Dienstplattformen für 3G-Netze

Johannes Martens

Hauptseminar "Dienste & Infrastrukturen mobiler Systeme"
Wintersemester 03/04
Institut für Informatik
Ludwig Maximilians Universität München
martens@informatik.uni-muenchen.de

Zusammenfassung. Diese Arbeit beschäftigt sich mit den verschiedenen Dienstplattformen in den Netzen der 3. Generation, die für den Aufbau eines Dienstes genutzt werden können. Zu Beginn werden die verschiedenen Betriebssysteme für mobile Endgeräte und das darauf aufbauende Mobile Station Application Execution Environment (MExE) vorgestellt. Nachdem die Client-Seite eines Dienstes betrachtet wurde, wendet sich diese Arbeit im zweiten Teil der Anbieterseite zu. Hier werden die Open Network APIs (Parlay/OSA) erläutert und danach das „Virtual Home Environment"-Konzept (VHE) aufgezeigt und anhand des „IST VESPER"-Projekts als Beispiel ausgeführt.

1 Einleitung

Um eine bessere Einordnung der angesprochenen Techniken zu ermöglichen und Verwirrung zu vermeiden, wird folgendes Rollenmodell und eine einheitliche Terminologie verwendet (siehe Abbildung 1):

- Der **Nutzer** besitzt ein oder mehrere verschiedene Endgeräte und greift über diese auf einen Dienst zu.
- Der **Netzbetreiber** ist für die Übertragungstechnik und die Erreichbarkeit des Nutzer verantwortlich. Er stellt die Verbindung zwischen Nutzer und Dienstanbieter her. Er hat meist mehrere Abkommen mit anderen Netzbetreibern. Hier sind in Deutschland die bekannten Mobilfunkanbieter *T-Mobile*, *Vodafone*, *E-Plus* und *O2*, aber auch Internet-Provider wie *AOL*, *T-Online* und Festnetztelefonieanbieter wie *T-Com* und *Arcor* zu nennen
- Der **Dienstanbieter** wiederum sammelt Daten von Inhaltsanbietern, bereitet diese auf und verknüpft sie in einer für den Nutzer sinnvollen Art und Weise miteinander. Bekannter Dienstanbieter in Deutschland sind *Jamba* und *Debitel*.
- Der **Inhaltsanbieter** liefert rohe Daten in Form von Bildern, Texten, Videos, etc. Beispiele hierfür sind *Reuters*, *dpa*, ...

Viele Netzbetreiber fungieren derzeit auch noch als Dienstanbieter, so dass momentan die unterschiedlichen Rollen nicht klar getrennt erscheinen. In Zukunft wird es allerdings viel mehr Dienstanbieter geben, die kein eigenes physisches Netz besitzen oder verwalten.

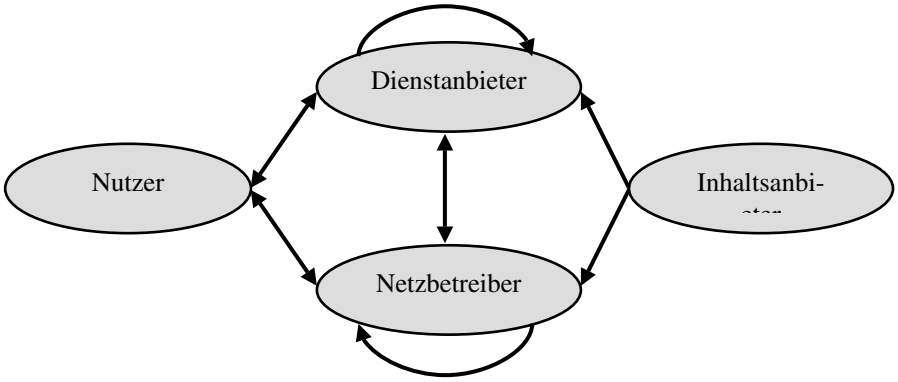

Abbildung 1. Verknüpfung der verschiedenen Rollen im Bereich mobile Dienste

Viele Anwendungen werden heutzutage noch für jedes Betriebssystem und jede Netztechnik einzeln geschrieben. Manchmal kommt auch Java schon zum Einsatz, wodurch teilweise die Codemobilität bezüglich unterschiedlicher OS erreicht wird. Daher betrachten wir in dem zweiten Kapitel die meist verbreiteten Betriebssysteme für mobile Endgeräte. Das dritte Kapitel beschäftigt sich dann mit dem „Mobile Station Application Execution Environment“ (MExE), welches ab Classmark 2 eine Art Middleware darstellt auf der Anwendungen betriebssystemübergreifend ablaufen können. Diese beiden genannten Kapitel beschäftigen sich mit den Techniken auf Nutzerseite. Im vierten Kapitel wird auf die Schnittstelle zwischen Netzbetreiber und Dienstanbieter eingegangen. Generell ist hier der Trend, weg von den separaten Netzen, für die jeder Dienst neu geschrieben und auf die Netzgegebenheiten angepasst werden musste, hinzu konvergenten Netzen zu erkennen. Diese Netze haben den Vorteil, dass die Dienstanbieter in einem sog. Dienst Netzwerk zusammengefasst sind und über eine wohldefinierte Schnittstelle mit dem Endgerät kommunizieren können, ohne die dazwischenliegenden Zugangsnetze zu kennen und einzeln berücksichtigen zu müssen. Techniken für solche Gateways sind Parlay/OSA, welche im vierten Kapitel behandelt werden. Somit kann ein Dienstanbieter direkt mit dem Nutzer kommunizieren ohne ein eigenes Netz besitzen zu müssen. Eine Abstraktionsebene höher setzt das „Virtual Home Environment“-Konzept an: Hierbei muss der Inhaltsanbieter nicht einmal mehr das Endgerät seines Kunden berücksichtigen. Das VHE wählt entsprechend vom Nutzer vorgegebener Policies, den zur Verfügung stehenden Netzen mit ihren garantierten Qualitätsgüte und den Möglichkeiten des Endgeräts die passenden Übertragungswege und -arten aus und transformiert die Information so, dass es am Endgerät angezeigt werden kann.

2 Betriebsysteme für mobile Endgeräte

In diesem Kapitel werden die momentan auf dem Sektor des Mobile Computing existierenden Betriebsysteme vorgestellt und anhand verschiedener für die Dienstentwicklung relevanter Parameter verglichen. Der Umfang dieser Arbeit lässt allerdings nur eine Auswahl der wichtigsten und am weitesten verbreiteten Systeme zu: „Microsoft Windows CE“, „Symbian OS“ und „Palm OS“. Die verschiedenen Linux-Ausprägungen werden in dieser Arbeit nicht weiterbetrachtet, da noch kein System länger als ein Produktzyklus zum Einsatz kam und daher die Zukunft von Linux im mobilen Endgerätebereich momentan noch nicht erkennbar ist.

2.1 Microsoft Windows CE

Mit dem Erscheinen der 2003er Pocket PC-Reihe wurde der offizielle Produktname von *Windows CE*[1] in *Windows Mobile 2003 für Pocket PC* geändert. Windows Mobile ist eine Dachmarke für alle mobilen Geräte auf Basis von Windows CE. Im Folgenden werden jedoch nicht die vollen offiziellen Produkttitel verwendet außer es besteht einen Verwechslungsgefahr mit anderen Produkten. „Microsoft Windows CE“ existiert derzeit in verschiedenen Ausprägungen:

- **Pocket PC:** Pocket PC 2002 basiert noch auf der Version Windows CE 3.0, während der Nachfolger Pocket PC 2003 auf Windows CE 4.2 mit dem .NET-Framework aufbaut. Zielplattform ist der „klassische“ Pocket PC ohne Tastatur und mit Intel StrongARM oder Xscale Prozessor. Eine Erweiterung um Telefonfunktionen stellt die Variante „Pocket PC 2003 Phone Edition“ dar.
- **Handheld PC 2000** basiert auf Windows CE 3.0 und ist für mobile Endgeräte mit einer Hardwaretastatur und einem größeren Display (ab 640 x 240 Pixel) als Pocket PCs ausgelegt.
- **Smartphone** zielt auf den reinen Handymarkt. Die Zielplattformen gleichen heutigen Mobiltelefone in der Größe, den Eingabe- und den Darstellungsmöglichkeiten. Als Basis dient hier Version 4.2 .NET.
- **Windows CE embedded** ist eine stark skalierbare Ausprägung von Windows CE. Einsatzgebiete sind eingebettete Systeme, wie z. B. VoIP-Telefone[2]. Diese Variante wird hier aber nicht weiter betrachtet.

