Entwicklung eines Konzepts für die Integration von MHP-Inhalten in eine linuxbasierte TV-Plattform

(Im Rahmen des Digitalen Video Projekts)

Diplomarbeit

zur Erlangung des akademischen Grades

Diplom-Informatiker

vorgelegt der Fakultät für Informatik und Automatisierung

der Technischen Universität Ilmenau

von

Andreas Regel

Ilmenau, 1. Dezember 2003

Fachgebiet Prozessinformatik

Betreuer:	Dipl.-Inf. Detlef Streitferdt
Verantwortlicher Hochschullehrer:	Prof. Dr.-Ing. Ilka Philippow
Inventarierungsnummer:	2003-12-01/078/IN98/2232

Zusammenfassung

Mit der Einführung des digitalen Fernsehens wurde die Basis für interaktive Mehrwertdienste geschaffen, die den Nutzer intensiver in das Fernsehgeschehen einbinden sollen. Mit der Multimedia Home Platform (MHP) liegt seit einigen Jahren ein internationaler Standard für diese Art von Diensten vor, der ständig weiterentwickelt wird und im Bereich des interaktiven Fernsehens wohl die Plattform der Zukunft darstellt.

Diese Diplomarbeit beschäftigt sich im Rahmen des Digitalen Video Projekts, welches im Fachgebiet Prozessinformatik an der Technischen Universität Ilmenau durchgeführt wird, mit der Integration der Multimedia Home Platform in eine bestehende, linuxbasierte Plattform für das digitale Fernsehen, die von Hause aus keine Unterstützung für die MHP beinhaltet. Im Verlauf der Arbeit wird eine Architektur in Form einer Systemfamilie entwickelt, die ausgehend von Kundenanforderungen die Generierung verschiedener Systeme erlaubt. Damit werden der Empfang, die Ausführung, die Verwaltung, die Steuerung und die Darstellung von MHP-Applikationen in verschiedenen Hardware- und Softwareumgebungen ermöglicht. Als Abschluss der Arbeit wird ein Prototyp implementiert, der die Realisierbarkeit der entwickelten Architektur demonstriert.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

Die Einführung des digitalen Fernsehens stellt einen weiteren Meilenstein in der Entwicklung eines Mediums dar, das seinen Siegeszug in den 30er Jahren des 20. Jahrhunderts begann. Mit der Digitalisierung des Fernsehsignals wird allerdings nicht nur dessen Bild- und Tonqualität erhöht oder eine bis dahin nie da gewesene Programmvielfalt ermöglicht. Die Art und Weise der Übertragung erlaubt es zudem, über die reinen Bild- und Toninformationen hinaus, beliebige Daten in das Fernsehsignal einzubetten. Dies ermöglicht die Integration einer ganz neuen Art von Zusatzangeboten, den so genannten Mehrwertdiensten, die dem Fernsehen zu einer nie da gewesenen Attraktivität verhelfen sollen.

Die Palette der Mehrwertdienste reicht von einfachen, textbasierten Angeboten bis hin zum „Interaktiven Fernsehen“, einem Begriff, der zeitgleich mit der Digitalisierung des Fernsehens geprägt wurde. Dieser umfasst multimediale Anwendungen, die den Zuschauer aktiv am Fernsehgeschehen beteiligen, indem sie unter anderem Zusatzinformationen auf Knopfdruck und Mitspielmöglichkeiten bei bestimmten Sendungen bieten. Da der Standard, nach dem das digitale Fernsehen übertragen wird, keinerlei Angaben über solche Dienste enthielt, mussten zuerst Plattformen für das interaktive Fernsehen entwickelt werden. Nach ersten proprietären und daher untereinander inkompatiblen Lösungen verschiedener Unternehmen wurde schließlich im Jahr 2000 ein europäischer Standard vorgestellt, der die Wünsche aller Interessensgruppen in sich vereint. Dieser Standard, der als Multimedia Home Platform (MHP) bezeichnet wird, bietet von allen bisherigen Plattformen für interaktive Mehrwertdienste die umfassendsten Möglichkeiten und wird in Zukunft wohl nicht nur in Europa die anderen Lösungen ersetzen.

Das Digitale Video Projekt, in dessen Rahmen diese Diplomarbeit angesiedelt ist, beschäftigt sich mit der Forschung zu Methoden der Systemfamilienentwicklung, in deren Umgebung ein System entwickelt wird, das eine neue und innovative Nutzung des Mediums Fernsehen erlauben soll. Dies umfasst Funktionalitäten, die weit über das „Hören“ und „Sehen“ hinaus gehen, wie die Integration eines vollwertigen digitalen Videorekorders und die Nutzung von Titeldatenbanken oder eines

elektronischen Programmführers. Zu einem solchen System, das alle Bereiche des digitalen Fernsehens umfassen soll, gehört natürlich auch die Unterstützung der zukünftigen Dienste des interaktiven Fernsehens und damit der Multimedia Home Platform.

Zum Zeitpunkt dieser Arbeit existiert keine MHP-Implementierung, die sich nahtlos in das System des Digitalen Video Projekts einbetten lässt. Um dies zu erreichen müssten an den vorhandenen Lösungen Änderungen auf Quelltextebene durchgeführt werden, was mit hohen Kosten verbunden wäre, da keine dieser Implementierungen frei verfügbar ist. Deshalb ist eine Lösung dieser Problematik vorzuziehen, die ohne solche Modifikationen auskommt.

Ziel dieser Diplomarbeit ist es deshalb ein Konzept zu entwickeln, das die Integration der MHP in die Architektur des Digitalen Video Projekts erlaubt, wobei auf vorhandene Implementationen der MHP zurückgegriffen werden kann. Für das zu entwickelnde System soll eine modulare und flexible Architektur in Form einer Systemfamilie entworfen werden. Dies ermöglicht die Konfigurierung eines kundenspezifischen Produkts auf Basis von ausgewählten Merkmalen und Parametern, was eine einfache Anpassung an die Fähigkeiten und Beschränkungen verschiedener Hardware- und Softwareumgebungen erlaubt. Zum Abschluss dieser Arbeit soll ein Prototyp implementiert werden, der die Realisierbarkeit der entwickelten Architektur demonstriert.

2. Grundlagen

In diesem Kapitel werden technische Grundlagen abgehandelt, die zum weiteren Verständnis notwendig sind. Außerdem wird kurz auf das Projekt eingegangen, in dessen Rahmen diese Arbeit angesiedelt ist.

2.1. Digitales Fernsehen nach dem DVB-Standard

Im Jahr 1991 wurde eine Initiative gestartet, die sich zum Ziel gesetzt hatte, die Entwicklung des digitalen Fernsehens in Europa voranzutreiben. Es sollte ein einheitliches Format für die unterschiedlichen Übertragungsmedien definiert werden. Daraus ging 1993 das DVB-Projekt hervor. DVB steht für Digital Video Broadcasting und definiert den Standard, nach dem das digitale Fernsehen gesendet wird. Nacheinander wurden 3 Varianten des Standards entwickelt und spezifiziert: DVB-S für die Satelliten-, DVB-C für Breitbandkabel- und DVB-T für die terrestrische Übertragung.

Der DVB-Standard hat sich in den letzten Jahren in Europa und zunehmend auch weltweit für die digitale Fernsehübertragung durchgesetzt. So wurde im Jahr 1996 die europaweite Ausstrahlung digitaler Fernsehsignale über Satellit gestartet. Inzwischen können auf diesem Übertragungsweg schon mehr digitale Programme empfangen werden als analoge. Bei der Übertragung via Breitbandkabel oder terrestrisch geht die Entwicklung in Deutschland etwas langsamer voran. Seit 2001 werden hier digitale Programme ins Kabelnetz eingespeist. Der terrestrische Empfang ist seit August 2003 im Ballungsraum Berlin möglich, die analoge Ausstrahlung wurde im gleichen Zeitraum komplett eingestellt. Bis zum Jahr 2010 soll laut Initiative Digitaler Rundfunk der analoge Sendebetrieb in Deutschland komplett eingestellt und damit die Umstellung auf das digitale Fernsehen vollzogen sein.

Die Einführung des Digitalfernsehens bringt eine Reihe von Vorteilen mit sich. Dazu gehört eine gegenüber der analogen Übertragung gesteigerte Bild- und Tonqualität. Das digitale Signal ist weit weniger anfällig gegen Störungen, die beim analogen Fernsehsignal sofort durch Rauschen und Strei-

fen im Bild sichtbar werden. Ein weiterer Vorteil ist die Steigerung der Anzahl der TV-Programme bei gleicher Zahl von Sendefrequenzen. Dies wird durch eine Datenreduktion nach dem MPEG-2 Verfahren möglich, welches die benötigte Bandbreite eines Fernsehsignals ohne sicht- und hörbare Verschlechterung der Qualität stark verringert. Damit ist es möglich auf einem herkömmlichen Fernsehkanal bis zu 10 digitale Fernsehprogramme unterzubringen[1].

Das digitale Fernsehsignal ist aber nicht nur auf Bild- und Tonsignale beschränkt. Aufgrund der Wahl des MPEG-2 Verfahrens können zusätzlich beliebige Daten eingebettet im DVB-Signal übertragen werden. Damit können vielfältige digitale Mehrwertdienste realisiert werden, die multimediale Inhalte und Interaktivität bis hin zu einem vollwertigen Internetzugang bieten.

2.2. Aufbau des DVB-Datenstroms

Das DVB-Projekt spezifizierte die Nutzung des MPEG-2 Verfahrens für die Kodierung der Audio- und Videosignale und die Erzeugung sowie Zusammensetzung der verschiedenen Datenströme des Fernsehsignals. Da praktische Anforderungen und die Möglichkeit ökonomischer Implementierungen im Vordergrund standen, wurde aufgrund seiner großen Vielfalt nicht der komplette MPEG-2 Standard verwendet. Es mussten Syntax und mögliche Parameter eingeschränkt werden, welche in den so genannten Implementation Guidelines [4] festgehalten wurden.

Die kodierten Video-, Audio- und Zusatzdaten werden als Elementary Streams (ES) bezeichnet. Ein Elementarstrom wird in Pakete unterschiedlicher Länge zerlegt. Diese Pakete starten mit einem Header, der unter anderem Zeitmarken zur Dekodierung und zur Wiedergabe enthalten kann. Die Folge der Elementarstrom-Pakete bildet einen Packetized Elementary Stream (PES). Vor der Übertragung des DVB-Signals werden die Pakete der Elementarströme mehrerer Fernsehprogramme mittels eines Multiplexers zusammengeführt. Diese bilden einen Transport Stream (TS), der nach dem Hinzufügen von Informationen zur Fehlerkorrektur und der Modulation auf einem Übertragungskanal gesendet wird.

2.2.1. Transportstrom

Der Transportstrom setzt sich aus den paketierten Elementarströmen der Audio-, Video- und Zusatzdaten sowie den programmspezifischen Informationen zusammen. Dabei ist der Transportstrom nicht

[1] Die Übertragungskapazität eines Fernsehkanals beträgt bis zu 40 Mbit/s. Bei MPEG-2 Komprimierung wird eine gute Qualität bei Fernsehauflösung (PAL: 720 x 576, NTSC: 720 x 480) bei etwa 4 bis 5 Mbit/s erreicht.

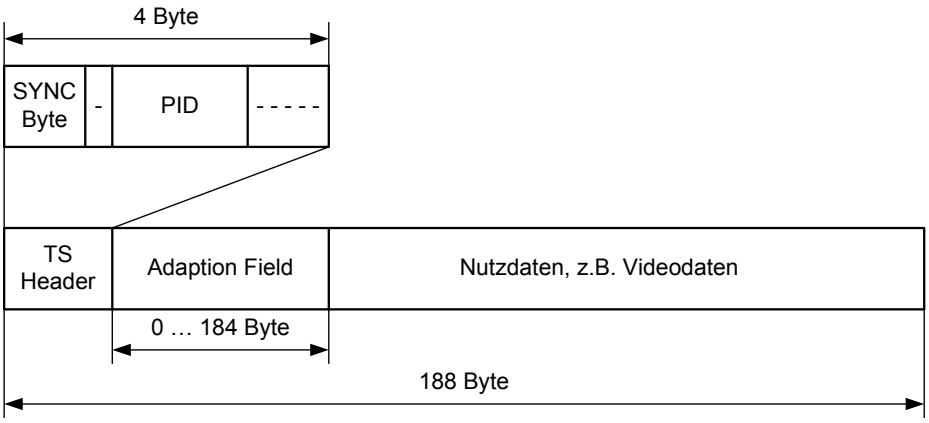


Abbildung 2.1.: Ein TS-Paket mit Header und optionalem Adaption Field (aus [15])

auf die Daten eines einzelnen Fernsehprogramms beschränkt. Aufgrund der Bandbreite von bis zu 40 Mbit/s können bis zu 10 Programme in einem einzigen Transportstrom untergebracht werden.

Wegen möglicher streckentypischer Übertragungsfehler weist der Transportstrom eine robuste Struktur in Form von kurzen Paketen einer festen Länge von 188 Byte auf. Jedes Paket des Transportstroms ist auf die Daten einer Komponente (Video, Audio oder Zusatz) beschränkt.

Die Pakete des Transportstroms beginnen jeweils mit einem Header von 4 Byte, womit 184 Byte für Nutzdaten verbleiben (Abbildung 2.1). Ein Teil der Nutzdaten kann einem Adaption Field zugewiesen sein, das unter anderem die Systemzeit in Form der Program Clock Reference (PCR) enthält. Der TS-Header beginnt mit einem Synchronisationsbyte, welches immer den Wert 47hex besitzt. Neben Bits zur Fehlererkennung und Prioritätenvergabe beinhaltet der Header die 13 bit große Packet Identification (PID). Mit dieser lassen sich die Daten eines TS-Pakets den verschiedenen Elementarströmen des Fernsehsignals zuordnen. Weiterhin wird das Vorhandensein des Adaption Fields durch entsprechende Bits im TS-Header angezeigt.

Die Zuordnung der Daten aus einem PES-Paket auf die 188 Byte großen Pakete des Transportstroms zeigt Abbildung 2.2. Der MPEG-2 Standard sieht vor, dass der PES-Header gleich nach dem TS-Header folgt, es sei denn das Transportstrom-Paket enthält ein Adaption Field. In dem Fall wird der PES-Header direkt nach diesem übertragen. Der Beginn eines PES-Pakets wird im TS-Header durch Setzen des Payload Unit Start Bits angezeigt. Die PES-Paketdaten werden fortlaufend auf weitere TS-Pakete aufgeteilt. Sollten Nutzdaten des PES-Pakets nicht mit dem Ende des TS-Pakets abschließen, wird der Payload des Transportstrom-Pakets mit der entsprechenden Menge an Stopfbytes aufgefüllt.

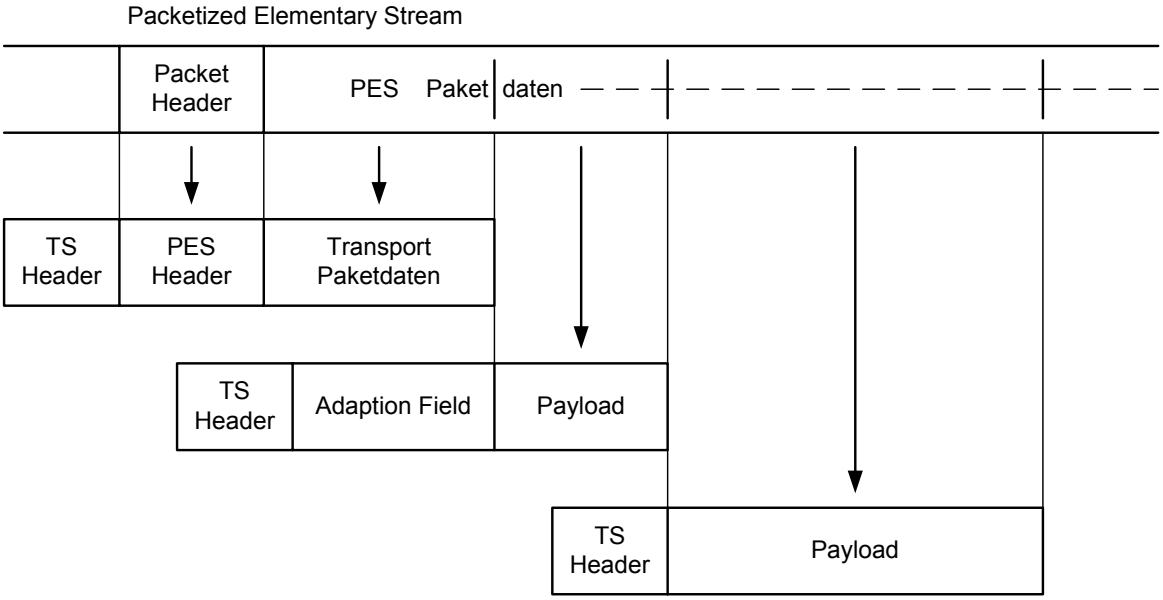


Abbildung 2.2.: Zuordnung der Daten aus einem PES-Paket auf die Transportstrom-Pakete (aus [15])

2.2.2. Programmspezifische Informationen

Ergänzend zu den Elementarströmen der Audio-, Video- und Zusatzdaten werden im Transportstrom Angaben zu dessen Aufbau und programmbegleitende Informationen übertragen. Dies geschieht in der nach MPEG-2 definierten Program Specific Information (PSI). Darin wird angegeben, aus welchen und wie vielen Elementarströmen sich ein Programm zusammen setzt und unter welchen PIDs deren einzelne Pakete im Transportstrom zu finden sind. Die im TS-Header enthaltene, 13 Bit große PID kann bis zu 8192 verschiedene Werte annehmen. Jedem Teilstrom in einem Programm, z. B. einem Audio- oder Videostrom, ist genau eine PID zugeordnet.

Im MPEG-2 Standard ist spezifiziert, dass die programmspezifischen Informationen in Form von vier Tabellen übertragen werden. Jede Tabelle setzt sich aus einzelnen Segmenten (Sections) zusammen, wobei dies je nach Art der Tabelle bis zu 256 Sections sein können. Die einzelnen Segmente einer Tabelle können eine maximale Größe von 4096 Byte besitzen. Die Zuordnung der Sections auf die Transportstrom-Pakete erfolgt ähnlich der der PES-Pakete mit dem Unterschied, dass das Ende des einen und der Beginn des folgenden Segments in ein und demselben TS-Paket enthalten sein können. In diesem Fall schließt sich dem TS-Header das Pointerfield, welches einen Zeiger darstellt, an. Dieser gibt an, wo im Payload die nächste Section beginnt.

Die PSI-Tabellen werden zyklisch mit teilweise festgelegten oder variablen PIDs übertragen (siehe Tabelle 2.1). Die Wiederholrate der Tabellen ist nicht im Standard festgelegt. Sie muss jedoch hoch

Tabelle	PID
Program Association Table (PAT)	0x0000
Program Map Table (PMT)	variabel
Conditional Access Table (CAT)	0x0001
Network Information Table (NIT)	variabel (MPEG-2)/0x0010 (DVB)
Bouquet Association Table (BAT)	0x0011
Service Description Table (SDT)	0x0011
Event Information Table (EIT)	0x0012
Running Status Table (RST)	0x0013
Time and Date Table (TDT)	0x0014
Time Offset Table (TOT)	0x0014
Stuffing Table (ST)	0x0010 bis 0x0014

Tabelle 2.1.: PSI-Tabellen und ihre PIDs

genug sein (etwa 10 bis 50 mal pro Sekunde), damit sich der Dekoder schnell auf das gewünschte Programm einstellen kann.

Die im MPEG-2 Standard definierten Tabellen der Program Specific Information werden im Folgenden näher beschrieben.

Die *Programm Association Table (PAT)* enthält eine Liste der im Transportstrom enthaltenen Programme mit Angabe der Programmnummer und der jeweiligen PID für die zugehörige Program Map Table.

Für jedes Programm ist eine *Programm Map Table (PMT)* im DVB-Datenstrom vorhanden. Diese beinhaltet eine Liste aller PIDs der Elementarströme, die dem Programm angehören. Dies umfasst sowohl Audio- und Videoströme als auch Datenströme, über die Teletext oder andere Mehrwertdienste übertragen werden. Die einzelnen Elementarströme werden durch die in den Einträgen der PMT enthaltenen Deskriptoren näher beschrieben.

Sobald wenigstens ein Programm im Transportstrom verschlüsselt ist und damit nur bedingten Zugriff erlaubt, wird die *Conditional Access Table (CAT)* übertragen. Sie stellt Informationen bereit, die für die Entschlüsselung der verwürfelten Datensätze benötigt werden. Dazu gehört unter anderem die Kennzeichnung des verwendeten Verschlüsselungssystems.

Die *Network Information Table (NIT)* enthält Informationen über den Übertragungskanal wie Fre-

quenzbänder, Orbitalposition des Satelliten und Transpondernummern.

Der detaillierte Aufbau der CAT und der NIT ist nicht durch den MPEG-2 Standard spezifiziert. Dies geschieht im DVB-Standard.

Über die programmspezifischen Informationen hinaus führte der DVB-Standard im Rahmen der Service Information (SI) weitere Tabellen ein. Diese enthalten Informationen, die mehr für den Zuschauer gedacht sind, z. B. für eine elektronische Programmzeitschrift oder zur Videorekordersteuerung. Die folgenden Tabellen wurden im DVB-Standard spezifiziert:

- Die *Bouquet Association Table (BAT)* enthält Informationen über die Programme eines Anbieters.
- Die *Service Description Table (SDT)* führt die Namen und Parameter der Programme eines Transportstroms ein.
- Die *Event Information Table (EIT)* stellt Informationen für eine elektronische Programmzeitschrift mit Titel, Anfangszeit, Dauer, Inhaltsangabe und weitere Angaben zu den Sendungen eines Programms bereit.
- In der *Running Status Table (RST)* kann z. B. mitgeteilt werden, ob eine bestimmte Sendung gerade läuft oder sich eventuell verzögert.
- Aus der *Time and Date Table (TDT)* und der *Time Offset Table (TOT)* können das Datum und die aktuelle Uhrzeit entnommen werden.
- Die *Stuffing Table (ST)* besitzt keinen relevanten Inhalt und wird nur für das Überschreiben nicht mehr gültiger Tabellen verwendet.

Für weiter ins Detail gehende Informationen über die PSI- und SI-Tabellen sei auf die Diplomarbeit von Sven Schäpe verwiesen [17].

2.3. Digitales Video Projekt

Die vorliegende Arbeit ist eingebettet in das Digitale Video Projekt (DVP) des Fachgebiets Prozessinformatik an der Technischen Universität Ilmenau.

Ziel des Digitalen Video Projekts ist die Entwicklung eines Systems, das eine neue und innovative Nutzung des Mediums „Fernsehen“ erlaubt. Ein Grundgedanke hierbei war das ausschließliche Ver-

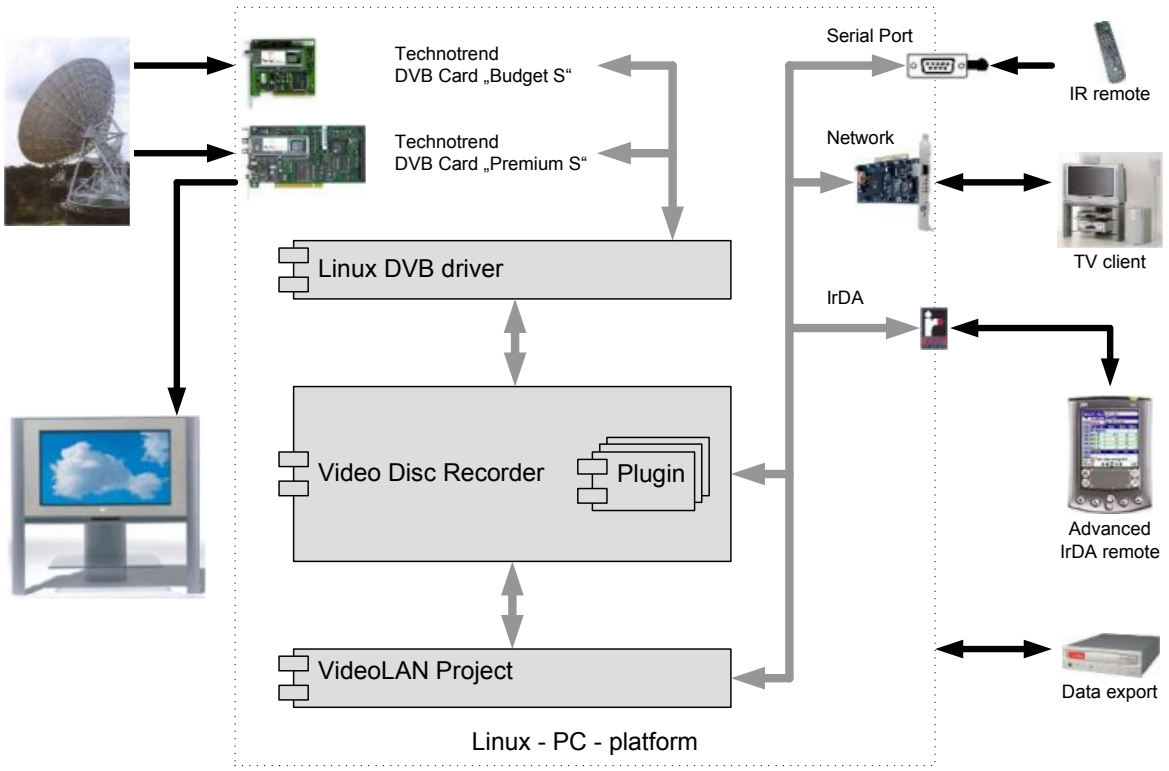


Abbildung 2.3.: Struktur des Digitalen Video Projekts

wenden von Standardkomponenten. Abbildung 2.3 gibt einen Überblick über die Struktur und die Komponenten des Projekts.

Als Endergebnis der Entwicklung soll ein digitaler Videorecorder oder besser ein Multimedia Center entstehen. Im Vordergrund steht dabei eine sehr nutzerfreundliche Bedienung, die verschiedene Nutzergruppen zufrieden stellen soll. Weiterhin soll eine flexible, modulare Architektur entstehen, die es erlaubt verschiedenste Funktionen und Ausstattungsmerkmale zu integrieren und effizient nach den Bedürfnissen der Nutzer anzupassen. Aus diesem Grund wurde das Konzept als Systemfamilie entworfen.

Das zu entwickelnde System des DVP soll nicht nur den Empfang und die Wiedergabe von digitalen Fernsehsignalen gewährleisten, sondern zusätzlich Funktionalitäten bieten, die weit über die des „normalen“ Fernsehens hinaus gehen.

Beispielsweise soll es möglich sein, Fernsehsendungen auf Festplatte aufzuzeichnen, diese per Videoschnitt zu bearbeiten und auf andere Medien wie CD, Video-CD und DVD zu exportieren. Neben dem Empfang der reinen Audio- und Videodaten ist es ebenfalls möglich dem Nutzer programmbegleitende Informationen in Form einer elektronischen Programmzeitschrift (Electronical Program

Guide, kurz EPG) zur Verfügung zu stellen.

Das System ist allerdings nicht nur als Einzelplatzlösung gedacht. Über eine Netzwerkverbindung können beliebig viele TV-Clients bedient werden, was als Basis für eine Video-on-Demand Lösung genutzt werden kann. Florian Meffert hat sich mit dieser Thematik in seiner Diplomarbeit auseinandergesetzt [12].

Die Steuerung des Gerätes erfolgt im Normalfall über eine Infrarotfernbedienung. Ebenfalls möglich ist die Verwendung einer erweiterten IrDA-Fernbedienung (Advanced IrDA Remote Control) unter Ausnutzung der IrDA-Schnittstelle der PC-Hardware. Die Realisierung dieser Variante war Gegenstand der Diplomarbeit von Ralph Dietzel [2].

Das Digitale Video Projekt beinhaltet eine Reihe von Hard- und Softwarekomponenten. Die momentan eingesetzte Hardware setzt sich zusammen aus:

- einem Standard PC (mit Netzwerkkarte, IrDA-Schnittstelle),
- einer Technotrend DVB Karte „Premium S“,
- einer Technotrend DVB Karte „Budget S“.

Neben diesen Hardwarekomponenten wird die folgende Software eingesetzt:

- SuSE Linux 8.0 (Kernel 2.4)
- Linux DVB Treiber 1.0
- VDR 1.2

Die Komponente VDR stellt das zentrale Element des Systems dar und wird deshalb im folgenden Abschnitt etwas näher betrachtet.

2.4. Video Disc Recorder

Mit der Verfügbarkeit von Festplatten großer Kapazitäten jenseits der 10 GByte und MPEG-Encodern/Decodern wurde die Idee des vollständig digitalen Videorecorders geboren.

Inzwischen existieren auf dem Markt verschiedene kommerzielle Produkte, die diese Funktionalität besitzen. Diese Geräte nutzen aber oft die gegebenen Möglichkeiten nur eingeschränkt oder bieten einige gar nicht erst an, wie beispielsweise das Schneiden der Aufnahmen auf der Festplatte oder das

Archivieren dieser auf externen Medien. Ein weiterer großer Nachteil kommerzieller Systeme ist die Abgeschlossenheit gegenüber Erweiterungen.

Mit dem Video Disc Recorder (VDR) [18] hat Klaus Schmidinger eine Software geschaffen, die es erlaubt einen normalen Standard-PC mit dem Betriebssystem Linux in einen vollwertigen digitalen Videorecorder zu verwandeln. Die einzige Hardware, die dafür zusätzlich benötigt wird, ist eine PCI-DVB-Karte für Satelliten-, Kabel oder terrestrischen Empfang. Diese wird von VDR über das API des Linux DVB Treibers [13] angesprochen.

Das VDR-Projekt ist ein Open-Source-Projekt, welches unter Anwendung der GNU General Public License (GPL) [8] freigeben wurde. Kurz zusammengefasst sichert diese Lizenz, dass ein frei zur Verfügung gestellter Quellcode auch immer und für jedermann frei bleibt. Die Quellcodeoffenheit bedeutet, dass jeder, der Interesse hat, an der Weiterentwicklung der Software arbeiten kann, die Quellen sind jedermann frei zugänglich. Diese Eigenschaft war für die Verwendung von VDR im Digitalen Video Projekt enorm wichtig, da es sich früher oder später als unumgänglich erweisen wird, einige Modifikationen daran vorzunehmen.

Der Video Disc Recorder bietet eine große Zahl von Funktionen, unter anderem sind das die Folgenden:

- Verwendung des On-Screen-Displays (OSD) der DVB-Karte für Rückmeldungen und Menü, diese können in verschiedenen Sprachen angezeigt werden
- Steuerung per Infrarotfernbedienung oder Tastatur
- Unterstützung von bis zu 4 DVB-Karten in einem System; Damit ist die Aufnahme mehrerer Sendungen auf verschiedenen Kanälen gleichzeitig möglich
- Auswertung und Präsentation der EPG-Daten in verschiedenen Modi („Was läuft jetzt/Was kommt als nächstes?“)
- Aufnahmeprogrammierung per Electronical Program Guide oder manuell, einzelne oder wiederholende Programmierungen sind möglich
- Speicherung der Aufzeichnungen auf Festplatte, dabei erfolgt eine automatische Aufteilung in Dateien < 2GByte
- Unterstützung von Mehrkanalton und Dolby Digital Ton

- verschiedene Wiedergabemodi: Normal, Pause, schneller Vor- und Rücklauf, Sprung an definierte Stelle, Sprung um 1 Minute
- Unterstützung des Editierens von Aufnahmen.
- Netzwerkunterstützung (SVDRP): Verwaltung von Programmierungen und Aufzeichnungen via Telnet

Mit der innerhalb eines Jahres entwickelten und im Juni 2003 freigegebenen Version 1.2 wurde der Video Disc Recorder um einige neue und besondere Funktionalitäten erweitert. So ist das gleichzeitige Aufzeichnen einer Sendung und Wiedergeben einer Aufnahme mit nur einer einzigen DVB-Karte möglich. Damit kann das so genannte Time-Shift-Verfahren, also die zeitversetzte Wiedergabe einer Aufzeichnung, eingesetzt werden, auch wenn nur eine DVB-Karte im PC installiert ist. Die zweite große Neuerung der neuen VDR-Version ist die Schaffung einer Plugin-Schnittstelle, auf die im Folgenden näher eingegangen wird.

Plugin-Schnittstelle

Bis einschließlich zur Version 1.0 des Video Disc Recorder war eine Erweiterung des Funktionsumfangs nur mit teilweise umfangreichen Änderungen des Quelltextes von VDR möglich. Diese wurden auch in großer Zahl in Form von Patches angeboten, welche allerdings einige Nachteile mit sich brachten. Nicht nur die Übersichtlichkeit des Quelltextes verschlechterte sich, auch wurde das Finden der Ursachen beim Auftreten von Fehlern enorm erschwert. Außerdem ergaben sich oft Schwierigkeiten beim Kombinieren verschiedener Patches.

Aus diesem Grund wurde mit dem Entwicklerzweig 1.1 eine Plugin-Schnittstelle in das VDR-Projekt integriert, die mit der stabilen Version 1.2 offiziell eingeführt wurde. Die Schnittstelle definiert eine Reihe von Basisklassen für verschiedene Funktionalitäten wie beispielsweise die Wiedergabe und den Empfang von Daten, für die Nutzung des On-Screen-Displays der DVB-Karte und die Unterstützung neuer Arten von Fernsteuerungen. Plugins in Form von dynamisch ladbaren Bibliotheken können Funktionalitäten dieser und weiterer Arten bereitstellen, indem sie neue Klassen und Objekte von diesen Basisklassen ableiten.

Inzwischen existiert eine große Zahl von Plugins für den Video Disc Recorder [19]. Sie realisieren die verschiedensten Funktionalitäten wie zum Beispiel Folgende:

- Wiedergabe verschiedener Medien (DVD, Video CD, MP3, DivX)

- Unterstützung neuer Empfangs- und Wiedergabehardware (DXR3-MPEG-2-Karten, analoge TV-Karten)
- Ausgabe von Statusinformationen auf LCDs (Text/Grafik)
- Streaming der DVB-Daten über Netzwerk
- Teletext, Untertitel
- uvm.

Auch die vorliegende Arbeit nutzt die Plugin-Schnittstelle für die Integration der Multimedia Home Platform in den VDR.

3. Warum MHP? – Mehrwertdienste im digitalen Fernsehen

In diesem Kapitel werden verschiedene Mehrwertdienste des digitalen Fernsehens bzw. Plattformen für solche vorgestellt und deren Möglichkeiten zusammengefasst. Auf die Multimedia Home Platform wird dabei im Speziellen eingegangen und die Frage geklärt, warum dies der Standard ist, auf den in Zukunft gesetzt werden sollte.

Mit dem Begriff Mehrwertdienste werden Angebote der Sendeanstalten, Gerätehersteller oder Dritter bezeichnet, die eine Nutzung des Mediums Fernsehen über die reinen Bild- und Tonsignale hinaus erlauben. Mit der Übertragung des DVB-Signals nach dem MPEG-2 Standard werden diese Dienste ermöglicht, da dieser die Einbettung von Daten beliebiger Art in den Transportstrom erlaubt.

Die Palette der Mehrwertdienste umfasst sowohl einfache, unidirektionale Angebote, beschränkt auf Textinformationen oder nur einfache Grafiken, als auch Dienste, die multimediale Inhalte aufbereiten und durch eine bidirektionale Kommunikation Interaktivität bis hin zum vollwertigen Internetzugang bieten. Im Folgenden wird deshalb eine Unterteilung in einfache und erweiterte Dienste vorgenommen.

3.1. Einfache Mehrwertdienste

Der DVB-Standard umfasst einige einfache Zusatzdienste. Da ist zum einen der Teletext zu nennen, der vom analogen Fernsehen übernommen wurde. Es wurden aber auch neue Angebote speziell für den neuen Standard entwickelt: Der elektronische Programmführer mit Informationen zu den übertragenen Sendungen und die DVB-Untertitel mit deutlich erweiterten Möglichkeiten. Die zwei Ersteren werden nun näher vorgestellt.

3.1.1. Teletext

Das Teletext-System, in Deutschland unter dem Namen Videotext bekannt, wurde für das analoge Fernsehen entwickelt um in der Austastlücke des Fernsehbildes unsichtbar Daten zu übertragen. Diese können beim Empfänger in Form von Grafiken und Texten auf dem Bildschirm sichtbar gemacht werden, den entsprechenden Dekoder vorausgesetzt. Diese Form der Datenübertragung wurde bei der Entwicklung des digitalen Fernsehens berücksichtigt und in den DVB-Standard aufgenommen.

