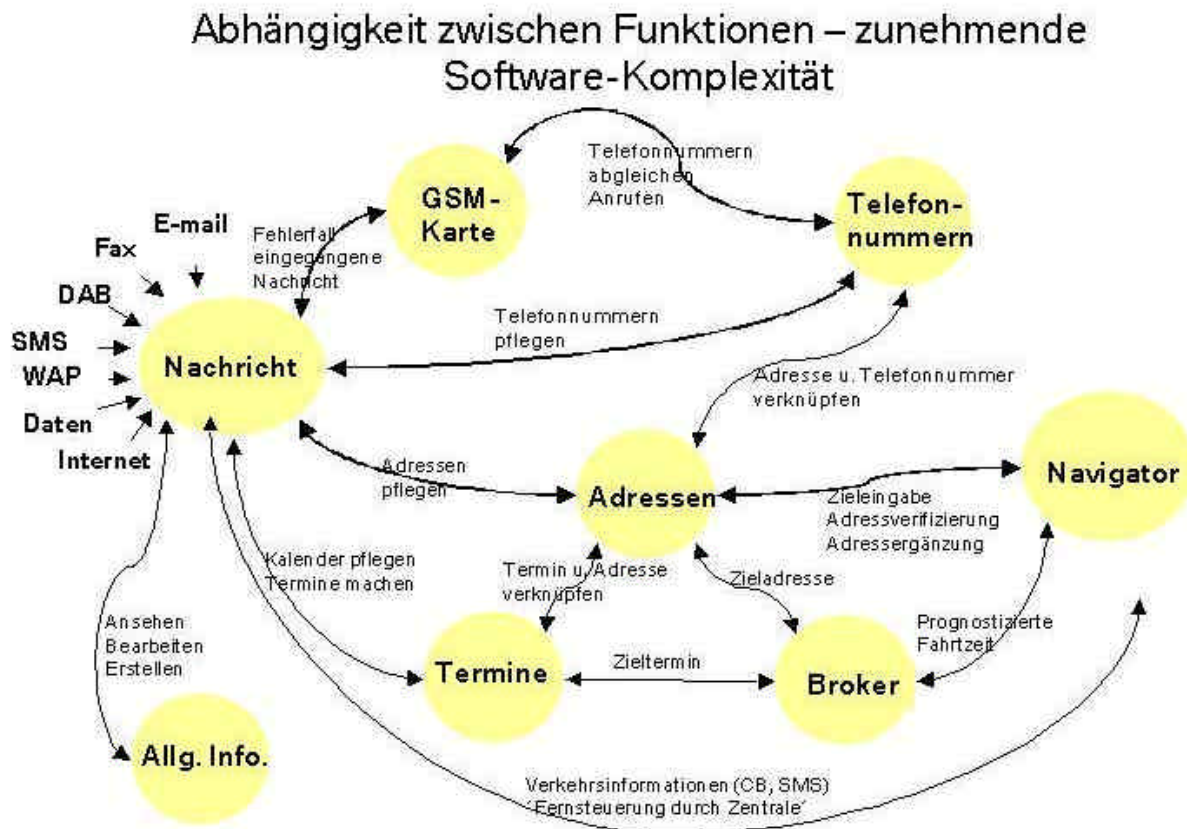


Abbildung 4: Verknüpfung von Funktionen und steigenden Software-Komplexität
 Figure 4: Dependency of functions increases software-complexity

Um dies zu konkretisieren hilft ein Blick auf die Entwicklung der Fahrzeugarchitektur im Infotainmentbereich (Abbildung 5). Hier hat eine Entwicklung stattgefunden, die von völlig unabhängigen Endgeräten zu einer Vernetzung von Geräten und damit Funktionen über ein Bussystem (MOST) geführt hat. MOST (**Media Oriented Systems Transport**, [1]) ist ein Standard, der neben dem Multimedia-Busprotokoll ein Objektmodell spezifiziert und damit die Steuergeräte durch Funktionsblöcke abstrahiert. Das Modell erlaubt Ressourcensharing, wodurch eine Funktion nur einmal für mehrere Nutzer im Busverbund implementiert werden muss, als auch funktionelle Partitionierung, d.h. Funktionen können von einem Steuergerät in ein anderes verlagert werden. Auch das Kombinieren von Funktionen zu höherwertigen Anwendungen wird durch eine hierarchische Strukturierung unterstützt. MOST ist damit ein erster Schritt zu einer offenen Systemarchitektur, die Systemkomplexität verringert und Unabhängigkeit von der Hardware ermöglicht.