Generell gesagt fügen die verschiedenen genannten Ausprägungen lediglich zusätzliche Anwendungen und Funktionalitäten für spezielle Aufgabengebiete der Basis (Windows CE) hinzu. Im folgenden werden die Eigenschaften von Windows CE 4.2 .NET - so weit bekannt - genauer beleuchtet. Die Geräteauswahl, auf denen Windows CE als Betriebssystem arbeitet, reicht von Pocket PC’s (z.B. Fujitsu-

[1] “CE” ist gegenüber immer wieder auftretender Gerüchte offiziell keine Abkürzung, sondern steht für die folgenden Designziele bei der Entwicklung von Windows CE: “Compact“, "Connectable", „Compatible", "Companion" und "Efficient." [1]

CE als Betriebssystem arbeitet, reicht von Pocket PC's (z.B. Fujitsu-Siemens Loox 610 BT und HP h1930) ohne Telefonfunktionalität bis zu Smartphones (z.B. Motorola MPx200). Abbildung 2 beschreibt den generellen Aufbau von Windows CE .NET 4.2. Diese Architektur ist vollkommen modular aufgebaut und ermöglicht dadurch das Weglassen einiger nicht benötigter Bereiche für besondere Anwendungsszenarien.

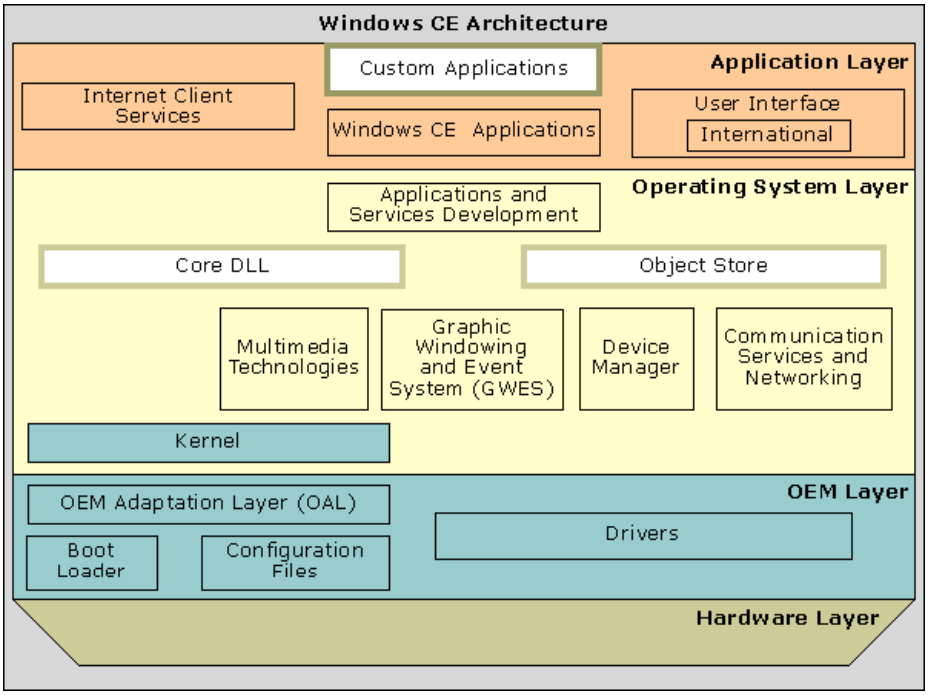

Abbildung 2. Windows CE .NET 4.2 Architektur [22].

Windows CE ist ein 32-Bit-, Multitasking- und Multithreading-Betriebssystem mit integrierter Energieverwaltung. Es beinhaltet Standardsoftware wie einen Browser (Microsoft Internet Explorer), ein E-Mail-Programm (Microsoft Outlook) und verschiedene Viewer (Pocket Versionen von Microsoft Word und Excel). Ebenfalls enthalten ist eine Komponente, die alle Informationen zwischen einem PC und dem mobilem Endgerät synchronisiert (Microsoft Active Sync). Dem Nutzer bietet das Betriebssystem eine ähnliche Umgebung wie an seinem Windows PC, wodurch die Eingewöhnungsphase meist verkürzt wird. Für den Anwendungsentwickler existiert eine große Anzahl von APIs, die meist stark den Windows-APIs für PCs ähneln, wodurch sich erneut Synergieeffekte ergeben. Diese werden durch die Integration des .NET-Frameworks noch verstärkt, da hierbei Anwendungen einmal geschrieben werden und automatisch für verschiedene Betriebssysteme, die alle das Framework implementiert haben (z.B. Pocket PC 2003 und Windows XP), einsetzbar sind.

In der Vergangenheit mussten Windows CE Anwendungen zusätzlich für den jeweiligen Prozessor angepasst bzw. kompiliert werden. Die neueren Versionen der

CE-Geräte unterstützen einen prozessorneutralen Maschinencode namens CEF (Common Executable Format). Ist eine Applikation für eine CEF-kompatible Plattform geschrieben, so wird auf der Plattform der CEF-Code in Maschinencode übersetzt und ausgeführt. Diese Kompatibilität geht auf Kosten der Leistung, daher ist CEF-Code etwa 20% langsamer als prozessorspezifischer Maschinencode. Es gibt für CEF-Code keine Just-in-Time-Compiler (JIT) wie für Java [4]. Leider basieren wie oben aufgelistet nicht alle Versionen auf dem selben Windows CE Kern. Daraus ergibt sich das Problem, dass gewisse Anwendungen, obwohl sie unter Pocket PC 2003 funktionieren, unter Pocket PC 2002 oder Handheld PC 2000 ihren Dienst verweigern und umgekehrt.

2.2 Symbian OS

Symbian wurde im Juni 1998 von den Unternehmen Ericsson, Nokia, Motorola und Psion gegründet. Es ist einen Weiterentwicklung des Betriebssystem EPOC der Firma Psion. Mittlerweile haben die folgenden Handyhersteller Symbian OS lizenziert: Fujitsu, Motorola, Nokia, Panasonic, Samsung, Sanyo, Sendo, Siemens und Sony Ericsson. Diese repräsentieren über 75% des weltweiten Mobiltelefonsektor, weswegen Symbian OS ein großes Wachstumspotential zugesprochen wird [21]. Ende 2003 existieren bereits 11 Mobiltelefone von verschiedenen Herstellern mit Symbian OS als Betriebssystem und weitere sechs sind angekündigt. Unter den bereits verfügbaren Mobiltelefone befinden sich sogar zwei für 3G-Netze (Fujitsu F2051 und F2102V), die bereits jetzt schon in NTT DoCoMo's FOMA (Freedom Of Mobile multimedia Access) 3G-Netzwerk im Einsatz sind.

Im Gegensatz zu Windows CE ist Symbian rein für den Einsatz in Mobiltelefone und Smartphones konzipiert. Dies ist spiegelt sich auch in der Architektur von Symbian OS (siehe Abbildung 3) wieder: Hier ist der Bereich für Telephonie tief in das Betriebsystem integriert. Es existieren mehreren Benutzerschnittstellen, die aber alle auf der Basis von Symbian OS aufbauen:

- **Series 60:** Zielplattform sind hier Mobiltelefone mit numerischen Tastenfeld (z.B. Nokia 7650, 3650, N-Gage und Siemens SX1).
- **Series 80:** Diese Schnittstelle ist speziell für Mobiltelefone mit großer Tastatur und Bildschirm ausgelegt (z.B. Nokia 9200 Communicator).
- **UIQ:** Anwendung findet diese Schnittstelle in Mobiltelefone mit berührungssensitiven Bildschirmen, bei denen eine Eingabe über einen Stift möglich ist (z.B. Sony Ericsson P800 und BenQ P30).