Der Videotext setzt sich aus vielen (bis ca. 800) Seiten zusammen, die wiederum Unterseiten besitzen können. Diese werden fortlaufend zyklisch übertragen, so kann es 30 Sekunden und länger dauern, bis die gewünschte Seite angezeigt wird, wobei die Wartezeit durch Zwischenspeicherung der Seiten im Empfangsgerät verkürzt werden kann. Eine Teletext-Seite beinhaltet 24 Zeilen zu je 40 Zeichen. Zum Zeichenvorrat gehören neben Buchstaben und Zahlen auch grafische Blöcke verschiedener Formen mit denen sich Symbole und Grafiken einfacher Art zusammensetzen lassen, wobei 8 verschiedene Farben möglich sind.

Der Teletext wird vorrangig für die Übertragung von Textinformationen genutzt. Dazu gehören Hintergrundinformationen zu einzelnen Fernsehsendungen, Untertitel für Hörgeschädigte, aktuelle Nachrichten und Wettervorhersagen sowie Serviceangebote.

Die Übertragung der Teletextseiten im DVB-Datenstrom geschieht wie im Folgenden beschrieben. Nach der DVB-Spezifikation wird ein Videotext-Elementarstrom erzeugt und in den Transportstrom eingebracht. Die Videotext-Daten werden magazin- und zeilenweise aufbereitet und zunächst als Packetized Elementary Stream zusammengestellt. Dabei muss die Länge des Nutzdatenteils der PES-Pakete einem Vielfachen von 184 Byte entsprechen. Nach optionalen Headern folgt die eigentliche Teletext-Information in Blöcken von 44 Byte (Abbildung 3.1). Diese beginnen mit Angaben über die Magazin- und Zeilennummer gefolgt von der darzustellenden Information mit 40 Byte Länge für die 40 Zeichen einer Zeile. Die PES-Pakete des Teletextstroms werden in TS-Pakete aufgeteilt und genauso wie Audio- oder Videodaten im Transportstrom des DVB-Signals übertragen. Die PID des Teletextstroms ist als PID für Private Streams in der Program Map Table des jeweiligen Programms hinterlegt und kann somit vom Empfänger ausgewertet werden.

Der Videotext wird wohl in Zukunft ein wenig an Bedeutung verlieren, da er einen rein unidirektionalen Informationsweg vom Sender zum privaten Haushalt darstellt. Die grafischen Fähigkeiten sind wegen des festen Zeichenvorrats zudem sehr eingeschränkt, was den Teletext im Vergleich zu neueren Verfahren etwas altbacken erscheinen lässt. Außerdem ist das Erscheinungsbild fest vorge-

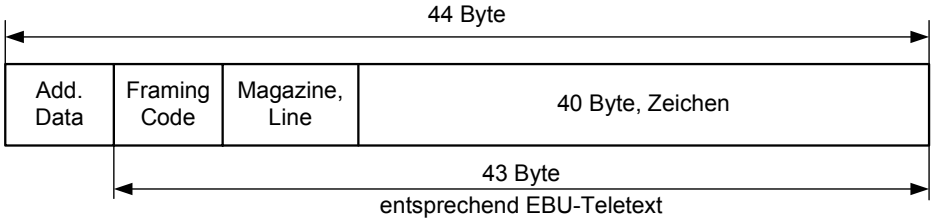


Abbildung 3.1.: Aufteilung der Teletext-Daten in einen 44-byte-Block (aus [15])

geben, die fehlende Strukturierung der übermittelten Informationen verhindert eine übersichtlichere und nutzerfreundlichere Aufbereitung. Auch die Untertitel zu Fernsehsendungen, die eine besondere Anwendung des Videotextes darstellen, werden wohl mit der Zeit durch die DVB-Untertitel ersetzt werden, da diese einige erweiterte Möglichkeiten, wie Mehrsprachigkeit, beliebige Positionierung im Bild und Unterstützung von Grafik mit sich bringen [5].

3.1.2. Programminformationen aus der EIT

Bei der Entwicklung des digitalen Fernsehens wurde die Möglichkeit geschaffen Informationen zum gesendeten Programm als Ergänzung zu den Bild- und Tondaten zu übertragen. Dies erfolgt in einer der im DVB-Standard definierten Tabellen der Service Information, der Event Information Table (EIT). Die EIT kann strukturierte Informationen zu jeder einzelnen Sendung (Event) enthalten, die für die Erstellung einer elektronischen Programmzeitschrift (Electronical Program Guide, kurz EPG) verwendet werden können.

Jeder Eintrag der EIT beinhaltet die Startzeit und die Dauer einer Sendung. Daran können sich Deskriptoren[1] verschiedener Art anschließen, die weiter ins Detail gehende Informationen enthalten. Dazu zählen der Titel der Sendung sowie eine Beschreibung über deren Inhalt, wobei diese Informationen in mehreren Sprachen angegeben werden können. Des Weiteren sind Angaben über die Art bzw. das Genre sowie das empfohlene Mindestalter einer Sendung möglich.

Die Informationen aus der Event Information Table stellen einen einfachen aber direkten Weg dar, den Zuschauer mit Zusatzinformationen zu den ausgestrahlten Sendungen zu versorgen. Im Gegensatz zum Teletext, der auch diese Art von Informationen enthalten kann, werden sie in einer strukturierten Form übertragen. Damit ist die Aufbereitung und Darstellung der Daten nicht fest vorgegeben, sondern der Software des Empfangsgerätes überlassen. Es können sehr nutzerfreundliche und übersichtliche Auflistungen erstellt werden, wie beispielsweise eine „Was läuft jetzt?“- oder „Was kommt

[1] Ein Deskriptor ist eine Folge von Daten, deren Bedeutung durch eine Identifikationsnummer bestimmt wird.

Abbildung 3.2.: Aufbereitung der EIT-Daten durch VDR, die „Was läuft jetzt?“-Liste (links) und die Detailinformation zu einer Sendung (rechts)

als nächstes?“-Liste über alle empfangbaren Programme. Auch die Software VDR bietet diese Möglichkeit, wie in Abbildung 3.2 zu sehen ist. Die EPG-Daten besitzen nur einen Nachteil: Sie sind auf relativ wenige Informationen in Textform beschränkt, Bilder oder gar kurze Videosequenzen sind nicht möglich.

3.2. Erweiterte Mehrwertdienste

Seit der Einführung des digitalen Fernsehens haben sich Sendeanstalten Gedanken darüber gemacht, wie sie ihr Angebot durch Mehrwertdienste erweitern könnten. Im Vordergrund stand dabei die Erschließung neuer Einnahmequellen aber auch die Erhöhung der Attraktivität des Angebots zur Steigerung von Marktanteilen. Aufgrund der Möglichkeiten der neuen Technik, wie das schnelle Überspringen von Werbeunterbrechungen bei der zeitversetzten Wiedergabe, könnte die herkömmliche Fernsehwerbung zunehmend an Bedeutung verlieren. Deshalb sind diese Schritte für die Fernsehsender notwendig.

Bei dieser Entwicklung wurde ein neuer Begriff geprägt: Das interaktive Fernsehen. Dies umfasst eine Vielzahl von Anwendungen, die das Fernsehen für die Zuschauer attraktiver machen könnten. Wegen neuer technischer Komponenten, wie dem Rückkanal für die Datenübertragung vom Empfänger zum Sender, und der ständigen Steigerung der Rechenleistung und der Zunahme der Fähigkeiten der Endgeräte werden heute Dienste ermöglicht, die früher undenkbar waren. Im Folgenden sind einige davon angegeben:

- Multimediale Zusatzdienste, das heißt die Kombination von Textinformationen mit Bildern

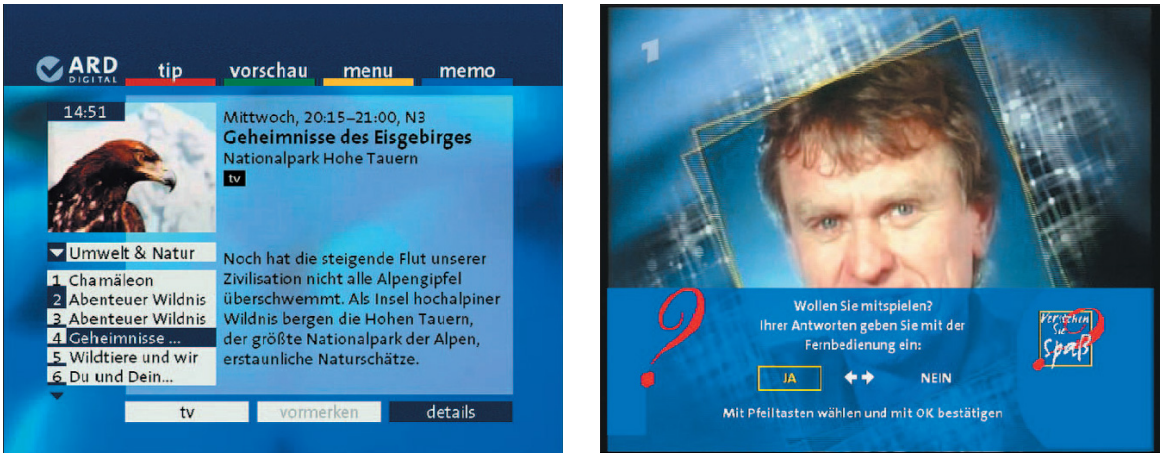


Abbildung 3.3.: Interaktive Angebote von ARD digital: Ein erweiterter Programmführer (links) und Mitspielmöglichkeit bei einer TV Show (rechts). (aus [1])

und Audio- bzw. Videosequenzen. Dazu zählen beispielsweise Hintergrundinformationen zu Fernsehbeiträgen, eine im Gegensatz zur EIT erweiterte elektronische Programmzeitschrift, Nachrichten und Wetterinformationen. Es sind auch Dienste möglich, die den Zuschauer mit einbeziehen, wie kleine Videospiele oder Lernprogramme als Zusatzdienst beim Bildungsfernsehen.

- E-Commerce: Diese Dienste ermöglichen einfaches und schnelles Einkaufen von zu Hause aus, die Abwicklung und Bezahlung der Bestellung findet sofort über den Rückkanal statt.

- Angebote von Werbeträgern: Es können gezielte, auf den Zuschauer angepasste Werbeeinblendungen realisiert werden.

- Verbindung von Fernsehen und Internet: Es können neben der Ausstrahlung von Fernsehsendungen auch multimediale Inhalte aus dem Internet übertragen werden. Dies kann in Verbindung mit dem Rückkanal zu einem vollwertigen Internetzugang erweitert werden.

- echtes interaktives Fernsehen, das heißt die Beeinflussung und Teilnahme des Zuschauers am Fernsehprogramm. Dazu zählen unter anderem die Ausstrahlung von Filmen auf Abruf (Video on Demand) sowie die Teilnahme an Umfragen oder Gewinnspielen.

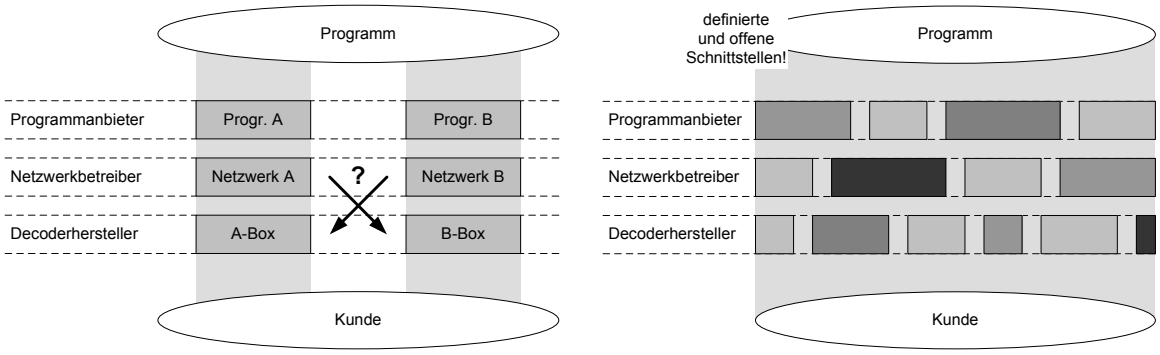


Abbildung 3.4.: Der vertikale Markt und seine Probleme. Zum Vergleich dazu der horizontale Markt mit seinen offenen Schnittstellen. (aus [1])

3.2.1. Marktstrukturen

Für die Schaffung und Nutzung der erweiterten Mehrwertdienste existieren im DVB-Standard keinerlei Vorgaben bis auf die durch das MPEG-2 Verfahren spezifizierten Wege der Datenübertragung. Deshalb mussten Plattformen geschaffen werden, die sowohl die Entwicklung und Verbreitung als auch das Dekodieren und Präsentieren der Anwendungen des interaktiven Fernsehens ermöglichen sollten.

Aufgrund der fehlenden Standards haben verschiedene Sendeanstalten und Endgerätehersteller eigene proprietäre Lösungen für die Integration interaktiver Anwendungen ins digitale Fernsehen entwickelt und vermarktet. Damit entstand eine so genannte vertikale Marktstruktur. Dieser Begriff steht für eine Situation, in der alle Ebenen der Wertschöpfungskette, also das Programm, die Verschlüsselung, die Verteilwege und das Empfangsgerät, kommerziell und technisch in den Händen eines Unternehmens liegen. Ein Beispiel dafür ist die Plattform von Premiere World bestehend aus dem Programmpaket und der D-Box als Empfangsgerät mit dem Betriebssystem Betanova von BetaResearch.

Der vertikale Markt bringt einige Probleme mit sich. Da alle Marktsegmente vom Programm zum Nutzer von einem Unternehmen besetzt sind, kann sich sich ein Anbieter eine Vormachtstellung auf diesem Gebiet erarbeiten, was sich nachteilig auf die Weiterentwicklung der Produkte und die Höhe der Preise auswirken kann. Außerdem ergeben sich wegen der Nicht-Offenheit der Schnittstellen grundlegende und auch gewünschte Inkompatibilitäten zwischen den Plattformen verschiedener Anbieter. Damit soll verhindert werden, dass sich ein Unternehmen die mühsam ausgebaute Decoderinfrastruktur eines anderen Anbieters zunutze macht. Dies benachteiligt letztendlich den Kunden, da er für jede Plattform einen eigenen anbieterspezifischen Decoder benötigt.

Der horizontale Markt für digitales Fernsehen verfolgt ein ganz anderes Ziel. Hier soll es jedem Anbieter möglich sein, seine Programme und Diensteangebote auf allen Geräten darstellen zu können. In einer solchen durch Konkurrenz bestimmten Marktstruktur sorgt der Wettbewerb zwischen den Unternehmen für Innovationen und niedrige Preise. Erst dies wird zu einer Akzeptanz des neuen Mediums beim Kunden führen.

3.2.2. Verfügbare Plattformen

Einige der Plattformen für interaktive Mehrwertdienste werden im folgenden vorgestellt, wobei auf die Multimedia Home Platform als Standard der Zukunft besonders Wert gelegt wird.

Proprietäre Standards

Eine der weit verbreitetsten Plattformen für interaktives Digitalfernsehen stellt das Betriebssystem OpenTV dar, welches von dem gleichnamigen Joint-venture Unternehmen zwischen der französischen Firma Thomson und Sun entwickelt wurde. Es ist auf weit über 10 Millionen Set-Top-Boxen weltweit installiert und wird beispielsweise von ARD digital für ein vielfältiges multimediales Free-TV-Angebot verwendet, welches einen elektronischen Programmführer und Mitspielmöglichkeiten bei verschiedenen Sendungen bietet. Laut Hersteller ist OpenTV weltweit führend im Bereich Entwicklung von interaktiven Medien-Lösungen.

Weitere proprietäre Lösungen für interaktive Anwendungen sind das Betriebssystem Mediahighway vom Canal+, welches vor allem in Frankreich eingesetzt wird, sowie PowerTV von der Scientific-Atlantas Inc. Alle diese Systeme besitzen ausreichende Fähigkeiten die oben genannten Anwendungen zu realisieren. Ihr größter Nachteil ist jedoch, dass die Lösungen untereinander nicht kompatibel sind. Damit sind Dienste, die für eine Plattform entwickelt wurden, nicht auf einem Empfangsgerät lauffähig, das eine andere Lösung bevorzugt.

MHEG

Aufgrund der bestehenden Gefahr einer Monopolbildung, falls sich einer der proprietären Standards durchsetzen würde, und der Vielzahl von verfügbaren Plattformen wurde der Ruf nach einem internationalen Standard für die Präsentation multimedialer und hypermedialer Informationen laut. Zu diesem Zweck wurde die Institution MHEG[2] von der ISO ins Leben gerufen. Diese Gruppe hat eine

[2] Multimedia and Hypermedia Expert Group

gleichnamige API definiert, die festlegt, wie multimediale Präsentationen und deren Benutzterinteraktionen formuliert werden können.

MHEG ist rein deklarativ, das heißt es wird eine Beschreibungssprache angeboten, die die freie Modellierung von Inhalten im digitalen Fernsehen ermöglicht. Die MHEG-Engine, die auf einer Set-Top-Box installiert ist, stellt eine Art Browser dar, der multimediale Inhalte zusammen fügt und sich durch den Nutzer interaktiv steuern lässt. Zusätzlich kann die Position der visuellen Elemente im Darstellungsbereich mit absoluten Koordinaten festgelegt werden, was für potentielle Werbekunden interessant ist.

MHEG, genauer MHEG 5, wird zwar seit 1998 unter anderem in Großbritannien eingesetzt, konnte aber die vorhandenen proprietären Lösungen für interaktive Anwendungen nicht ablösen. Das liegt zum Großteil daran, dass es sich nur um eine deklarative Beschreibung von Inhalten handelt. Die Definition von neuem Verhalten und über den Standard hinausgehenden Funktionalitäten, wie das bei den anderen Plattformen durch die Einführung neuer Klassen und Methoden geschehen kann, ist bei MHEG nicht möglich. Auch wird der Standard seit einiger Zeit nicht mehr weiterentwickelt [11].

MHP

Das europäische DVB-Projekt war wegen oben genannter Gründe an einer horizontalen Marktstruktur interessiert. Deshalb wurde Anfang 1997 damit begonnen einen europäischen Standard für interaktive Angebote im digitalen Fernsehen zu entwickeln. Dabei ist es gelungen, den Konsens zwischen allen Beteiligten (Programm- und Diensteanbieter, Netzwerkbetreiber und Endgerätehersteller) herzustellen und den jahrelangen Streit um eine einheitliche Softwareschnittstelle bei den Digitalempfängern zu beenden. Als Ergebnis dieser Bemühungen wurde im Jahr 2000 die erste Version der Multimedia Home Platform (MHP) bei der europäischen Standardisierungsinstitution ETSI eingereicht.

Da die Multimedia Home Platform den Standard der Zukunft für interaktive Anwendungen im digitalen Fernsehen darstellt und im weiteren Verlauf dieser Diplomarbeit eine wichtige Rolle spielt, wird im nächsten Abschnitt auf einige Details der MHP eingegangen.

3.3. Die Multimedia Home Platform

3.3.1. Einführung

Die Multimedia Home Platform (MHP) definiert eine Schnittstelle zwischen interaktiven Anwendungen des Digitalfernsehens und den Geräten, auf denen diese ausgeführt werden. Die Schnittstelle entkoppelt die Applikationen unterschiedlicher Anbieter von der spezifischen Hard- und Software verschiedener Implementationen der MHP. Damit können alle Typen von Empfangsgeräten angesprochen werden, von Low-End bis zu High-End Set-Top-Boxen.

Beim Entwurf der MHP stand stets die Nutzung von Interaktivität im Vordergrund. Das umfasst sowohl die lokale Interaktivität auf dem Empfangsgerät als auch echte Interaktivität, die das Senden von Informationen zum Inhalteanbieter einschließt, wofür ein Rückkanal erforderlich ist. Mit der Interaktivität sollen dem Zuschauer eine Menge neuer Benutzungsmöglichkeiten eröffnet werden. Das beginnt bei einem elektronischen Programmführer, der Bilder bzw. kurze Videosequenzen aus Filmen und andere Informationen enthält und führt bis zur aktiven Beteiligung des Zuschauers an interaktiven Spielshows oder der Verfügbarkeit von Zusatzinformationen aus dem Internet auf Knopfdruck. Des Weiteren wurden bei der Entwicklung der MHP weitere interaktive Anwendungen angedacht, wie zum Beispiel Home-Shopping, Home-Banking, Pay-per-View, Video on Demand, E-Mail und Computerspiele.

Zur Abbildung der geplanten Dienste und Anwendungen wurden drei Profile definiert, welche die Klassifizierung der Fähigkeiten der MHP-Implementationen auf den Empfangsgeräten erlauben:

- Das *Enhanced Broadcast Profile* ist das Basisprofil von MHP. Es wird für Systeme verwendet, in denen Applikationen und Daten von den Sendeanstalten empfangen werden aber kein Rückkanal existiert. Anwendungen, die für dieses Profil entworfen werden, bieten eine lokale Interaktivität mit den Nutzer, wie beispielsweise elektronische Programmführer und Informationsdienste.

- Das *Interactive Broadcast Profile* beinhaltet das Enhanced Broadcast Profile und fügt Unterstützung für einen bidirektionalen Datenkanal hinzu, wobei eines der Systeme verwendet wird, die im DVB-Standard für Rückkanäle spezifiziert wurden. Dies erlaubt das Senden von Daten zum Programmanbieter und ermöglicht unter anderem e-Commerce- und Video-on-Demand-Applikationen.

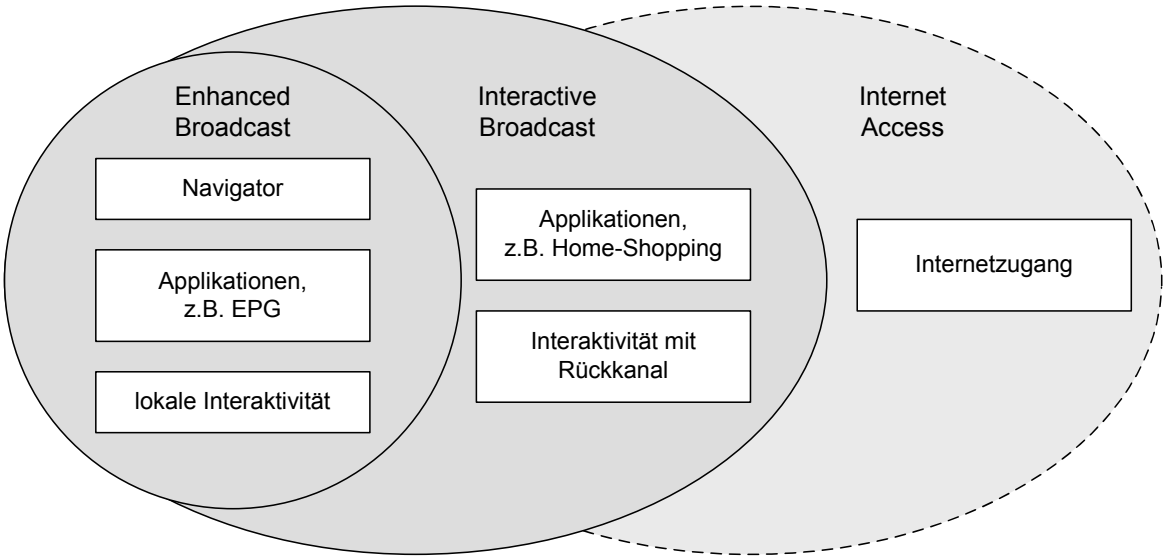


Abbildung 3.5.: Profile für MHP-Implementationen (aus [20])

- Das *Internet Access Profile* umfasst das Interactive Broadcast Profile. Es fügt Anwendungen auf dem Empfangsgerät hinzu, welche den Zugriff auf die meisten der bekannten Internetdienste, wie WWW, E-Mail und Newsforen erlauben. Dieses Profil erlaubt die Verknüpfung von interaktiven Applikationen mit den Diensten des Internets.

Von diesen drei Profilen wurden die ersten beiden in der MHP-Spezifikation 1.0 umgesetzt und im Mai 2000 dem europäischen Standardisierungsgremium ETSI vorgelegt. Die Version 1.0 wurde inzwischen über die Versionen 1.0.1 und 1.0.2 in die MHP-Spezifikation 1.0.3 überführt, welche im Juni 2003 freigegeben wurde. Bei diesem Vorgang wurden eine Menge Korrekturen eingefügt, die Fehler und Inkonsistenzen beseitigen, die zum Großteil erst durch die Entwicklung von MHP-Implementationen sichtbar wurden. Parallel zu diesem Prozess wurde an der Version 1.1 der Spezifikation gearbeitet. Diese bringt vor allem eine detaillierte Spezifikation des Internet Access Profile sowie des Persistent Storage, also der Speicherung von Daten auf einem Datenträger im Empfangsgerät, mit sich.

3.3.2. MHP-Architektur

Um eine hohe Plattformunabhängigkeit zu erreichen und einen horizontalen-Markt für MHP-Implementationen zu schaffen wurde die Möglichkeit der Informationsverarbeitung in einem Schichtenmodell vorgesehen. Die MHP-Spezifikation legt in erster Linie die Grenzschicht zwischen der

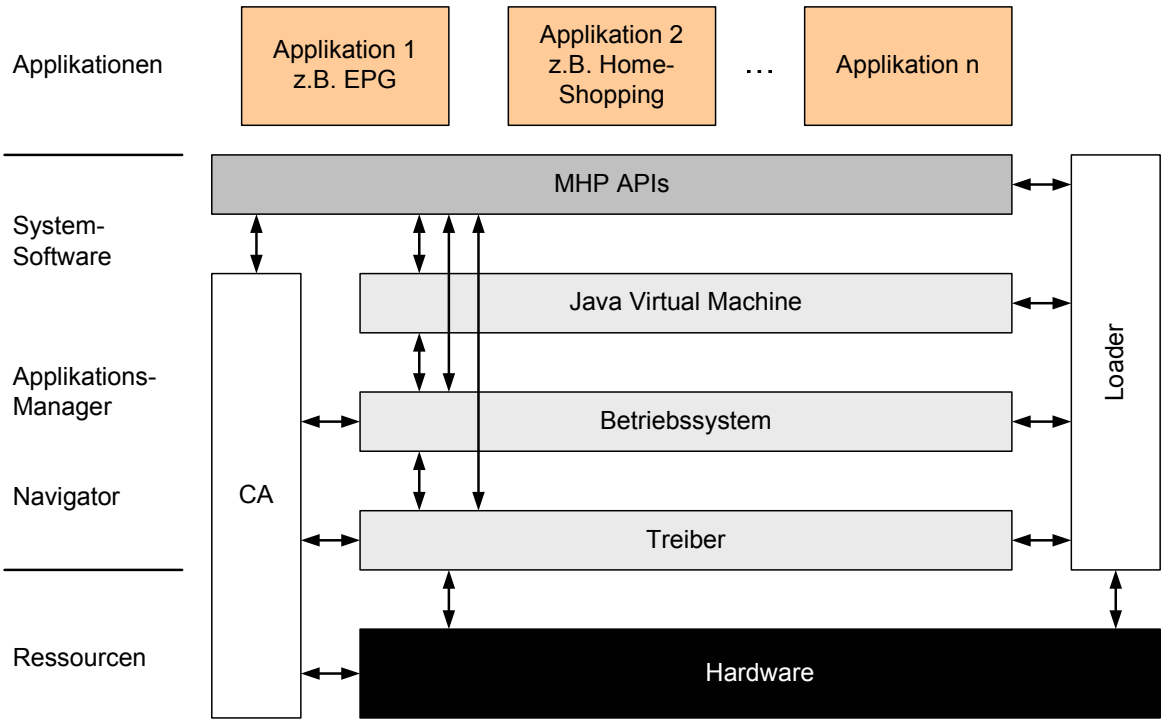


Abbildung 3.6.: Allgemeine MHP-Architektur (aus [20])

Systemsoftware und den Applikationen fest. Es wird explizit darauf hingewiesen, dass die Spezifikation nur Schnittstellen zwischen diesen Elementen beschreibt und nicht deren Implementation.

In Abbildung 3.6 wird eine allgemeine Form der MHP-Architektur gezeigt. In der obersten Schicht befinden sich eine oder mehrere Applikationen, die auch parallel ablaufen können. Diese werden im Datenstrom des DVB-Signales übertragen. Dagegen sind die Elemente der weiteren Schichten resident im Empfangsgerät vorhanden. Dies erfordert allerdings einen gewissen Speicherplatz in der MHP für die einzelnen APIs.

Die Systemsoftware ist in mehrere Schichten aufgeteilt. Zu ihr zählen die Java Laufzeitumgebung in Form der Java Virtual Machine, das Betriebssystem sowie die Treiber, die den Zugriff auf die Hardware ermöglichen. Ein weiterer Bestandteil der Systemsoftware ist der Application-Manager, der verantwortlich für das Starten und Stoppen von Applikationen ist und deren Aktivitäten überwacht. Außerdem gehört ein je nach Implementation unterschiedlich ausgestatteter Navigator zu den Bestandteilen dieser Schicht. Der Navigator soll dem Nutzer eine Übersicht über die Dienste und Programme eines Senders bieten, damit sich dieser aus der Vielzahl der Angebote die ihn interessierenden aussuchen kann.

Typischerweise ist ein Loader zum Nachladen von Updates für die Systemsoftware vorhanden.

Dieser ist aber nicht Bestandteil der MHP-Spezifikation, sondern kann von jedem Hersteller anders implementiert werden. Mit dem Loader kann die Zukunftssicherheit der Implementation gewährleistet werden. Ein weiteres Element der Architektur ist die Conditional Access Einheit, die für den Zugriff auf verschlüsselte Inhalte verantwortlich ist.

In der untersten Schicht der Architektur befinden sich schließlich die Ressourcen der Multimedia Home Platform. Dazu gehören Elemente der Hardware des Empfangsgerätes wie Tuner, MPEG-Decoder, Eingabe/Ausgabe-Schnittstellen und Grafiksystem.

3.3.3. MHP-Applikationen

Es gibt zwei Typen von MHP-Applikationen: DVB-HTML- und DVB-J-Applikationen. Der erste Typ ist nicht sehr verbreitet, was zum einen daran liegt, dass die Spezifikation von DVB-HTML erst mit Version 1.1 des MHP-Standards komplettiert wurde. Andererseits ist dieser Typ vielen Sendeanstalten, Geräteherstellern und Applikationsentwicklern zu komplex und zu schwer zu implementieren. Eine DVB-HTML Applikation entspricht einer Menge von HTML-Seiten, die als Teil eines Programms gesendet werden.

Der zweite und weitaus populärere Typ von MHP-Applikationen sind die DVB-J-Applikationen. Diese sind in der Programmiersprache Java geschrieben unter Verwendung der MHP-Schnittstelle. Die verschiedenen APIs dieser Schnittstelle sind in Anhang A angegeben. Die DVB-J-Applikationen werden als eine Menge von im Bytecode vorliegenden Klassendateien im Datenstrom übertragen.

DVB-J-Applikationen liegen als Xlets vor. Diese wurden von Sun in der JavaTV Spezifikation eingeführt, welche ein allgemeines API für interaktive TV-Plattformen beinhaltet. Die Xlets sind ein Konzept ähnlich den Applets in Internetanwendungen. Das heißt sie sind selbst nicht ausführbar sondern erfordern einen Container in Form der Applikationsverwaltung, also des Application-Managers, der das Starten und Beenden des Xlets übernimmt.

Applikations-Signalisierung

Bevor der Application-Manager eine MHP-Anwendung ausführen kann, sind einige vorausgehende Schritte erforderlich. Er muss als erstes erfahren, dass eine Applikation überhaupt existiert, danach muss geklärt werden, ob der Nutzer diese Anwendung zur Zeit ausführen darf und schließlich muss der Manager Zugriff auf alles erhalten, was für die Ausführung der Applikation notwendig ist, also die Klassendateien und andere Daten.

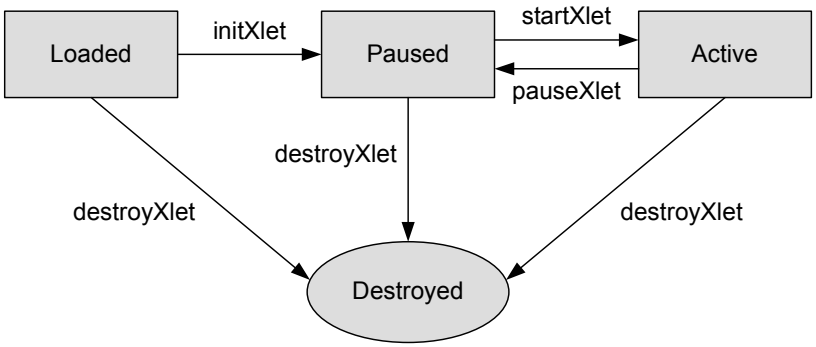


Abbildung 3.7.: Lebenszyklus eines Xlets (aus [7])

Für die Übertragung der eben genannten Informationen führt die MHP eine neue Tabelle der Service Information, die Application Information Table (AIT), ein. Diese Tabelle wird in jedem Service, der ein oder mehrere MHP-Applikationen enthält, gesendet und beinhaltet einen Eintrag pro übertragener MHP-Anwendung. Die Informationen der AIT werden in Form von Deskriptoren angegeben.

Die AIT enthält alle Informationen, die ein Empfänger benötigt um die Applikation auszuführen und dem Nutzer die verfügbaren Applikationen in geeigneter Weise mitzuteilen. Dies beinhaltet den Namen der Anwendung, die Position ihrer Dateien (Klassen und Daten) und mögliche Parameter, die der Applikation beim Start übergeben werden. Da der Name nicht unbedingt eindeutig sein muss, besitzt jede MHP-Applikation einen eindeutigen Bezeichner, der sich aus der Organization ID und der Application ID zusammen setzt.

Für die Kennzeichnung des Laufzeitverhaltens einer Applikation existieren in der AIT verschiedene Indikatoren. Einer davon legt fest, ob eine Anwendung beim Umschalten auf ein neues Programm automatisch gestartet oder beendet werden soll oder ob der Nutzer diese manuell per Knopfdruck ausführen darf. Außerdem können Applikationen existieren, die auch beim Umschalten auf einen neuen Sender weiter ausgeführt werden. Ein programmübergreifender EPG stellt eine Anwendung dieses Konzepts dar.

Lebenszyklus einer MHP-Applikation

Jede MHP-Applikation muss ein bestimmtes Java Interface (`javax.tv.xlet.Xlet`) implementieren, damit sie vom Application-Manager ausgeführt werden kann. Die Xlet Schnittstelle stellt vier Methoden zur Verfügung, mit denen die Anwendung von anstehenden Zustandsänderungen in Kenntnis gesetzt werden kann.

Das Diagramm in Abbildung 3.7 zeigt den Lebenszyklus einer MHP-Applikation und welche Zustandsübergänge von den Methoden ausgelöst werden. Nach dem extrahieren einer Applikation aus dem Datenstrom befindet sich diese im „Loaded“-Zustand. Folgende Übergänge sind daraufhin möglich:

- Mit dem Aufruf der *initXlet*-Methode wird die Applikation veranlasst sich selbst zu initialisieren. Es erfolgt ein Übergang in den „Paused“-Zustand.
- Danach kann ein Aufruf von *startXlet* erfolgen, mit dem der Applikation mitgeteilt wird, dass sie in den Zustand „Active“ übergehen und mit ihrer Ausführung beginnen soll. Jetzt sind auch Ausgaben auf dem Bildschirm möglich.
- Mit der Methode *pauseXlet* wird die Applikation veranlasst sich wieder in den „Paused“-Zustand zu begeben, dabei falls nötig Bildschirmausgaben zu entfernen und den Ressourcenbedarf zu minimieren. Danach ist wieder ein Aufruf von startXlet möglich.
- Der Aufruf der *destroyXlet*-Methode ist von jedem Zustand aus möglich. Damit wird die Applikation darauf hingewiesen, dass sie in naher Zukunft beendet wird. Sie sollte deshalb gegebenenfalls Statusinformationen speichern und Ressourcen sobald wie möglich freigeben.

Im Normalfall werden diese Methoden vom Application-Manager aufgerufen um den Lebenszyklus der Applikation zu beeinflussen, falls durch bestimmte Ereignisse, wie zum Beispiel das Umschalten auf ein neues Programm, eine Änderung des Laufzeitverhaltens der Anwendung nötig wird. Jedoch kann auch die Applikation selbst einen Zustandsübergang durchführen und den Manager davon in Kenntnis setzen.

Übertragung der Dateien einer MHP-Applikation

Die Dateien einer Applikation werden als DSM-CC Object Carousel im MPEG-2 Transportstrom übertragen. Der DSM-CC[3] Standard ist Teil der MPEG-2 Spezifikation und stellt ein sehr umfangreiches und komplexes Protokoll für die Übertragung von Daten dar. Es ist aufgrund besonderer Fähigkeiten, wie z. B. dynamische Updates, ideal für die Übertragung von MHP-Applikationen geeignet. Die Bezeichnung Karussell stammt daher, dass die Objekte wiederholt übertragen werden, damit ein Empfänger immer alle Daten einer Applikation erhalten kann, egal wann mit dem Empfang begonnen wird.