Eine wesentliche Herausforderung bleibt aber weitgehend ungelöst: Das permanente Ergänzen von zusätzlicher Funktionalität mit kurzen Lebenszyklen oder das Realisieren von Software-Upgrades in kurzen Zeitabständen kann damit noch nicht ausreichend unterstützt werden.

Um zukünftige Anforderungen zu erfüllen ist ein offenes Plattformdesign notwendig, in dem Funktionen in Form von Softwarekomponenten jederzeit dynamisch geladen bzw. gelöscht werden können und über eine Standardschnittstelle (Standard API) mit einem Software-Plattform kommunizieren, die alle Aufgaben der Interaktion zwischen den Komponenten übernimmt.

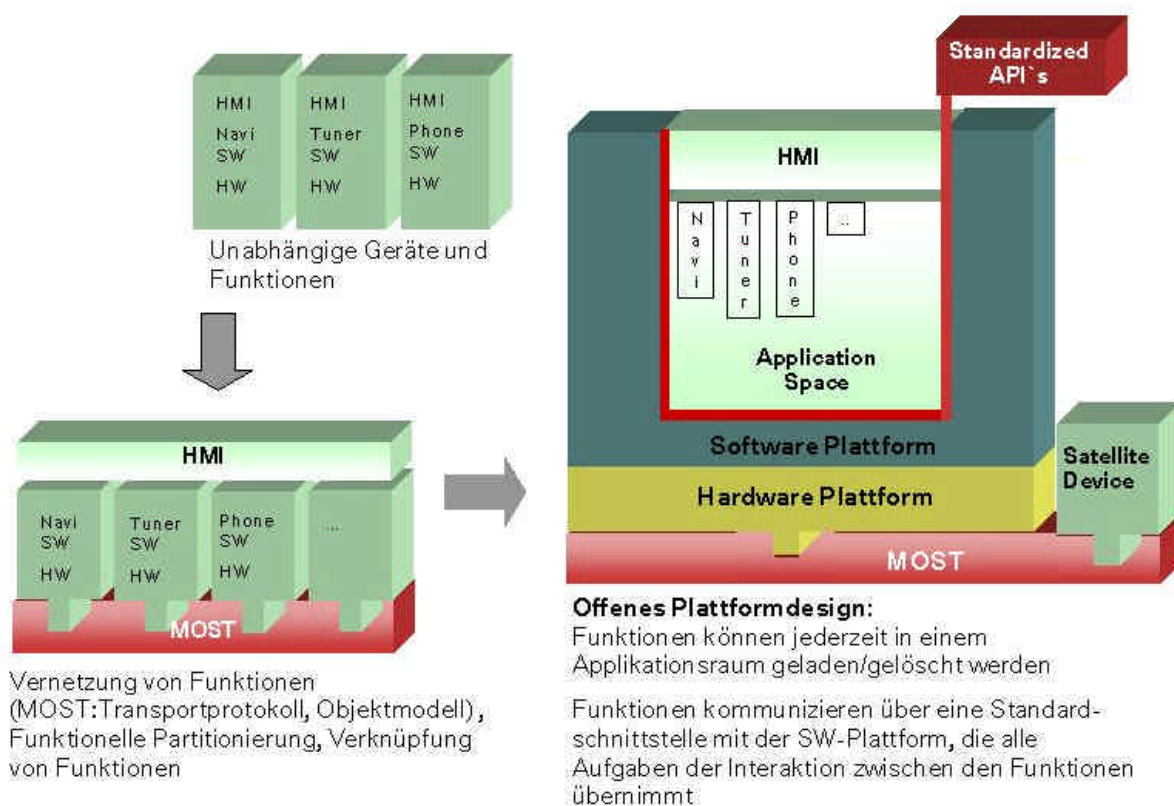


Abbildung 5: Entwicklung der Fahrzeugarchitektur im Infotainmentbereich
 Figure 5: Development of the vehicle-infotainment architecture

Die Standardisierung einer solchen Schnittstelle bzw. einer Softwareplattform auf Anwendungsebene ist damit fahrzeugrelevant und darf nicht, wie bisher, ausschließlich Firmen überlassen bleiben, die im Informations- bzw. Consumerbereich den Markt beherrschen. Die Fahrzeughersteller müssen ihre Anforderungen in entsprechenden, offenen Standardisierungsgremien wie OSGi (Open Services Gateway Initiative, [2]) einbringen und den Standardisierungsprozess mit vorantreiben.