Symbian OS besitzt eine abstrakte API für 2G und 3G Standards, welche den Herstellern eine weltweiten Absatzmarkt in allen Netzen für ihre Produkte ermöglicht. Darüber hinaus bietet es eine umfangreiche Messaging-Umgebung (MMS, EMS, SMS und Email über POP3, IMAP4 und SMTP), Unterstützung der Kommunikation im WAN-Bereich (TCP, IPv4, IPv6 und WAP) und lokalen Bereich (Infrarot, Bluetooth und USB) und Möglichkeiten zur Internationalisierung (Unicode).

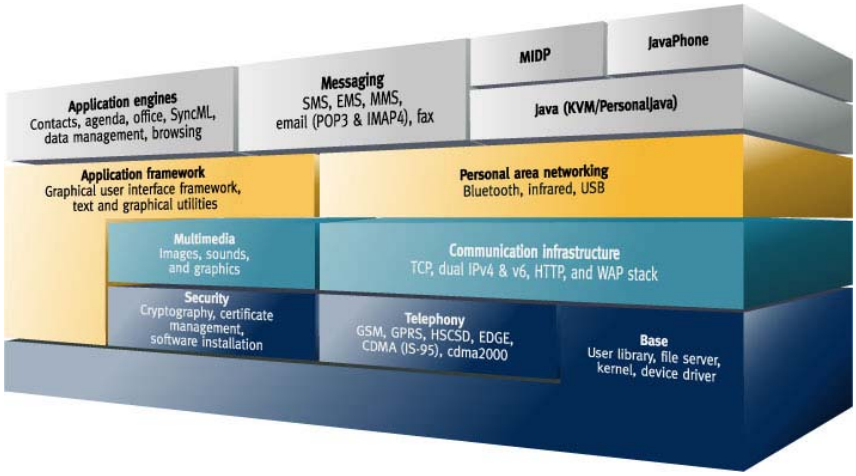

Abbildung 3. Symbian OS v7.0 Architektur. Quelle: Symbian

Der Anwendungsentwickler muss sich bei Symbian OS nicht um die verwendete Hardware kümmern. Das Betriebssystem garantiert die sichere Ausführung seiner Anwendungen. Allein auf die verwendete Benutzerschnittstelle, die manchmal allerdings auch Funktionserweiterungen der API mit sich bringt, muss sich der Entwickler festlegen.

2.3 Palm OS

Palm OS liegt momentan in der Version 5.0 vor. Die größte Neuerung ist sicher der Wechsel der Hardwareplattform von der 68000 Motorola Familie zu den neuen ARM-Prozessoren. Dennoch garantiert die Umgebung PACE (siehe Abbildung 4), dass Software für die Versionen 4.x und 3.x weiterhin auf neuen Geräten lauffähig ist. Mit den neuen Prozessoren kamen auch neue Display-Auflösungen: Neben den alten 160x160 unterstützt Palm OS jetzt auch 320x320. Um alte Anwendungen dennoch sinnvoll auf neuen Geräten anzeigen zu können, wurden neue sog. „High-Density-APIs“ eingeführt, die die Pixelangaben der Anwendungen dem entsprechend umrechnen. Für Echtzeit kritische oder rechenintensive Anwendungen kann man nativen ARM-Code in das Programm einbetten [6].

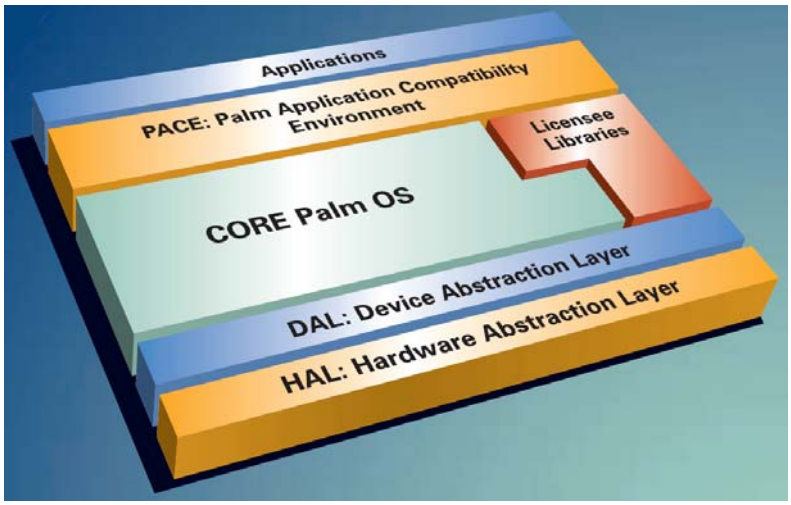

Abbildung 4. Palm OS 5.0 Architektur. Quelle: PalmSource

Neben der graphischen Komponente (Farb-Displays und höhere Auflösungen) wurde auch die Soundausgabe stark überarbeitet, um den wachsenden Multimedia-Anforderungen Rechnung zu tragen. Die Synchronisierung mit dem PC geschieht über sogenannte „HotSync Conduits“. Hierfür stehen umfangreiche Entwicklungskits für Windows und Mac OS zur Verfügung. Ebenfalls neu hinzugekommen ist der „Web Browser 2.0“, welcher weg von den bisherigen proprietären Protokollen nun HTML, Javascript und SSL 3.0 unterstützt. Dieser ist wegen besserer Performance komplett in nativem ARM-Code implementiert und benötigt entgegen der alten Version nicht mehr die Palm.net-Proxies, was die Entwicklung von Web-basierten Anwendungen enorm beschleunigt. Des weiteren bietet Palm OS eine Menge von APIs und Treibern die 802.11b, GSM, CDMA, Bluetooth und 3G Kommunikation auf Systemebene unterstützen [7].

3 Mobile Execution Environment (MExE)

Es gibt derzeit eine Reihe von realisierten Dienstplattformen, um Mehrwertdienste für Mobiltelefone anzubieten. Nachfolgend werden die relevanten Aspekte des durch das 3rd Generation Partnership Project (3GPP) spezifizierten MExE präsentiert [9].

3.1 Allgemeines über MExE

Es ist abzusehen, dass für die durch mobile Endgeräte zu erwartende Informationsvielfalt unterschiedliche und auf die jeweiligen Bedürfnisse angepasste Geräte produziert werden. MExE soll den Spagat schaffen, einen gemeinsamen Standard zu defi-

nieren, damit die gebotenen Informationen und Anwendungen auf unterschiedlichen Endgeräten verwendet werden können.

Der Wunsch nach einem einheitlichen Standard für die Darstellung von Informationen und die Ausführung von Anwendungen auf mobilen Endgeräten ist jedoch unrealistisch. Schon jetzt gibt es zahlreiche Standards wie beispielsweise WAP und HTTP, die auf mobilen Endgeräten verfügbar, jedoch nicht untereinander kompatibel sind. In der MExE–Spezifikation werden drei unterschiedliche Classmarks (Klassifikationen) vorgeschlagen und beschrieben [10].

In jedem Classmarks wird festgelegt, wie die Übermittlung der Fähigkeiten eines Endgerätes stattfindet und wie der Server darauf reagiert. Im MExE-Kontext wird dazu der Begriff „*capability negotiation*“ eingeführt. Es dient dem Zweck, Informationen und Applikationen individuell auf die Bedürfnisse des anfragenden mobilen Endgerätes anzupassen. Dies geschieht entweder dadurch, dass für jedes Classmark eine eigene Version der Information oder Applikation vorliegt, oder diese dynamisch je nach Anfrage und Eigenart des anfragenden Endgerätes aus den Informationen generiert. Für diese „*capability negotiation*“ wird das http–Protokoll (Classmark 2 / 3) bzw. WAP–Protokoll (Classmark 1) verwendet.

Weiterhin erfolgt eine sogenannte ''*content negotiation*'', die sich auf das zu übertragende Format des Inhalts bezieht. So kann serverseitig ein für das Endgerät optimiertes Format gesendet werden. Beispielsweise bekommt ein Classmark 1-Endgerät (WAP) ein WML–Dokument geliefert, während ein Classmark 2-Endgerät ein HTML–Dokument bekommt [8].