[3] Digital Storage Media - Command & Control [10]

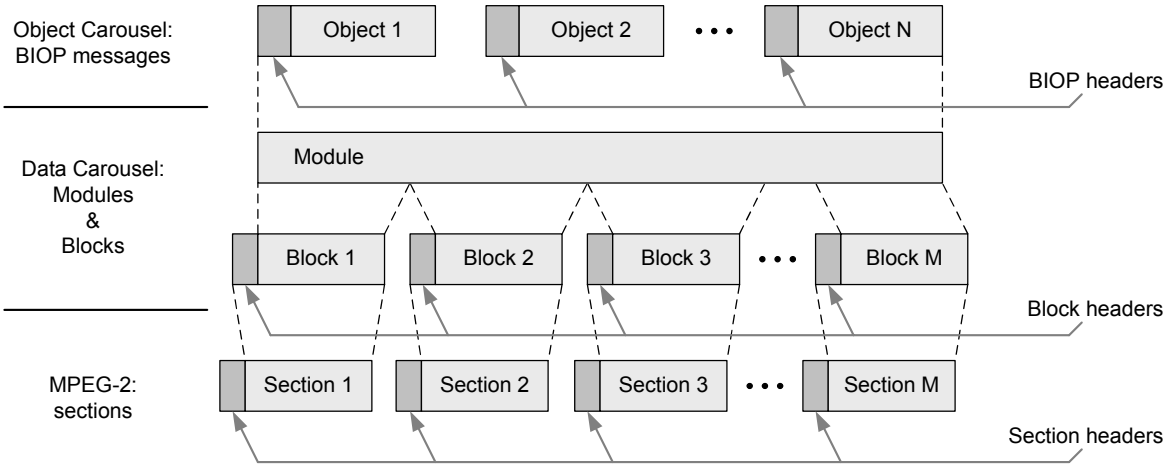


Abbildung 3.8.: Dekomposition der Objekte in Module, Blöcke und Sections (aus [7])

Das DSM-CC Object Carousel ermöglicht die Übertragung strukturierter Objekte vom Sender zum Empfänger. Die Objekte können Dateien, Verzeichnisse oder andere Datenströme sein. Die eigentlichen Dateien und Verzeichnisse liegen auf einem Server des Senders und werden von diesem in einen MPEG-2 Transportstrom unter Verwendung des Object Carousel Protokolls eingefügt. Dieser Vorgang wird in Abbildung 3.8 verdeutlicht.

Die Objekte des Object Carousels werden in BIOP[4]-Messages eingeschlossen. Eine BIOP-Nachricht enthält einen Header, der die Kennzeichnung des Objekttyps enthält sowie die Identifikation der Objekte durch den eindeutigen Object Key erlaubt.

Die BIOP-Messages werden in Modulen des so genannten Datenkarussells (Data Carousel) übertragen. Dabei werden ein oder mehrere BIOP-Nachrichten zu einem Modul variabler Länge zusammengefasst. Nach der Spezifikation des DSM-CC Data Carousel wird jedes Modul je nach Größe in ein oder mehrere Blöcke gleicher Länge aufgespalten. Ein solcher Block, der einen Download-Data-Block repräsentiert, wird in eine MPEG-2 Section eingefügt, welche dann im Transportstrom des DVB-Signals übertragen wird. Ein weiterer Typ von Blöcken des Data Carousel ist die Download-Info-Indication. Diese enthält eine Auflistung der Module eines Datenkarussells sowie weitere Übertragungsparameter.

3.3.4. Grafikreferenzmodell

Die grafische Ausgabe einer MHP-Applikation wird durch die Kombination von drei hintereinander liegenden Schichten gebildet, wie es Abbildung 3.9 verdeutlicht. Die Hintergrundschicht (Back-

[4]Broadcast Inter-ORB Protocol

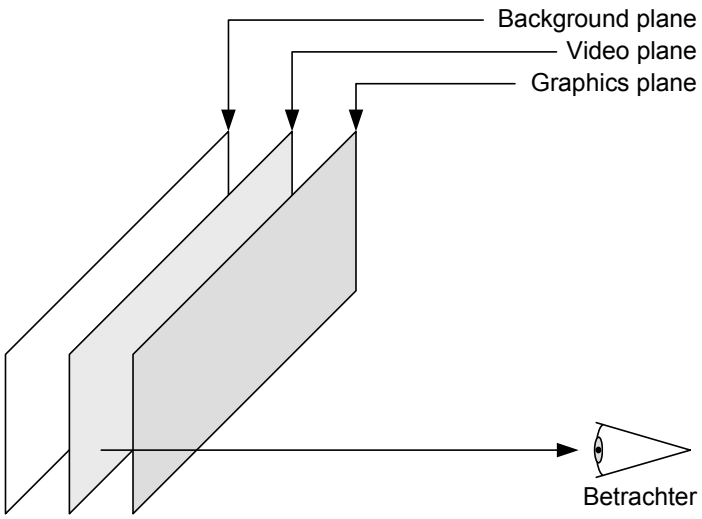


Abbildung 3.9.: Die Schichten des MHP-Grafikreferenzmodells (aus [20])

ground plane) kann in der Regel nur eine Farbe anzeigen, manchmal ist auch ein Bild möglich. Über dieser liegt die Videoschicht (Video plane), welche, wie der Name schon sagt, zur Darstellung von Videodaten, meist des aktuellen Fernsehsignals, dient. Allerdings sind hierbei die Möglichkeiten der meisten Set-Top-Boxen auf eine Vollbild- oder manchmal eine Viertelbild-Darstellung an festen Koordinaten beschränkt. Ganz oben liegt die Grafikschicht (Graphics plane). Das ist die Schicht, in der die Grafikoperationen der MHP ausgeführt werden.

Beim Zusammenfügen der drei Schichten zu einem Gesamtbild findet nicht nur eine einfache Kombination statt. Es können auch die Transparenzen der einzelnen Schichten festgelegt werden, wodurch sich neue Möglichkeiten ergeben. Beispielsweise lassen Elemente der Grafikschicht, wenn sie nur Teile des Bildschirms ausfüllen, ein dahinter liegendes Videobild an transparenten Stellen durchscheinen. Neben vollständiger Transparenz sind auch semi-transparente Bereiche möglich, die ein dahinter liegendes Videobild zwar sichtbar belassen aber dabei abschwächen.

Die Multimedia Home Platform bietet dem Anwendungsentwickler viele Werkzeuge für die Darstellung von Videobildern, Grafiken, Texten, Buttons und weiteren grafischen Komponenten auf dem Bildschirm, wobei eine freie Positionierung möglich ist. Für die so genannten „Lightweight Components“, also Grafikprimitiven wie Linien, Rechtecke und Texte, wird auf die Komponenten des Java AWT zurückgegriffen. Die „Heavyweight Components“ des Java AWT, wie sie scrollbare Listen und Buttons darstellen, waren für die MHP nicht geeignet, da deren Aussehen implementationsabhängig ist.

Für ein einheitliches „Look and Feel“ der MHP-Applikationen auf allen Plattformen auch bei der Verwendung von „Heavyweight Components“ mussten somit Erweiterungen am API vorgenommen werden. Außerdem musste eine Möglichkeit für die intuitive Verwaltung der Schichten des Grafikreferenzmodells geschaffen werden. Zur Lösung beider Probleme konnte auf die Java Erweiterungen des HAVi[5] Level 2 User Interface zurückgegriffen werden. Somit waren auf diesem Gebiet keine Eigenentwicklungen notwendig.

3.3.5. Sicherheitsaspekte

Mit der Integration eines Rechners in die Multimedia Home Platform und der Nutzung ablauffähiger Programme als Serviceinhalte musste bei der Entwicklung der MHP ein hoher Grad an Sicherheit gewährleistet werden. Durch den Ablauf von Programmen, die vorher übertragen werden, entsteht ein Gefahrenpotenzial, wie es von der Computerwelt und insbesondere aus dem Internet bekannt ist. Dort gehören Virenattacken und Hackerangriffe zur täglichen Praxis. Bei der Entwicklung der MHP-Spezifikation wurden deshalb Vorkehrungen getroffen, die die Sicherheit gewährleisten.

Alle MHP-Applikationen laufen in einer so genannten Sandbox ab. Ein Security Manager überwacht die Aktionen der Applikationen und die Zugriffe auf die Ressourcen des Empfangsgeräts und verhindert diese gegebenenfalls. Dieses Prinzip ist von der Java-Sandbox bekannt. Einer darin ablaufenden Applikation ist es nur erlaubt Inhalte auf dem Bildschirm auszugeben, nicht aber beispielsweise die Weitergabe von sicherheitsrelevanten Daten des Nutzers nach außen.

Für viele Applikationen des Fernsehbereichs sind diese Sicherheitskriterien zu restriktiv. Deshalb wurde die Möglichkeit des Signierens von MHP-Anwendungen geschaffen. Während unsignierte Applikationen in einer maximal gesicherten Sandbox ablaufen, können einer signierten Applikation spezifische Rechte zugewiesen werden. Dazu zählen zum Beispiel der Zugang zum nicht-flüchtigen Speicher des Empfangsgerätes, das Umschalten auf ein anderes Programm und die Nutzung des Rückkanals.

3.4. Fazit

Die Zukunft des Digitalfernsehens liegt zweifelsohne in den Applikationen der interaktiven Mehrwertdienste. Diese bieten weit gehende neue Möglichkeiten den Zuschauer aktiv am Fernsehgesche-

[5] Home Audio Video Interoperability

hen zu beteiligen, sei es das Mitspielen bei bestimmten Sendungen oder die Verfügbarkeit von Hintergrundinformationen auf Knopfdruck. Nicht nur der Zuschauer, sondern auch die Sendeanstalten und Netzwerkbetreiber können von den neuen Möglichkeiten profitieren. Die mit der Integration von Serviceangeboten anderer Unternehmen wie Home-Shopping oder Home-Banking verbundenen Mieteinnahmen kommen ihnen zugute. Außerdem ergeben sich durch die freie Positionierbarkeit grafischer Elemente neue Formen der Produktwerbung.

Mit der Multimedia Home Platform steht die beste Plattform für das interaktive Fernsehen zur Verfügung. Sie bietet von allen verfügbaren Lösungen die wohl umfassendsten Möglichkeiten für Anwendungen dieser Art und sie wird ständig weiterentwickelt. Während sich die proprietären Plattformen oft auf eine Art der Interaktivität beschränken, vereinigt die MHP sowohl lokale als auch echte Interaktivität in einem Standard. Bei dessen Entwicklung waren viele Vertreter der verschiedenen Interessensgruppen der Fernsehwelt beteiligt. Die MHP wird deswegen europa- und zunehmend auch weltweit akzeptiert und stellt damit die beste Wahl dar, wenn sich das digitale Fernsehen zu einem horizontalen Markt entwickeln soll.

Das Digitale Video Projekt hat sich zum Ziel gesetzt ein offenes System zu entwickeln, das alle Möglichkeiten des Digitalen Fernsehen unterstützen soll. Dazu zählen natürlich auch die interaktiven Anwendungen der Multimedia Home Platform. Allerdings existiert keine Implementation der MHP, die sich nahtlos in die modulare Architektur des DVP integriert. Die momentan existierenden Lösungen sind zudem nicht frei verfügbar sondern mit teilweise immensen Kosten verbunden, insbesondere wenn sie im Quelltext vorliegen müssen, was für eine nahtlose Integration in ein bestehendes System aber zwingend notwendig ist. Da das Open-Source-Prinzip im Digitalen Video Projekt einen großen Stellenwert besitzt und das System sich zum größten Teil aus im Quelltext verfügbaren Komponenten zusammen setzt, ist auch für die Integration von MHP-Anwendungen eine offene Lösung vorzuziehen.

Deshalb wird im weiteren Verlauf dieser Arbeit ein Konzept vorgestellt, das die Integration der Multimedia Home Platform in das Digitale Video Projekt unter Verwendung vorhandener MHP-Implementierungen erlauben soll. Das schließt folgende Funktionen ein:

- das Extrahieren der MHP-Applikationen aus dem DVB-Datenstrom,
- das Ausführen und Verwalten dieser Applikationen,
- die Steuerung der Anwendungen mittels Fernbedienung bzw. Tastatur,

- die Darstellung von deren Bildschirmausgaben auf dem Bildschirm

Das Ergebnis soll sich nahtlos in die modulare Architektur des vorhandenen Systems, also in die Software VDR, integrieren lassen. Des Weiteren soll es selbst einen modularen Aufbau besitzen und flexibel an sich ändernde Anforderungen anpassbar sein. Damit soll es eine Grundlage für darauf aufbauende Arbeiten darstellen, wie das beispielsweise die Entwicklung einer Open-Source MHP-Implementierung sein könnte.

4. Konzept für die Integration von MHP-Inhalten

In diesem Kapitel wird ein Konzept erarbeitet, das die Integration der Multimedia Home Platform in das System des Digitalen Video Projekts erlaubt, unter Berücksichtigung der im vorangegangenen Abschnitt erklärten Zielsetzung. Zu Beginn werden die zu realisierenden Fähigkeiten des zu erstellenden Systems und dessen Hard- und Softwarevoraussetzungen erarbeitet. Daran anschließend folgt die Modellierung der Merkmale aus diesen Anforderungen, wobei Abhängigkeiten und Einschränkungen ermittelt und erläutert werden. Nach diesem Schritt wird die Architektur des Systems entworfen. Dies beginnt mit der Zuordnung der ermittelten Merkmale auf die Pakete und Komponenten des Systems. Im Anschluss daran wird die Architektur des Gesamtsystems vorgestellt und wie sich diese in das Digitale Video Projekt integriert. Danach folgt die Erläuterung des Aufbaus und der Funktionsweise der einzelnen Pakete und Komponenten des Systems der MHP-DVP-Integration.

4.1. Anforderungsanalyse

4.1.1. Anwendungsfälle des MHP-Systems

In Kapitel 3.3 wurde bereits auf die Architektur und Eigenschaften der Multimedia Home Platform eingegangen. Daraus können direkt die Anwendungsfälle eines MHP-Systems abgeleitet werden. Diese sind in Abbildung 4.1 dargestellt und können in zwei Gruppen eingeteilt werden, die nutzerbezogenen und die systeminternen Anwendungsfälle. Während Erstere vom Nutzer des Systems ausgelöst oder wahrgenommen werden, repräsentiert der zweite Typ von Anwendungsfällen die vom System ausgeführten Aktionen, die im Hintergrund abgearbeitet werden und nicht direkt vom Nutzer beeinflussbar sind.

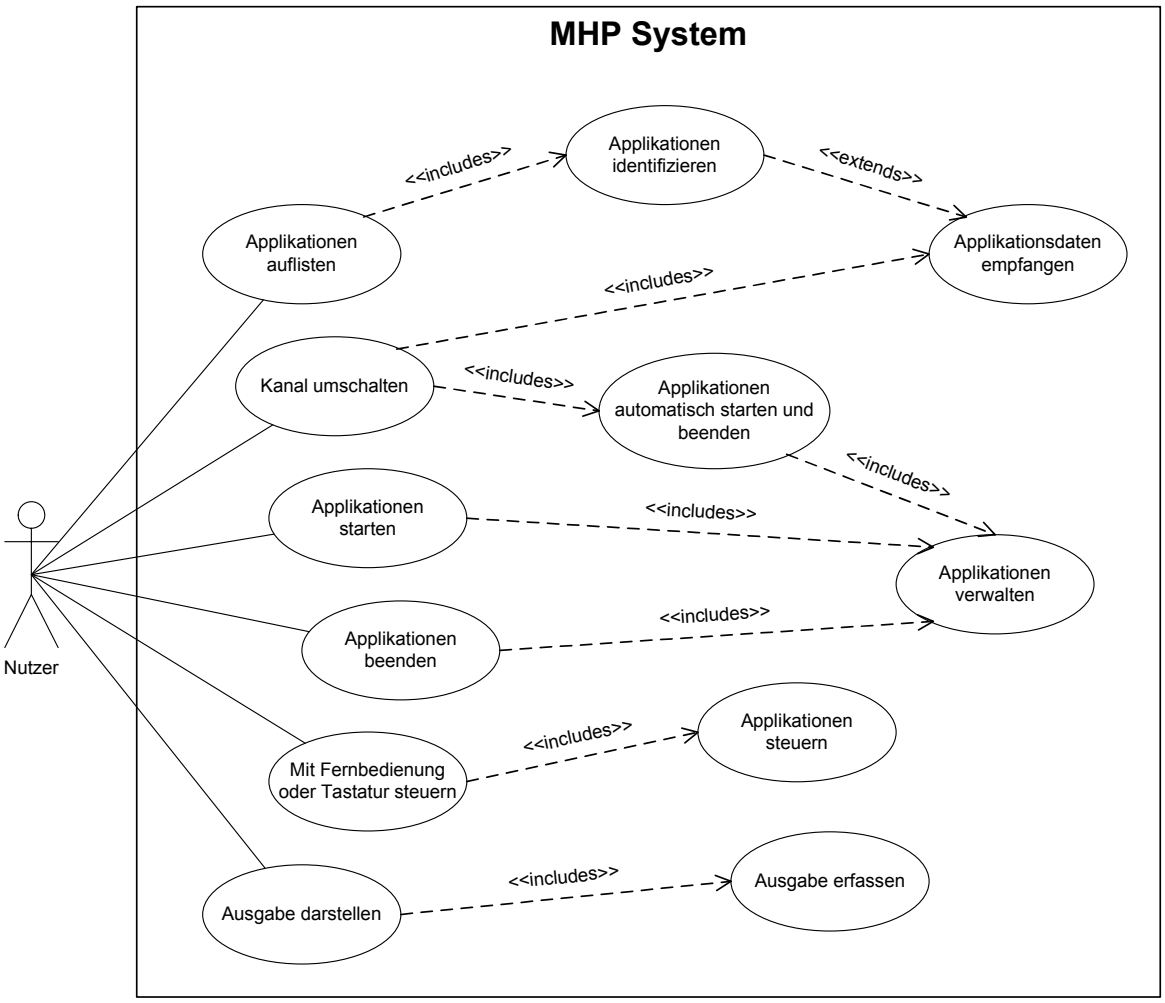


Abbildung 4.1.: Anwendungsfälle des MHP-Systems

Nutzerbezogene Anwendungsfälle

Dem Nutzer des ins Digitale Video Projekt integrierten MHP-Systems soll die Möglichkeit geboten werden, bestimmte Aktionen am System auszulösen und direkt über deren Ergebnisse informiert zu werden. Die für diesen Aspekt verantwortlichen Anwendungsfälle stellen die Schnittstelle des Systems zu dessen Nutzer dar. Sie werden im Folgenden näher erläutert.

Kanal umschalten: Es soll dem Anwender des Systems erlaubt sein den empfangenen Fernsehkanal zu wechseln. Falls zu diesem Zeitpunkt gerade eine oder mehrere MHP-Anwendungen ausgeführt werden, müssen diese gegebenenfalls vor dem Umschalten auf den neuen Kanal beendet werden. Nachdem auf das neue Programm gewechselt wurde und auf diesem gesendete MHP-Inhalte empfangen wurden, kann es erforderlich sein einige dieser Applikationen automatisch zu starten.

Applikationen auflisten: Dieser Anwendungsfall liefert dem Nutzer eine Liste der MHP-Applikationen, die im aktuellen Programm verfügbar sind und deren Ausführung im Moment erlaubt ist.

Applikationen starten: Neben dem automatischen Start der Anwendungen wird es dem Nutzer ermöglicht, per Knopfdruck die MHP-Applikationen manuell auszuführen.

Applikationen beenden: Dies erlaubt dem Anwender das Beenden von gerade ausgeführten MHP-Applikationen, die in der Regel auch vom Nutzer manuell gestartet wurden.

Mit Fernbedienung oder Tastatur steuern: Falls gerade eine MHP-Anwendung ausgeführt wird, muss es dem Nutzer möglich sein, deren Verhalten mittels Tastendrücken auf einer Fernbedienung oder Tastatur zu beeinflussen.

Ausgabe darstellen: Nahezu jede Applikation der MHP verfügt über grafische Ausgaben, die der Information über ihren Fortschritt und der Miteinbeziehung des Nutzers dienen. Diese Ausgaben müssen dem Anwender des Systems in irgendeiner Form präsentiert werden, was Aufgabe dieses Anwendungsfalls ist.

Systeminterne Anwendungsfälle

Neben den vom Anwender ausgelösten und von ihm wahrgenommenen Aktionen existieren im System Prozesse, die im Hintergrund ausgeführt werden und vom Nutzer nicht direkt beeinflusst werden können. Diese sind aber für die Abarbeitung der nutzerbezogenen Anwendungsfälle unbedingt erforderlich.

Applikationsdaten empfangen: Nach dem Umschalten auf einen neuen Kanal wird dessen Datenstrom auf das Vorhandensein von MHP-Inhalten untersucht. Falls MHP-Applikationen gefunden werden, werden deren Dateien aus dem Fernsehsignal extrahiert und lokal abgespeichert. Parallel dazu werden im Anwendungsfall *Applikationen identifizieren* die Namen und Parameter der Anwendungen ermittelt und für die spätere Verwendung abgespeichert.

Applikationen verwalten: Dieser Anwendungsfall umfasst das Laden, Starten und Beenden von MHP-Applikationen sowie das Überwachen ihrer Aktionen bei der Ausführung. Das Starten und Beenden kann dabei manuell vom Nutzer des Systems oder automatisch, z. B. bei einem Kanalwechsel, ausgelöst werden.

Applikationen steuern: Das Verhalten der Applikationen soll wie oben erklärt durch Tastendrücke des Nutzers beeinflusst werden können. Damit dies ermöglicht wird, müssen diese Ereignisse auf irgendeine Art und Weise an die MHP-Anwendung weitergeleitet werden, wofür dieser Anwendungs-

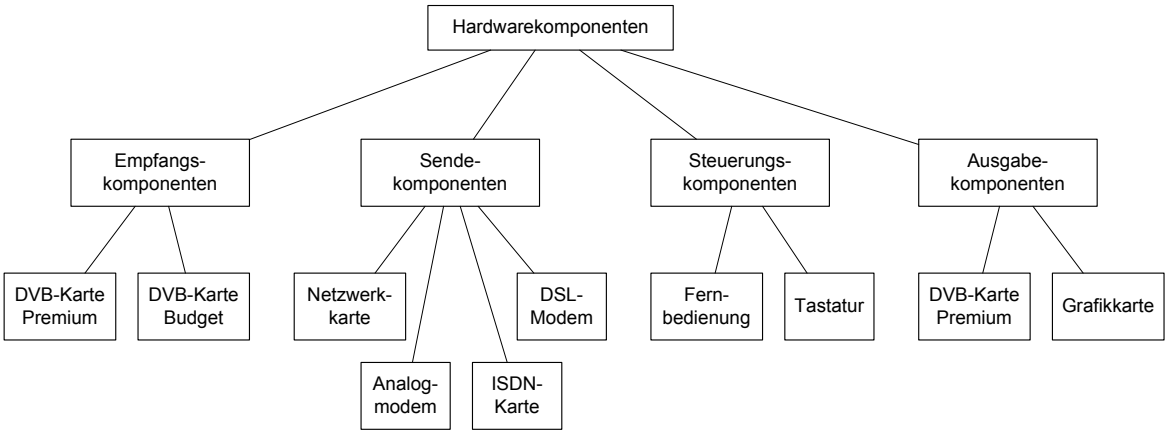


Abbildung 4.2.: Hardwarekomponenten des MHP-Systems

fall verantwortlich ist.

Ausgabe erfassen: Bevor die grafische Ausgabe der MHP-Applikationen dem Nutzer zur Verfügung gestellt werden kann, muss sie erst durch diesen Prozess ermittelt werden.

4.1.2. Anforderungen an die Hardware

Die Integration der Multimedia Home Platform in eine linuxbasierte Standard-PC-Plattform, wie sie das Digitale Video Projekt darstellt, erfordert neben einem Rechner der mittleren Leistungsklasse eine ganze Reihe von Hardwarekomponenten. Diese sind in Abbildung 4.2 dargestellt, wobei die meisten dieser Komponenten im Grundsystem des DVP schon enthalten sind.

Für den Empfang des Fernsehsignals und damit auch der MHP-Applikationen wird eine *Empfangskomponente* in Form einer DVB-PCI-Karte benötigt. Diese muss nicht unbedingt über einen MPEG-2-Decoder wie die „Premium“-Modelle verfügen, eine so genannte „Budget“-Karte genügt vollkommen. Im DVP sind sogar beide Typen in der Variante für den Satellitenempfang enthalten.

Falls die zu integrierende MHP-Umgebung das „Interactive Broadcast“-Profil und damit die Übermittlung von Daten zum Sender über einen Rückkanal unterstützen soll, ist mindestens eine der *Sendekomponenten* erforderlich, über die eine Verbindung auf Basis des Internet Protokolls hergestellt wird. Dafür kommen Analogmodem, ISDN-Karte, DSL-Modem oder Netzwerkkarte in Frage, weitere Typen sind möglich. Das System des Digitalen Video Projekts besitzt eine Netzwerkkarte für diese Zwecke.

Damit eine Beeinflussung des Verhaltens des MHP-Systems und seiner Applikationen ermöglicht wird, muss dieses über eine *Steuerungskomponente* verfügen, wie sie Infrarot-Fernbedienung und

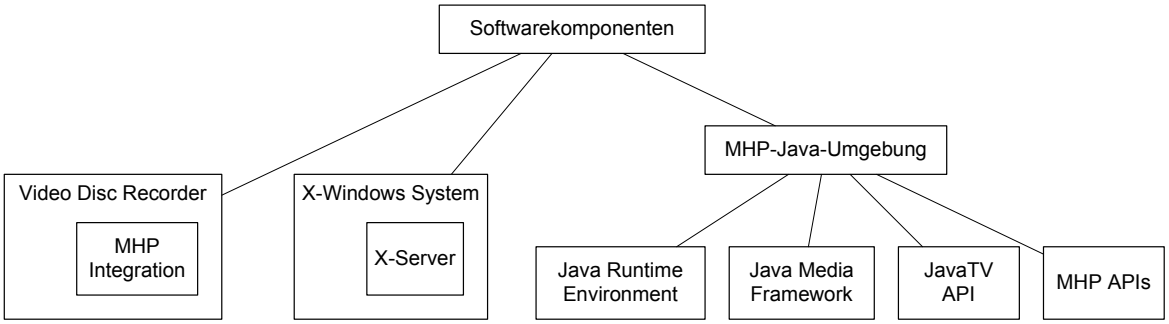


Abbildung 4.3.: Softwarekomponenten des MHP-Systems

Tastatur darstellen. Das DVP-System umfasst beide dieser Komponenten.

Die Darstellung der grafischen Ausgabe der MHP-Anwendungen auf dem Fernseher erfolgt über die *Ausgabekomponente* des Systems. Dabei kann es sich zum einen um eine Grafikkarte mit TV-Ausgang handeln. Diese Art der Darstellung bietet nahezu unbegrenzte grafische Möglichkeiten, wird aber von der zugrunde liegenden Software VDR nicht als Ausgabegerät unterstützt. Die deshalb im weiteren Verlauf der Arbeit vorzuziehende Lösung ist die Darstellung der MHP-Applikationen über den Videoausgang einer DVB-Karte in der Premium-Ausführung. Diese Variante bringt aber einige Beschränkungen mit sich, die bei der Merkmalserfassung im Detail analysiert werden.

4.1.3. Anforderungen an die Software

Neben der Hardware sind für das zu entwickelnde MHP-System einige Softwarekomponenten erforderlich, welche in Abbildung 4.3 aufgeführt werden. Da ist als erstes die Software *Video Disc Recorder* zu nennen, die die Basis des Digitalen Video Projekts darstellt und damit schon vorhanden ist. Allerdings ist für die Integration der MHP eine Erweiterung des VDR in Form eines Plugins vonnöten.

Für die Darstellung der grafischen Ausgabe von Java-Anwendungen und damit auch von MHP-Applikationen kommt unter Linux die grafische Oberfläche *X-Windows* zum Einsatz, welche auf einer Client/Server-Architektur basiert. Der so genannte X-Server stellt allen Programmen (Clients) eine Schnittstelle zur Verfügung, um z. B. Buchstaben, Pixel, Linien, Flächen, Fenster usw. auf den Bildschirm zu bringen. Diese Schnittstelle ist systemunabhängig definiert und netzwerkfähig. Unter Linux kommt die freie Variante von X-Windows, XFree86, zum Einsatz.

Die wohl wichtigste Softwarekomponente ist die *MHP-Java-Umgebung*, die für die Ausführung von DVB-J-Applikationen der MHP unbedingt erforderlich ist. Der detaillierte Umfang und Aufbau

der Komponenten der MHP-Java-Umgebung ist in Anhang A angegeben. Sie setzt sich aus folgenden Teilkomponenten zusammen:

- das Java Runtime Environment (JRE),
- das Java Media Framework (JMF),
- das JavaTV API,
- die MHP APIs.

Das JRE, die Java Laufzeitumgebung, wird für die Ausführung von Java-Programmen grundsätzlich immer benötigt. Das für die MHP vorausgesetzte JRE muss mindestens in der Version 1.1.8 vorliegen. Die Java Laufzeitumgebung wird von verschiedenen Unternehmen in diversen Varianten angeboten. Die jeweils aktuellste und den anderen Herstellern als Grundlage dienende Ausgabe des JRE kommt natürlich von der Firma Sun, aus deren Feder die Programmiersprache stammt und die sich für deren Weiterentwicklung verantwortlich zeigt. Die von den anderen Unternehmen angebotenen Varianten der Laufzeitumgebung bieten oftmals im Bereich der Geschwindigkeit bzw. des Speicherbedarfs einige Vorteile gegenüber der originalen Ausgabe von Sun. Allerdings hinken diese Abwandlungen des Originals meist einige Versionen hinterher, was aber für die MHP nicht relevant ist, da hierfür die schon einige Jahre alte Version 1.1.8 vollkommen ausreichend ist. Als ein Beispiel sei hier die Implementationen des JRE von IBM genannt.

Für die Ausführung von Applikationen der Multimedia Home Platform werden zusätzlich zur Laufzeitumgebung noch einige andere Java Pakete benötigt. Dazu zählen das Java Media Framework (JMF) und das JavaTV API. Beide Erweiterungen wurden von Sun entwickelt und werden auf deren Webseite angeboten. Das API des JMF, welches mindestens in der Version 1.0 vorliegen muss, dient der Präsentation und Steuerung von Audio- und Videodaten sowie anderer zeitbasierter Medien. JavaTV stellt Schnittstellen und Klassen für die allgemeinen Elemente einer Plattform für interaktives Fernsehen bereit, unabhängig von der Art der Übertragung.

Die wohl wichtigste Erweiterungen des Java Grundsystems, die für die Multimedia Home Platform benötigt werden, stellen die APIs der MHP-Schnittstelle dar: DAVIC für die Ressourcenverwaltung und eine erweiterte Unterstützung der MPEG- und DVB-Konzepte, das API der HAVi Level 2 GUI für das Ansprechen des MHP-Grafikmodells und die Ereignisverwaltung sowie die DVB-APIs, die DVB-spezifische Erweiterungen beinhalten. Während die ersten drei Komponenten der MHP-Java-Umgebung frei verfügbar sind und sogar den Zugriff auf deren Quelltext zulassen, existieren von

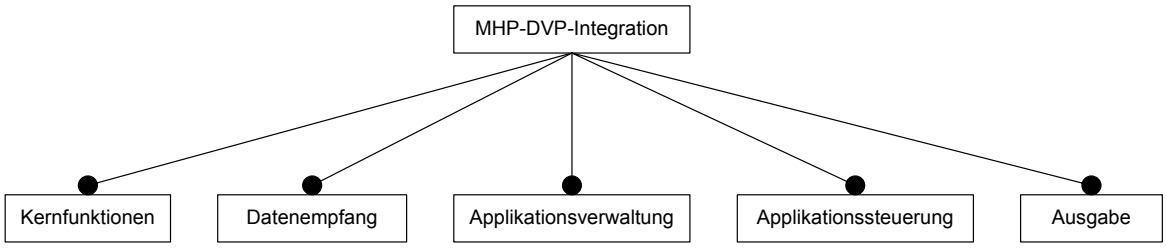


Abbildung 4.4.: Merkmale der MHP-DVP-Integration

den MHP APIs zum Zeitpunkt dieser Diplomarbeit keine kostenlosen Varianten. Diverse Hersteller bieten die so genannte MHP-Middleware für die verschiedensten Systeme, auch für Linux, gegen entsprechende Lizenzgebühren an. Die meisten dieser Produkte für die PC-Plattform basieren auf der Referenzimplementierung des Instituts für Rundfunktechnik (IRT), welche eine hundertprozentige Umsetzung des MHP-Standards darstellt.

4.2. Merkmalserfassung

Im vorhergehenden Abschnitt wurden die Fähigkeiten des zu entwickelnden Systems in Form von Anwendungsfällen dargelegt. Des Weiteren wurden die Voraussetzungen beschrieben, die an die Hard- und Softwareumgebung gestellt werden. Aus diesen Anforderungen werden nun die Merkmale und Submerkmale des Systems der MHP-DVP-Integration modelliert und außerdem wird festgelegt welche dieser Merkmale obligatorisch sind und an welchen Stellen sich Variabilitäten ergeben. Des Weiteren werden Parameter ermittelt, die die durch die Merkmale repräsentierten Funktionalitäten beeinflussen. Das Gesamtmodell, das aus den folgenden Überlegungen entsteht, ergibt sich durch die Zusammenfassung der Merkmaldiagramme in den Abbildungen 4.4 bis 4.9. Die Notation dieser Diagramme wurde dem Buch „Generative Programming“ von Czarnecki/Eisenecker entnommen [3], zusätzlich werden die Parameter als gestrichelte Rechtecke eingefügt.

Die Abbildung 4.4 fasst die obligatorischen Merkmale der MHP-DVP-Integration zusammen. Diese besitzen jeweils einige Submerkmale und Besonderheiten, die neben ihrer Funktionalität in den folgenden Abschnitten erläutert werden.

4.2.1. Das Merkmal Kernfunktionen

Als *Kernfunktionen* werden die grundlegenden Prozesse und Aspekte des Systems bezeichnet, die für die anderen Hauptmerkmale vorausgesetzt werden. In Abbildung 4.5 sind diese Prozesse als

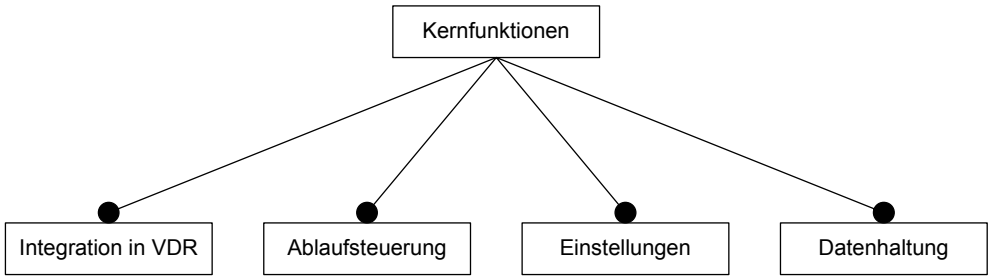


Abbildung 4.5.: Submerkmale des Merkmals Kernfunktionen

Submerkmale der Kernfunktionen zusammengefasst.

Als erstes Merkmal ist hierbei die *Integration* des Systems *in den Video Disc Recorder* zu nennen, was unter Nutzung von dessen Plugin-Schnittstelle geschieht. Das Merkmal sorgt zum einen dafür, dass die MHP-Inhalte vom Menüsystem des VDR aus abrufbar sind. Dies umfasst eine Liste der momentan verfügbaren Applikationen. Des Weiteren muss das System von bestimmten Ereignissen, wie z. B. das Umschalten auf einen neuen Kanal, in Kenntnis gesetzt werden.

Unter der *Ablaufsteuerung* wird die Steuerung der zeitlichen Abfolge der durch die anderen Merkmale angebotenen Funktionalitäten verstanden. Dies umfasst unter anderem die wiederholte Ausführung der Ausgabe und das Starten und Stoppen des Datenempfangs als Reaktion auf bestimmte Ereignisse.

Die Speicherung von Informationen über die im Fernsehsignal übertragenen MHP-Applikationen und deren Daten wird als *Datenhaltung* verstanden. Diese Informationen werden beispielsweise von der *Applikationsverwaltung* verwendet um festzustellen, ob ein automatischer Start einer Anwendung nötig ist.

Die meisten Merkmale der MHP-DVP-Integration sind durch Parameter beeinflussbar. Deren aktuelle Werte liegen in internen Datenstrukturen gespeichert vor und können über die Menüstruktur des VDR verändert werden. Dieser Aspekt des Systems wird unter dem Merkmal *Einstellungen* zusammengefasst.