3. OSGi – Standardisierung einer offenen Software-Plattform



Die **Open Services Gateway Initiative** ist eine Allianz aus über 50 Firmen, die sich die Standardisierung eines offenen Plattformdesigns zum Ziel gesetzt hat: OSGi spezifiziert eine javabasierte Software-Plattform (Framework), die eine dynamische Integration von Softwarekomponenten ermöglicht und die Kommunikation zwischen diesen Komponenten steuert. Mit der Spezifizierung zusätzlicher Basisdienste sollen u.a. die Anforderungen unterschiedlicher Märkte, wie beispielsweise der Fahrzeugindustrie berücksichtigt werden. OSGi ist in Arbeitsgruppen gegliedert, wobei speziell die „Vehicle-Expert Group“ dafür verantwortlich ist, die Anforderungen aus der Fahrzeugindustrie in den Standardisierungsprozess einzusteuern (Abbildung 6).

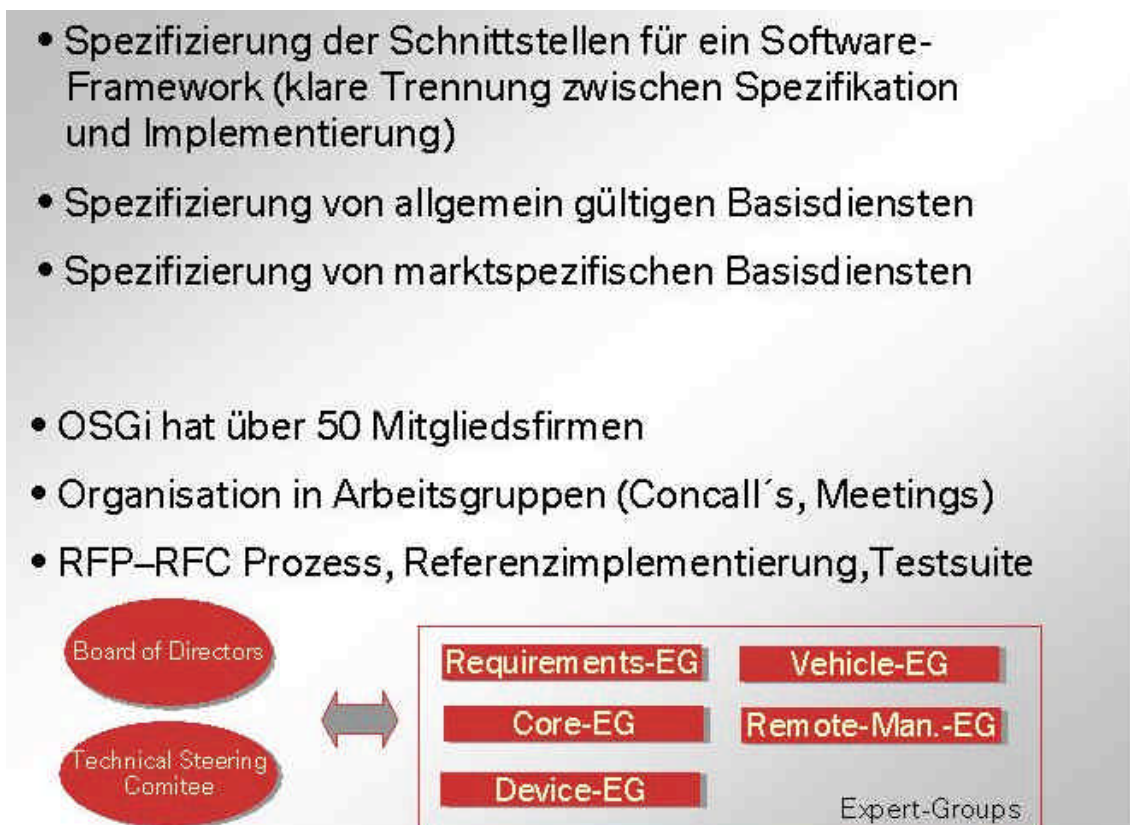


Abbildung 6: Aufgaben und Organisation der OSGi Alliance
Figure 6: Objectives and organization of the OSGi Alliance