3.2 Die MExE Kategorien (Classmarks)

Wie oben skizziert ist das Ziel von MExE, Entwicklern, Dienstanbietern, Netzbetreibern und Herstellern Konventionen und Standards vorzugeben, mit denen sie Applikationen entwickeln und Inhalte auf diversen mobilen Endgeräten darstellen können. Die für die einzelnen Kategorien entwickelten Anwendungen sollen auf allen MExE-Geräten dieser jeweiligen Kategorie zu verwenden sein und die Informationen auf allen MExE-Geräten dieser Kategorie lesbar sein.

3.2.1 Classmark 1 (WAP)

Classmark 1 basiert auf WAP (Wireless Application Protocol). Daher kommt es mit begrenzten Ein- und Ausgaberessourcen wie einer 4 x 18 Buchstaben großen Anzeige und einer numerischen Tastatur aus.

WAP ist als weltweiter Standard eingeführt der es ermöglichen soll, auf der Empfängerseite mit geringen Hardwareanforderungen sowie geringen Bandbreiten auszukommen und dennoch Daten in akzeptabler Form zu präsentieren. So muss beispielsweise WML (Wireless Markup Language) strengeren syntaktischen Anforderungen als beispielsweise HTML genügen, um keine unnötigen Daten transportieren zu müssen. WAP–Anwendungen werden in WML geschrieben. Bevor WML über das drahtlose Netz übertragen und auf das mobile Endgerät geschickt wird, wird WML komprimiert, um die Datenmenge weiter zu reduzieren.

Classmark 1, welches im wesentlichen WAP ist, hat folgende Layer-Charakteristika:

User Input Layer (UIL): WML-Forms um Informationen eingeben zu können. Diese bestehen im Wesentlichen aus Texteingabefeldern, Auswahlfeldern und Buttons zum Bestätigen der Eingaben. Links können nach dem Domainnamen und der aufzurufenden Datei mit Informationen angereichert werden, die übertragen werden. z.B.: http://wap.mvv-muenchen.de/efa.wml?haltestelle=Oettingenstraße

Presentation Layer (PL): `Hier sind die Sprachen` WML und WMLScript zu nennen. Informationen werden bei WAP mittels WML komprimiert übertragen und diese von dem mobilen Endgerät dargestellt. WML gehört zu der SGML-Familie und ist daher HTML sehr ähnlich. WMLScript ist eine Scriptsprache, die eingebettet in WML–Seiten verwendet wird. Sie ist Teil der WAP–Spezifikation. WMLScript ist eine abgespeckte Version von JavaScript. WML-Seiten haben WMLScripts aber nicht direkt eingebettet, sondern verweisen lediglich auf URLs, die die WMLScript enthalten. WMLScript wird in Bytecode übersetzt, bevor es an den WAP–Browser geschickt wird. Mit WMLScript wird, ähnlich wie mit JavaScript ein Teil der Logik auf den Client übertragen und dort ausgeführt. Eine Anwendung hierfür ist beispielsweise die Verifikation von Benutzereingaben.

Business Logic Layer (BLL): Der BLL ist die hardwarespezifische Umsetzung des WAP–Browsers und dessen Interpretation von WML bzw. WMLScript. Hier findet die Umsetzung der Logik statt.

3.2.2 Classmark 2 (PersonalJava)

Classmark 2 basiert auf dem PersonalJava-Environment (PJAE) von Sun. Sie benötigt eine Hardwareumgebung, die wesentlich mehr Ressourcen, wie Rechenleistung, Speicherplatz und Netzwerkbandbreite fordert sowie höhere Anforderungen an die grafische Darstellung stellt, dafür aber mächtigere und flexiblere Anwendungen zulässt. Die Classmark 2 Definition beinhaltet PersonalJava mit dem JavaPhone API. Die PersonalJava Umgebung ist eine Java Umgebung, die für mobile Endgeräte angepasst bzw. optimiert wurde, so dass auf diesen Web-Inhalte und Java Applets darstell- oder ausführbar sind. PersonalJava basiert auf J2SE (Standard Java 2) mit einigen Standard-Java-Klassen und einigen auf die Bedürfnisse mobiler Endgeräte angepassten Java-Klassen, wie beispielsweise eine modifizierte Version des AWT, welche auf kleinen Geräte ohne Mausunterstützung und mit wesentlich kleineren Anzeigen als bei einem PC auskommen muss [14]. Die JavaPhone API ist eine Java-Erweiterung, die insbesondere folgende Funktionen anbietet: Zugriff auf die Telefonfunktionen, die Adressbuchfunktion, SMS–Nachrichten, eine Kalenderfunktion, Benutzerprofile, Batteriestatusüberwachung und das Installieren von Anwendungen [13]. Abbildung 5 zeigt die verschiedenen Schnittstellen auf die eine Classmark 2-Anwendung zurückgreifen kann und deren Abhängigkeiten untereinander. Classmark 2 (sowie auch Classmark 3) benötigen das HTTP-Protokoll, um *`capability`*- und *`content negotiation`* durchzuführen. Auch das Herunterladen von Dateien und Anwendungen erfolgt hierbei über das HTTP-Protokoll.

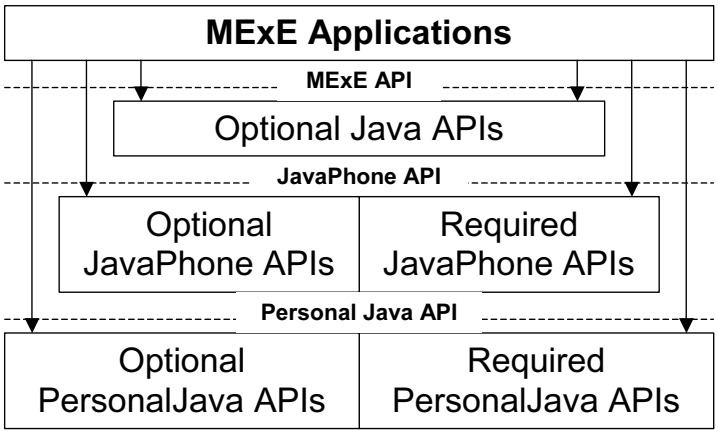

Abbildung 5. MExE-Classmark 2 Architektur. Quelle [10]

User Input Layer: Der UIL besteht aus den Eingabekomponenten des AWT wie Checkbox, Button, TextField und TextArea. Damit können einfache GUIs erstellt werden, die mit Tastatur, einer Mausemulation oder einem Eingabestift bedient werden können. Die Texteingabefelder können mehrzeiligen Text aufnehmen.

Presentation Layer: Der PL besteht aus Komponenten des AWT wie TextField, TextArea, Label und Scrollbar. Diese Elemente unterstützen (wie oben) einfache GUIs auf der Ausgabeebene. Es ist möglich, farbige Grafiken und Tabellen in begrenztem Umfang darzustellen.

Business Logic Layer: Der BLL hat die Mächtigkeit des PJAE und der JavaPhone API.

`PJAE - PersonalJava Environment`

Das PJAE ist eine Java Umgebung, die in Java geschriebene Software ausführen kann. PJAE ist speziell für Software auf netzwerkfähigen mobilen Endgeräten zugeschnitten. PersonalJava basiert auf dem JDK 1.1.6 und wurde um folgende Aspekte erweitert:

- Double Buffering – Optimierung des Updates bei grafischen Operationen, die die Anzeige verändern.
- Eingabemöglichkeiten ohne Maus: dazu gehören Schnittstellen, die beschreiben, wie Benutzer, die keine Maus besitzen, mit dem System interagieren können.
- Timer API – Klassen, die Ereignisse zeitgesteuert auslösen. Die JavaPhone-API stellt alle für ein Mobiltelefon wichtigen Telephonie-Funktionen zur Verfügung. Hierfür gibt es in Java keine Alternative.

3.2.3 Classmark 3 (J2ME)

Classmark 3 - Geräte basieren auf der „*Java 2 Platform, Micro Edition*“ (J2ME). Dabei wird die sog. „*Connected Limited Device Configuration*” (CLDC) mit dem “*Mobile Information Device Profile*“ (MIDP) verwendet. *J2ME* ist eine Version von

Java, die auf mobile Endgeräte angepasst wurde. *CLDC* besteht aus einer JavaVM (in diesem Falle die KVM) und APIs, die dafür bestimmt sind, eine Umgebung zu erstellen, die mit limitierten Hardwareanforderungen, wie etwa geringer Rechenleistung, geringem Speicherplatz und geringer Bandbreite, auskommt [11]. Die Abbildung 6 zeigt die Rückgriffmöglichkeiten einer MExE-Anwendung auf die verschiedenen APIs und deren Abhängigkeiten untereinander. *MIDP* bietet die Kernfunktionalität, einschließlich der Benutzerschnittstelle, Netzanbindung und lokaler Datenhaltung – zusammengefasst als eine Standard Java Laufzeitumgebung mit APIs [12].