4.2.2. Das Merkmal Datenempfang

Unter dem Merkmal *Datenempfang* werden alle Aktivitäten des Systems zusammengefasst, die für das Extrahieren und das Speichern von Informationen über die MHP-Applikationen sowie deren Dateien verantwortlich sind. Diese sind als Submerkmale in Abbildung 4.6 dargestellt.

Der Datenstrom, der die MHP-Applikationen enthält, liegt als Transportstrom vor. Dieser wird

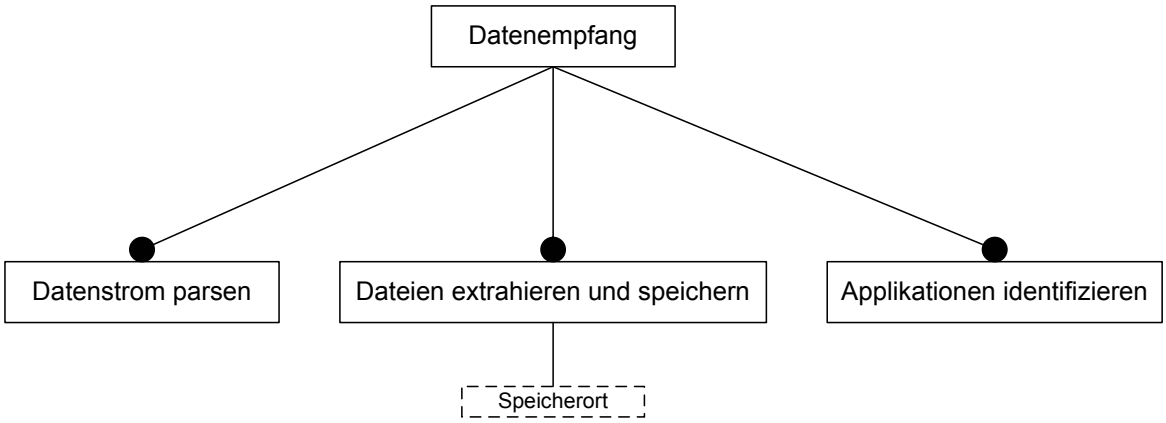


Abbildung 4.6.: Submerkmale des Merkmals Datenempfang

in MPEG-Sections zerlegt, aus denen dann die Blöcke und Module des Datenkarussells gebildet werden. Aus den Modulen werden im letzten Schritt die Objekte des Object Carousel gewonnen. Der gesamte Vorgang der Wandlung vom Transportstrom zu den Objekten wird mit dem Merkmal *Datenstrom parsen* beschrieben.

Die Nachrichten des Objektkarussells bilden die Dateien und Verzeichnisse der MHP-Applikation. Diese werden aus dem Datenstrom heraus gelöst, entsprechend ihrer Object-Keys zu einer Baumstruktur angeordnet und schließlich auf der Festplatte des PCs in dem durch den Parameter *Speicherort* festgelegten Verzeichnis abgespeichert. Diese Aktionen werden unter dem Merkmal *Dateien extrahieren und speichern* zusammengefasst.

Das letzte Submerkmal des Datenempfangs ist *Applikationen identifizieren*. Dies beinhaltet das Analysieren der Einträge der Application Information Table, aus denen Informationen über die verfügbaren MHP-Anwendungen, wie deren Name und Parameter, gewonnen und in den Strukturen der Datenhaltung gespeichert werden.

Der gesamte Vorgang des Datenempfang hat zyklisch zu erfolgen. Damit wird sichergestellt, dass sich ändernde oder neue Daten sofort empfangen und den Applikationen zur Verfügung gestellt werden können. Dieser Vorgang des Aktualisierens wird beispielsweise für Nachrichten- oder Börsenticker benötigt.

4.2.3. Das Merkmal Applikationsverwaltung

Die in Abbildung 4.7 dargestellte *Applikationsverwaltung* umfasst alle Aktivitäten des Systems, die für das Starten, Stoppen und für Zuständsänderungen während der Ausführung von MHP-

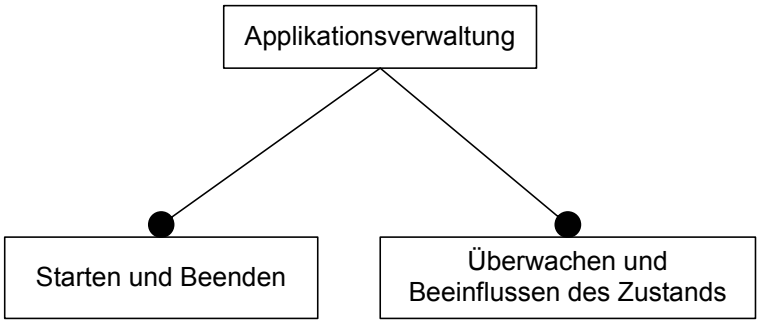


Abbildung 4.7.: Submerkmale des Merkmals Applikationsverwaltung

Anwendungen benötigt werden. Diese durch den Application Manager vorgenommen Aktionen werden im Folgenden beschrieben.

Beim Umschalten auf einen neuen Kanal kann es nötig sein eine momentan ausgeführte MHP-Applikation automatisch zu beenden oder eine gerade empfangene Anwendung automatisch zu starten, wenn dies in den Einträgen der AIT gefordert wird. Des Weiteren muss der manuelle Start durch Auswahl aus einer Liste verfügbarer Applikationen sowie das manuelle Beenden durch Knopfdruck ermöglicht werden. Alle diese Aktivitäten werden im Merkmal *Starten und Beenden* zusammengefasst.

Während eine MHP-Applikation ausgeführt wird, können Zustandsänderungen nötig werden, seien sie durch Aktionen des Nutzers oder durch Änderungen im Laufzeitverhalten hervorgerufen. Diese werden vom Application Manager über die Methoden der Xlet-Schnittstelle vorgenommen. Außerdem kann die Anwendung Parameter besitzen, deren Werte sie dann vom Application Manager über eine festgelegte Schnittstelle erfragen muss. Diese Aktionen der Applikationsverwaltung werden mit dem Merkmal *Überwachung und Beeinflussen des Zustands* beschrieben.

4.2.4. Das Merkmal Applikationssteuerung

Um das Verhalten der MHP-Applikationen während deren Ausführung beeinflussen zu können muss das System Funktionalitäten besitzen, welche die vom Nutzer vorgenommene Betätigung von Tasten auf einer Steuerungskomponente wahrnehmen und diese an die Anwendung weiterleiten. Diese Vorgänge werden mit dem Merkmal *Applikationssteuerung* beschrieben, welches mit seinen Submerkmalen in Abbildung 4.8 gezeigt wird.

Das Wahrnehmen der Steuersignale wird in dem Merkmal *Tastendruck erfassen* zusammengefasst. Dieser Vorgang wird zum größten Teil von internen Prozessen des Video Disc Recorder übernom-

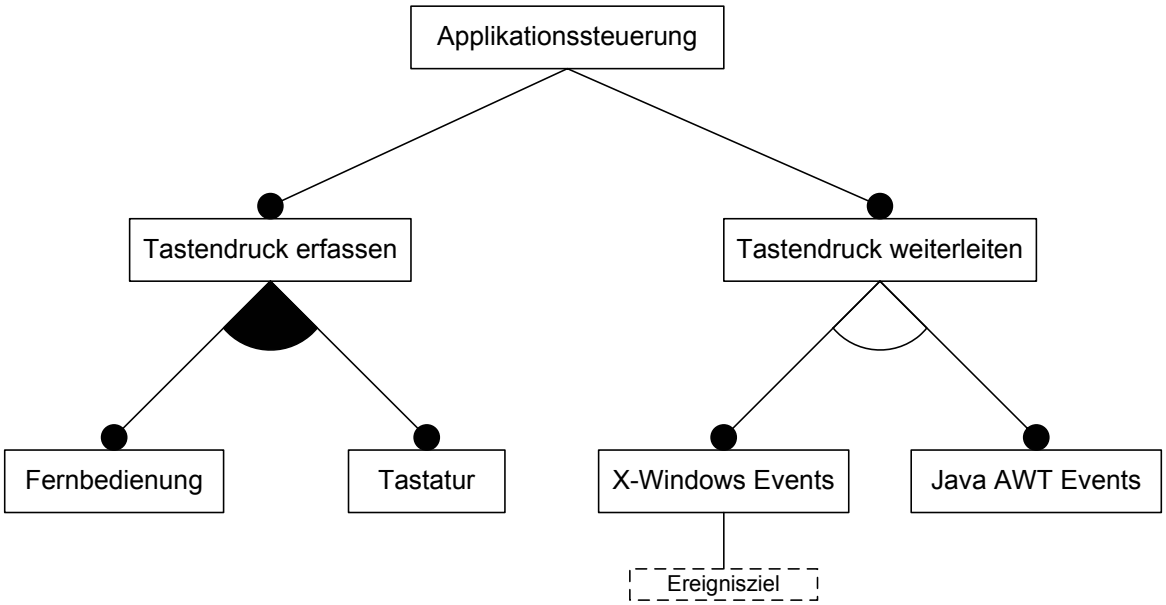


Abbildung 4.8.: Submerkmale des Merkmals Applikationssteuerung

men. Als Quelle für die Signale kann eine Infrarotfernbedienung, eine Tastatur oder jede weitere durch die cRemote-Schnittstelle des VDR angebundene Steuerungskomponente dienen. Dem System werden die Codes dieser Signale über einen durch VDR vorgenommenen Funktionsaufruf mitgeteilt.

Damit eine Beeinflussung des Verhaltens der gerade ausgeführten MHP-Applikation stattfinden kann, müssen die erfassten Steuersignale in irgendeiner Art und Weise an diese gesendet werden. Diesen Vorgang repräsentiert das Merkmal *Tastendruck weiterleiten*. Das Senden der Tastencodes kann in zwei verschiedenen Varianten erfolgen, weitere sind aber denkbar:

- *X-Windows Events*: In diesem Fall erfolgt das Senden der Tastencodes in Form von Ereignissen des X-Windows Systems entweder an den X-Server, in dem die Ausgabe der MHP-Applikation dargestellt wird, oder direkt an das Fenster, das deren Ausgabe enthält. Wie und wohin das Ereignis gesendet wird, bestimmt der Parameter *Ereignisziel*.

- *Java AWT Events*: Bei dieser Variante sendet der in der Java VM laufende Application Manager die Tastencodes als AWT Ereignisse an das Fenster der MHP-Anwendung. Vorher muss der Video Disc Recorder, bzw. der dafür zuständige Teil des Plugins, die Signale an den Applikation Manager übermittelt haben, was beispielsweise über eine Netzwerk- oder Shared-Memory-Verbindung geschehen kann.

Welche Variante für die Weiterleitung der Tastensignale genutzt werden kann, ist größtenteils

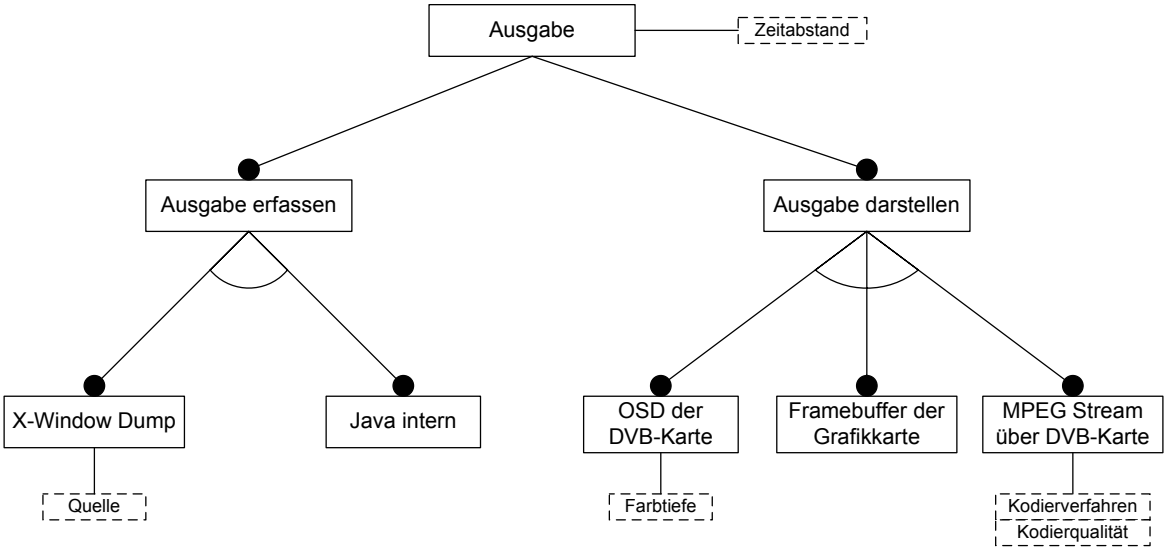


Abbildung 4.9.: Submerkmale des Merkmals Ausgabe

abhängig von den Softwarevoraussetzungen. Nur wenn die Ausgabe der Applikationen durch die MHP-Java-Umgebung mittels X-Windows erfolgt, ist die erste Variante möglich. Dagegen erfordert der zweite Fall unter Umständen kleine Modifikationen an der MHP-Java-Umgebung, so dass Teile davon im Quelltext vorliegen müssen.

4.2.5. Das Merkmal Ausgabe

Damit der Nutzer überhaupt etwas von den MHP-Applikationen zu sehen bekommt, sind einige Prozesse erforderlich, die mit *Ausgabe* beschrieben werden. Das Merkmal steht für den Vorgang, bei dem der grafische Output einer Anwendung dem Nutzer auf dem Bildschirm präsentiert wird. Dieser Prozess wird bei der Ausführung einer MHP-Applikation ständig in einem festen Zeitabstand wiederholt durchgeführt, der durch den gleichnamigen Parameter bestimmt wird. Die Ausgabe wird durch zwei Teilprozesse vorgenommen, welche durch die Submerkmale *Ausgabe erfassen* und *Ausgabe darstellen* repräsentiert werden, die in Abbildung 4.9 aufgeführt sind.

Der Prozess des Erfassens der Ausgabe ermittelt den grafischen Output einer MHP-Applikation und stellt ihn als Bild im RGBA-Format zur Verfügung. Dies kann auf zwei verschiedene Arten erfolgen:

- *X-Window Dump:* In diesem Fall wird die mittels X-Windows dargestellte Ausgabe der MHP-Applikation erfasst, indem ein Speicherabzug auf dem Framebuffer des X-Servers durchge-

führt wird. Dies erfolgt über den Aufruf der Funktion XGetImage des X-Windows-API, die, festgelegt durch den Paramter *Quelle*, entweder den Inhalt des kompletten Bildschirms oder den eines bestimmten Fensters im XWD-Format zurück liefert.

- *Java intern:* Eine weitere Variante ist die java-interne Ausgabeerfassung. Hier wird der grafische Output der Applikation innerhalb der MHP-Java-Umgebung ermittelt. Dies geschieht durch Methoden der Programmiersprache Java, deren Ergebnis in einem internen Bildformat bereitgestellt wird. Die Pixeldaten müssen dann noch an das VDR-Plugin übermittelt werden, was z. B. über eine lokale Netzwerk- oder Shared-Memory-Verbindung geschehen kann.

Beide Varianten erfordern gegebenenfalls eine Konvertierung der Pixeldaten in das RGBA-Format, damit diese problemlos von der Ausgabedarstellung weiterverarbeitet werden können. Welche Variante schließlich für die Ausgabeerfassung in Frage kommt, hängt von den gleichen Voraussetzungen ab, die schon für die Wahl der Applikationssteuerung zugrunde lagen.

Bei der Darstellung der Ausgabe werden die im RGBA-Format vorliegenden Pixeldaten auf dem Fernseher präsentiert, damit diese vom Nutzer dem Systems wahrgenommen werden können. Auch für diesen Prozess existieren verschiedene Varianten:

- *OSD der DVB-Karte:* Der DSP, der auf den PCI-DVB-Karten zu finden ist, erlaubt die Überblendung des Videosignals mit einem On-Screen-Display. Dies kann pixelgenau angesprochen werden und eignet sich deshalb prinzipiell gut für die Darstellung der MHP-Ausgabe. Jedoch ist das OSD der DVB-Karten einigen Beschränkungen unterworfen. Zum einen können maximal 256 verschiedene Farben gleichzeitig dargestellt werden, was zeitraubende Farbreduktionen notwendig macht. Des Weiteren ist die Größe des On-Screen-Displays durch den vorhandenen Speicher auf der DVB-Karte beschränkt, welcher auf den aktuellen Modellen 2 MB groß ist, wovon aber nur etwa 90 KB dem OSD zur Verfügung stehen. Damit kann bei 256 darstellbaren Farben lediglich ein etwa 350x250 Pixel großer Bereich angesprochen werden, was noch nicht mal einem Viertel des gesamten Bildschirms entspricht. Bei verringerter Anzahl der Farben ist ein entsprechend größerer Bereich möglich, allerdings verschlechtert sich dann die Qualität der Darstellung der MHP-Applikation. Die gewünschte Farbtiefe und damit auch die maximale Größe des Bereiches wird mit dem gleichnamigen Parameter festgelegt.

- *MPEG-Stream über DVB-Karte:* Um die oben genannten Beschränkungen des On-Screen-Displays zu umgehen kann für die Ausgabe der MHP das Video-Device der DVB-Karte ver-

wendet werden. Diesem können MPEG-kodierte Videoströme oder als I-Frames vorliegende Einzelbilder übergeben werden, welche von dem DSP der Karte dekodiert und direkt auf dem Fernseher dargestellt werden. Dabei müssen keine Beschränkungen von Farbtiefe oder Auflösung hingenommen werden, die Ausgabe der MHP-Applikationen kann bildschirmfüllend erfolgen. Diese Variante besitzt nur einen Nachteil: Für die Kodierung der Pixeldaten nach dem MPEG-Verfahren ist eine bestimmte Rechenleistung vonnöten, was aber auf heutigen PCs kein Problem mehr darstellt. Abhängig von der Leistung des Systems sollte das Kodierverfahren (MPEG-1 oder MPEG-2) und die Kodierqualität gewählt und mit den gleichnamigen Parametern festgelegt werden.

- *Framebuffer der Grafikkarte:* Die umfassendsten Möglichkeiten bietet die Darstellung der MHP-Applikationen über die Grafikkarte des PCs. Über einen Video-Ausgang, den viele Modelle inzwischen besitzen, kann das Bild auch auf einem Fernseher ausgegeben werden. Diese Variante ist wie die MPEG-Ausgabe keinen Beschränkungen unterworfen und erfordert auch keine zusätzliche Rechenleistung. Somit ist die Grafikkarte prinzipiell die beste Variante, wenn es nur um die Darstellung von MHP-Anwendungen geht. Allerdings existiert im API des Video Disc Recorder keine Möglichkeit die Grafikkarte direkt anzusprechen. Dies wäre nur über eine Integration als generelles Ausgabegerät über die cDevice-Schnittstelle möglich. Dann müsste die Grafikkarte aber alle Ausgabefunktionen der DVB-Karte übernehmen, wozu auch das Dekodieren des Fernsehsignals gehört, was wieder eine zusätzliche Rechenbelastung darstellt.

Die Wahl der zu bevorzugenden Variante der Ausgabedarstellung hängt zum großen Teil von den Möglichkeiten der Hardware ab. Zum Zeitpunkt dieser Diplomarbeit lässt der Entwicklungsstand des VDR-Projekts jedoch ohne umfangreiche Erweiterungen nur eine Nutzung der DVB-Karte als Ausgabegerät zu, weshalb im Folgenden diese beiden Varianten vorrangig behandelt werden.

4.3. Entwurf der MHP-DVP-Integration

In den Kapiteln 4.1 und 4.2 wurden die Anforderungen erfasst und die Merkmale des Systems der MHP-DVP-Integration erarbeitet. Im Folgenden wird eine Architektur entworfen, die die Realisierung der dort formulierten Fähigkeiten und Funktionalitäten ermöglicht.

Der Entwurf der Architektur wird in mehreren Schritten erfolgen. Als erstes wird festgestellt, aus welchen Komponenten sich das System zusammensetzen soll und wie die einzelnen Merkmale

darauf abgebildet werden. Daran anschließend wird die Gesamtarchitektur des Systems vorgestellt und erklärt, wie sich diese in die Rechnerarchitektur des Digitalen Video Projekts integriert. Der abschließende dritte Teil des Architekturentwurfs ist sogleich der umfangreichste und beinhaltet die detaillierte Vorstellung der einzelnen Pakete des Systems bis hinunter auf Klassenebene. Es wird untersucht, wie die Komponenten miteinander kommunizieren und welche Datenflüsse zwischen ihnen auftreten. Des Weiteren werden die Prozesse und Aktivitäten betrachtet, die innerhalb der Komponenten stattfinden.

Für die Bezeichner von Paketen, Modulen, und Klassen sowie deren Attribute und Methoden werden im Folgenden ausschließlich englische Begriffe verwendet.

4.4. Abbildung der Merkmale auf Pakete und Module

Die in Kapitel 4.2 ermittelten Merkmale der MHP-DVP-Integration werden nun so auf Pakete und Module des Systems abgebildet, dass eine Architektur entsteht, die flexibel an sich ändernde Anforderungen angepasst werden kann. Des Weiteren soll eine intuitive Möglichkeit geschaffen werden, die in den Merkmalen *Applikationssteuerung* und *Ausgabe* vorhandenen Variabilitäten zur Compile- bzw. Laufzeit des Systems auflösen zu können.

Die Zuordnung der Merkmale auf Pakete und Module des Systems ist in Abbildung 4.10 angegeben. Darin fällt auf, dass nicht jedem Merkmal des Merkmalmodells eine Komponente zugeordnet wird. Das liegt zum einen daran, dass einige der Merkmale nur Konzepte beschreiben, die das System berücksichtigen muss, die aber keiner Aktivität entsprechen. Dies gilt beispielsweise für die *Integration in VDR*, die durch die Wahl der richtigen Basisklassen und die Nutzung der von VDR angebotenen Schnittstellen sichergestellt wird. Andere Merkmale, wie z. B. *Tastendruck erfassen* werden vom Video Disc Recorder selbst erledigt und können automatisch durch die Integration des Systems in VDR erfüllt werden. Zu guter Letzt kann eine Komponente die Funktionalitäten mehrerer Merkmale in sich vereinen.

Jedes der direkten Merkmale der MHP-DVP-Integration[1] wird auf genau ein Paket des Systems abgebildet:

- *Kernfunktionen* auf *CoreFunctions*,
- *Datenempfang* auf *DataReceiving*,

[1] vgl. Abbildung 4.4

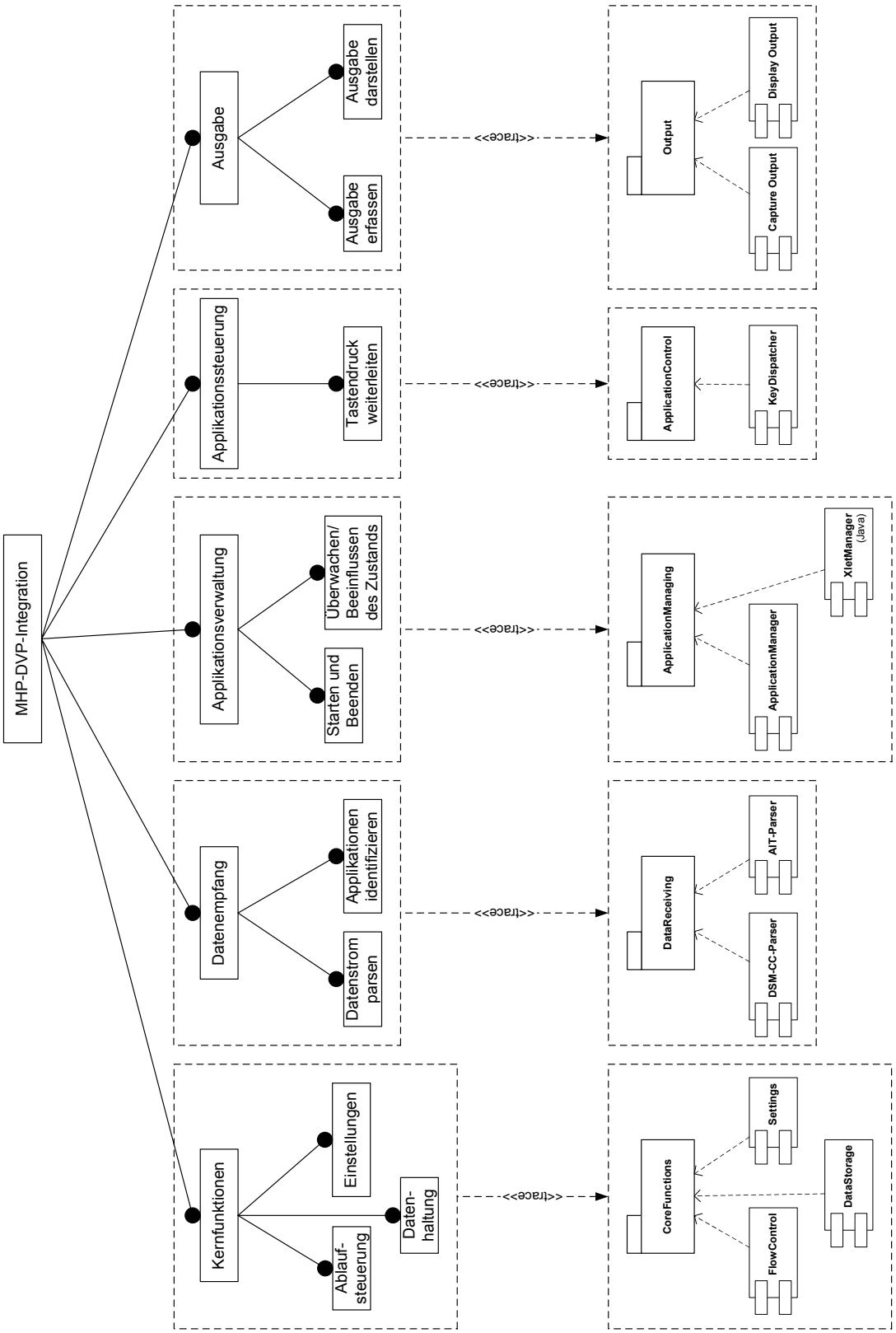


Abbildung 4.10.: Abbildung der Merkmale auf Pakete

- *Applikationsverwaltung* auf *ApplicationManaging*,
- *Applikationssteuerung* auf *ApplicationControl*,
- *Ausgabe* auf *Output*.

Jedes dieser Pakete besitzt noch ein oder mehrere Module, welche die Submerkmale der Hauptmerkmale des Systems repräsentieren und deren Zuordnung im Folgenden vorgenommen wird.

Die drei Submerkmale der *Kernfunktionen* können direkt durch die Module des Pakets *CoreFunctions* umgesetzt werden:

- *FlowControl* (repräsentiert das Merkmal *Ablaufsteuerung*),
- *DataStorage* (stellt die Funktionalitäten der *Datenhaltung* bereit),
- *Settings* (repräsentiert die *Einstellungen* des Systems).

Von den drei Untermerkmalen des *Datenempfangs* können zwei direkt durch Module des Pakets *DataReceiving* repräsentiert werden: Der *DSM-CC-Parser* übernimmt die Aufgabe des Merkmals *Datenstrom parsen*, der *AIT-Parser* stellt eine Umsetzung des zweiten Submerkmals, *Applikationen identifizieren*, dar. Die Funktionalitäten des verbliebenen Merkmals *Dateien extrahieren und abspeichern* werden teilweise vom *DSM-CC-Parser* übernommen, der weitere Teil der Abarbeitung wird zwar von diesem ausgelöst, findet aber im Modul *DataStorage* des Pakets *CoreFunctions* statt.

Die Umsetzung der Submerkmale der *Applikationsverwaltung* kann nicht direkt erfolgen. Deren Funktionalitäten werden zusammengefasst und von zwei Modulen des Paketes gemeinsam übernommen, dem *ApplicationManager* und dem *XletManager* Während der *ApplicationManager* innerhalb des VDR-Plugins realisiert wird, stellt der *XletManager* einen Teil der MHP-Java-Umgebung dar. Zwischen den beiden Modulen besteht eine ständige Verbindung, über welche diese miteinander kommunizieren können.

Das Submerkmal *Tastendruck erfassen* der *Applikationssteuerung* wird nicht in einem eigenen Modul des Pakets *ApplicationControl* realisiert. Mit der Integration des Systems in VDR durch das Ableiten einer Basisklasse wird das Paket automatisch über Tastendrücke von Fernbedienung oder Tastatur in Kenntnis gesetzt und muss diese nur noch weiterleiten. Dies geschieht durch den *KeyDispatcher*, welcher das Merkmal *Tastendruck weiterleiten* direkt repräsentiert. Die Varianten des Sendens der Tastensignale an die MHP-Applikation werden innerhalb des Moduls auf Klassenebene realisiert. Wie dies geschieht wird im weiteren Verlauf dieser Arbeit noch betrachtet.

Die zwei Submerkmale der *Ausgabe* können wieder direkt auf Module des Pakets *Output* abgebildet werden:

- *Capture Output* (repräsentiert das Merkmal *Ausgabe erfassen*) und
- *Display Output* (repräsentiert das Merkmal *Ausgabe darstellen*).

Wie beim Modul *KeyDispatcher* werden auch hier die Varianten der Erfassung und Darstellung der Ausgabe innerhalb der entsprechenden Module realisiert, worauf das Kapitel 4.6 näher eingeht.

4.5. Architektur des Gesamtsystems

Die im vorangegangenen Abschnitt ermittelten Pakete und Module der MHP-DVP-Integration bilden zusammen mit weiteren Komponenten die Gesamtarchitektur des Systems, die in Abbildung 4.11 dargestellt ist. Zwischen den Komponenten des Systems findet ein ständiger Datenaustausch statt.

Wie in der Abbildung zu erkennen ist, werden die meisten Pakete und Module der MHP-DVP-Integration innerhalb des Video Disc Recorders realisiert, wobei von dessen Plugin-Schnittstelle Gebrauch gemacht wird. Diese Komponenten können mit Hilfe der vom VDR angebotenen Klassen und Schnittstellen die Hardwarekomponenten des Rechners beeinflussen bzw. von denen eingehende Daten erhalten. Des Weiteren findet ein Datenaustausch zwischen den Paketen des VDR-Plugins und den Modulen der anderen Softwarekomponenten des Systems statt, was im Falle der MHP-Java-Umgebung über den zur MHP-DVP-Integration gehörenden *XletManager* erfolgt. Dagegen findet die Kommunikation mit dem X-Windows System direkt über dessen API statt. Der Datenfluss zwischen diesen beiden Softwarekomponenten, der die grafische Ausgabe der MHP-Applikation und die Ereignisse des X-Windows Systems repräsentiert, erfolgt unbeeinflusst von den Paketen des VDR-Plugins.

Die Details der Datenflüsse zwischen den Komponenten werden in Kapitel 4.6 noch einmal näher betrachtet.

Erweiterung der DVP-Rechnerarchitektur

Der Rechner des Digitalen Video Projekts muss über bestimmte Hardware- und Softwarekomponenten verfügen, damit er seine Funktion als digitaler Videorecorder erfüllen kann. Dazu zählen die zwei DVB-Karten, die den Empfang des Satellitensignals und die Ausgabe auf dem Fernsehgerät übernehmen. Diese werden über das API des Linux DVB Treibers vom Video Disc Recorder angesprochen, der wohl wichtigsten Softwarekomponente der DVP-Architektur. Die Steuerung des VDR

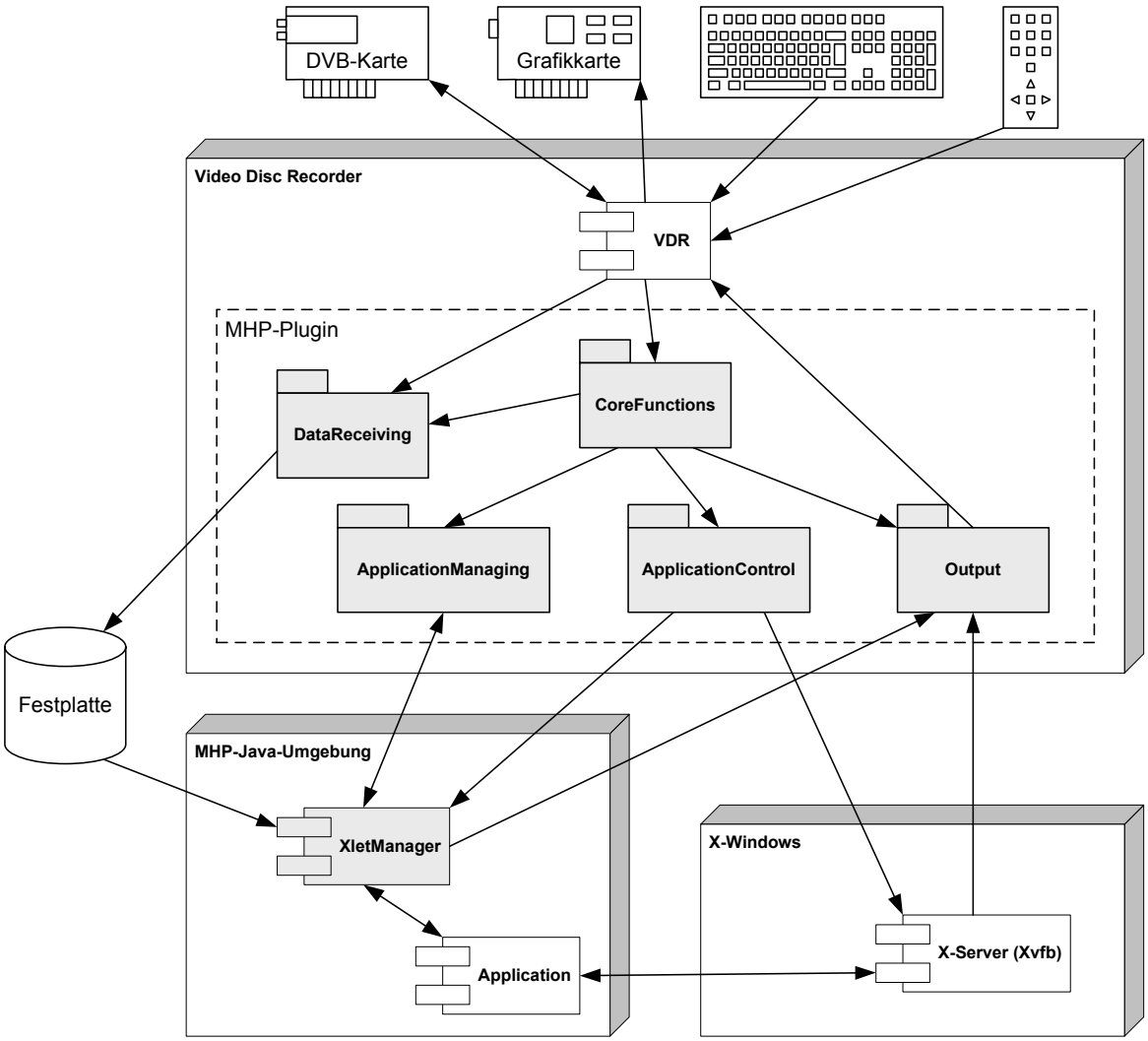


Abbildung 4.11.: Gesamtarchitektur der MHP-DVP-Integration. Die selbst zu entwickelnden Bestandteile sind grau eingefärbt.