3.1 Welche Vorteile bringt OSGi ?

Als Ergebnis der Standardisierungsmaßnahmen durch OSGi ergeben sich Vorteile, die sich weitgehend mit den zukünftigen Herausforderungen an die Fahrzeughersteller decken. OSGi

- erlaubt das dynamische Laden von Funktionen - und bietet damit die notwendige Flexibilität bei der Nutzung von Applikationen im Fahrzeug
- spezifiziert eine standardisierte Schnittstelle zum Software-Framework und Basisdienste – und bildet so die Grundlage zur Verkürzung von Entwicklungszeiten durch den Applikationsentwickler
- bietet ein Sicherheitsmodell auf Basis von Java 2 – und ermöglicht bei gemeinsamer Nutzung von Standardprotokollen für die Übertragung ein durchgängiges Sicherheitskonzept
- integriert Geräte-Management – und unterstützt damit im Zusammenspiel mit der Plug'n Play-Funktionalität die automatische Suche nach einem optimalen Gerätetreiber
- reduziert aufgrund der Standardisierung Abhängigkeiten vom Plattform-lieferanten und Service-Provider

3.2 Grundlagen der OSGi Architektur

Die OSGi Architektur basiert auf einem Komponentenmodell ([3]). Die Softwarekomponenten sind weitgehend unabhängig voneinander und haben klar definierte Schnittstellen und Abhängigkeiten zum Software-Framework. Das spezifizierte Framework erhält die Komponentenstruktur, indem es jeder Komponente einen eigenen „Lebensraum“ (Namens – und Datenbereich) zur Verfügung stellt. Zentrale Aufgaben des Frameworks ist zum einen die Steuerung der Interaktion zwischen den Komponenten und zum anderen die dynamische Verwaltung. Das heißt der Lebenszyklus jeder Softwarekomponente wird durch das Framework gesteuert; eine Komponente kann dynamisch installiert bzw. gestartet oder gestoppt bzw. deinstalliert werden.

3.2.1 Bundles und Services

Eine Softwarekomponente ist in der OSGi-Standardisierung mit dem Namen *Bundle* belegt. Bundles sind Softwarepakete, welche Bibliotheken, Ressourcen oder gemeinsam nutzbare, unabhängige Elemente einer Applikation, in OSGi *Services* genannt, enthalten können (siehe Abbildung 7). Basisdienste sind in OSGi standardisierte Services.

- Ein Bundle ist ein Softwarepaket welches Bibliotheken, Ressourcen oder gemeinsam nutzbare, unabhängige Elemente einer Applikation (Services) enthält.
(ähnlich wie DLL, JPG, COM-Objekte)
- Ein Bundle ist ein Java-Archive (JAR-Datei)
(Zip-basiertes Java-Format)
- Das Framework kann Bundles dynamisch installieren, starten updaten, stoppen oder deinstallieren.
- Ein Bundle registriert keinen, einen oder mehrere Services
Ein Service ist als Java-Interface spezifiziert und kann in mehreren Bundles unterschiedlich implementiert sein
- Suchmechanismen können genutzt werden, um von anderen Bundles registrierte Services zu finden

Abbildung 7: Der OSGi Bundle-Begriff
Figure 7: OSGi Bundles

So kann ein Bundle beispielsweise einen Positions-Service mitführen, dessen Methoden die Möglichkeit bieten, die Position des Fahrzeugs festzustellen. Ist der Positions-Service ein Basisdienst innerhalb der OSGi Standardisierung, dann ist er als Positions-Interface mit seinen Methoden spezifiziert. Der Positions-Service ist somit eine Implementierung dieses Interfaces.

Das Framework bietet nun die Möglichkeit, den Positions-Service in einer Registry zu verwalten und ermöglicht anderen Bundles über einen Suchmechanismus diesen Service zu finden und zu nutzen. Ein typisches Szenario: Ein Bundle, welches einen Park-Service enthält, wird geladen. Die Park-Service Methoden ermöglichen freien Parkraum im Umfeld des Fahrzeugs zu ermitteln, hierzu wird aber die Position des Fahrzeugs benötigt. Ist der Positions-Service registriert, dann kann der Park-Service über das Framework auf ein Service-Objekt des Positions-Service zugreifen und über seine bekannten, da standardisierten, Methoden die Position des Fahrzeugs ermitteln (Abbildung 8). Diese Möglichkeit ändert die Applikations-Entwicklung radikal: Applikationen entwickeln bedeutet weitgehend das Verstehen, Finden und Nutzen von vorhandenen Softwarekomponenten ([4]). Letztendlich entstehen die Applikationen erst durch die Interaktion der Komponenten auf dem Framework. Die Softwarekomponenten können damit von unterschiedlichen Applikationen genutzt werden, wie in Abbildung 9 für eine Navigations- und Location-Based Advertising-Applikation dargestellt.