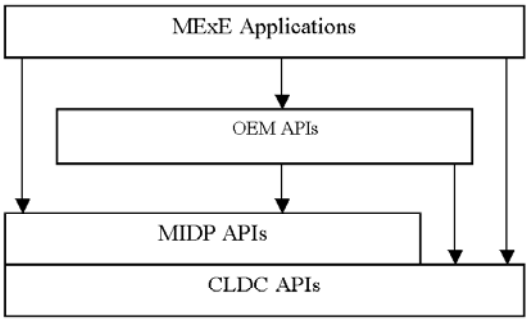

Abbildung 6. Funktionale Architektur eines Classmark 3 MExE-Endgeräts

User Input Layer: Der UIL besteht aus Eingabekomponenten des MIDP wie ChoiceGroup und TextField. Diese sind wenige und eingeschränkte Klassen, die nicht die Mächtigkeit derer von Classmark 2 haben. Mit ihnen lassen sich einzeilige Texte (TextField) eingeben und aus einem Menü auswählen (ChoiceGroup).

Presentation Layer: Der PL besteht auch aus Java-Komponenten des MIDP wie StringItem, Ticker, ImageItem und Gauge. Auch diese sind auf die beschränkten Hardwareressourcen abgestimmt. StringItem kann einen einfachen Text darstellen. Ticker kann einen einfachen Text als Laufschrift darstellen. ImageItem stellt einfache s/w-Grafiken dar und Gauge biete die Darstellung einfacher Balkendiagramme.

Business Logic Layer: Der BLL hat die Mächtigkeit der J2ME Plattform, welche als wesentlichen Bestandteil die KVM enthält. Die KVM ist eine JavaVM, die speziell auf die Bedürfnisse sehr eingeschränkter mobiler Endgeräte zugeschnitten ist. Diese sind geringer Speicherbedarf, geringer Energiebedarf und geringe Rechenleistung. Die Mindestanforderungen an die Hardware für eine MIDP-Umgebung (und damit für Classmark 3) sind folgende:

- Display
  - Bildschirmgröße 96 x 54 Pixel
  - Anzeigenfarbtiefe 1 Bit
  - Pixeldimensionen 1x1
- Tastatur
  - Einhand-Eingabefeld oder
  - QWERTY(Z) - Tastatur oder

    - Touch Screen
- Speicher
    - 128 KB beständiger Speicher für MIDP Komponenten
    - 8 KB beständiger Speicher für Daten, die von den Anwendungen erzeugt und verwendet werden
    - 32 KB Speicher für die Java Laufzeitumgebung
- Netzanbindung

3.3 Unterschiede zwischen den Classmarks

Der wesentliche Unterschied zwischen Classmark 1 und den Classmarks 2 / 3 ist das zugrundeliegende Protokoll. Classmark 1 verwendet WAP, Classmark 2 und 3 verwenden HTTP zur Übertragung der Daten.

Der Hauptunterschied zwischen Classmark 2 und Classmark 3 besteht im Umfang der Java Version. Classmark 2 basiert auf dem JDK 1.1.6 und den JavaPhone-Klassen. CLDC (Classmark 3) benutzt die KVM, also eine wesentlich eingeschränktere JavaVM. Intuitiv anzunehmen wäre, dass die Classmarks eine Hierarchie bilden. Dem ist allerdings **nicht** so. Bei Classmark 2 sind die Hardwareanforderungen wesentlich höher als bei J2ME CDLC (Classmark 3). Classmark 2 benötigt beispielsweise eine Anzeige, die das AWT darstellen kann. Allein diese Bedingung verlangt nach leistungsfähigerer Hardware als die meisten der bisher bekannten Mobiltelefone und Classmark 3-Geräte sie bieten. Eines der Geräte, das Classmark 2 unterstützt, ist beispielsweise der Nokia Communicator 9210.

3.4 Einsatzgebiete der verschiedenen Classmarks

Classmark 1 eignet sich am besten für Mobiltelefone ohne größeres Display. Classmark 1 ist WAP, welches die meisten Hersteller bereits in ihre Geräte integriert haben.

Classmark 2 eignet sich am besten für Smartphones und PDAs mit Kommunikationsanbindung. Durch die höheren Anforderungen an die Hardware und das Display ist es kaum möglich und wenig sinnvoll, sie in einem herkömmlichen Mobiltelefon unterzubringen.

Classmark 3 eignet sich am besten für Mobiltelefone mit begrenzten Ressourcen. Da die Anforderungen an die Hardware gering gehalten sind, ist es gut möglich, kleinste Geräte basierend auf Classmark 3 zu bauen. Im Unterschied zu Classmark 1 ist es bei Classmark 3 möglich, Anwendungen zu entwickeln, die mehr Logik auf dem Telefon ausführen als mit es reinem WAP (Classmark 1) möglich wäre.

4 Parlay/OSA

Ursprünglich waren Parlay, welches von der “Parlay Group“ spezifiziert wurde, und OSA (Open Services Architecture), das von 3GPP ins Leben gerufen wurde, unterschiedliche Spezifikationen für eine offen standardisierte API zwischen Netzbetreibern und Dienstanbietern. Weitere Initiativen auf diesem Gebiet sind von ETSI (European Telecommunications Standards Institute) und JAIN (Java in Advanced Intelligent Networks) von Sun Microsystems, die aber den Rahmen dieser Arbeit sprengen würden. Daher beschränkt sich diese Arbeit auf die Spezifikation, die am meisten verbreitet und in ihrer Entwicklung schon weit fortgeschritten ist.

4.1 Die Architektur von Parlay/OSA

Parlay/OSA ist eigentlich nichts anderes als eine API zwischen Telekommunikationsnetzen und Dienstanbietern, die es Anwendungsentwicklern ohne Kenntnis über die Telekommunikationsinfrastruktur ermöglicht, mobile Dienste zu implementieren. Die Parlay/OSA APIs werden von der Parlay Group [15], einem nicht profit-orientierten Konsortium von mehr als 65 Firmen aus der Telekommunikation- und IT-Branche, definiert. Das „OSA“ wird angehängt, um zu zeigen, dass Parlay nur ein Teil der von 3GPP definierten Open Service Architecture ist, nämlich die API. Außerdem basiert Parlay/OSA auf einer Reihe von offenen Standards, wie z.B. CORBA, IDL, JAVA, UML and Web Services

4.1.1 Parlay/OSA APIs

Parlay/OSA bietet eine umfangreiche Menge von APIs für Kommuniaktionsanwendungen an, die folgenden Gebiete abdecken: Mobilität, Positionsbestimmung, Anrufkontrolle, Messaging und Policy-Management. Die folgenden Leistungsmerkmale werden unterstützt:

- Mobilität (Wie Anwendungen die Position des Endgeräts finden und welche Merkmale das Netzwerk unterstützt.)
- Endgeräteigenschaften
- Kontrolle der Datensitzung (für GPRS und andere 2.5G Anwendungen)
- Präsenz- und Erreichbarkeitsmanagement
- Gebührenabrechnung („Content-based charging“, Pre-Paid-Modelle, …)
- Anrufkontrolle (Wie werden Anrufe initiiert, wie warden sie von der Anwendung in das Netz geleitet und wie werden Multimedia-Anrufe eingeleitet)
- Messaging (E-Mail, Fax, Voice, ...)
- Verbingungsmanager (Quality of Service (QoS) und Konfiguration von „Virtual Private Networks“)
- Policy-Management (Wie Anwendungen mit Policies in Netzen verfahren)

4.1.2 Parlay/OSA Framework

Die Aufgabe des Frameworks ist es, sicherzustellen, dass die Kommunikationsinfrastruktur nicht unautorisierter Nutzung zur Verfügung steht. Dieses Framework ist in dem Parlay/OSA Gateway (siehe Abbildung 7) implementiert. Alle Anwendungen und Dienste, die die Parlay/OSA APIs nutzen möchten, müssen sich vorher beim Framework registrieren. Das Parlay/OSA Framework ist eine Softwarekomponente, die die Anwendungen authentifiziert und eine Objektreferenz zurückgibt. Diese beinhaltet den Zugriff auf die vom Netzbetreiber zugelassenen Funktionen.