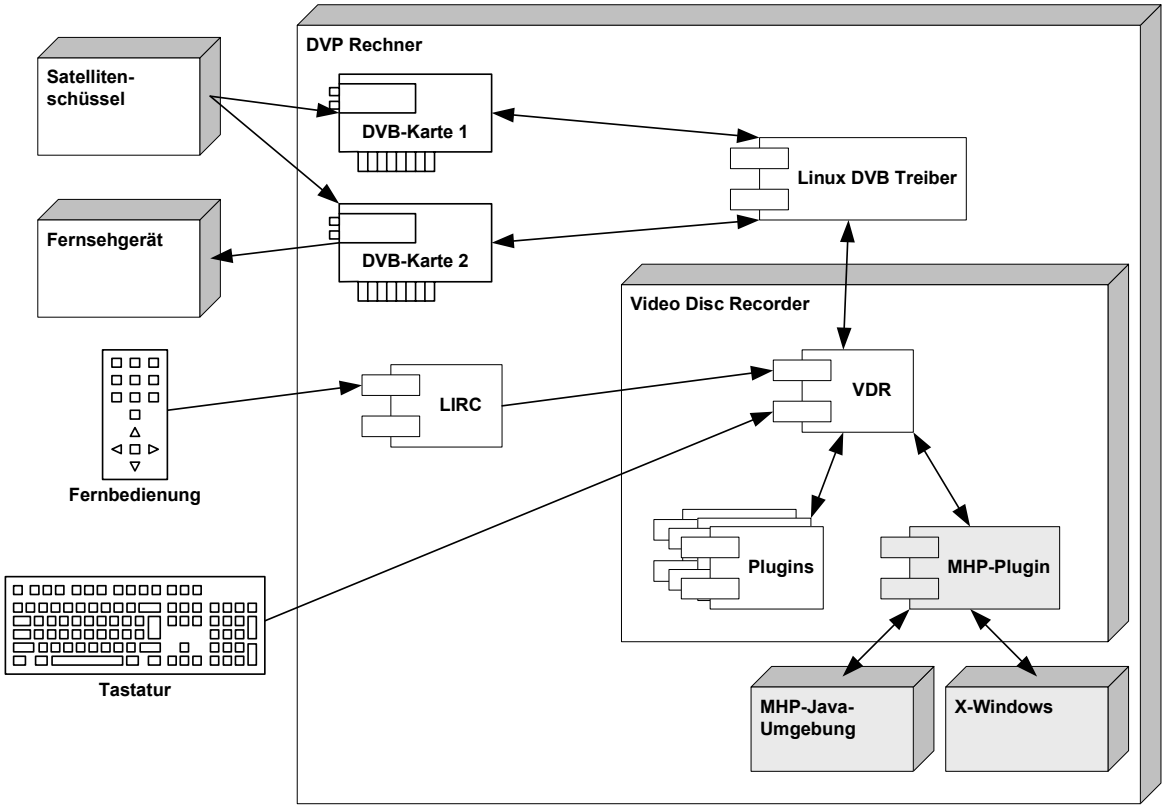


Abbildung 4.12.: Integration des Systems in die DVP-Rechnerarchitektur

kann über eine Tastatur oder eine mittels LIRC angesprochene Infrarotfernbedienung erfolgen. Für die Verwirklichung weiterer Zusatzfunktionen können diverse Plugins an den VDR angebunden sein.

Abbildung 4.12 zeigt die Architektur eines DVP-Rechners und wie diese durch die grau eingefärbten Teile der MHP-DVP-Integration erweitert wird. Die Hauptfunktionalität des Systems wird in einem VDR-Plugin realisiert, das die zusätzlich benötigten Softwarekomponenten der MHP-Java-Umgebung und des X-Windows-Systems mit dem Video Disc Recorder verbindet. Damit wird die Ausführung von MHP-Applikation in der VDR-Umgebung des DVP-Rechners ermöglicht.

4.6. Statische und dynamische Aspekte der Pakete des Systems

In den folgenden Abschnitten werden die Pakete der MHP-DVP-Integration einzeln im Detail vorgestellt. Dies beginnt jeweils mit der Architektur des Pakets in Form eines Komponentendiagramms, in dem die Module des Pakets angegeben sind sowie die Datenflüsse innerhalb des Pakets und zu den Modulen anderer Komponenten des Systems. Der Architektur folgt der Entwurf der statischen Struktur des Pakets, die als Klassendiagramm angegeben wird. Abschließend werden die dynamischen Aspekte des Pakets mit Hilfe von Verhaltensdiagrammen[2] beschrieben. Die Notation der Diagramme richtet sich in weiten Teilen nach der Unified Modeling Language (UML).

4.6.1. Das Paket CoreFunctions

Das Paket *CoreFunctions* bildet die Grundlage des MHP-Plugins. Es beinhaltet Funktionalitäten, die von den anderen Komponenten genutzt werden bzw. für das Ausführen von deren Prozessen erforderlich sind. Außerdem sorgt es für die Integration des Plugins in den Video Disc Recorder.

Funktionale Architektur

Das Paket setzt sich aus 3 Modulen zusammen: *FlowControl*, *DataStorage* und *Settings*. In Abbildung 4.13 wird angegeben, wie diese in den Video Disc Recorder eingebunden sind und welche Datenflusse zwischen ihnen auftreten.

[2] Als Verhaltensdiagramme werden in der UML Sequenzdiagramme, Kollaborationsdiagramme, Aktivitätsdiagramme und Zustanddiagramme bezeichnet.

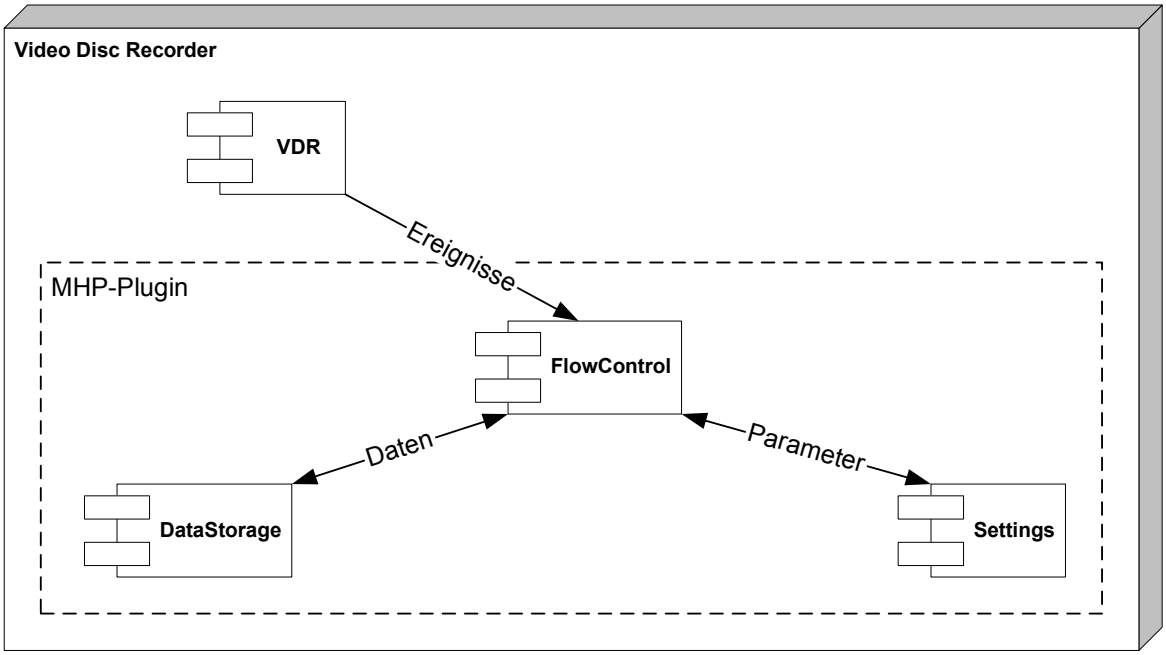


Abbildung 4.13.: Architektur des Pakets CoreFunctions

Das Modul *FlowControl* ist für die Steuerung des Kontrollflusses verantwortlich. Es reagiert auf Ereignisse die von den internen Prozessen des VDR gesendet werden, wie Tastensignale oder den Wechsel auf einen neuen Kanal, und startet oder beendet daraufhin Prozesse, die von den anderen Paketen des Systems angeboten werden. Das Modul *DataStorage* beinhaltet die Verwaltung verschiedener Daten, die von anderen Paketen benötigt oder von diesen angeboten werden. Das Modul *Settings* stellt verschiedene Parameter zur Verfügung, welche das Verhalten fast aller Komponenten des VDR-Plugins beeinflussen. Außerdem bietet das Modul dem Nutzer die Möglichkeit die Werte der Parameter über das Einstellungsmenü zu verändern. Die Einstellungen werden mittels von VDR bereitgestellten Mechanismen in einer Konfigurationsdatei gespeichert, die zeilenweise Einträge der Form „Parameter=Wert“ enthält.

Statische Struktur

Für die Verwirklichung der Funktionalitäten des Pakets *CoreFunctions* wurde eine Klassenstruktur entworfen, die in Abbildung 4.14 gezeigt wird. In dieser sind die Klassen nach den Modulen, denen sie angehören, geordnet.

Das Modul *DataStorage* setzt sich aus Klassen zusammen, die der Speicherung und Verwaltung von Datenstrukturen dienen. Die Klasse *cApplication* repräsentiert eine im DVB-Signal übertragene

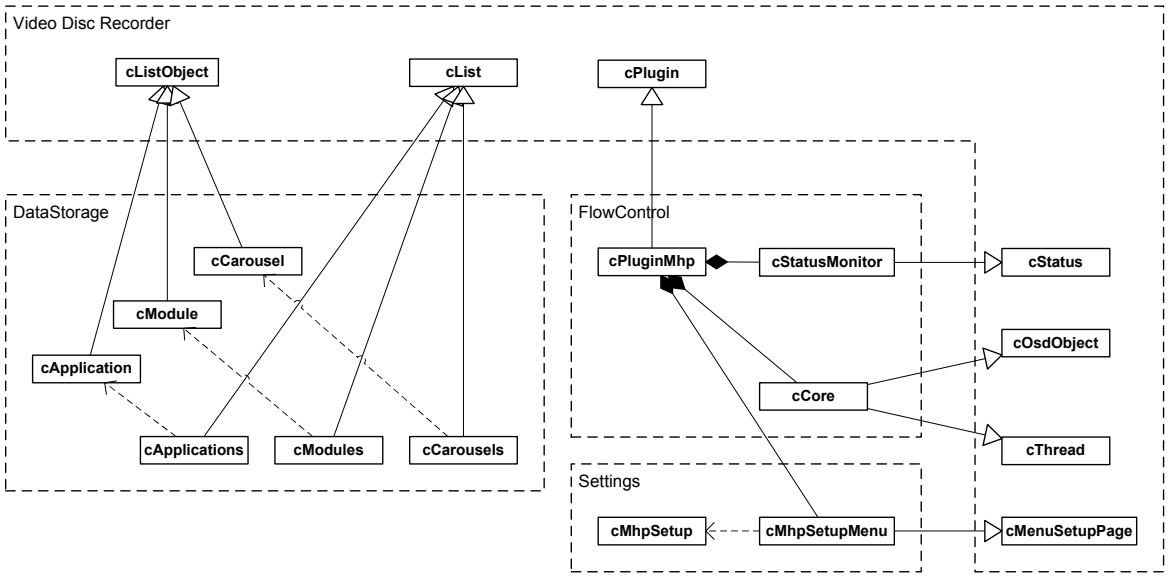


Abbildung 4.14.: Statische Struktur des Pakets CoreFunctions

MHP-Applikation und beinhaltet verschiedene Informationen wie Name, Parameter, Klassendatei und Startverhalten. Die Klasse *cCarousel* steht für die Daten eines DSM-CC Object Carousel, das heißt sie beinhaltet die aus dem Datenstrom extrahierten Objekte eines Objektkarussells. Die Module des DSM-CC Data Carousel werden durch die Klasse *cModule* repräsentiert. Alle drei Klassen sind Subklassen der VDR-Klasse *cListObject* und können damit als Elemente einer Liste vom Typ *cList* eingesetzt werden. Dies geschieht mit den Klassen *cApplications*, *cCarousels* und *cModules*, welche alle von der Basisklasse *cList* abgeleitet werden und damit dynamische, doppelt-verkettete Listen darstellen, die von VDR verwaltet werden und Objekte der vorher genannten Klassen enthalten.

Das Modul *FlowControl* beinhaltet drei Klassen: *cPluginMhp*, *cStatusMonitor* und *cCore*. Die Klasse *cPluginMhp* stellt die initiale Schnittstelle des MHP-Plugins zum Video Disc Recorder dar. Die Ableitung von der Klasse *cPlugin* sorgt dafür, dass das Plugin vom VDR dynamisch geladen und gestartet werden kann. Außerdem wird dem Hauptmenü ein neuer Eintrag hinzugefügt, über den die Nutzerinteraktion mit dem Plugin begonnen werden kann. Die Klasse *cStatusMonitor* ist für die Überwachung von Kanalwechseln und die Reaktion darauf verantwortlich. Dies geschieht automatisch, sobald eine Instanz der Klasse erzeugt wird, da *cStatusMonitor* von *cStatus* abgeleitet und die Methode *ChannelSwitch* überladen wird. Der Aufruf dieser Methode wird vom VDR vor und nach jedem Kanalwechsel vorgenommen. Die Klasse *cCore* stellt den Kern der Ablaufsteuerung dar, sobald die Interaktion des Plugins mit dem Nutzer begonnen hat. Sie bekommt vom Video Disc Re-

corder mitgeteilt, wenn eine Taste auf der Fernbedienung gedrückt wurde, was durch die Basisklasse *cOsdObject* sichergestellt wird. Aufgrund des Erbens von *cThread* wird der Klasse ermöglicht, im Hintergrund Aktivitäten durchzuführen, ohne den Ablauf des VDR zu stören.

Das Modul *cSettings* setzt sich aus Klassen zusammen, die die Verwaltung und Speicherung der von den Komponenten abgefragten Parametern vornehmen. Die Klasse *cMhpSetup* stellt einen einfachen Container für die einzelnen Parameter dar und beinhaltet selbst keine Funktionalitäten für deren Modifizierung. Dies wird durch die Klasse *cMhpSetupMenu* vorgenommen, welche eine Seite des im OSD dargestellten Menüs repräsentiert und dem Nutzer die Veränderung der Parameter der Container-Klasse erlaubt. Die Einbetting der Seite in die Menüstrukur des VDR wird durch die Klasse *cPluginMhp* sichergestellt.

Dynamische Aspekte

Die Hauptaktivitäten des Pakets *CoreFunctions* finden im Modul *FlowControl* statt. Die Tätigkeiten von *DataStorage* und *Settings* beschränken sich innerhalb des Pakets auf die Speicherverwaltung und die Änderung von Werten, was allerdings durch deren Basisklassen übernommen wird. Aus diesem Grund wird hier nicht näher darauf eingegangen.

Abbildung 4.15 illustriert die Aktivitäten der Ablaufsteuerung. In dem Diagramm können drei parallel und zyklisch ablaufende Teilprozesse erkannt werden. Zum einen findet eine Überwachung der Kanalwechsel statt. Auf ein solches Ereignis wird als erstes geprüft, ob im Moment ausgeführte MHP-Applikationen beendet werden müssen. Nachdem das Umschalten auf einen neuen Kanal vollzogen ist, wird der Prozess des Datenempfangs gestartet. Ist dieser abgeschlossen, wird, falls dies erforderlich ist, der automatische Start von neu empfangenen Applikationen vorgenommen.

Der zweite Prozess des Moduls reagiert auf die Betätigung von Tasten auf der Fernbedienung. Abhängig vom aktuellen Kontext und der Art der Taste erfolgt entweder ein Auslösen der Applikationsverwaltung, die daraufhin z. B. das manuelle Starten bzw. Beenden vornehmen kann, oder es findet eine Weiterleitung des Signals an die Applikationssteuerung statt.

Falls eine MHP-Applikation gestartet wird, beginnt der dritte Teilprozess mit seiner Aktivität. Solange die Anwendung ausgeführt wird, erfolgt in festen Zeitabständen eine ständige Erfassung und Darstellung der grafischen Ausgabe der Applikation, indem die entsprechenden Prozesse des Pakets Output gestartet werden.

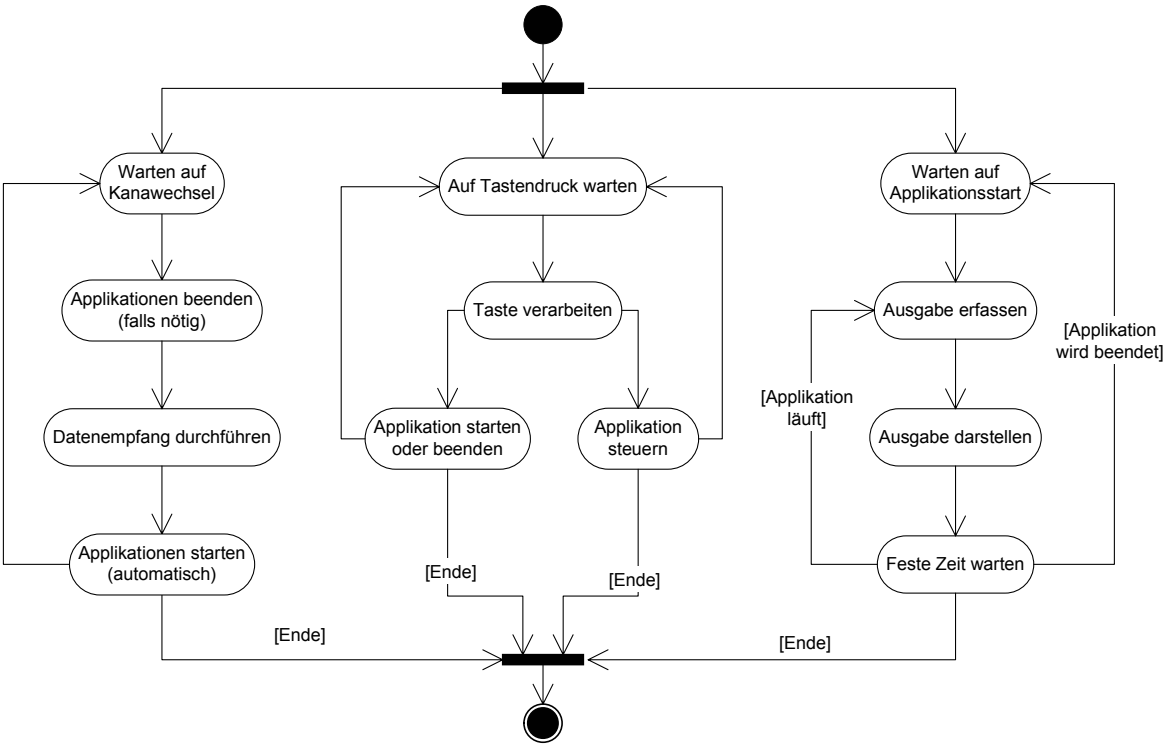


Abbildung 4.15.: Aktivitäten des Moduls FlowControl

4.6.2. Das Paket DataReceiving

Das Paket *DataReceiving* ist für den Empfang und die Speicherung aller Daten verantwortlich, die zu den MHP-Applikationen gehören. Dazu zählen die Dateien der Anwendung, sowie Informationen über deren Name, Parameter und Startverhalten.

Funktionale Architektur

Die funktionale Architektur des Pakets ist in dem Komponentendiagramm angegeben, das in Abbildung 4.16 gezeigt wird. Es enthält die drei Module des Pakets: *DataReceiver*, *DSM-CC-Parser* und *AIT-Parser*. Außerdem fällt auf, dass auch zwei Module der *CoreFunctions*, grau eingefärbt, in dem Diagramm enthalten sind. Diese bieten Funktionen oder Daten an, die von den Komponenten des *DataReceiving* genutzt werden.

Das Modul *DataReceiver* stellt die Verbindung des Pakets zu den internen Vorgängen des Video Disc Recorders dar. Der von der DVB-Karte empfangene Transportstrom des DVB-Signals wird vom VDR in die einzelnen TS-Pakete zerlegt. Die Pakete, die die richtigen PIDs aufweisen werden an den *DataReceiver* übergeben, welcher daraus die MPEG-2-Sections extrahiert. Das weitere Vor-

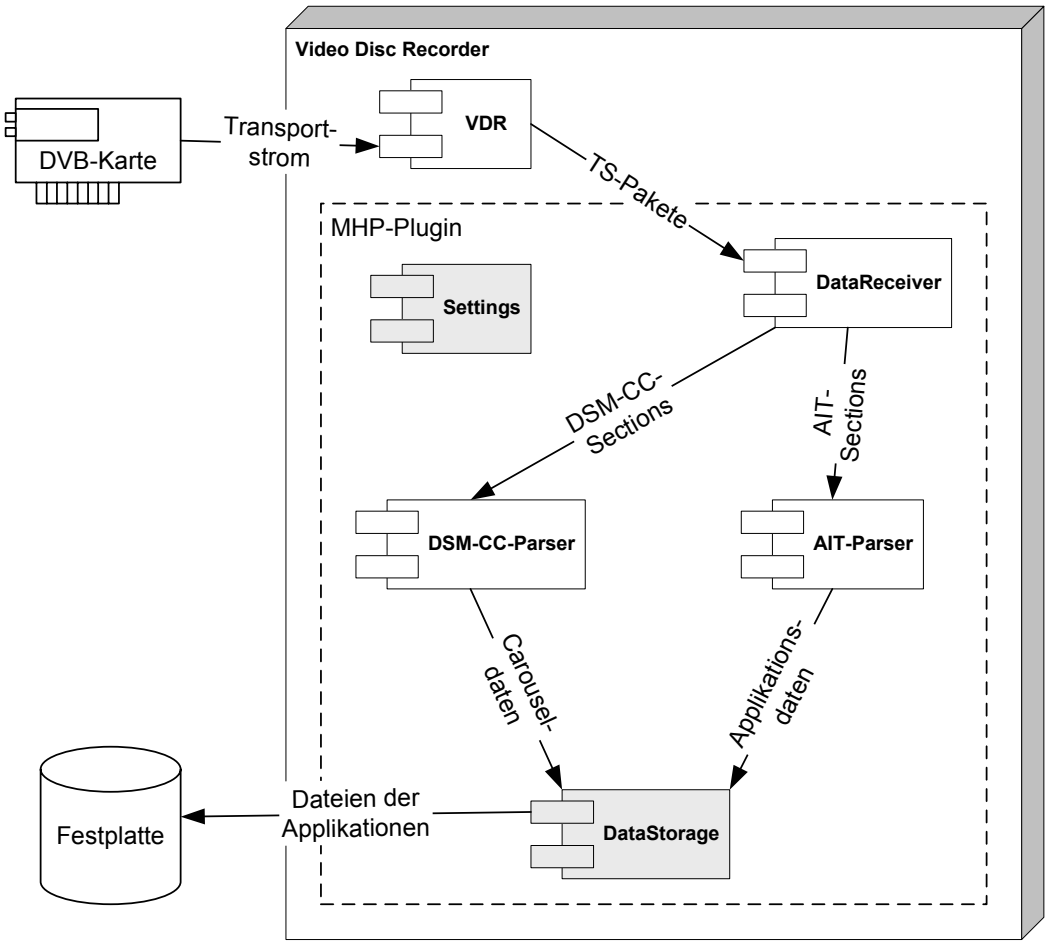


Abbildung 4.16.: Architektur des Pakets DataReceiving

gehen ist abhängig von der Art der Sections. Falls es sich um Segmente der Application Information Table handelt, werden sie dem *AIT-Parser* übergeben, liegen DSM-CC-Sections vor, werden diese an den *DSM-CC-Parser* weitergereicht. Der *AIT-Parser* extrahiert aus den Segmenten der AIT Informationen zu den im Datenstrom übertragenen Applikationen und speichert diese in den Objekten des *DataStorage*-Moduls. Die an den *DSM-CC-Parser* übergebenen Segmente werden in mehreren Schritten zu den Objekten des DSM-CC Object Carousel überführt, welche die Verzeichnisse und Dateien der MHP-Applikation repräsentieren. Diese Daten werden über die Klassen des Moduls *DataStorage* auf der Festplatte des Rechners abgespeichert.

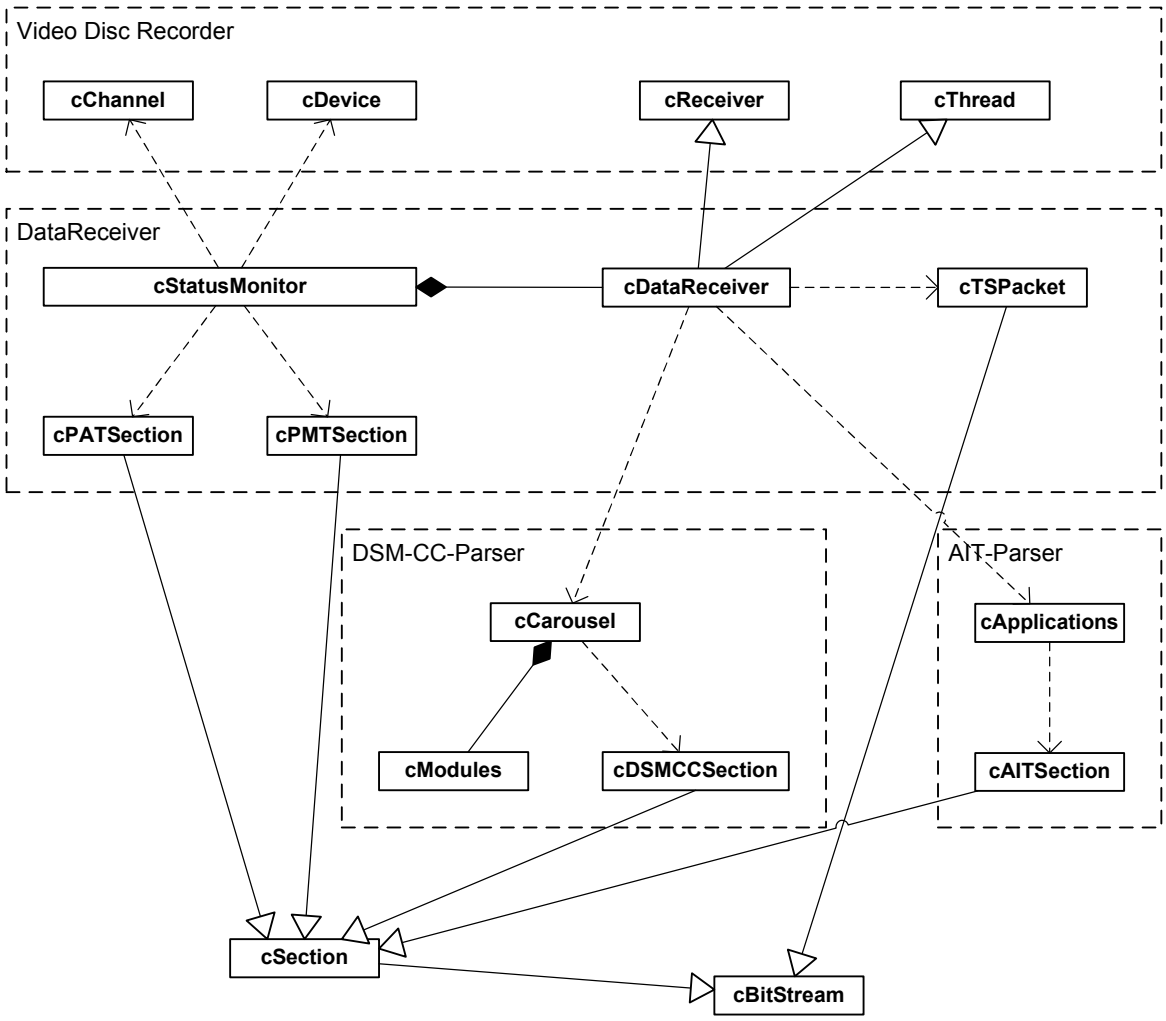


Abbildung 4.17.: Statische Struktur des Pakets DataReceiving

Statische Struktur

In Abbildung 4.17 ist die Klassenstruktur angegeben, die für die Bereitstellung der Funktionalitäten des Datenempfangs verantwortlich ist. Darin enthalten sind auch einige Klassen des *DataStorage*-Moduls, deren Aufbau und Eigenschaften dort zwar festgelegt wurden, deren Verhalten allerdings erst im Paket *DataReceiving* entworfen wird.

Das Paket *DataReceiving* besitzt zwei Hilfsklassen, *cBitStream* und *cSection*, von denen einige der anderen Klassen abgeleitet werden. Die Klasse *cBitStream* erlaubt das bitweise Durchlaufen eines beliebigen Datenpakets. Sie eignet sich daher insbesondere für die Verarbeitung von Headerdaten, deren Inhalte häufig in Folgen von unterschiedlich vielen Bits organisiert sind. Die Klasse *cSection* nutzt diese Funktionalität um den Zugriff auf die Headerdaten jeder beliebigen MPEG-2-

Section zu abstrahieren. Sie dient damit den Klassen *cPATSection*, *cPMTSection*, *cAITSection* und *cDSMCCSection* als Basis für die spezifische Analyse der entsprechenden Segmente. Eine weitere von cBitStream abgeleitete Klasse ist *cTSPacket*, die den Zugriff auf die Daten eines TS-Pakets vereinfacht.

Die Aktivitäten des *DataReceiver* werden von zwei Klassen übernommen. In der Klasse *cStatusMonitor* wird nach einem Kanalwechsel untersucht, ob der neue Datenstrom MHP-Applikationen enthält und auf welchen PIDs diese übertragen werden. Die VDR-Klassen *cDevice* und *cChannel* dienen dabei der Ermittlung der Service Identification des neuen Kanals und mit Hilfe der Klassen *cPATSection* und *cPMTSection* werden die Einträge der PAT und der PMT analysiert. Die Klasse *cDataReceiver* empfängt die TS-Pakete der ermittelten Datenströme über eine von der Basisklasse *cReceiver* angebotene Schnittstelle. Mit Hilfe von *cTSPacket* werden die Pakete analysiert und die MPEG-2-Sections extrahiert.

Das Modul *AIT-Parser* wird als Methode in der *cApplications*-Klasse realisiert. Mit der Klasse *cAITSection* werden die Einträge der Application Information Table analysiert und entsprechend *cApplication*-Objekte erzeugt und der Liste von Applikationen hinzugefügt.

Der *DSM-CC-Parser* wird auch in Form von Methoden der *DataStorage*-Klassen realisiert. Für die Analyse der DSM-CC-Segmente kommen die Klassen *cCarousel* und *cModule(s)* zum Einsatz. In diesen Klassen erfolgt die Überführung der Sections in Blöcke, Module und schließlich zu den Objekten des Object Carousels. Des Weiteren ist *cCarousel* für die Speicherung der Dateien der Applikation auf der Festplatte verantwortlich

Dynamische Aspekte

Abbildung 4.18 gibt einen Überblick über die Aktivitäten des Pakets *DataReceiving*. Sie beginnen mit dem Umschalten auf einen neuen Kanal und enden vor dem nächsten Kanalwechsel.

Der Datenempfang beginnt mit dem Ermitteln der Service Identification (SID) des aktuell empfangenen Kanals. Die SID stellt eine eindeutige Identifizierung des Dienstes dar und wird benötigt um die PID der Program Map Table in Erfahrung zu bringen. Dies geschieht im folgenden Schritt, dem Parsen der PAT. Aus den Einträgen der Tabelle wird derjenige herausgesucht, dessen SID mit der des aktuellen Kanals übereinstimmt. Der gefundene Eintrag enthält die PID der Program Map Table, die als nächstes empfangen und analysiert wird. Die PMT besitzt für jeden Datenstrom des Programms einen Eintrag, der dessen Typ, die PID der TS-Pakete und weitere Informationen in

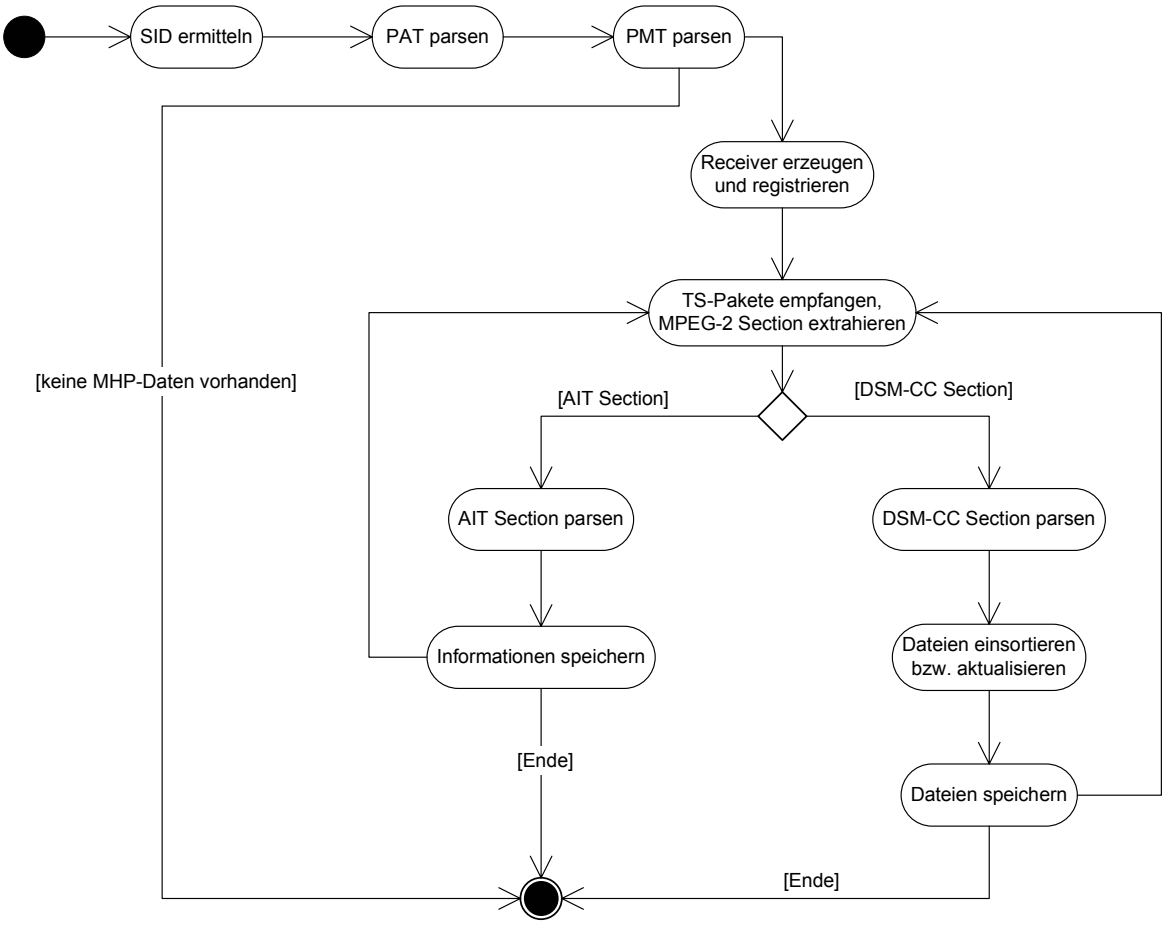


Abbildung 4.18.: Aktivitäten des Pakets DataReceiving

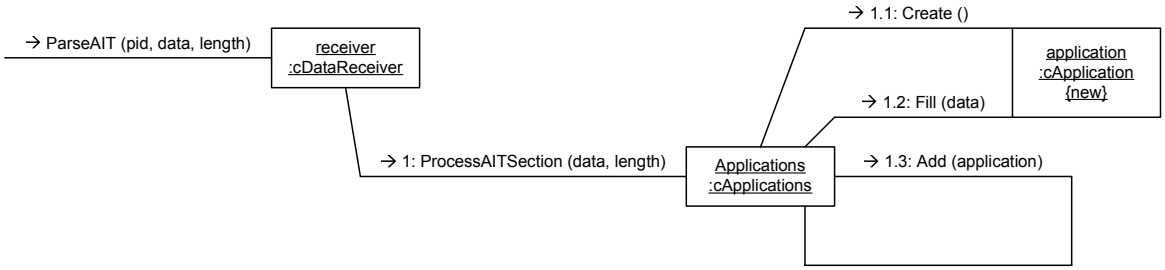


Abbildung 4.19.: Funktionsweise des AIT-Parsers

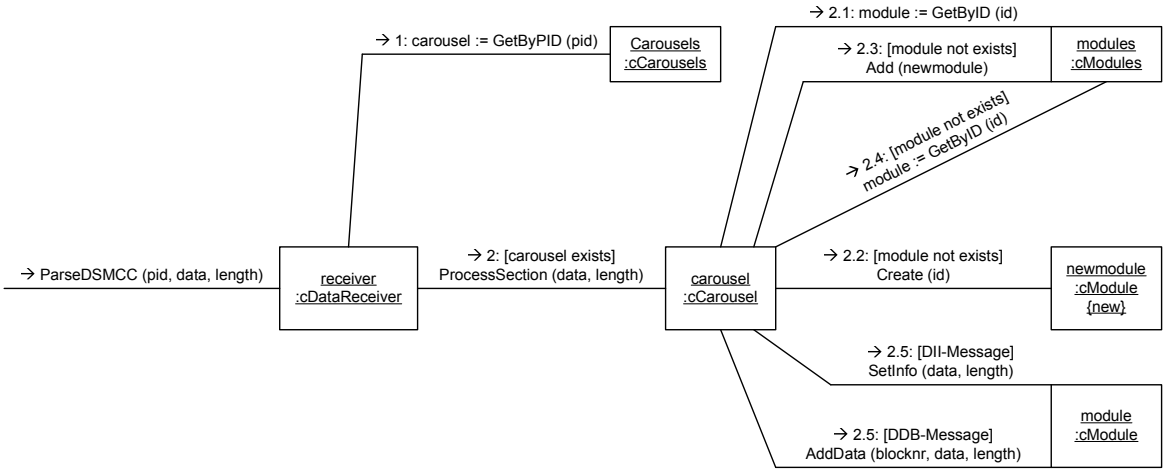


Abbildung 4.20.: Funktionsweise des DSM-CC-Parsers

Form von Deskriptoren beinhaltet. Die Typen der für den Empfang von MHP-Inhalten relevanten Datenströme besitzen die Werte 5 für die AIT und 11 für die DSM-CC-Ströme. Falls keine solchen Datenströme gefunden werden, ist der Datenempfang abgeschlossen. Andernfalls wird für die ermittelten PIDs eine *cDataReceiver*-Instanz erzeugt und im VDR registriert. Sie erhält von da an die TS-Pakete der gewünschten Datenströme und ist für deren Weiterverarbeitung verantwortlich. Die TS-Pakete transportieren einen Strom von MPEG-2-Sections. Sobald ein solches Segment komplett empfangen wurde, wird es an den zuständigen Parser weitergeleitet, eine AIT-Section wird somit an den AIT-Parser übergeben und eine DSM-CC-Section an den DSM-CC-Parser. Die Funktionsweise dieser Komponenten ist in den Abbildungen 4.19 und 4.20 angegeben.