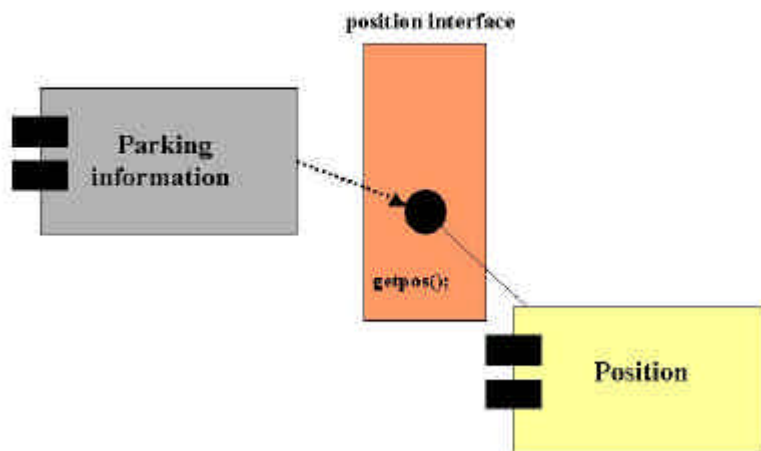


Abbildung 8: Der Park-Service nutzt den Positions-Service über ein standardisiertes Interface

Figure 8: A parking-service communicates with a position-service by using the standardized position-interface