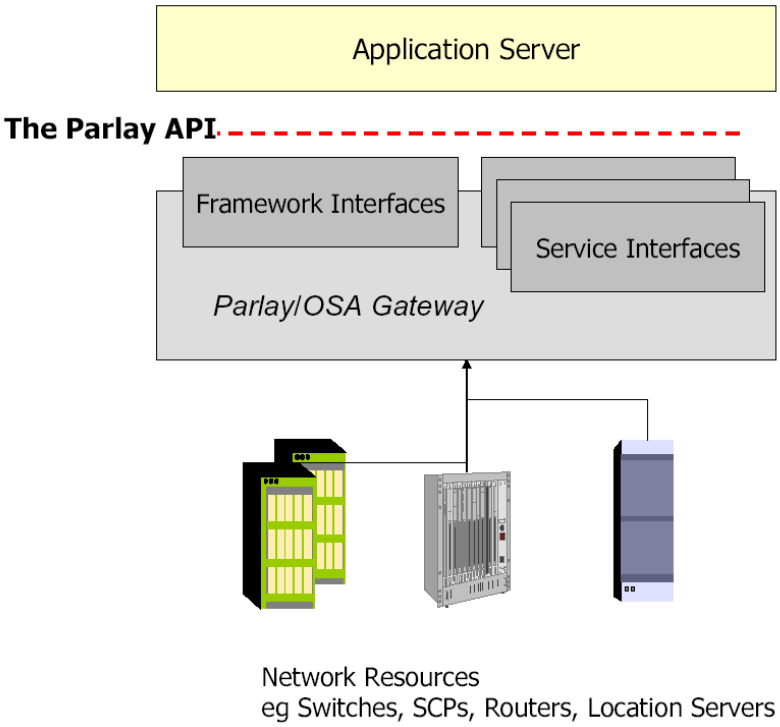

Abbildung 7. Parlay/OSA Gateway. Quelle: [16]

Das Gateway wird dann gemäß Abbildung 8 in das schon bestehende Netzwerk integriert. Normalerweise ist das Gateway unter der Kontrolle des Netzbetreibers. Es wird von der Anwendungsseite über TCP/IP angesprochen und verfügt aber über mehrere Schnittstellen zu den unterschiedlichen Netztypen auf der Übertragungsseite. Dies hat zur Folge, dass alle Anwendungen diesen Punkt passieren müssen. Somit werden auch die Anwendungen von spezifischen Anwendungen der Transportnetzwerke isoliert, was z.B. einen Wechsel von Protokollen auf unterer Ebene ohne Neuprogrammierung der Anwendungen ermöglicht.

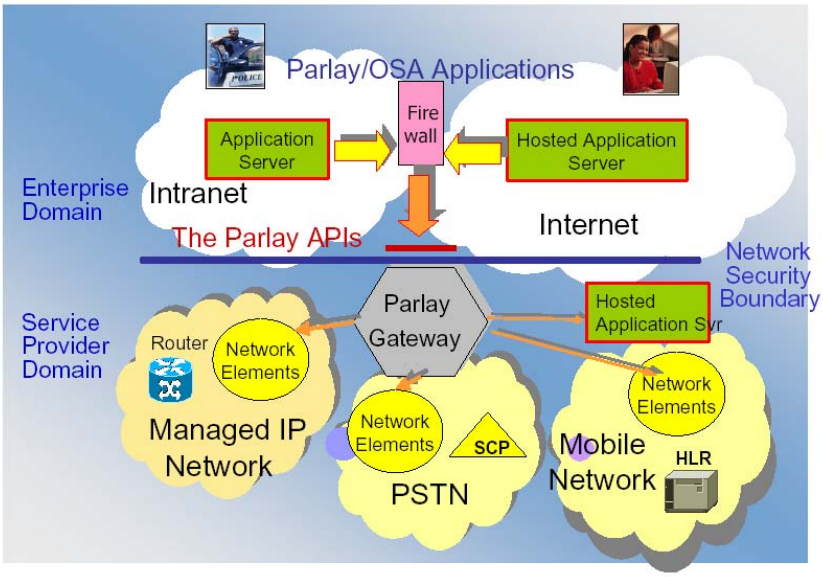

Abbildung 8. Parlay/OSA im Netzwerk. Quelle: [16]

4.2 Vorteile von Parlay/OSA

Es ermöglicht eine schnellere Dienstentwicklung, da sich die Entwickler rein auf den Dienst und die damit verbundene Dienstlogik konzentrieren können und netzwerkspezifische Eigenschaften und Unterschiede nicht mehr berücksichtigen müssen. Diese gewonnene Netzwerkunabhängigkeit ermöglicht es außerdem, die entwickelte Anwendung auf ein anderes Netz zu portieren, ohne dass Anpassungen oder Neuentwicklungen notwendig wären. Dies führt wiederum zu einer Herstellerunabhängigkeit, welche vor allem aus betriebswirtschaftlichen Aspekten sehr vorteilhaft ist. Durch die Möglichkeit, das zugrundeliegende Netzwerk vernachlässigen zu können, steht auch ein viel größerer Anwendungsentwicklerkreis zur Verfügung.

4.3 Entwicklung und Einführung von Parlay/OSA

Mitte 2003 existieren bereits 24 verschiedene Parlay/OSA Gateways auf dem Markt. Dazu kommen 58 Anwendungen, die auf Parlay/OSA aufbauen, mit einer Wachstumsrate von 45% jedes Quartal. Weiterhin sind auch 19 Application-Server speziell für Parlay/OSA auf dem Markt. Insgesamt existieren somit mehr als 150 Anwendungen, die in über 50 verschiedenen Netzen momentan getestet oder schon produktiv zum Einsatz kommen. Abbildung 9 zeigt die Historie und mögliche Entwicklungen von Parlay gemäß der Parlay Group. Momentan arbeitet die Parlay Group an Parlay

Web Services und Parlay-X, einer Spezifikation für einfach zu implementierende Web Services.

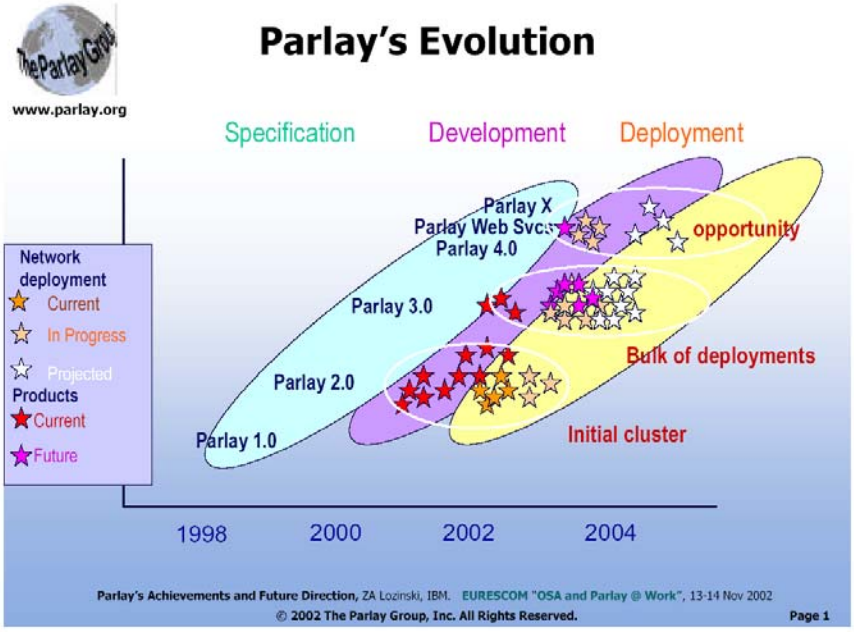

Abbildung 9. Entwicklung von Parlay. Quelle: [16]

5 Virtual Home Environment (VHE)

Das Virtuelle Home Environment stellt ein abstraktes Modell dar, welches von dem 3gpp-Konsortium definiert wurde. Hierbei werden nur die Anforderungen und Leistungsmerkmale genannt. Über welche Technik diese erreicht werden, ist in VHE nicht spezifiziert. Aus diesem Grund wird zunächst das abstrakte Konzept betrachtet und danach eine viel versprechende Implementierung – „Virtual Home Environment for Service Personalization and Roaming Users“(IST VESPER) vorgestellt.