Beim Parsen der AIT-Segmente wird für jeden darin enthaltenen Eintrag eine Instanz der Klasse *cApplication* erzeugt. Dieses Objekt wird anschließend mit den Angaben zu einer Applikation aus der AIT gefüllt und der Liste der im Datenstrom übertragenen MHP-Anwendungen, repräsentiert durch ein Objekt der Klasse *cApplications*, hinzugefügt. Ist dieser Prozess abgeschlossen wird mit dem

Empfang der nächsten MPEG-2-Section begonnen, es sei denn, der Datenempfang wird beendet.

Das Parsen der DSM-CC-Segmente ist weitaus komplizierter. Als erstes wird über die PID, auf der die Section empfangen wurde, das Object Carousel ermittelt, dem diese Daten angehören. Die entsprechende Instanz der Klasse cCarousel nimmt die weitere Verarbeitung der DSM-CC Nachricht vor. Die Nachricht enthält Daten für ein bestimmtes Modul. Deshalb wird zuerst die Instanz des entsprechenden Moduls ermittelt und zurückgeliefert. Sollte diese nicht existieren, wird ein neues Objekt der Klasse *cModule* erzeugt und der Liste der Module des Object Carousel hinzufgefügt. Die Daten der Nachricht werden abhängig von deren Typ dem nun vorhandenen Modul mitgeteilt. Dabei kann es sich entweder um Angaben über die Anzahl und die Größe der Blöcke des Moduls oder um die Daten der Blöcke selber handeln. Sind die Daten eines Moduls komplett, werden die darin enthaltenen Elemente des Objektkarussells extrahiert, welche die Dateien und Verzeichnisse der MHP-Applikation darstellen.

Sollten nach dem Parsen einer DSM-CC-Section neue bzw. aktualisierte Dateien und Verzeichnisse vorliegen, werden diese in die bestehende Verzeichnisstruktur aufgenommen und auf der Festplatte des Rechners gespeichert. Danach wird auch hier mit dem Empfang der nächsten MPEG-2-Section begonnen, es sei denn, der Datenempfang wird beendet.

4.6.3. Das Paket ApplicationManaging

Das Paket *ApplicationManaging* ist für das automatische oder manuelle Starten bzw. Beenden von MHP-Applikationen verantwortlich. Des Weiteren nimmt es Zustandsänderungen an den ausgeführten Anwendungen vor bzw. reagiert auf diese.

Funktionale Architektur

Das Paket, dessen Architektur in Abbildung 4.21 angegeben ist, setzt sich aus insgesamt drei Modulen zusammen. Zwei davon, *ApplicationListing* und *ApplicationManager*, sind Bestandteil des MHP-Plugins und werden damit innerhalb des Video Disc Recorder ausgeführt. Diese beiden benötigen die Daten bzw. Parameter, die von den Modulen *DataStorage* und *Settings* bereitgestellt werden. Das dritte Modul des Pakets, der *XletManager*, läuft dagegen in der MHP-Java-Umgebung ab.

Das Modul *ApplicationListing* benutzt die in *DataStorage* gespeicherten Informationen um dem Anwender des Systems eine Liste zu präsentieren, die alle auf dem aktuellen Kanal übertragenen

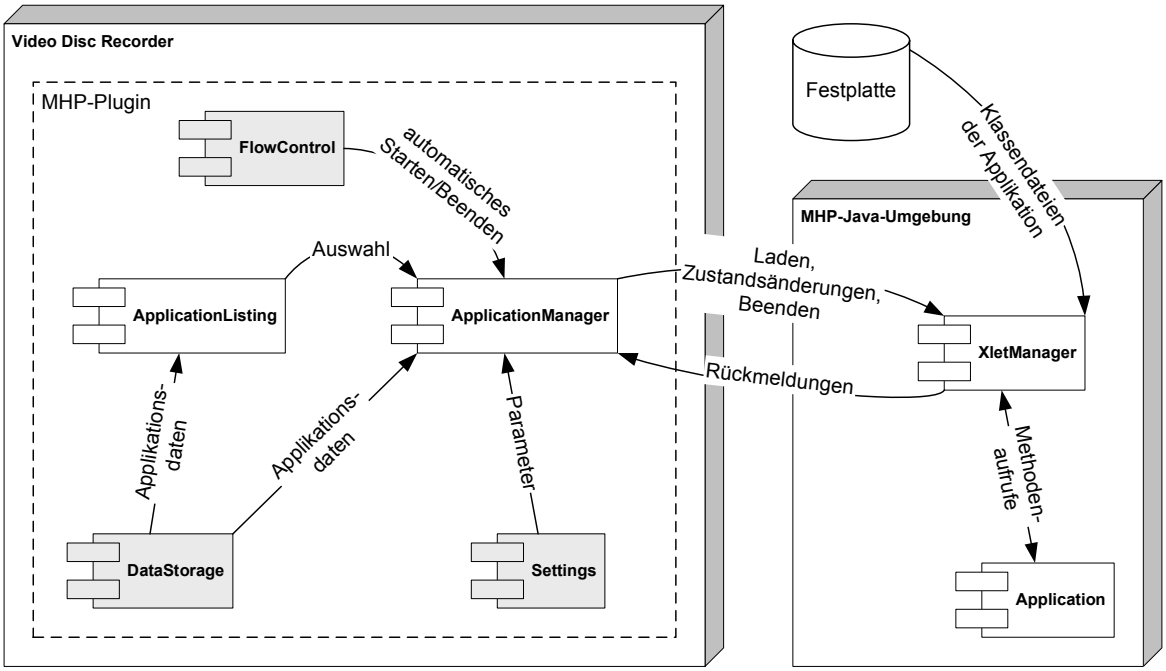


Abbildung 4.21.: Architektur des Pakets ApplicationManaging

MHP-Applikationen beinhaltet. Wählt der Nutzer eine der bereitstehenden Applikationen aus, wird diese an das Modul *ApplicationManager* übergeben, welches den Start der Anwendung über den *XletManager* vornimmt. Dies kann auch von *FlowControl* nach einem Kanalwechsel ausgelöst werden, worauf der gegebenenfalls nötige automatische Start von Applikationen erfolgt. Während diese ausgeführt werden, ist das Modul für die Beeinflussung von deren Lebenszyklus über das Senden von Zustandsänderungen und die Reaktion auf daraufhin erhaltene Rückmeldungen verantwortlich. Soll die MHP-Anwendung gestoppt werden, sei es auf Wunsch des Nutzers oder automatisch von der Ablaufsteuerung gefordert, wird dieser Vorgang auch vom *ApplicationManager* ausgelöst.

Das eigentliche Starten und Beeinflussen der Applikation findet aber durch den in der MHP-Java-Umgebung ausgeführten *XletManager* statt. Dieser steht in ständiger Verbindung mit dem *ApplicationManager*, über die die Kommandos für Starten, Zustandsänderungen und Beenden sowie Rückmeldungen über Fehler oder Erfolg ausgetauscht werden. Dafür wird im Folgenden eine Netzwerkverbindung über Sockets verwendet. Allerdings sind auch andere Varianten denkbar, wie der Austausch über einen Shared-Memory-Bereich. Die Beeinflussung der Applikation wird über die Methoden der Xlet-Schnittstelle vorgenommen. Über eine weitere Schnittstelle kann auch der *XletManager* selbst von Ereignissen der Applikation in Kenntnis gesetzt werden.

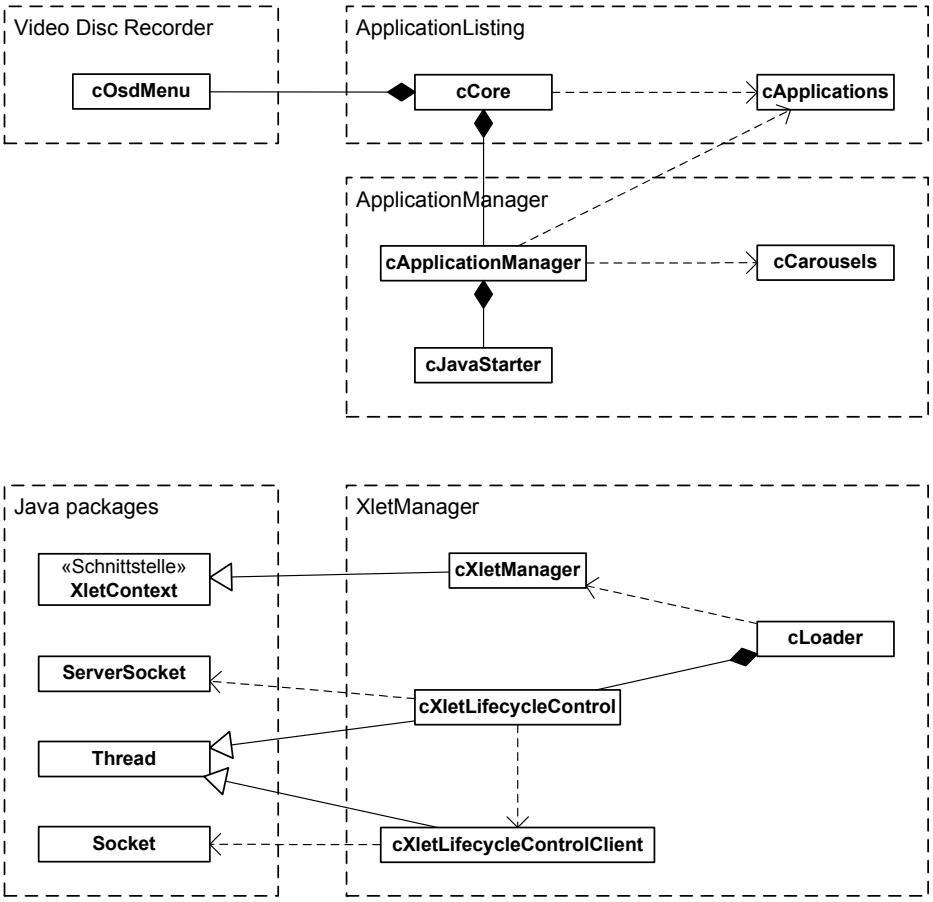


Abbildung 4.22.: Statische Struktur des Pakets ApplicationManaging

Statische Struktur

Für die Realisierung der Funktionalitäten des Pakets *ApplicationManaging* wurde eine Klassenstruktur entworfen, die in Abbildung 4.22 angegeben ist. Auch hier sind einige Klassen der *CoreFunctions* enthalten, die entweder Informationen anbieten oder für die Applikationsverwaltung um einige Funktionen erweiert werden.

Das Modul *ApplicationListing* wird über eine Erweiterung der Klasse *cCore* aus den *CoreFunctions* realisiert. Mit Hilfe der VDR-Klasse *cOsdMenu* kann mit einfachen Mitteln eine Menuseite aufgebaut und im On-Screen-Display der DVB-Karte angezeigt werden, die unter Verwendung der in *cApplications* gespeicherten Informationen eine Liste der momentan verfügbaren Applicationen präsentiert.

Die Funktionalität des *ApplicationManager* wird in zwei Klassen realisiert, *cApplicationManager* und *cJavaStarter*. Erstere bietet eine Schnittstelle für die Ablaufsteuerung, MHP-Applikationen zu

starten, ihren Zustand aufgrund von Nutzerergeignissen zu beeinflussen und sie zu beeenden. Dafür muss die MHP-Java-Umgebung gestartet werden, wofür die Klasse *cJavaStarter* verantwortlich ist. Des Weiteren ist *cApplicationmanager* dafür verantwortlich die Socket-Verbindung zum *XletManager* bei Bedarf aufzubauen und wieder zu schließen.

Der *XletManager* wird als Java-Applikation realisiert. Er setzt sich aus vier Klassen zusammen. Die Klasse *cLoader* enthält das Hauptprogramm, welches die Auswertung der Kommandozeilenparamter und das Starten der einzelnen Prozesse des Moduls übernimmt. Die Klasse *cXletManager* stellt den Kern des Moduls dar und führt die direkte Beeinflussung der MHP-Applikation über die Xlet-Schnittstelle durch. Über die aus der geerbten Schnittstelle *XletContext* übernommenen Methoden bietet *cXletManager* der gerade ausgeführten Anwendung eine Schnittstelle, über die Parameter erfragt und Zustandsänderungen bekanntgegeben werden können. Für die Verbindung des XletManagers mit dem MHP-Plugin sind die zwei übrigen Klassen verantwortlich. Dies geschieht mit Hilfe den von Java angebotenen *ServerSocket* und *Socket*. Die Klasse *cXletLifecycleControl* repräsentiert den Server, der die Anfragen des *ApplicationManager*-Moduls entgegennimmt. Die Verarbeitung der Anfragen und Weiterleitung der darin enthaltenen Kommandos wird von der Klasse *cXletLifecycleControlClient* übernommen, welche auch für das Senden von Rückmeldungen zurück an das MHP-Plugin verantwortlich ist.

Dynamische Aspekte

Die Aktivitäten des Pakets *ApplicationManaging* beginnen entweder nach einem Kanalwechsel, falls der automatische Start von MHP-Anwendungen vonnöten ist, oder nach dem Starten der Nutzerinteraktion mit dem MHP-Plugin durch Auswahl des Hauptmenüeintrags. Abbildung 4.23 gibt einen Überblick über die sich daran anschließenden Aktivitäten.

Wenn der manuelle Start einer MHP-Anwendung erfolgen soll, werden als erstes die verfügbaren Applikationen mit ihren Namen aufgelistet und für den Nutzer sichtbar im OSD-Menü des Video Disc Recorders angezeigt. Nun kann der Vorgang entweder abgebrochen werden, womit die Aktivitäten des Pakets fürs Erste beendet werden, oder die gewünschte Applikation wird mit der Fernbedienung ausgewählt, was zur Ausführung der Anwendung führt.

Die Ausführung einer Applikation beginnt, auch im Falle eines automatischen Starts, mit dem Starten der MHP-Java-Umgebung, deren Bestandteil der *XletManager* ist. Dies geschieht mit der Erzeugung eines Teilprozesses innerhalb des MHP-Plugins, in welchem die Java Virtual Machine

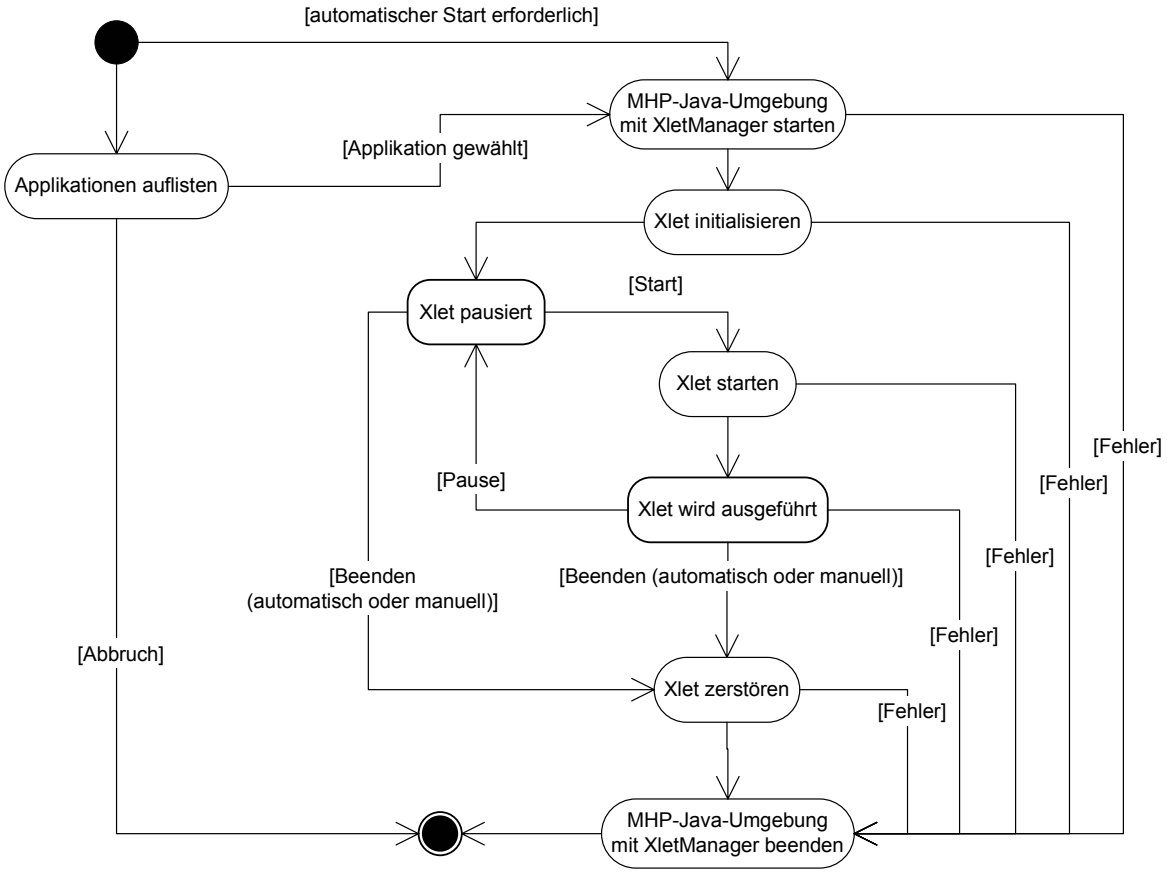


Abbildung 4.23.: Aktivitäten des Pakets ApplicationManaging

ausgeführt wird. Bei diesem Vorgang müssen neben dem Name der Klasse, die das Xlet enthält, und dem Speicherort der Dateien der Anwendung auch die unter Umständen vorhandenen Parameter der Applikation an den *XletManager* übergeben werden. Dies kann auf verschiedene Arten erfolgen: mittels Kommandozeilenparametern, über eine Socket-Verbindung bzw. einen Shared-Memory-Bereich oder indem diese Daten in eine Datei geschrieben werden, die vom *XletManager* ausgewertet wird.

Ist der Start der MHP-Java-Umgebung abgeschlossen, erfolgt die Initialisierung der Applikation, ausgelöst vom *ApplicationManager* und durchgeführt vom *XletManager*. Danach befindet sich das Xlet im „Pause“-Zustand. Nach dem nun folgenden Start des Xlets über den entsprechenden Methodenaufruf wird die Applikation ausgeführt. Durch einen vom Nutzer oder von internen Vorgängen der Applikation ausgelösten Übergang kann das Xlet wieder in den Zustand „Pause“ versetzt werden, woraufhin ein neuerlicher Start möglich ist.

Das Beenden der gerade ausgeführten MHP-Applikation kann entweder manuell oder automatisch durch einen Kanalwechsel geschehen. Liegt einer der Fälle vor wird als erstes ein Zerstören des Xlets

vorgenommen. Danach wird die MHP-Java-Umgebung und damit auch der *XletManager* beendet. Damit sind die Aktivitäten des Pakets abgeschlossen.

Sollte während den Aktivitäten bei der Ausführung einer MHP-Applikation ein Fehler auftreten, wird der gesamte Prozess sofort abgebrochen und mit dem Beenden der MHP-Java-Umgebung abgeschlossen.

4.6.4. Das Paket ApplicationControl

Das Paket *ApplicationControl* beeinflusst das Verhalten der ausgeführten MHP-Applikation, indem es vom Nutzer vorgenommene Tastendrücke auf Fernbedienung oder Tastatur an die Anwendung weiterleitet.

Funktionale Architektur

Das Paket, dessen Architektur in Abbildung 4.24 angegeben ist, führt ein neues Modul (*KeyDispatcher*) in das MHP-Plugin ein. Auch in diesem Paket werden Funktionalitäten und Daten der *CoreFunctions* genutzt. Des Weiteren kann es je nach verwendeter Steuerungsvariante[3] nötig werden den *XletManager* zu erweitern.

Die vom Nutzer ausgelösten Steuersignale in Form von Infrarot- und Tastensignalen werden durch den VDR registriert und in interne Tastencodes umgewandelt. Diese werden an das MHP-Plugin gesendet. Dort werden sie vom Modul *FlowControl* entgegengenommen, das daraufhin die Applikationssteuerung auslöst, indem die Tastencodes an das Modul *KeyDispatcher* übergeben werden. Der *KeyDispatcher* übernimmt das Weiterleiten der Ereignisse an die ausgeführte Applikation, zum Teil von den Parametern beeinflusst, die das Modul *Settings* liefert.

Für das Senden der Tastencodes an die MHP-Anwendung existieren nach der Merkmalserfassung zwei Varianten: X-Windows oder Java AWT Events. Beim ersten Fall werden die Tastencodes als Ereignisse direkt an den X-Server gesendet, in welchem die grafische Ausgabe der Applikation dargestellt wird. Das X-Windows System übernimmt folgend die Weiterleitung dieser Ereignisse an die Anwendung. Die zweite Variante erfolgt über den *XletManager*. Dieser erhält die Tastensignale vom *KeyDispatcher*, was über die gleiche Verbindung erfolgen kann, die schon vom *ApplicationManager* genutzt wird. Ein zum *XletManager* hinzugefügter Prozess leitet die Ereignisse dann als Java AWT Events an die Applikation weiter.

[3] vgl. Kapitel 4.2.4

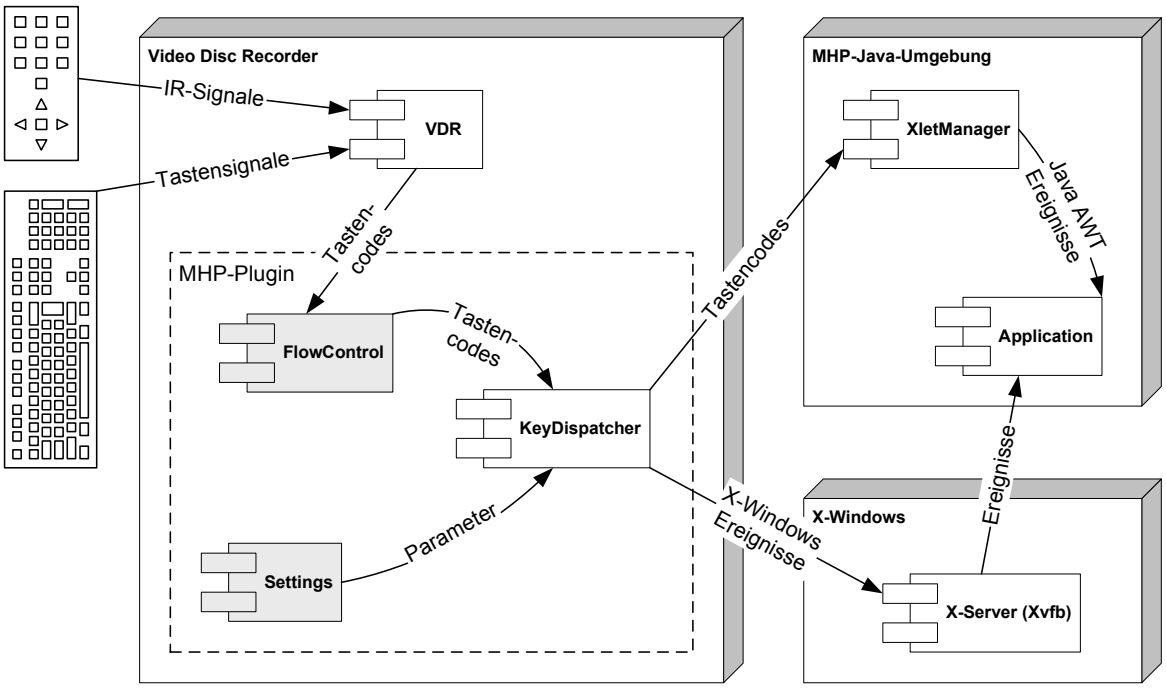


Abbildung 4.24.: Architektur des Pakets ApplicationControl

Statische Struktur

In Abbildung 4.25 wird die Klassenstruktur angegeben, die für die Realisierung der Funktionalitäten des Pakets entworfen wurde. Auch hier sind die Klassen nach den Modulen geordnet, denen sie angehören.

Die Unterstützung der Variabilitäten des Moduls *KeyDispatcher* wird über eine Klassenhierarchie erreicht. Die Basisklasse *cKeyDispatch* definiert die Schnittstelle, über die das Modul *FlowControl*, speziell die Klasse *cCore*, die Steuerung der Applikation auslösen kann. Dafür steht die Methode

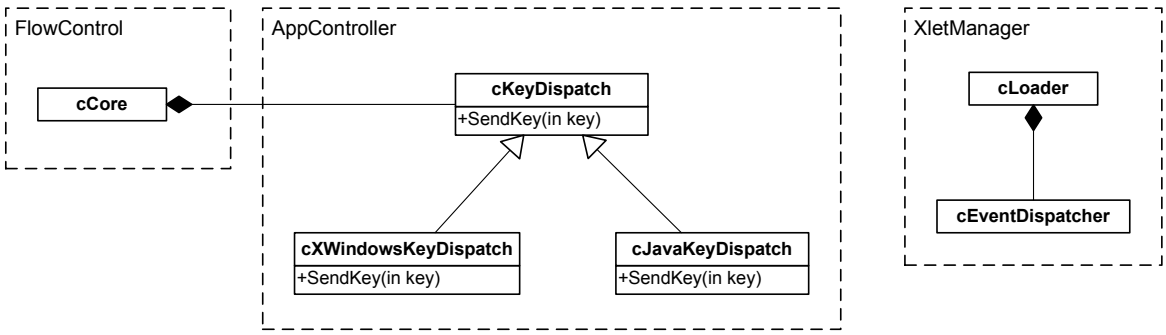


Abbildung 4.25.: Statische Struktur des Pakets ApplicationControl

SendKey zur Verfügung. Die von *cKeyDispatch* abgeleiteten Klassen realisieren die Steuerungsvarianten: *cXWindowsKeyDispatch* für das Senden von X-Windows Events und *cJavaKeyDispatch* für das Senden von Java AWT Ereignissen. Die Wahl der durch die Klassen repräsentierten Varianten kann in jedem Fall statisch bei der Kompilierung erfolgen, z. B. mit Hilfe von Compiler-Direktiven. Die Klassenhierarchie erlaubt es jedoch, eine dynamische Parametrisierung und damit die Wahl der Variante während der Laufzeit des Systems durchzuführen. Dies wird durch den Polymorphismus, ein Konzept der Objektorientierung, und das so genannte späte Binden ermöglicht, sofern bei der Deklaration der Schnittstelle geeignet vorgegangen wird[4].

Damit die Variante des Sendens von Java Ereignissen funktionieren kann, muss der *XletManager* noch erweitert werden. Dies geschieht mit der Klasse *cEventDispatcher*, die für das Weiterleiten der vom *KeyDispatcher* gesendeten Tastencodes in Form von Ereignissen des Advanced Windowing Toolkits (AWT) verantwortlich ist.

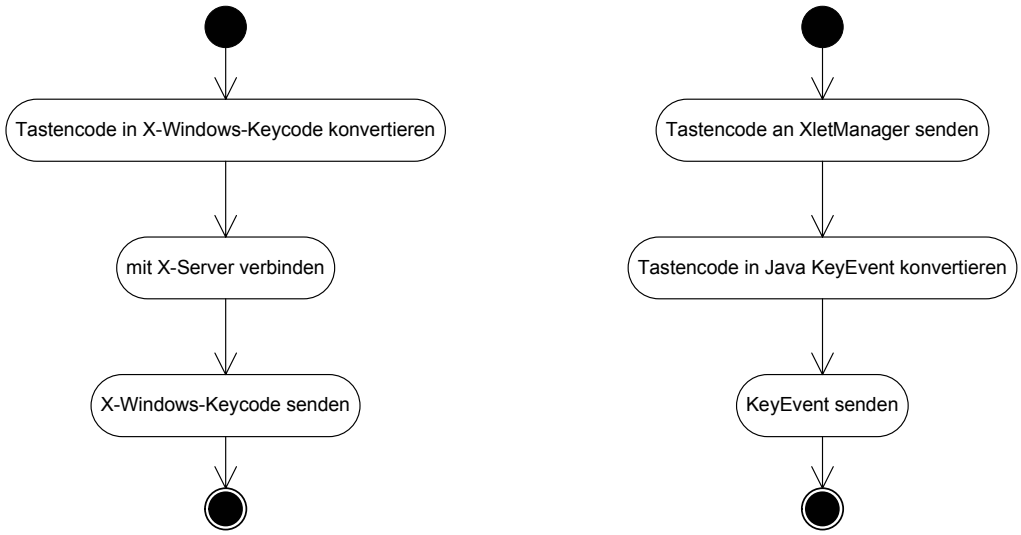
Dynamische Aspekte

Die Aktivitäten des Pakets *AppliactionControl* beschränken sich auf das Weiterleiten der Tastensignale an die gerade ausgeführte MHP-Applikation. Das Erfassen der Signale wird vom VDR und der Klasse *cCore* der Kernfunktionen übernommen. Diese löst nach deren Auftreten den Prozess der Applikationssteuerung aus, woraufhin eine der zwei Varianten des Sendens der Ereignisse gestartet wird, deren Arbeitsweise die Abbildung 4.26 zeigt.

Bevor das Senden der Ereignisse an den X-Server beginnen kann, muss der im VDR-internen Format vorliegende Tastencode in einen X-Windows-Keycode konvertiert werden. Allerdings ist das nicht immer direkt möglich, da nicht jeder Taste auf der Infrarotfernbedienung ein Keycode entspricht. Ein Beispiel dafür sind die Farbtasten (Rot, Grün, Gelb und Blau). Diese werden stattdessen auf die Funktionstasten F1 bis F4 abgebildet, was aber bei den meisten MHP-Implementierungen kein Problem darstellt, da diese häufig die Funktionstasten intern wieder durch die farbigen Tasten ersetzen. Liegt der Keycode vor, wird als nächstes mit Hilfe von Funktionen des X-Windows-API eine Verbindung zum X-Server aufgebaut. Über diese wird schließlich der Keycode als Tastendruck-Ereignis an das X-Windows System gesendet.

Für die Weiterleitung der Tastencodes als Java Events müssen diese als erstes an den *XletManager* gesendet werden. Dafür kann entweder die bestehende Verbindung des *ApplicationManagers* genutzt

[4] In der Programmiersprache C++ wird hierfür das Schlüsselwort *virtual* verwendet.



(a) (b)

Abbildung 4.26.: Arbeitsweise der Varianten des *KeyDispatcher*, das Senden von X-Windows Ereignissen (a) und Java AWT Ereignissen (b)

werden, oder eine eigene Art der Verbindung, die zuvor aufgebaut werden muss. Die weiteren Aktivitäten laufen im *XletManager* ab. Zunächst muss auch hier eine Konvertierung des Tastencodes in ein Java KeyEvent erfolgen. Dies ist in allen Fällen direkt möglich, da die APIs der MHP-Schnittstelle eine Erweiterung der Java Events um die auf Fernbedienungen vorkommenden Tasten beinhalten. Zum Abschluss der Operation wird das Ereignis an die Applikation gesendet.

4.6.5. Das Paket Output

Das Paket *Output* ist für die Präsentation der MHP-Anwendungen auf dem Fernseher verantwortlich. Dies umfasst die Erfassung und Darstellung der grafischen Ausgabe der Applikationen, wofür jeweils verschiedene Varianten existieren.

Funktionale Architektur

Die Architektur des Pakets wird in Abbildung 4.27 gezeigt. Wie zu erkennen ist, fügt das Paket dem MHP-Plugin zwei neue Module hinzu, *Capture Output* und *Display Output*. Auch dieses Paket kommt nicht ohne die Funktionalitäten und Daten aus, die von den *CoreFunctions* angeboten werden. Außerdem kann, je nach verwendeter Variante der Ausgabeerfassung[5], eine Erweiterung des

[5] vgl. Kapitel 4.2.5

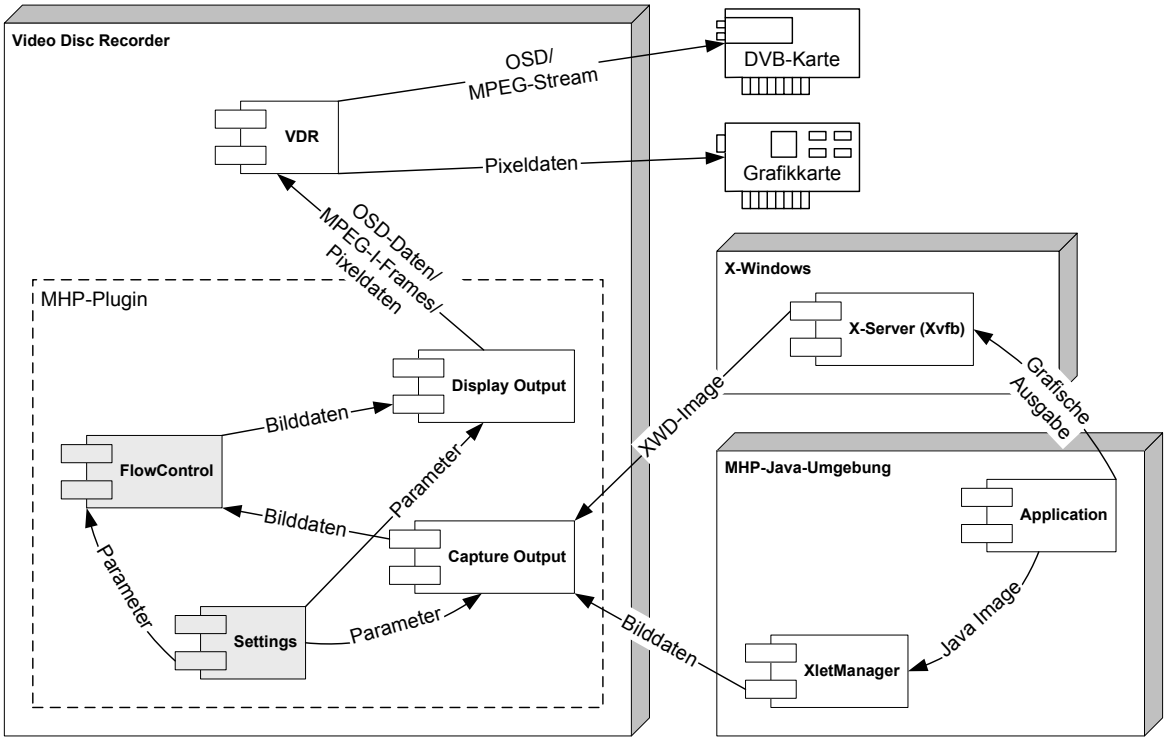


Abbildung 4.27.: Architektur des Pakets Output

XletManager nötig werden.

Das Modul *FlowControl* kümmert sich um die zeitgesteuerte Durchführung der Ausgabeprozesse und gibt die von der Erfassung zurückgelieferten Bilddaten an die Darstellung weiter. Das Modul *Settings* stellt den Komponenten des Pakets die Werte verschiedener Parameter zur Verfügung.

Die Erfassung des grafischen Outputs der MHP-Applikation wird von dem Modul *Capture Output* durchgeführt. Dafür wurden zwei Varianten ermittelt: Der X-Window Dump und die Java-interne Erfassung. Ersterer erfolgt durch Kommunikation mit dem X-Server, der die Applikation darstellt, indem ein Speicherabbild des Framebuffers erzeugt und in Form eines Bildes im XWD-Format zurückgeliefert wird. Für diese Variante werden die Funktionen des X-Windows API genutzt. Die Java-interne Erfassung wird dagegen vom *XletManager* durchgeführt. Dieser ermittelt die grafische Ausgabe der MHP-Applikation im Java-internen Format. Die Daten des Bildes werden an das Modul *Capture Output* gesendet, was über die gleiche Verbindung geschehen kann, die auch vom *ApplicationManager* genutzt wird.