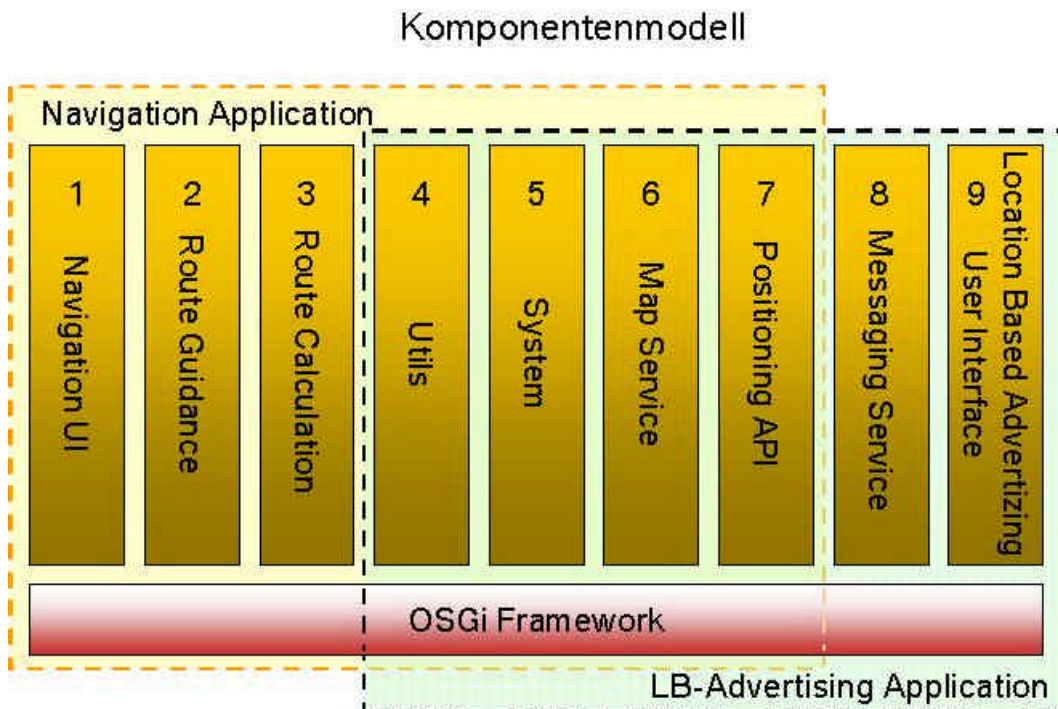


Abbildung 9: Das OSGi Komponentenmodell

Figure 9: OSGi component model

3.2.2 OSGi Basisdienste

In der OSGi Standardisierung sind eine Reihe von Basisdiensten spezifiziert, die dem Applikationsentwickler nicht nur die Arbeit erleichtern, sondern die speziell auch auf die Anforderungen bestimmter Märkte eingehen. Ohne auf die Bedeutung aller Basisdienste einzugehen (siehe hierzu [1]), sollen hier doch beispielhaft einige durch die Vehicle-Expert Group erarbeiteten Services erwähnt werden:

- *Position Service*: Spezifiziert eine Standardschnittstelle für die Erkennung der Position (Geografische Breite, Länge, Höhe, Bewegungsrichtung, Geschwindigkeit). Die Information wird als Measurement-Objekt (mit Genauigkeitsangabe) geliefert.
- *Measurement*: Definiert Measurement-Objekte. Ein Measurement-Objekt enthält einen Zeitstempel, einen Wert (double), einen Fehler (double), Einheiten im SI-System
- *Communication-Service*: ermöglicht das Herstellen von Verbindungen oder das Verschicken von Nachrichten zu einem Kommunikationspartner über Internet, eine Einwahlverbindung oder SMS (nachrichtenbasierte oder verbindungsorientierte Kommunikation)
- *State Management-Service*: Ermöglicht den Zugriff auf Sensoren und Aktuatoren im Fahrzeug. OSGi legt dabei nur den Mechanismus für den Zugriff fest ohne diesen Mechanismus auf eine spezielle Fahrzeugontologie (wie z.B. in MOST spezifiziert) abzubilden.

3.3 OSGi und andere technologische Ansätze

Oft wird OSGi mit anderen technologischen Ansätzen wie der Multimedia Home Plattform (MHP) oder MIDP (Mobile Information Device Profile) verglichen. Der wesentliche Unterschied zwischen diesen verschiedenen Ansätzen liegt in der unterschiedlichen Nutzung bzw. Ausgangssituation begründet:

- MIDP unterstützt z.B. die Nutzung des Mobiltelefons als „Gaming-Konsole“. Damit ist das Laden von Gaming-Applikationen bzw. allgemein von Komponenten, in MIDP „Midlets“ genannt, ein Thema, die Kommunikation zwischen diesen Komponenten aber keine notwendige Voraussetzung für die angestrebte Funktionalität.
- Ähnliches gilt für MHP: Bei MHP wird davon ausgegangen das ein breitbandigen Kommunikationskanal (DVB-T) für das Laden von Komponenten, hier mit „Xlets“ bezeichnet, permanent verfügbar ist und diese jederzeit lokalisiert und geladen werden können. Zudem ist MHP bisher primär auf Settop-Boxen mit speziellen Anwendungen zugeschnitten.

Dies bewirkt, dass in beiden Ansätzen die Themen „Lebenszyklus-Management“ und Komponenten-Interaktion wenig ausgeprägt bzw. überhaupt nicht existent sind.

Abbildung 10 zeigt eine schematische Darstellung der Unterschiede. Midlets und Xlets stellen damit nur einen Teilumfang der OSGi – Bundles dar. Die OSGi Standardisierung ermöglicht aber, diese und andere technologische Ansätze (WAP, Jini, UPnP, etc) zu integrieren ([2]).