Das abstrakte VHE-Konzept

Das Virtual Home Environment (VHE) ist eines der Schlüsselkonzepte für die Mobilfunknetze der dritten Generation. Das VHE-Konzept ermöglicht den Zugriff auf eine Vielfalt von Diensten, die den individuellen Anforderungen angepasst sind und unabhängig vom Endgerät und verwendeten Netzwerk zur Verfügung stehen. Einschränkungen bei Netzübergängen oder Wechsel des Endgeräts gehören somit der Vergangenheit an. Im Vergleich zum gegenwärtigen GSM-Netz bietet UMTS eine größere

Bandbreite und damit ein völlig neues Anwendungsspektrum. Dazu kommt die Möglichkeit, die Dienste dynamisch an den Aufenthaltsort des Nutzer anzupassen und so ortbezogene Information zu liefern. Durch die Möglichkeit, festzustellen, ob ein Nutzer erreichbar ist, lassen sich Kommunikationsdienste individuell konfigurieren. Das „always on"-Paradigma erlaubt es, permanent mit Dienst-Servern verbunden zu sein und so Informationen auf dem Endgerät stets auf dem aktuellen Stand zu halten. Das VHE-Konzept ermöglicht Nutzern Dienste auf unterschiedlichen Endgeräten und in Netzen verschiedener Anbieter auf die gewohnte Weise verwenden zu können. Die besondere Herausforderung liegt in der Umsetzung des abstrakten Konzeptes VHE in konkrete Lösungsvorschläge und der Behandlung der hohen Heterogenität durch unterschiedliche Netze, Endgeräte, Profile und Gateways [17].

Das Virtual Home Environment ermöglicht den Zugriff auf eine Vielfalt von Diensten. Folgende Informationen können zur Personalisierung und Adaptierung des Dienstes genutzt werden:

- Verwendetes Endgerät (Smartphone, Handheld, ...)
- Netze über die Übertragung erfolgt
- Aktuelle Position des Endgerätes
- Verbindungsgebühren
- Persönliche Profile

Diese Informationen werden über Policies dynamisch verknüpft, um eine optimale und vom Nutzer gewünschte Anpassung an das Endgerät und die Netzumgebung zur Verfügung zu stellen. Den Schlüssel liefern hier die persönlichen Profile, die zusammen mit den anderen vorhandenen Informationen das Verhalten des Endgerätes und die entsprechende Konfiguration auf der Anbieterseite (z.B. angeforderte Bandbreite und QoS) bestimmen.

Um auf Netzdaten wie Positionsinformationen zugreifen zu können, ist eine enge Kooperation zwischen den Netzbetreibern und Dienstanbietern erforderlich. Standardisierte Schnittstellen (Session Initiation Protocol (SIP), Parlay, 3GPP/OSA) werden benutzt, um Anwendungen Zugang zu diesen Informationen zu erlauben. Die Schnittstellen sichern die Investitionen in Dienstentwicklungen, da Anwendungen unabhängig vom jeweiligen Telekommunikationsnetzwerk und Anbieter entwickelt werden können [18].

„IST VESPER" als Implementierung von VHE

Die Abbildung 10 zeigt die Integration von VHE in bestehende Komponenten. Auf Serverseite ist die VHE API von den Diensten ansprechbar und kommuniziert ihrerseits über OSA/Parlay Gateways mit den angeschlossenen Netzen. Auf Endgerätseite bietet das VHE seine API wiederum direkt den Anwendungen an und kommuniziert aber selber über das Universal SIM Application Toolkit (USAT) oder via MExE mit den Kernfunktionen des Endgeräts.

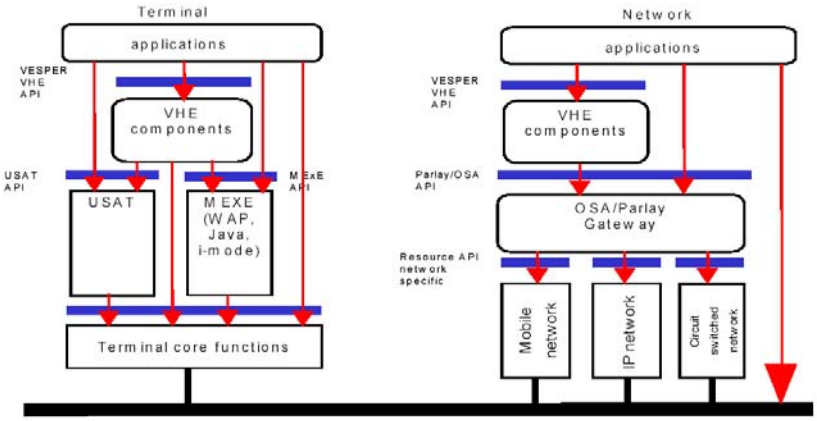

Abbildung 10. Einbettung der VHE-Architektur

Das VESPER-Projekt [20] wurde bereits anhand eines abstrakten „Multimedia delivery service“ (MDS), welcher als Dienst das Streaming von Videodaten anbietet, in Laborumgebung und später in echten Netzwerken getestet. Anhand dieses Beispiels wird im folgenden die weitere Architektur erklärt.

Der Inhaltsanbieter sendet direkt oder über einen Dienstanbieter lediglich seinen Videostream an die VHE-Komponenten. Er nimmt keinerlei Anpassungen oder ähnliches vor. Das VHE-System ist verantwortlich für die Anpassung an Netzwerk- und Endgeräteigenschaften.

Einen Überblick über beteiligten Systeme bei der Dienstadaption stellt Abbildung 11 dar. Die Hauptaufgaben - das Anpassen des Dienstes an von dem verwendeten Netz unterstützten QoS-Klassen, der Server- und Endgeräteeigenschaften und das Management der Benutzerinteraktion - bewältigt die sog. „Adaption-Component“. Alle VESPER-Komponenten nutzen eine CORBA-basierte Umgebung. Auf Endgerätseite wird die Verbindung über die „Connection-Component“ via OSA/Parlay aufgebaut. In diesem Beispiel werden Agenten eingesetzt, um das serverseitige Neukodieren für die entsprechenden Endgeräteigenschaften und das endgerätseitige Dekodieren des Videostreams zu realisieren. Dies fordert zusätzlich eine Agenten-Umgebung auf dem Server des Dienstanbieters und auf dem Endgerät des Kunden.

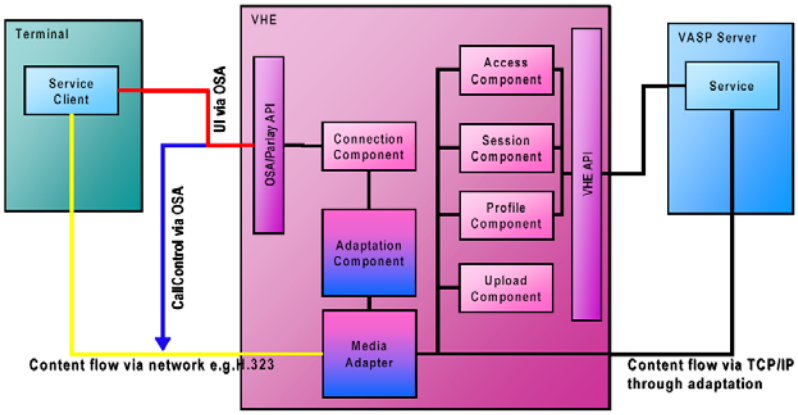

Abbildung 11. Adaption des Medienstroms

Das Management der Nutzerinteraktion mit dem Dienst veranschaulicht ein Beispiel (Abbildung 12). Um dieses Leistungsmerkmal nutzen zu können, muss jeder Dienst eine formale Beschreibung (Ausgabetexte, Eingabefelder, Buttons, ...) in Form eines „User Interface model“ (UIModel) in XML-Form bieten. [19]