Das Modul *Display Output* sorgt für die Darstellung der von *Capture Output* gelieferten Bilddaten auf dem Fernseher, wofür in der Merkmalserfassung drei Varianten ermittelt wurden. Eine davon, die

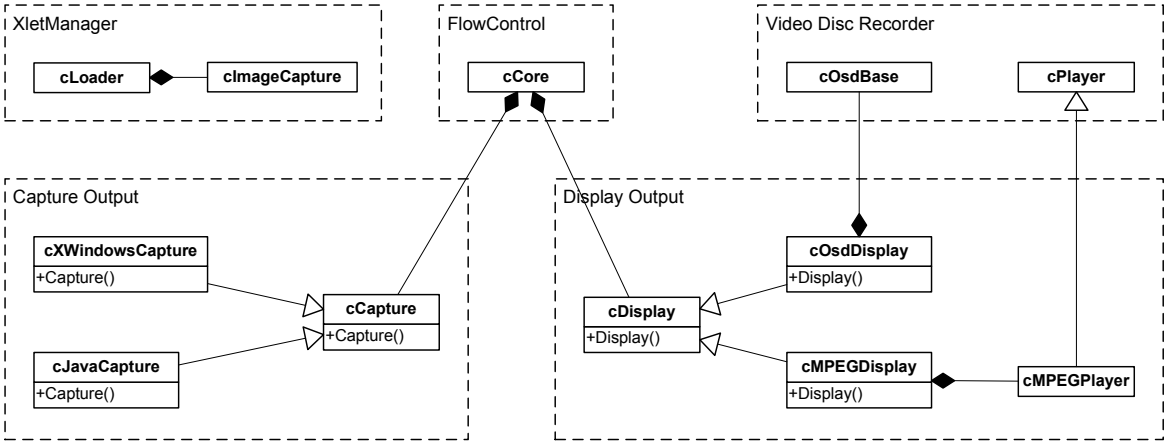


Abbildung 4.28.: Statische Struktur des Pakets Output

Ausgabe über die Grafikkarte, soll hier nur erwähnt und nicht weiter berücksichtigt werden, da der VDR keine Möglichkeit vorsieht den Framebuffer der Grafikkarte direkt anzusprechen. Die einzige Möglichkeit eine Grafikkarte als Ausgabegerät in den VDR zu integrieren ist die Verwendung der cDevice-Schnittstelle. In diesem Fall muss sie aber wie die DVB-Karte angesprochen werden, das heißt die Ausgabe ist nur per OSD oder MPEG-Stream möglich. Die anderen beiden Varianten der Darstellung erfolgen über die DVB-Karte des Rechners. An diese sendet das *Capture Output* Modul über die Methoden des VDR das Bild, das die Ausgabe der Applikation enthält, in Form von OSD-Daten oder als MPEG-kodiertes Einzelbild. Die DVB-Karte stellt diese dann im OSD bzw. über das Video-Device auf dem Fernseher dar.

Statische Struktur

Für die Realisierung der Funktionalitäten des Pakets wurde die Klassenstruktur entworfen, die in Abbildung 4.28 angegeben wird. Auch hier findet eine Ordnung der Klassen nach den Modulen, denen sie angehören, statt. Die Umsetzung der Variablilitäten bei der Erfassung und Darstellung der grafischen Ausgabe erfolgt nach den gleichen Prinzipien, die auch im Paket *ApplicationControl* angewendet werden. Unter Nutzung von Polymorphie und spätem Binden wird damit im Paket *Output* sowohl die statische als auch die dynamische Auswahl der Variante ermöglicht.

Die Realisierung der Funktionalitäten des Moduls *Capture Output* erfolgt über eine Klassenhierarchie, deren Basis die Klasse *cCapture* darstellt. Diese definiert die Schnittstelle, über die das Modul *FlowControl* mit der Klasse *cCore* die Erfassung der Ausgabe einleitet. Die Methode *Capture*, die das Ergebnis der Operation als Bild im RGBA-Format zurückliefert, steht dafür zur Verfügung. Die

Subklassen von *cCapture* implementieren das Verhalten der Schnittstelle entsprechend der Variante, die sie repräsentieren. Die Klasse *cXWindowsCapture* realisiert die Erfassung über einen X-Window Dump und die Klasse *cJavaCapture* die Java-interne Ermittlung der Ausgabe. Die zweite Variante erfordert noch eine Erweiterung des *XletManager*-Moduls in Form der Klasse *cImageCapture*. Diese ist für die Erfassung der Ausgabe der Applikation als Java Image und das Senden der Bilddaten zurück an das MHP-Plugin verantwortlich.

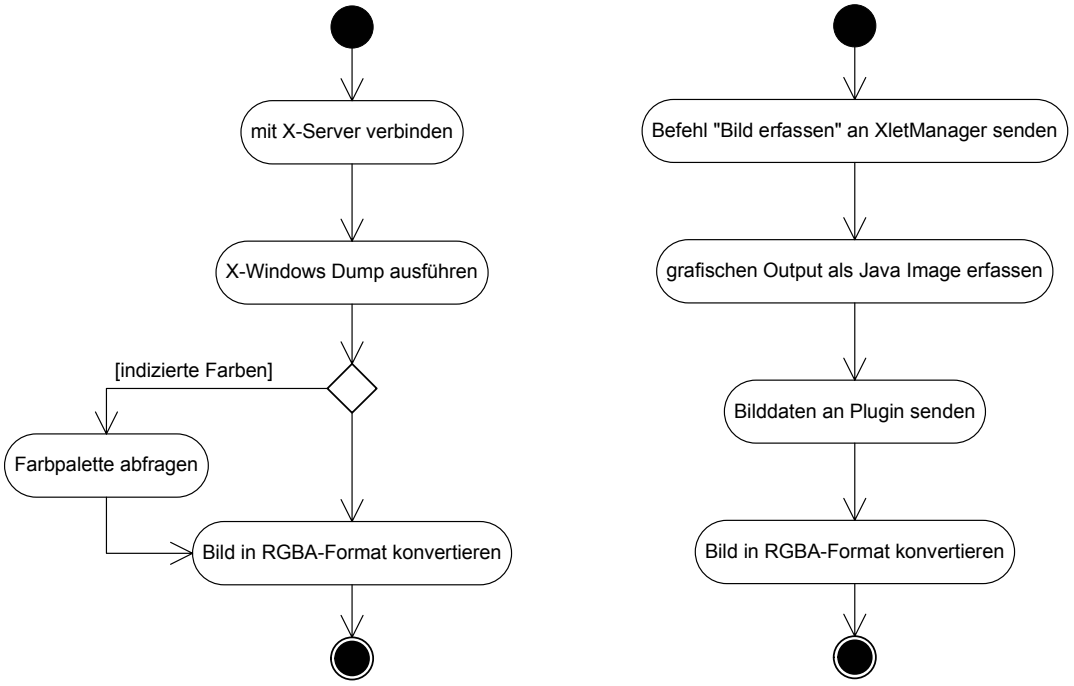
Auch das Modul *Display Output* wird als Klassenhierarchie umgesetzt. Die Basisklasse *cDisplay* definiert mit der Methode *Display* die Schnittstelle, die von der Ablaufsteuerung genutzt wird um die Darstellung der Ausgabe auszulösen. Die Methode erwartet das auszugebende Bild im RGBA-Format. Die Varianten der Darstellung werden durch die Subklassen von *cDisplay* realisiert, die das Verhalten der Methode implementieren. Die Klasse *cOsdDisplay* verwirklicht die Darstellung der Ausgabe im OSD der DVB-Karte, wobei auf die Methoden der VDR-Klasse *cOsdBase* zurückgegriffen wird. Die Ausgabe als MPEG-kodierte Einzelbilder über das Video-Device der DVB-Karte übernimmt die Klasse *cMpegDisplay*. Dabei wird die eigentliche Darstellung der MPEG-I-Frames von *cMpegPlayer* durchgeführt, unter Verwendung der von der VDR-Klasse *cPlayer* angebotenen Funktionen.

Dynamische Aspekte

Die Aktivitäten des Pakets *Output* umfassen die Erfassung und Darstellung der grafischen Ausgabe der MHP-Applikationen. Die wiederholte zeitliche Abfolge dieser Prozesse wird von der Klasse *cCore* der *CoreFunctions* gesteuert. Im Folgenden wird die Funktionsweise der verschiedenen Varianten der Ausgabeerfassung und -darstellung erläutert.

Das Aktivitätsdiagramm in Abbildung 4.29 zeigt die Arbeitsweise der zwei Varianten der Ausgabeerfassung als X-Window Dump oder mittels Java-interner Funktionen.

Das Erfassen des grafischen Output über das API des X-Windows Systems beginnt mit dem Verbindungsaufbau des MHP-Plugins zum X-Server, in dem die Ausgabe der Applikation dargestellt wird. Danach wird mit der Funktion *XGetImage* ein Speicherabzug von dessen Framebuffer durchgeführt, welcher im XWD-Format geliefert wird. Handelt es sich dabei um ein Bild, das indizierte Farben verwendet, falls der X-Server beispielsweise mit einer Farbtiefe von 8 Bit läuft, muss noch ein Abfragen der Farbpalette erfolgen. Bevor die Operation beendet ist und die Daten des X-Window Dump zurückgeliefert werden, findet noch eine Konvertierung des Bildes in das RGBA-Format statt.



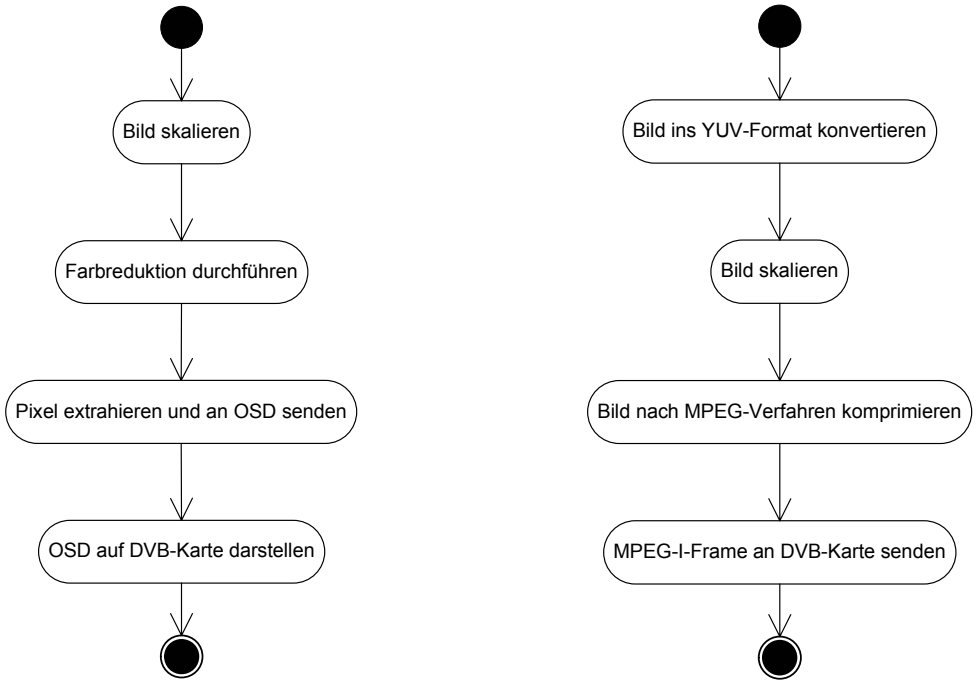
(a) (b)

Abbildung 4.29.: Arbeitsweise der Varianten der Ausgabeerfassung, mittels X-Window Dump (a) und Java-interner Erfassung über den XletManager (b)

Bei der Java-internen Erfassung muss zunächst an den *XletManager* der Befehl zum Erfassen der Ausgabe gesendet werden. Das kann über eine bestehende Verbindung geschehen erfolgen, gegebenenfalls muss diese zuvor noch aufgebaut werden. Der *XletManager* erzeugt daraufhin ein Abbild der Applikation in Form eines Java Image. Anschließend werden die Daten des Bildes an das MHP-Plugin zurück gesendet, wo abschließend eine Konvertierung in das RGBA-Format vorgenommen wird.

In Abbildung 4.30 ist die Arbeitsweise der Varianten der Ausgabedarstellung, über das OSD bzw. das Video-Device der DVB-Karte, in Form von Aktivitätsdiagrammen angegeben.

Die Darstellung der Ausgabe im OSD der DVB-Karte beginnt mit der Skalierung des übergebenen Bildes auf die Größe des On-Screen-Displays. Diese ist abhängig vom verfügbaren Speicher auf der Karte und der gewünschten Farbtiefe der Darstellung. Nach der Skalierung erfolgt die Reduktion der Anzahl der im Bild vorkommenden Farben entsprechend der Farbtiefe des OSD. Mit Hilfe der dabei erhaltenen Farbpalette werden die einzelen Pixel des Bildes wieder ins RGBA-Format konvertiert und über die Methoden der Klasse *cOsdBase* in das OSD gezeichnet. Dies wird anschließend mit dem Aufruf der Funktion *Flush* auf der DVB-Karte dargestellt, womit die Operation beendet ist.



(a) (b)

Abbildung 4.30.: Arbeitsweise der Varianten der Ausgabedarstellung, über das OSD (a) und das Video-Device (b) der DVB-Karte

Bei der Darstellung der Ausgabe als MPEG-Stream über das Video-Device der DVB-Karte muss zuerst eine Konvertierung der im RGBA-Format vorliegenden Daten in das vom MPEG-Encoder verwendete YUV-Format erfolgen. Danach wird eine Skalierung des Bild auf die Auslösung des Fernsehsignals[6] vorgenommen. Anschließend erfolgt die Komprimierung nach dem MPEG-Verfahren, die als Ergebnis ein I-Frame liefert. Dieses wird zum Abschluss der Operation an das Video-Device der DVB-Karte gesendet und damit auf dem Fernseher dargestellt.

4.7. Bindung der Variabilitäten

Bei der Generierung eines Systems nach den Anforderungen des Kunden können die im Merkmalmodell festzulegenden Variabilitäten auf zwei Arten im generierten MHP-Plugin gebunden werden. Zum Einen ist eine statische Bindung während des Kompilierens möglich. Diese hat den Vorteil, dass nur die wirklich benötigten Quelltextteile eingebunden werden, was den Speicherbedarf des Systems gering hält. Diese Variante eignet sich eher für Systeme, die nach Kundenwünschen generiert wer-

[6] bei PAL: 720x576

den. Im Gegensatz dazu steht die dynamische Bindung. Hier wird der gesamte Quelltext kompiliert und zum Gesamtsystem zusammengefügt. Die Festlegung der Variabilitäten erfolgt erst während der Laufzeit über Parameter der *Settings*-Komponente. Diese Variante eignet sich eher zu Demonstrationszwecken. Der Vorgang des statischen Bindens der Variabilitäten wird im Folgenden noch einmal an einem Beispiel erläutert.

Angenommen der Kunde wünscht aufgrund gegebener Hardwarevoraussetzungen, dass die Darstellung der grafischen Ausgabe der MHP-Applikationen als MPEG-Stream über das Video-Device der DVB-Karte erfolgt. In diesem Fall wählt er die entsprechende Variante im Merkmalmodell aus. Des Weiteren legt der Kunde die Standardwerte der Parameter des Systems fest. Ein Generator erzeugt aus den festgelegten Variabilitäten eine Header-Datei, welche die gewünschte Variante „MPEG-Ausgabe“ in Form einer Compiler-Direktiven spezifiziert und von allen Quelltextdateien des MHP-Plugins, die die Ausgabe betreffen, eingebunden wird. Im gleichen Schritt werden die Standardwerte der Parameter in eine Konfigurationsdatei geschrieben, die später zur Laufzeit von dem MHP-Plugin über Mechanismen des VDR eingelesen wird. Beim Kompilieren des Systems werden die Quelltextdateien vom Präprozessor analysiert, die durch die Direktiven festgelegten Fragmente extrahiert und an den Compiler zur weiteren Verarbeitung übergeben. Dieser erstellt daraus den fertigen Maschinencode des MHP-Plugins, welches bei der Ausgabedarstellung nur den Code der MPEG-Ausgabe enthält.

Die Erläuterung der Aktivierung der Merkmale auf Quelltextebene sowohl für die statische als auch für die dynamische Bindung erfolgt in Kapitel 5.2.2.

4.8. Bewertung der Architektur

Aus dem in Kapitel 4.2 vorgestellten Merkmalmodell wurde in den vorangegangenen Abschnitten eine Architektur für die MHP-DVP-Integration entworfen, die sich durch einen modularen Aufbau auszeichnet. Dieser garantiert eine intuitive Anpassbarkeit an Kundenwünsche und gegebene Hardware- und Softwareumgebungen, sowie die einfache Wart- und Erweiterbarkeit bei sich ändernden Anforderungen.

Beim Entwurf der Architektur wurde auf eine möglichst direkte Abbildung der Merkmale auf die Komponenten des Systems Wert gelegt, was auch in den meisten Fällen gelungen ist. Des Weiteren wurde durch die Verwendung fester Schnittstellen eine weitgehende Unabhängigkeit der Pakete untereinander erreicht. Dies sorgt für für die einfache Wartbarkeit und Erweiterbarkeit des Systems.

Das Merkmalmodell beinhaltet eine Reihe von Variabilitäten und Parametern. Durch deren Festlegung, abhängig von den spezifischen Anforderungen des Kunden, kann eine ganze Familie von Systemen für die MHP-DVP-Integration erzeugt werden. Allein durch die Wahl der Varianten bei Applikationsteuerung und Ausgabe werden 36 verschiedene Systeme ermöglicht[7]. Diese unterscheiden sich zum Teil erheblich voneinander in ihren Hardware- und Softwarevoraussetzungen. Die Zahl der Systeme kann durch die Variation der die Merkmale beeinflussenden Parameter um ein Vielfaches gesteigert werden. Da sowohl bei der Applikationsteuerung als auch bei der Ausgabe weitere Varianten denkbar wären und diese wegen der modularen und flexiblen Architektur auch einfach hinzugefügt werden könnten, ist der Zahl von erzeugbaren Systemen nach oben keine Grenze gesetzt.

Die prinzipielle Realisierbarkeit der entworfenen Architektur für die MHP-DVP-Integration wird durch die prototypische Implementierung des Systems gezeigt, auf die im folgenden Kapitel eingegangen wird.

[7] Dies ergibt sich aus der Anzahl von Kombinationen bei 3 Varianten für Tastendruck erfassen, 2 Varianten für Tastendruck weiterleiten, 2 Varianten der Ausgabeerfassung und 3 Varianten der Ausgabedarstellung.

5. Prototypische Implementierung

Nachdem im vorhergehenden Kapitel die Architektur für die MHP-DVP-Integration entworfen wurde, soll im Folgenden der entwickelte Prototyp vorgestellt werden, der diese Architektur realisiert.

5.1. Hardware- und Softwareumgebung

Die Implementierung wurde auf dem Standard-PC des Digitalen Video Projekts mit Pentium 4 1.5 GHz, 512 MB RAM und 40 GB Festplatte durchgeführt. Des Weiteren verfügt der Rechner über zwei DVB-Karten, in einer Premium- und einer Budget-Ausführung, die über eine angeschlossene Satellitenantenne mit dem DVB-Signal versorgt werden. Die Ausgabe erfolgt auf einen an die DVB-Karte angeschlossenen Fernseher. Die Steuerung kann sowohl über Tastatur als auch mit einer Infrarotfernbedienung erfolgen. Als Betriebssystem kommt SuSE Linix 8.0 zum Einsatz. Die weitere zugrunde gelegte Software ist der Linux DVB Treiber Version 1.0.1 und VDR in der Version 1.2.5. Die Entwicklung des Plugins erfolgte mit der Programmiersprache C++, für den XletManager wurde Java verwendet.

5.1.1. Benötigte Softwarepakete

Wie in Kapitel 4.1.3 festgestellt, benötigt das System der MHP-DVP-Integration neben dem im Digitalen Video Projekt obligatorischen Video Disc Recorder noch einige weitere Softwarekomponenten. Außerdem benötigen einige der Pakete des Systems bestimmte Funktionalitäten, die von externen Bibliotheken angeboten werden. Im Folgenden wird beschrieben, welche konkreten Softwarekomponenten in welcher Version verwendet werden.

Als X-Windows System kommt das freie XFree86 in der Version 4.3.0 zum Einsatz, welches im Lieferumfang des auf dem DVP-Rechner installierten SuSE Linux bereits enthalten ist. Dies beinhaltet auch den X-Server Xvfb, dessen Ausgabe ohne die Nutzung einer Grafikkarte in einen virtuellen Framebuffer im Hauptspeicher des Rechners abgelegt wird.

Als Java-Laufzeitumgebung wird die Originalversion von Sun verwendet. Die Variante, die von IBM angeboten wird[1], bot zwar nach kurzen Tests in einigen Bereichen eine bessere Performance, allerdings traten Inkompatibilitäten zu den anderen Komponenten der MHP-Java-Umgebung auf, die zu Fehlern in der Darstellung führten. Deshalb wurde schließlich die Version 1.4.2 der Laufzeitumgebung von Sun verwendet. Diese kann genau wie das Java Media Framework in der Version 2.1.1 und JavaTV in der Version 1.0 von der Java-Homepage von Sun[2] bezogen werden. Für die letzte Komponente der MHP-Java-Umgebung, die APIs der MHP-Schnittstelle, gelten einige Besonderheiten, auf die später noch einmal näher eingegangen wird.

Die Ausgabedarstellung benötigt zwei externe Bibliotheken, die für das Kodieren nach dem MPEG-Verfahren vor der Darstellung über das Video-Device und die Farbreduktion vor der OSD-Ausgabe verantwortlich sind. Ersteres wird von der Bibliothek *libavcodec* übernommen. Diese ist Bestandteil des „FFmpeg video and audio converter“[3], welcher in Version 0.4.8 verwendet wird und eine Reihe von Encodern und Decodern für die verschiedensten Audio- und Video-Formate zur Verfügung stellt. Für die Farbreduktion wird das Paket *ImageMagick*[4] in der Version 5.5.4 verwendet. Dieses beinhaltet auch eine Bibliothek für die Programmiersprache C/C++, die neben Konvertern für die verschiedensten Grafikformate auch Funktionen zur Bildbearbeitung wie Skalierung und Farbreduktion bietet.

5.1.2. Einschränkungen bei der MHP-Java-Umgebung

Für die Realisierung eines Prototyps der MHP-DVP-Integration wurde auch eine Implementierung der MHP-Schnittstelle benötigt. Allerdings existierte zum Zeitpunkt des Beginns dieser Arbeit keine frei verfügbare Version der MHP APIs, weshalb nach Alternativen gesucht werden musste. Dabei wurden verschiedene Entwicklungsumgebungen für MHP-Applikationen untersucht, die in eingeschränkten Testversionen aus dem Internet heruntergeladen werden konnten. Zwar beinhalten die Programme meist die kompletten Java Pakete der MHP-Schnittstelle, in der Regel können diese aber nicht aus der Entwicklungsumgebung extrahiert und für eigene Zwecke verwendet werden. Die einzige der untersuchten Entwicklungsumgebungen, bei der dies möglich war, ist GEAR von der SNAP2 Corporation[5].

[1] http://www-106.ibm.com/developerworks/
[2] http://java.sun.com/
[3] http://ffmpeg.sf.net/
[4] http://www.imagemagick.org/
[5] http://www.snaptwo.com/

SNAP2 GEAR beinhaltet alle Pakete der MHP-Schnittstelle, die für die grafische Ausgabe der MHP-Applikationen benötigt werden. Dazu zählen die komplette HAVi Level 2 GUI und Teile der DVB und DAVIC API. Leider fehlen in GEAR viele Pakete, die für das Ausführen der Anwendungen vonnöten wären, die von den Fernsehsendern heutzutage ausgestrahlt werden. Dies umfasst alle Pakete, die die Steuerung der DVB-Karte, beispielsweise den Zugriff auf Service Information, MPEG-2 Section Filter und Tuner, erlauben. Damit trotzdem eine Demonstration des entwickelten Prototyps durchgeführt werden kann, wurde unter Verwendung der SNAP2 GEAR Pakete eine eigene MHP-Applikation in Form eines kleinen Quiz realisiert, welche von einigen der grafischen Widgets der MHP, wie Button, Text und Bild, Gebrauch macht.

Kurz vor Abschluss der vorliegenden Arbeit wurde von einem Projekt namens XleTView[6] erfahren, das den Beginn einer freien Implementierung der MHP-Schnittstelle unter der GNU GPL darstellt und dessen erste Version Mitte Juli 2003 freigegeben wurde. XleTView beinhaltet alle Klassen der MHP-Schnittstelle, allerdings ist deren Implementierung in vielen Fällen noch unvollständig. Beispielsweise im Bereich der grafischen Ausgabe ist es SNAP2 GEAR noch unterlegen. Aus diesem Grund wurde schließlich für den Prototyp der MHP-DVP-Integration eine Kombination aus SNAP2 GEAR und den Paketen von XleTView verwendet. Damit wird es sogar möglich einige der von den Sendeanstalten übermittelten MHP-Applikationen mit Einschränkungen auszuführen.

5.2. Implementierung des Prototyps

5.2.1. Realisierte Fähigkeiten

Bei der Implementierung des Systems der MHP-DVP-Integration konnten nicht alle Teile des Merkmalmodells berücksichtigt werden, da die vollständige Umsetzung der Pakete der Architektur den Rahmen dieser Arbeit deutlich überschritten hätte. Allerdings ist für das Zeigen der prinzipiellen Realisierbarkeit keine komplette Umsetzung der Architektur notwendig. Im folgenden wird beschrieben, welchen Umfang die Implementierung der einzelnen Pakete besitzt und an welchen Stellen Einschränkungen vorgenommen wurden.

Das Paket *CoreFunctions* konnte fast vollständig implementiert werden. Natürlich mussten an den Stellen Teile weggelassen werden, die Funktionen der anderen Pakete auslösen, die selbst aber nicht realisiert wurden.

[6] http://sourceforge.net/projects/xletview

Die umgesetzten Funktionen des Pakets *DataReceiving* beschränken sich auf den Empfang und die Speicherung der Daten, die zum Zeitpunkt des Kanalwechsels, und damit beim Starten des Datenempfangs, verfügbar sind. Sollten während dem Verweilen auf einem Programm neue Applikationen oder aktualisierte bzw. neue Dateien übertragen werden, können diese nicht berücksichtigt werden.

Die Implementierung des Pakets *ApplicationManaging* umfasst nur die Fähigkeit zum manuellen Starten und Beenden von MHP-Applikationen, was deren Auswahl aus einer im OSD dargestellten Liste beinhaltet. Die automatische Durchführung dieser Vorgänge vor bzw. nach einem Kanalwechsel wurde nicht umgesetzt. Außerdem wurde auf die Unterstützung des Pause-Zustands verzichtet.

Das Paket *ApplicationControl* wurde in seinen Grundzügen komplett umgesetzt. Nur bei den Varianten der Weiterleitung des Tastendrucks wurden Eingeschränkungen vorgenommen. Hier wurde nur das Senden über das X-Windows Systems implementiert. Da das Erfassen der Tastensignale von verschiedenen Steuerungskomponenten unabhängig vom MHP-Plugin durch den VDR durchgeführt wird, waren bei diesem Teil des Pakets keine Beschränkungen notwendig.

Auch die Implementierung des Pakets *Output* ist ist nur in seinen Variabilitäten beschränkt worden, die grundlegenden Funktionalitäten wurden komplett realisiert. Die Erfassung der Ausgabe beschränkt sich auf die Umsetzung des X-Window Dump. Bei deren Darstellung wurden beide Varianten, die die DVB-Karte als Ausgabegerät nutzen, implementiert. Damit wird die Demonstrierung der dynamischen Aktivierung der Merkmale durch den Wechsel der Darstellungsvariante während der Laufzeit des Systems ermöglicht.

5.2.2. Aktivierung der variablen Merkmale

Bei der Erzeugung eines Systems für die MHP-DVP-Integration müssen die gewählten variablen Submerkmale der Applikationssteuerung und Ausgabe im Quelltext entsprechend aktiviert werden. Dies kann auf verschiedene Arten erfolgen.

Eine Möglichkeit der Aktivierung der Merkmale ist die statische Parametrisierung während des Kompilieren des Systems. Dabei wird in einer gesonderten Header-Datei namens „config.h“ über die Präprozessordirektiven *#define* und *#undef* festgelegt, welche Merkmale das kompilierte Programm beinhalten soll. Dies zeigt das folgende Listing am Beispiel der Ausgabedarstellung, in dem die Darstellung per MPEG-kodiertem Einzelbild aktiviert wird.

```
#undef DISPLAY_OSD
#define DISPLAY_MPEG
```

Die Header-Datei muss dann in die ensprechenden Implementierungsdateien eingebunden werden. Dort wird durch Einbettung der variablen Quelltextfragmente zwischen die Direktiven *#ifdef* und *#endif* festgelegt, welche Quelltextteile für die Kompilierung des System verwendet werden sollen. Das folgende Listing zeigt dies am Beispiel der Erzeugung der Instanz für die Ausgabedarstellung.

```
#include "config.h"

[...]

#ifdef DISPLAY_OSD
    cOsdDisplay * display;
    display = new cOsdDisplay(MhpSetup.displayOsdBpp);
#endif
#ifdef DISPLAY_MPEG
    cMpegDisplay * display;
    display = new cMpegDisplay(MhpSetup.displayMPEGQuality);
#endif
```

Die Auswertung der Direktiven wird durch den Präprozessor übernommen, der die nicht gewünschten Quelltextfragmente ausblendet und das Ergebnis an den Compiler weiterleitet.

Neben der Aktivierung der Merkmale während des Kompilieren des Systems erlaubt die entworfene Klassenstruktur auch die dynamische Parametrisierung. Damit ist die Festlegung der variablen Merkmale auch während der Laufzeit möglich. Wie dies funktioniert, zeigt das folgende Listing am Beispiel der Ausgabedarstellung.

```
1  // Deklaration der Variable
2  cDisplay * display;
3
4  [...]
5
6  // Erzeugen der Instanz
7  switch (MhpSetup.displayType)
8  {
9      case kDisplayOsd:
10         display = new cOsdDisplay(MhpSetup.displayOsdBpp);
11         break;
12     case kDisplayMPEG:
13         display = new cMpegDisplay(MhpSetup.displayMPEGQuality);
14         break;
15 }
16
17 [...]
18
19 // Auslösen der Darstellung
```

```
20 display->Display(image);
```

Die Variable *display*, über die der Zugriff auf die Ausgabedarstellung erfolgt, wird als Zeiger auf die Basisklasse *cDisplay* deklariert (Zeile 2). Der Parameter *MhpSetup.displayType*, der in den *Settings* gespeichert wird, legt die Art der Darstellung fest. Abhängig von dessen Wert wird entweder eine Instanz der Klasse *cOsdDisplay* (Zeile 10) oder der Klasse *cMpegDisplay* (Zeile 13) erzeugt und der Variable *display* zugewiesen. Dabei werden auch gleich die die Darstellung beeinflussenden Parameter Farbtiefe und Kodierqualität übergeben. Das Auslösen der Ausgabedarstellung erfolgt über den Zugriff auf die Schnittstelle der Basisklasse (Zeile 20). Der Methodenpolymorphismus sorgt dabei automatisch dafür, dass die richtige Version der Funktion entsprechend der Art der Darstellung ausgeführt wird.

Für den entwickelten Prototyp wurde die dynamische Parametrisierung verwendet. Das erlaubt den Wechsel der Darstellungsvariante während der Laufzeit und dient hauptsächlich zu Demonstrationszwecken. Für Systeme, die nach Kundenwünschen durch Auswahl der Merkmale zusammengestellt werden, ist jedoch die statische Parametrisierung in der Regel ausreichend, da bekannt ist, welche Varianten genutzt werden können. Außerdem ist der Speicherbedarf des Systems geringer, da nur die Teile kompiliert werden, die auch vonnöten sind.

5.2.3. Programmstruktur

Um eine bessere Übersichtlichkeit und einfache Wart- und Erweiterbarkeit sicherzustellen wurde die Implementierung der Komponenten des Systems auf mehrere Quellcodedateien verteilt. Die Folgenden zwei Tabellen geben an, in welchen Implementierungsdateien des MHP-Plugins und des XletManagers welche Klassen bzw. Funktionen zu finden sind.

Implementierungsdateien des MHP-Plugins

Datei	Inhalt/Klassen
application.c/.h	cApplication, cApplications, cApplicationManager
bit.c/.h	cBitStream
capture.c/.h	cCapture, cXWindowsCapture
carousel.c/.h	cCarousel, cCarousels
control.c/.h	cKeyDispatch, cXWindowsKeyDispatch

Datei	Inhalt/Klassen
core.c/.h	cCore
display.c/.h	cDisplay, cOsdDisplay, cMpegDisplay, cMpegPlayer
global.c/.h	Hilfsfunktionen für Fehler-, Info-, Debug-Meldungen und für die Zeitmessung
javastart.c/.h	cJavaStarter
mhp.c/.h	cPluginMhp
module.c/.h	cModule, cModules
receiver.c/.h	cDataReceiver
section.c/.h	cSection, cPATSection, cPMTSection, cDSMCCSection, cAITSection
setup.c/.h	cMhpSetup, cMhpSetupMenu
status.c/.h	cStatusMonitor
ts.c/.h	cTSPacket
types.h	Datentyp für internes RGB-Bildformat

Implementierungsdateien des XletManagers

Datei	Inhalt/Klassen
cLoader.java	cLoader
cXletLifecycleControl.java	cXletLifecycleControl
cXletLifecycleControlClient.java	cXletLifecycleControlClient
cXletManager.java	cXletManager
loadxlet.sh	Shellscript zum Starten der Java VM mit dem XletManager

5.3. Ergebnis

In den folgenden Abschnitten werden einige Teilaspekte des Ergebnisses der Entwicklung des Prototyps näher beleuchtet. Dazu zählen die Integration in den Video Disc Recorder, die Bedienung des Systems und eine Analyse der Performance über Zeitmessungen einzelner Prozesse.

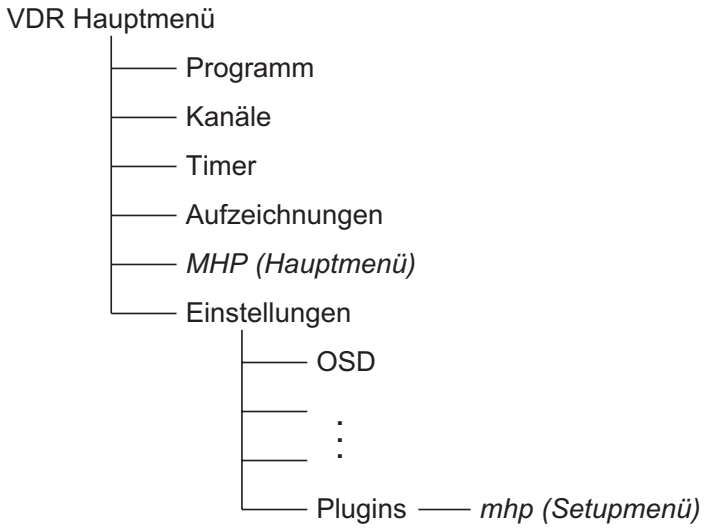


Abbildung 5.1.: Integration des MHP-Plugins in die Menüstruktur des VDR

5.3.1. Integration in VDR

Damit das entwickelte Plugin namens *mhp,* das den Hauptteil der MHP-DVP-Integration darstellt, in den Video Disc Recorder eingebunden werden kann, muss der VDR mit einem Parameter gestartet werden, der für das Laden des Plugins sorgt: `runvdr -Pmhp`. Nach dem Start von VDR mit diesem Parameter integriert sich das Plugin automatisch in dessen Menüstruktur. Abbildung 5.1 zeigt, an welchen Stellen dies der Fall ist.

Die Hauptfunktionen des Plugins verbergen sich unter dem Eintrag „MHP“, der im Hauptmenü des VDR eingefügt wird. Nach dessen Auswahl und anschließender Bestätigung wird ein Untermenü geöffnet, das die Liste der momentan verfügbaren MHP-Anwendungen zeigt, wie es in Abbildung 5.2 zu sehen ist. Beim Setupmenü des Plugins, das innerhalb der Plugineinstellungen eingegliedert wird, handelt es sich um eine Menüseite, die umfangreiche Einstellmöglichkeiten der Parameter des Systems bietet. Diese ist in Abbildung 5.3 dargestellt. Hier kann unter anderem auch die Variante der Ausgabedarstellung gewechselt werden.