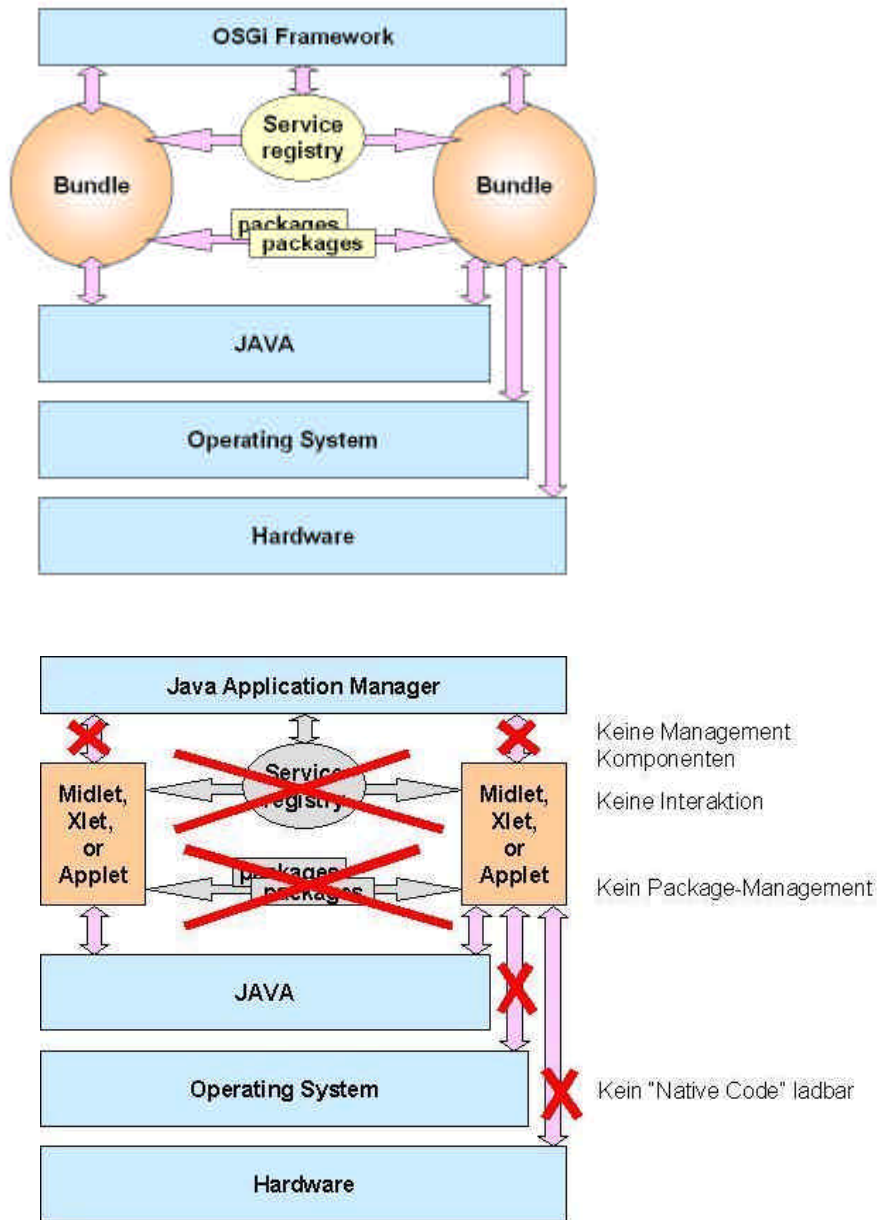


Abbildung 10: Vergleich zwischen OSGi (Bundles), MIDP (Midlets) und MHP (Xlets)
 Figure 10: Comparison between OSGi (Bundles), MIDP (Midlets) und MHP (Xlets)

4 **Fazit**

Die zunehmende Vernetzung von Fahrzeug-Funktionen, die steigenden Software-Anteile und das Zusammenwachsen mit einer externen, durch die Consumer-industrie geprägten Welt erhöhen die Systemkomplexität im Fahrzeug. Um auf die ständig neuen Anforderungen zeitgerecht reagieren zu können muss eine Infotainment-Plattform Architektur

- für den Anwendungsentwickler standardisierte Schnittstellen zu Verfügung stellen und Ressourcensharing auf Applikationsebene ermöglichen, um kurze Entwicklungszeiten realisieren zu können
- einen Mechanismus zur Verfügung stellen, der einfach und schnell zusätzliche Funktionalität und damit neue Applikationen auf der Plattform verfügbar macht und der den Lebenszyklus der Applikationen steuert.

OSGi ist auf den besten Weg diese Anforderungen zu erfüllen und bietet den Fahrzeugherstellern die Möglichkeit, Einfluss auf die Standardisierung zu nehmen und den Standardisierungsprozess voranzutreiben.

Literatur

- [1] MOST-Cooperation: <http://www.mostcooperation.com>
- [2] OSGi-Alliance: <http://www.osgi.org>
- [3] Kirk, Chen; Gong Li: Programming Open Gateways with Java Embedded Server Technology, Addison-Wesley, August 2001
- [4] Szyperski, C., "Component Software - Beyond Object-Oriented Programming ACM Press, New York, 1998.