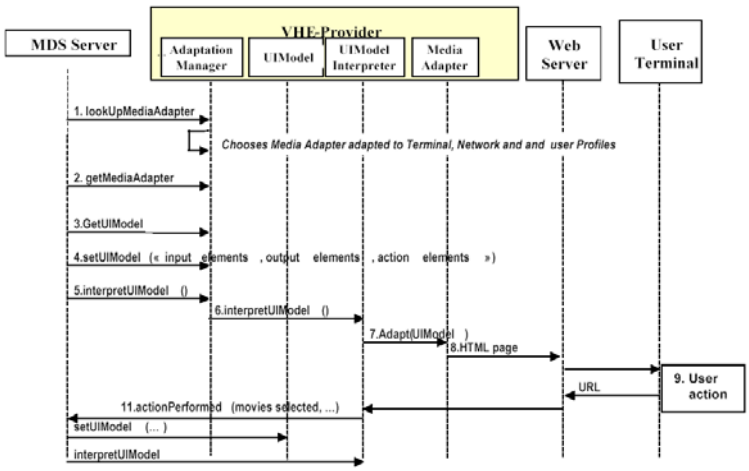

Abbildung 12. MDS Interaktion mit den VHE Adaptationskomponenten

Als Beispiel wird angenommen, der Nutzer hätte den Dienst über einen Web-Browser initiiert:

1. Der Dienst sucht nach einem Medienadapter. VHE-Server bietet dem Dienst eine Liste von verfügbaren Adaptern für das aktuell genutzte Netzwerk und Endgerät.
2. MDS wählt den passendsten Adapter.
3. Dienst fragt nach UIModel-Objekt.

4. MDS sendet Beschreibung der Benutzerschnittstelle an dieses Objekt.
5. MDS fordert die Adaptationskomponente auf das UIModel zu interpretieren.
6. Interpretation des UIModel
7. Aufbereiten einer HTML-Seite gemäß den Endgeräteigenschaften
8. Senden der HTML-Seite an einen Web-Server, der diese wiederum an das Endgerät übermittelt.
9. Benutzer interagiert mit der angezeigten HTML-Seite (z.B. Auswahl eines Buttons).
10. Web-Server leitet Anfrage an UIModel-Interpreten weiter.
11. UIModel-Interpret sammelt alle Angaben, bereitet diese auf und ruf die Funktion des MDS zum Behandeln von Nutzeraktionen auf.

6 Zusammenfassung und Ausblick

Bei den verschiedenen Betriebssystemen zeichnet sich momentan kein klarer Favorit ab. Microsoft versucht gegenwärtig seinen Rückstand aufgrund des verspäteten Markteintritts aufzuholen, was im Handheld-Bereich teilweise gelingt, aber im Smartphone-Markt bisher noch nicht von Erfolg gekrönt war. Palm war lange Zeit der Marktführer und hat dies aber aus heutiger Sicht aufgrund fehlender Multimediaeigenschaften der Endgeräte im Consumer-Sektor verspielt. Hier wird die Zukunft zeigen, ob die hohen Anteile von Palm OS in Firmen dies wettmachen. Symbian OS hat zwei große Vorteile: Mehr als 70% aller verfügbaren Mobiltelefone stammen von Symbian OS-Lizenznehmern und das Betriebssystem ist rein für Smartphones und Mobiltelefone ausgelegt. Dennoch ist fraglich, wie weit sich Symbian durchsetzen wird, da viele Lizenznehmer ihre Eigenentwicklungen noch weiterpflegen und sich auch in anderen Betriebssystemlagern (Linux oder Windows CE) umsehen. Aus diesen Gegebenheiten lässt sich der Bedarf an einer einheitlichen Middleware ableiten. Bisher ist hier JAVA stark favorisiert worden. Doch Microsoft versuch mit seiner .NET Initiative auch hier Fuß zu fassen und Java als Standard abzulösen. Dadurch ergeben sich schon jetzt Probleme für MExE, da Microsoft Java nicht mehr direkt unterstützt, sondern nur noch das eigene .NET. Die Classmarks von MExE müssen in naher Zukunft neu überarbeitet werden, da sie bald nicht mehr die verbreiteten Endgeräte in Bezug aud Anzeige und Rechenleistung abdecken.

Viele der technischen Fragen auf dem Bereich des Virtual Home Environment sind bereits gelöst worden, so dass dem Einsatz von technischer Seite aus nichts mehr im Weg stehen würde, aber noch keine Mehrwertdienste existieren, die diese Möglichkeiten ausschöpfen würden. Ein weiteres Problem besteht allerdings auf betriebswirtschaftlicher Seite: Es werden ganz neue Abrechnungsmodelle und Rahmenverträge zwischen Kunden und Netzbetreiber, Dienstanbieter und Inhaltsanbieter benötigt. VHE bietet aber dem Dienstanbieter kurze Entwicklungszeiten und vereinfachte Implementierung, die Beschränkung auf das Kerngeschäft und eine Herstellerunabhängigkeit durch offene Schnittstellen und Verwendung von Standards - und damit ver-

bundene Sicherung von Investitionen. Dadurch lassen sich neue Dienste besonders auch von kleineren Anbietern ohne ein bestehendes eigenes Netz einer großen Nutzeranzahl offerieren.

7 Referenzen

1. “The Meaning of "CE" in Windows CE”, Microsoft Knowledge Base, http://support.microsoft.com/default.aspx?scid=kb;EN-US;Q166915
2. „Führende Hersteller bringen Windows CE .NET-basierte VoIP-Geräte auf den Markt“, http://www.microsoft.com/germany/ms/presseservice/meldungen.asp?ID=530597
3. „Comparison of Windows CE .NET 4.2, Pocket PC 2002, and Windows Mobile 2003 Software for Pocket PCs”, http://www.microsoft.com/downloads/details.aspx?FamilyID=111fe6d5-b0e1-4887-8070-be828e50faa9&DisplayLang=en
4. Windows CE 3.0 Application Programming, Nick Gratten, Marshall Brain, Prentice Hall PTR, ISBN: 0130255920
5. Symbian OS Homepage, http://www.symbian.org
6. “Palm OS 5 Development Overview”, http://www.palmos.com/dev/support/docs/palmos50/
7. “Palm OS 5”, http://www.palmsource.com/de/enterprise/Palm_OS5_Overview.pdf
8. Lars Schrix, „Toolkit für die dritte Generation“, NET 10/01, http://www.net-im-web.de/pdf/2001_10_s64.pdf
9. „Mobile Execution Environment (MExE) service description; Stage 1“, 3G 22.057 v5.4.0
10. "Mobile Station Application Execution Environment (MExE);Functional description; Stage 2", version 5.0.0, 3GPP TS 23.057: 2002-06.
11. “Connected Limited Device Configuration (JSR-36)”, http://java.sun.com/products/cldc/
12. “J2ME Mobile Information Device Profile (MIDP)”, http://java.sun.com/products/midp/
13. "JavaPhone API", http://java.sun.com/products/javaphone/
14. "PersonalJava Application Environment", http://java.sun.com/products/personaljava/
15. Parlay Homepage, http://www.parlay.org
16. Zygmunt Lozinski, “Parlay/OSA – a new way to create wireless Services”, submitted to “IEC Mobile Wireless Data”, 15 May 2003, revised 1 June 2003, http://www.parlay.org/specs/library/IEC_Wireless_A_New_Way_to_Create_Wireless_Services.pdf
17. "Virtual Home Environment", TS 23.127, 3GPP project
18. "The Virtual Home Environment", TS 22.121, 3GPP project
19. “Virtual Home Environment for multimedia services in 3rd generation networks”, http://mdslab.iit.unict.it/~orazio/Papers/networking2002.pdf
20. IST VESPER Homepage, http://www.ee.surrey.ac.uk/CCSR/IST/Vesper/
21. „Personal Digital Assistant and Mobile Phone Predictions, 2003“, Dataquest, November 2002
22. „About Windows CE .NET” Microsoft Developer Network, http://msdn.microsoft.com/library/en-us/wcemain4/html/cmconAboutWindowsCE.asp