5.3.2. Bedienung

Die für den Nutzer wahrnehmbaren Aktionen des Systems beginnen mit der Auswahl des Hauptmenüeintrags und der Betätigung der OK-Taste, worauf eine Liste der momentan vefügbaren MHP-Applikationen angezeigt wird. In dieser Liste wird auch gekennzeichnet, welche der Anwendungen aus dem Datenstrom extrahiert wurden („from stream“) und bei welchen es sich um lokale, auf der

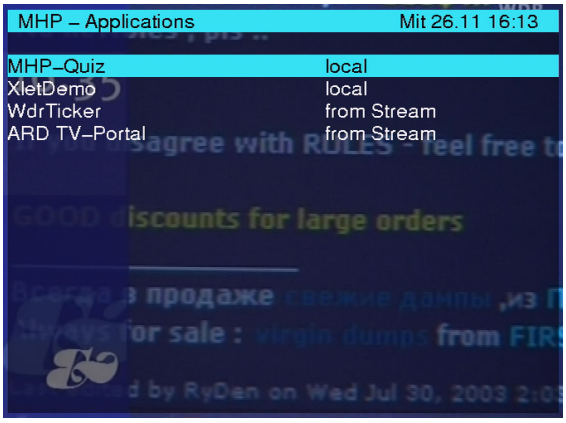


Abbildung 5.2.: Liste der momentan verfügbaren MHP-Applikationen

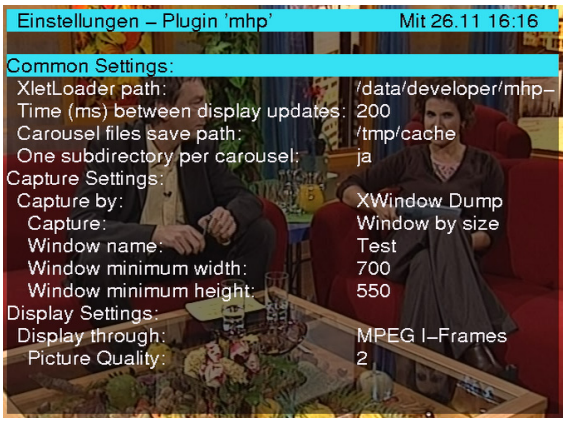


Abbildung 5.3.: Einstellungsmenü des MHP-Plugins

Festplatte gespeicherte Applikationen handelt („local“). Mit der Wahl eines dieser Einträge und der Betätigung der OK-Taste wird mit dem Start der Applikation begonnen. Sollte dies nicht möglich sein, da noch nicht alle Dateien der Applikation empfangen wurden oder ein Fehler bei deren Starten aufgetreten ist, wird eine Fehlermeldung ausgegeben und der Vorgang abgebrochen.

Wenn der Start der Applikation erfolgreich verlaufen ist, beginn der Ausgabeprozess mit der fortlaufenden Durchführung von Erfassung und Darstellung des grafischen Output der ausgeführten Anwendung. Dessen Ergebnis wird in Abbildung 5.4 am Beispiel des selbsterstellten MHP-Quiz gezeigt.

Während die Applikation ausgeführt wird werden die meisten Tastenbetätigungen des Nutzers an diese weitergeleitet und führen in der Regel zu Änderungen in deren Laufzeitverhalten. Die Betätigung der Zurück-Taste führt dagegen zum Beenden der Applikation, woraufhin auch der Ausgabeprozess seine Arbeit einstellt. Die vom Nutzer des Systems wahrnehmbaren Aktionen sind damit vorerst abgeschlossen und können zu einem späteren Zeitpunkt beliebig oft wiederholt werden.

5.3.3. Zeitmessungen und Optimierungsmöglichkeiten

Während das System der MHP-DVP-Integration eine Applikation ausführt, verbringt es die meiste Zeit in den Routinen der Ausgabeerfassung und -darstellung. Deshalb werden in den folgenden Abschnitten die implementierten Varianten der Ausgabe auf ihren Zeitbedarf hin untersucht. Dabei wird neben der insgesamt benötigten Zeit auch die Dauer von Teilprozessen angegeben. Außerdem wird auf Besonderheiten eingegangen und auf Optimierungsmöglichkeiten hingewiesen.

Die Messungen wurden auf dem Rechner des Digitalen Video Projekts durchgeführt. Als MHP-

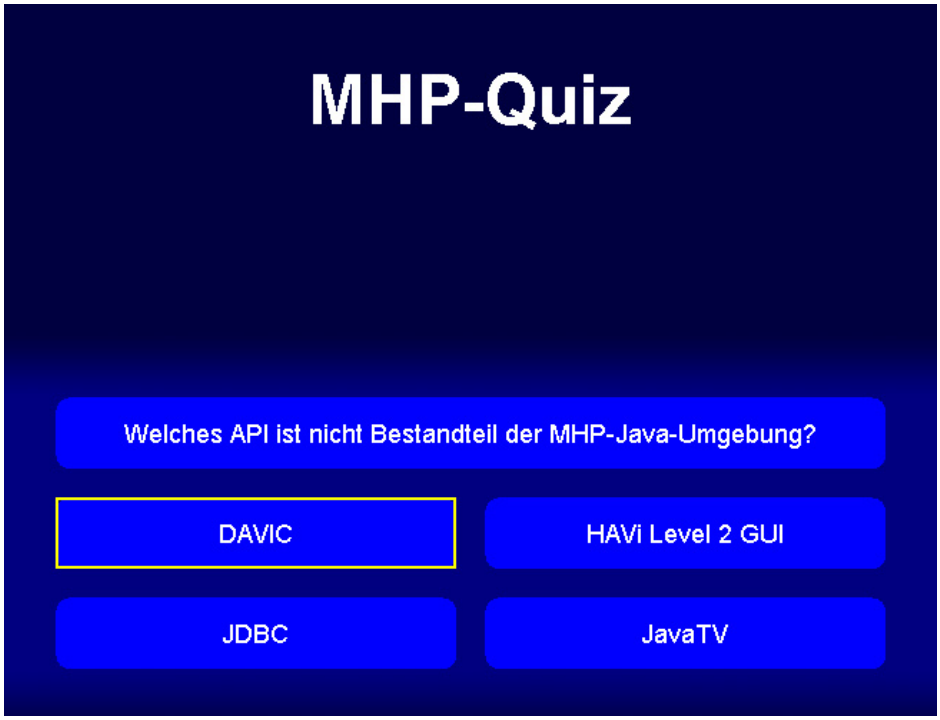


Abbildung 5.4.: Beispielapplikation MHP-Quiz

Applikation kam in allen Fällen das selbstentwickelte MHP-Quiz zum Einsatz. Die Ergebnisse sind Durchschnittswerte aus etwa 20 bis 30 Einzelmessungen.

Ausgabeerfassung mit X-Window Dump

Die Zeitmessung der Ausgabeerfassung mittels X-Window Dump wurde bei verschiedenen Farbtiefen des X-Server durchgeführt. Die Ergebnisse werden in Tabelle 5.3 angegeben. Darin fällt auf, dass die Dauer der Durchführung des X-Window-Dump abhängig von der Farbtiefe des X-Servers ist und mit zunehmender Farbtiefe ansteigt. Das liegt zum größten Teil an der jeweils zunehmenden Men-

		Farbtiefe des X-Servers		
		8 bit	16 bit	24 bit
gesamter Zeitbedarf		14 ms	25 ms	58 ms
davon	Durchführen des X-Window Dump	7 ms	18 ms	51 ms
	Konvertieren in internes RGBA-Format	7 ms	7 ms	7 ms

Tabelle 5.3.: Zeitbedarf der Ausgabeerfassung mittels X-Window Dump

		Kodierqualität (kleiner ist besser)		
		1	5	15
gesamter Zeitbedarf		230 ms	335 ms	700 ms
davon	Konvertieren nach YUV	11 ms	11 ms	11 ms
	Skalieren	26 ms	26 ms	26 ms
	Kodieren nach MPEG-Verfahren	19 ms	19 ms	19 ms
	Darstellen über Video-Device	170 ms	275 ms	640 ms

Tabelle 5.4.: Zeitbedarf der Ausgabedarstellung als MPEG-I-Frames

ge von Daten, die bei diesem Vorgang kopiert werden müssen. Den besten Kompromiss zwischen Darstellungsqualität und Geschwindigkeit stellt eine Farbtiefe von 16 bit dar. Das Konvertieren der erhaltenen Daten in das interne RGBA-Format ist im Gegensatz zum X-Window Dump immer gleich zeitaufwändig.

Da der Zeitbedarf der Ausgabeerfassung mittels X-Window Dump insbesondere bei 8 und 16 bit Farbtiefe relativ gering ist, gibt es hier nur wenig Bedarf an einer Optimierung. Eine Möglichkeit wäre, statt dem Umweg über das X-Windows API, ein direkter Zugriff auf den Framebuffer des X-Servers über dessen Shared-Memory-Schnittstelle um die Pixeldaten der Applikationsausgabe zu erhalten.

Ausgabedarstellung als MPEG-I-Frames

Die Zeitmessung der Ausgabedarstellung als MPEG-I-Frames wurde bei verschiedenen Kodierqualitäten des Encoders durchgeführt. Die Ergebnisse werden in Tabelle 5.4 angegeben. Hierbei ist zu erkennen, dass die Dauer der ersten drei Schritte des Vorgangs, das Konvertieren aus dem RGBA-Format ins YUV-Format, das Skalieren auf Vollbildgröße (720x576) und das Kodieren nach dem MPEG-Verfahren, unabhängig von der Qualitätseinstellung ist und nur einen kleinen Teil des gesamten Zeitbedarfs ausmachen. Die meiste Zeit des Prozesses wird von der abschließenden Darstellung über das Video-Device der DVB-Karte beansprucht, die mit abnehmender Qualität länger dauert. Der Grund dafür ist eine Eigenart des MPEG-Decoders der DVB-Karte. Dieser beginnt erst mit der Kodierung, wenn mindestens 400 KB an Daten vorliegen. Da die Einzelbilder in der Regel aber deutlich kleiner sind (im Schnitt 10 KB bis 50 KB), werden diese vom Treiber automatisch so oft an die Karte gesendet, bis die Mindestmenge erreicht wird. Damit müssen bei veringerter Kodierqualität und

		Farbtiefe des OSD		
		2 bit	4 bit	8 bit
gesamter Zeitbedarf		970 ms	590 ms	370 ms
davon	Erzeugen des ImageMagick-Bildes	68 ms	68 ms	68 ms
	Skalieren	16 ms	10 ms	6 ms
	Farbreduktion	810 ms	450 ms	245 ms
	Übergeben an OSD-Klassen des VDR	42 ms	26 ms	13 ms
	Darstellen im OSD der DVB-Karte	30 ms	30 ms	30 ms

Tabelle 5.5.: Zeitbedarf der Ausgabedarstellung im OSD

damit kleinerer Größe der Einzelbilder diese öfter gesendet werden, was zum Teil deutlich länger dauert, in Extremfällen, z. B. bei einfarbigen Bildern, sogar bis zu 2 Sekunden.

Eine Möglichkeit den Zeitbedarf der Darstellung zu verringern wäre das Senden der Daten als zusammenhängenden Videostrom statt in Form von Einzelbildern. Damit sollte das wiederholte Senden entfallen können, wenn erst einmal mit der Dekodierung begonnen wurde. Eine weitere Zeitersparnis würde das Weglassen der Skalierung bringen, da die Ausgabe der meisten MHP-Applikationen schon in der richtigen Größe vorliegt.

Ausgabedarstellung im OSD

Die Zeitmessung der Ausgabedarstellung im On-Screen-Display der DVB-Karte wurde bei verschiedenen Farbtiefen und damit auch verschiedener Auflösung des OSD durchgeführt. Die Ergebnisse werden in Tabelle 5.5 angegeben.

Das Erzeugen des Bildes im internen Format der ImageMagick-Bibliothek benötigt in allen drei Fällen die gleiche Zeit. Die Dauer des folgenden Skalierens auf die Größe des On-Screen-Displays fällt mit zunehmender Farbtiefe, da dieser Prozess abhängig von der Zahl der Pixel des OSD ist, die auch mit steigender Farbtiefe abnimmt. Die anschließend stattfindende Farbreduktion nimmt den größten Zeitbedarf der Ausgabedarstellung in Anspruch und ist genau wie das Skalieren abhängig von der Zahl der Pixel. Das gleiche gilt für das Übergeben der Pixeldaten an die OSD-Klassen des Video Disc Recorder. Das eigentliche Darstellen im OSD der DVB-Karte hängt nur davon ab, wie viele Pixel sich von einem Bild zum nächsten ändern. Dessen Dauer kann von 0 ms, wenn sich nichts ändert, bis etwa 100 ms für ein vollständiges Update reichen.

Das größte Potenzial für Geschwindigkeitssteigerungen liegt bei der Farbreduktion. Die Verwendung einer anderen Bibliothek mit einem schnelleren Algorithmus, der auf Assemblerebene optimiert und an spezifische Eigenschaften der CPU angepasst ist, würde mit Sicherheit einige Einsparungen bringen. Des Weiteren würde dann auch die Erzeugung des ImageMagick-Bildes wegfallen können.

Abschlussanalyse der Geschwindigkeit

Aus den Zeitmessungen der Ausgabeprozesse ergibt sich eine für den ersten Prototyp des Systems der MHP-DVP-Integration zufriedenstellende Geschwindigkeit. Bis auf einige Fälle liegt der Zeitbedarf meist unter den im MHP-Standard maximal erlaubten 500 ms. Bei Umsetzung der vorgeschlagenen Optimierungen sollte dieser Wert in allen Fällen deutlich zu unterbieten sein. Damit könnte das System im Bereich der Geschwindigkeit durchaus mit den zur Zeit im Handel verfügbaren Endgeräten mithalten.

6. Schlussbemerkungen und Ausblick

6.1. Bewertung der Ergebnisse

Ziel dieser Diplomarbeit war die Entwicklung eines Konzeptes die Integration der Multimedia Home Platform in das linuxbasierte System des Digitalen Video Projekts. Dies umfasst den Empfang, die Ausführung, die Verwaltung, die Steuerung und die Darstellung der MHP-Applikationen. Als Umsetzung dieses Konzeptes sollte eine Systemfamilie entworfen werden, die sich durch eine modulare und flexible Architektur auszeichnet. Für die Demonstration der Realisierbarkeit der Architektur sollte zum Abschluss ein erster Prototyp der MHP-DVP-Integration implementiert werden.

Im Verlauf der Entwicklung entstand ein Merkmalmodell, welches eine große Zahl von Variabilitäten und Paramtetern besitzt. Abhängig von deren Festlegung, beispielsweise aufgrund von Kundenvorgaben, können eine Vielzahl von Systemen generiert werden, die individuell an die gegebenen Hardware- und Softwarevoraussetzungen angepasst sind. Obwohl sich viele dieser Systeme nur in kleinen Details unterscheiden, reicht die Palette der Anwendungen von einem MHP-Entwicklersystem, das mit Tastatur gesteuert wird und die Ausgabe der Applikationen auf die Grafikkarte umleitet, bis zu einem mit Infrarotfernbedienung gesteuerten MHP-Consumer-Endgerät, dessen Ausgabe auf dem Fernseher dargestellt wird.

Die aus dem Merkmalmodell entwickelte Architektur der Systemfamilie erfüllt die an sie gestellten Anforderungen. Sie besitzt einen modularen Aufbau, der eine flexible Anpassung an sich ändernde Anforderungen erlaubt. Die einzelnen Pakete und Module der Architektur wurden weitgehend unabhängig voneinander entworfen, die Kommunikation findet über definierte Schnittstellen statt. Bei der Generierung des Systems erlaubt dies die einfache Kombination der Module entsprechend dem Merkmalmodell. Die Architektur erhebt keinen Anspruch auf Vollständigkeit. An einigen Stellen, wie im Bereich des Datenempfangs und der Applikationsverwaltung, mussten aufgrund des Umfangs des MHP-Standards Einschränkungen vorgenommen werden.

Der entwickelte Prototyp bietet keine vollständige Implementierung der Architektur, da dies den

Rahmen dieser Arbeit deutlich überschritten hätte. Beispielsweise im Bereich des Datenempfangs und der Applikationsverwaltung konnten nur deren grundlegende Aspekte umgesetzt werden. Auch bei der Implementierung der verschiedenen Varianten bei Applikationssteuerung und Ausgabe wurden Abstriche gemacht. Des Weiteren ist aufgrund einer fehlenden kompletten Implementierung der MHP-Schnittstelle die fehlerfreie Ausführung der im Datenstrom übertragenen Applikationen nicht gewährleistet. Bei allen genannten Einschränkungen ist der Prototyp jedoch für die Demonstration der Realisierbarkeit der Architektur vollkommen ausreichend.

Zum Abschluss kann festgestellt werden, dass trotz der genannten Abstriche die im Vorfeld definierten Ziele der Diplomarbeit erreicht werden konnten.

6.2. Weitergehende Entwicklungen

Als Abschluss der Arbeit werden nun verschiedene Richtungen für die Weiterentwicklung des Systems der MHP-DVP-Integration aufgezeigt.

Ein erster Schritt der weiteren Entwicklung sollte die Vervollständigung des implementierten Prototyps sein, bis alle Funktionalitäten der Architektur realisiert wurden. Dies schließt auch die Integration von Fehlerbehandlungsroutinen und die Umsetzung der in Kapitel 5.3.3 genannten sowie weiterer Optimierungen mit ein. Des Weiteren sollte eine Erweiterung des Video Disc Recorder um eine intuitive Unterstützung des Frambuffers von Grafikkarten vorgenommen werden. Damit wären keinerlei Einschränkungen bei der Darstellung der grafischen Ausgabe der MHP-Applikationen mehr hinzunehmen.

Wurden alle Komponenten der Architektur vollständig implementiert, sollte mit der Untersuchung begonnen werden, ob der Systemfamilie weitere Merkmale hinzugefügt werden können. Dabei sind neben weiteren Varianten der Applikationssteuerung und Ausgabe auch komplett neue Merkmale denkbar, die z. B. erst mit der nächsten Version der MHP erforderlich werden könnten.

Neben Erweiterungen bei Architektur und Implementierung des Systems der MHP-DVP-Integration sollte die Entwicklung einer vollständigen, quelltextoffenen MHP-Implementierung in Betracht gezogen werden. Als Basis für diese Entwicklung könnte das XleTView-Projekt dienen [22]. Damit würde die fehlerfreie Ausführung der im Datenstrom übertragenen Applikationen ermöglicht, ohne dass Zusatzkosten für eine Impementierung der MHP-Schnittstelle von Drittherstellern anfielen.

A. Überblick über die DVB-J APIs

In diesem Anhang sind alle Java Packages aufgeführt, die Bestandteil der DVB-J-Platform von MHP 1.0.3 und MHP 1.1 sind. Die Paketbeschreibungen wurden den Javadoc Dokumentationen entnommen. Bei einigen Paketen werden vom DVB-Konsortium signifikante Elemente weggelassen, was als eine „DVB-Untermenge“ bezeichnet wird.

A.1. Standard Java APIs (DVB-Untermenge)

Der Kern der Java-Plattform der MHP-Spezifikation wird aus aus den Standard Packages des JDK 1.1.8 (java.lang, java.util, java.io, usw.) gebildet, wobei einige dieser Pakete Beschränkungen unterliegen bzw. weggelassen wurden, hauptsächlich im Bereich der GUI.

A.2. JMF 1.0 APIs

Die Multimedia Home Platform benutzt das API des Java Media Framework für die Präsentation und Steuerung von Audio- und Videodaten sowie anderer zeitbasierter Medien.

Paket	Beschreibung
`javax.media`	Beinhaltet Klassen und Schnittstellen für die Konfiguration und Verwaltung von Geräten, Medien und Ereignissen.
`javax.media.protocol`	Bietet Klassen und Schnittstellen für die Referenzierung verschiedener Datenquellen.

A.3. Java Secure Sockets Extension APIs

Die Java Secure Sockets Extension (JSSE) ist eine Menge von Java Paketen, die eine sichere Kommunikation über das Internet ermöglichen. Dafür werden die Protokolle SSL (Secure Sockets Layer) und TLS (Transport Layer Security) zur Verfügung gestellt. Die Erweiterungen beinhalten Funktionalitäten für Datenverschlüsselung, Server-Authentifizierung und Nachrichten-Integrität.

Paket	Beschreibung
`javax.net`	Beinhaltet optionale Klassen für Netzwerkanwendungen. Dazu zählen Factories für die Erstellung und Konfiguration von Sockets.
`javax.net.ssl`	Stellt Klassen für die sichere Kommunikation über Sockets mit dem SSL-Protokoll zur Verfügung.
`javax.security.cert`	Bietet Klassen und Schnittstellen für die Auswertung und Verwaltung von Zertifikaten

A.4. JavaTV 1.0 APIs (DVB-Untermenge)

JavaTV ist eine Menge von allgemeinen, TV-bezogenen Paketen. Sie beinhalten die allgemeinen Elemente einer Plattform für interaktives Fernsehen, unabhängig von der Art der Übertragung. Das bedeutet, dass die APIs sowohl für DVB- als auch für nicht-DVB-Systeme anwendbar sind.

Paket	Beschreibung
`javax.tv.graphics`	Bietet Mechanismen für Alpha-Blending und die Ermittlung des Wurzelcontainers eines Xlets
`javax.tv.locator`	Stellt Mittel für die Referenzierung von Daten und Ressourcen zur Verfügung, auf die mit den JavaTV APIs zugegriffen werden kann.
`javax.tv.media`	Beinhaltet Steuerungselemente und Ereignisse für die Verwaltung von Echtzeit-Medien in Fernsehumgebungen.

Paket	Beschreibung
javax.tv.net	Stellt den Zugriff auf IP-Datagramme, die im Übertragungskanal gesendet werden, zur Verfügung.
javax.tv.service	Bietet Mechanismen für den Zugriff auf die Service Informationen des DVB-Signals und beinhaltet APIs für deren Elemente.
javax.tv.service.guide	Stellt APIs für die Unterstützung von elektronischen Programmführern bereit. Das beinhaltet komplette Programmpläne, einzelne Sendungen und deren Altersbeschränkungen.
javax.tv.service.navigation	Bietet Klassen und Schnittstellen für die Navigation durch die verschiedenen Dienste und hierarchischen Informationen.
javax.tv.service.selection	Beinhaltet einen Mechanismus um einen Dienst für die Präsentation auszuwählen.
javax.tv.service.transport	Stellt Zusatzinformationen über die Transportmechanismen der übertragenen Inhalte zur Verfügung, die durch die SI-Daten beschrieben werden.
javax.tv.util	Bietet Klassen und Schnittstellen für die Erzeugung und Verwaltung von Timern, also zeitgesteuerten Aufzeichnungen.
javax.tv.xlet	Bietet Schnittstellen für die Kommunikationen zwischen den Applikationen und dem Application Manager.

A.5. DAVIC APIs

Paket	Beschreibung
org.davic.media	Beinhaltet Erweiterungen für das JMF für die Steuerung von fernsehorientierten Audio- und Videoinhalten.
org.davic.mpeg	Bietet Klassen für bekannte MPEG-Konzepte.

Paket	Beschreibung
org.davic.mpeg.dvb	Bietet Klassen für bekannte MPEG-Konzepte, wie sie bei DVB Verwendung finden.
org.davic.mpeg.sections	Erlaubt den Zugriff auf den MPEG-2 Section-Filter.
org.davic.net	Bietet eine allgemeine Referenzierung von Inhalten.
org.davic.net.ca	Stellt eine Schnittstelle für das Conditional Access System zur Verfügung.
org.davic.net.dvb	Bietet eine DVB-spezifische Referenzierung von Inhalten.
org.davic.net.tuning	Erlaubt den Zugriff auf den Tuner und die Wahl des MPEG-Multiplex.
org.davic.resources	Bietet ein Framework für die Ressourcenverwaltung.

A.6. DVB APIs

Paket	Beschreibung
org.dvb.application	Bietet den Zugriff auf die Liste der im aktuellen Kontext verfügbaren Applikationen und die Möglichkeit diese zu starten.
org.dvb.dsmcc	Erlaubt den erweiterten Zugriff auf Dateien, die im Datenstrom übertragen werden.
org.dvb.event	Bietet den Zugriff auf Eingabeereignisse des Nutzers, bevor diese vom Ereignismechanismus des java.awt Pakets verarbeitet werden.
org.dvb.io.ixc	Bietet Unterstützung für die Kommunikation zwischen Applikationen
org.dvb.io.persistent	Enthält Erweiterungen des java.io Pakets für den Zugriff auf Dateien des persistenten Speichers.
org.dvb.lang	Beinhaltet die Merkmale des Kerns der MHP, die nicht im java.lang Paket enthalten sind.
org.dvb.media	Enthält DVB-spezifische Erweiterungen des Java Media Framework.
org.dvb.net	Bietet allgemeine Netzwerkunterstützung.

Paket	Beschreibung
org.dvb.net.ca	Enthält Erweiterungen für das Conditional Access API von DAVIC.
org.dvb.net.rc	Beinhaltet APIs für eine bi-direktionale Kommunikation über den Rückkanal.
org.dvb.net.tuning	Enthält Erweiterungen für das Tuning API von DAVIC.
org.dvb.si	Erlaubt den Zugriff auf die Service Informationen.
org.dvb.test	Bietet Testapplikationen die Möglichkeit Nachrichten während der Ausführung aufzuzeichnen.
org.dvb.ui	Bietet erweiterte Grafikfunktionalitäten.
org.dvb.user	Erlaubt den Zugriff auf Einstellungen des Endnutzers

A.7. HAVi Level 2 GUI APIs

Paket	Beschreibung
org.havi.ui	Enthält Klassen und Schnittstellen für die Erstellung von Benutzeroberflächen und zum Zeichnen von Grafiken und Bildern.
org.havi.ui.event	Bietet Klassen und Schnittstellen für die Behandlung von Ereignissen, die von den AWT Komponenten gesendet werden.

A.8. Zusätzliche APIs von MHP 1.1

Paket	Beschreibung
org.dvb.application.inner	Bietet Unterstützung für die Einbettung anderer interaktiver Applikationen in die Nutzerschnittstelle einer DVB-J-Applikation.
org.dvb.application.plugins	Erweitert die MHP um die Einbindbarkeit von applikationsübergreifenden Plugins.
org.dvb.application.storage	Erlaubt die Speicherung von Applikationen.
org.dvb.dom.bootstrap	Bietet DVB-J-Applikationen einen Einstiegspunkt für die W3C DOM APIs.

Paket	Beschreibung
org.dvb.dom.inner	Bietet Unterstützung für die Einbettung von DVB-HTML-Applikationen in die Nutzerschnittstelle einer DVB-J-Applikation.
org.dvb.internet	Bietet einen Mechanismus für die Steuerung von Internet-Clients, die auf einer MHP vorhanden sein können, wie z.B. ein Webbrowser oder ein Email-Client.
org.dvb.smartcard	Enthält DVB-spezifische Unterstützung für Smartcard-Lesegeräte.

B. Installationsanleitung

Die im folgenden aufgeführten Schritte zur Installation der im Laufe der Diplomarbeit entstandenen Software beziehen sich auf SuSE Linux ab Version 8.0.

Systemvoraussetzungen

Bevor mit der Installation des MHP-Plugins begonnen werden kann müssen folgende Softwarekomponenten auf einem Standard PC mit DVB-Karte installiert sein:

- Standardinstallation von SuSE Linux inklusive Entwicklungswerkzeuge (C/C++ Compiler, make, abhängige Bibliotheken)
- Linux DVB Treiber 1.0.1 oder höher
- VDR 1.2.5 oder höher

Benötigte Pakete

Zu einer Standardinstallation von SuSE Linux müssen noch folgende und deren abhängige Pakete nachinstalliert werden, soweit diese nicht schon vorhanden sind.

Pakete	Beschreibung
`XFree86, XFree86-devel`	X-Windows System XFree86 und Entwicklungsbibliotheken
`xextra, XFree86-Xvfb`	X-Server mit hardwareunabhängiger Ausgabe
`ImageMagick, ImageMagick-devel`	Bibliothek für die Bildverarbeitung
`zlib, zlib-devel`	Bibliothek für das Entpacken von mit gzip komprimierten Daten

Folgende Pakete, die nicht zum Lieferumfang von SuSE Linux gehören, werden zusätzlich noch benötigt und müssen gegebenenfalls aus dem Internet heruntergeladen werden.

Datei	Quelle	Beschreibung
`ffmpeg-0.4.8.tar.gz`	`ffmpeg.sf.net`	„FFmpeg video and audio converter“
`j2re-1_4_2-linux-i586.bin`	`sun.java.com`	Sun Java Runtime Environment
`vdr-mhp-0.1.0.tgz`		MHP-Plugin
`xletdemos.tgz`		Demo-MHP-Anwendungen
`xletmanager.tgz`		XletManager inklusive JavaTV, JMF und den MHP APIs

Installationsschritte

1. Java: Die Datei `j2re-1_4_2_02-linux-i586.bin` aus dem Verzeichnis `/opt` heraus ausführen. Die Lizentbedingungen mit „yes“ akzeptieren.

   ```
   cd /opt
   ./j2re-1_4_2_02-linux-i586.bin
   ```

2. FFmpeg: Die Datei`ffmpeg-0.4.8.tar.gz` in das `/usr/local/src` -Verzeichnis entpacken, ins FFmpeg-Verzeichnis wechseln und FFmpeg compilieren.

   ```
   tar -xzf ffmpeg-0.4.8.tar.gz -C /usr/local/src
   cd /usr/local/src/ffmpeg-0.4.8
   ./configure --enable-shared
   make
   make install
   ```

3. Die Dateien `xletdemos.tgz` und `xletmanager.tgz` nach `/usr/local/src` entpacken.

```
tar -xzf xletdemos.tgz -C /usr/local/src
tar -xzf xletmanager.tgz -C /usr/local/src
```

4. MHP-Plugin: Die Datei `vdr-mhp-0.1.0.tgz` ins Plugin-Verzeichnis des Video Disc Recorder entpacken und das Verzeichnis in `mhp` umbenennen

   ```
   cd /usr/local/src/vdr/PLUGINS/src
   tar -xzf vdr-mhp-0.1.0.tgz
   mv mhp-0.1.0 mhp
   ```

   Ins VDR-Verzeichnis wechseln und das MHP-Plugin compilieren.

   ```
   cd /usr/local/src/vdr
   make plugins
   ```

5. Bevor das MHP-Plugin ausgeführt werden kann, müssen unter Umständen noch die Verzeichnisangaben im Startscript des XletManagers (`loadxlet.sh`) angepasst werden. Außerdem muss vor dem Aufruf des Plugins der XServer Xvfb gestartet worden sein. Dies geschieht folgendermaßen:

   ```
   Xvfb :1 -screen 0 800x600x16
   ```

   Wenn dies geschehen ist, kann das MHP-Plugin mit dem Befehl

   ```
   runvdr -Pmhp
   ```

   gestartet werden.

Abkürzungsverzeichnis

AIT	Application Information Table
API	Application Programming Interface
AWT	Advanced Windowing Toolkit
BAT	Bouquet Association Table
BIOP	Broadcast Inter-ORB Protocol
CA	Conditional Access
CAT	Conditional Access Table
DSM-CC	Digital Storage Media - Command & Control
DVB	Digital Video Broadcasting
DVP	Digital Video Project
EIT	Event Information Table
EPG	Electronical Program Guide
ES	Elementary Stream
ETSI	European Telecommunication Standards Institute
GNU	GNU is Not Unix
GPL	General Public License
HAVi	Home Audio Video Interoperability
IEC	International Electrotechnical Commission
IrDA	Infrared Data Association
ISO	International Standard Organization
JMF	Java Media Framework
JRE	Java Runtime Environment
MHP	Multimedia Home Platform

MHEG	Multimedia and Hypermedia Experts Group
MPEG	Motion Picture Experts Group
NIT	Network Information Table
OSD	On Screen Display
PAT	Program Association Table
PCR	Program Clock Reference
PES	Packetized Elementary Stream
PID	Packet Identification
PMT	Program Map Table
PSI	Program Specific Information
RST	Running Status Table
SDT	Service Description Table
SI	Service Information
SID	Service Identification
ST	Stuffing Table
SVDRP	Simple VDR Protocol
TDT	Time and Date Table
TOT	Time Offset Table
TS	Transport Stream
VDR	Video Disc Recorder
VoD	Video-on-Demand

Literaturverzeichnis

[1] Dipl.-Ing. Klaus Merkel und Gerald Breuning. *Die Migration des „Free Universe Networks“ (F.U.N.)*. Sonderdruck aus Fernseh- und Kino-Technik, 2001.

[2] Ralph Dietzel. *Konzeption und Entwicklung einer Systemfamilie für eine Universal- Fernbedienung auf Basis eines Palm-Handhelds*. Technische Universität Ilmenau, April 2003.

[3] Krzysztof Czarnecki und Ulrich W. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, 2000.

[4] European Telecommunication Standards Institute. *Technical Report 101 154: Digital Video Broadcasting (DVB); Implementation guidelines for the use of MPEG-2 Systems, Video and Audio in satellite, cable and terrestrial broadcasting applications*, Juli 2000.

[5] European Telecommunication Standards Institute. *European Telecommunication Standard 300 743: Digital Video Broadcasting (DVB); Subtitling systems*, Oktober 2002.

[6] European Telecommunication Standards Institute. *European Telecommunication Standard 300 468: Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems*, Januar 2003.

[7] European Telecommunication Standards Institute. *Technical Specification 101 812: Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.0.3*, Juni 2003.

[8] Free Software Foundation. GNU General Public License. `http://www.gnu.org/copyleft/gpl.html`, 1991.

[9] International Organization for Standardization/International Electrotechnical Commission. *International Standard ISO/IEC 13818-1: Information Technology - Generic coding of moving pictures and associated audio: Systems*, November 1994.

[10] International Organization for Standardization/International Electrotechnical Commission. *International Standard ISO/IEC 13818-6: Information Technology - Generic coding of moving pictures and associated audio – Part 6: Extensions for DSM-CC*, 1994.

[11] Andreas Kraft. Multimedia and Hypermedia Expert Group. `http://www.mheg.org`, 2003.

[12] Florian Meffert. *Konzeption einer Videodatenverteilung im Rahmen des Digitalen Video Projektes*. Technische Universität Ilmenau, Februar 2003.

[13] Marcus Metzler and Ralph Metzler. *The Linux DVB API*, August 2002.

[14] Steve Morris. The MHP Tutorial. `http://mhp-interactive.org`, 2003.

[15] Rudolf Mäusl. *Fernsehtechnik: Vom Studiosignal zum DVB-Sendesignal*. Hüthig Verlag Heidelberg, third edition, 2003.

[16] Anne Preiß. *Systemfamilienbasierte Dokumentation des Digitalen Video Projekts (SW-Modul „vdr 1.1.20“)*. Technische Universität Ilmenau, März 2003.

[17] Sven Schaepe. *Be- und Verarbeitung von Service Informationen im DVB-Stream*. Technische Universität Ilmenau, 2003.

[18] Klaus Schmidinger. Video Disc Recorder. `http://cadsoft.de/vdr/`, Oktober 2003.

[19] Klaus Schmidinger. Video Disk Recorder: Plugins. `http://cadsoft.de/vdr/plugins.htm`, Oktober 2003.

[20] Dipl.-Ing. Robert Sedlmeyer. *Multimedia Home Platform – Standard 1.0.1*. Sonderdruck aus Fernseh- und Kino-Technik, 2001.

[21] Detlef Streitferdt. Digital Video Project. `http://www.theoinf.tu-ilmenau.de/~streitdf/DVP/`, Oktober 2003.

[22] Martin Svedén. SourceForge.net: Project Info - XleTView. `http://sourceforge.net/projects/xletview`, November 2003.

Thesen der Diplomarbeit

- Mit der intensiven Einbeziehung des Zuschauers können interaktive Mehrwertdienste die Attraktivität des Fernsehens deutlich steigern.
- Die Multimedia Home Platform stellt, als weltweit anerkannter Standard, den besten Weg zur Bildung einer horizontalen Marktstruktur für das digitale, interaktive Fernsehen dar.
- Zum Zeitpunkt der Diplomarbeit existiert keine Implementierung der MHP, die sich nahtlos in das vorhandene System des Digitalen Video Projekts integriert.
- Eine Systemfamilie ist die optimale Realisierungsvariante für die MHP-DVP-Integration.
- Die im Rahmen dieser Diplomarbeit entwickelte Systemfamilie und deren Architektur erlauben die einfache und flexible Generierung einer Vielzahl von Systemen für die MHP-DVP-Integration, die sich optimal an gegebene Hardware- und Softwareumgebungen anpassen.
- Die Hauptkomponente des entwickelten Prototyps, das Plugin *mhp*, stellt eine Erweiterung der DVP-Systemsoftware VDR dar.
- Der Prototyp zeigt nicht nur die Realisierbarkeit der entworfenen Architektur, sondern kann unter Zuhilfenahme einer vollwertigen MHP-Implementierung für eine vollständige Integration der Multimedia Home Platform in die Architektur des Digitalen Video Projekts verwendet werden.

Ilmenau, _____ _____

Andreas Regel

Erklärung

Hiermit erkläre ich, dass ich die Diplomarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe.

Ilmenau, _____ _____

Andreas Regel