Technische Universität Ilmenau
Fakultät für Informatik und Automatisierung
Institut für Theoretische und Technische Informatik
Fachgebiet Prozessinformatik

Diplomarbeit

Konzeption und Entwicklung einer Systemfamilie für eine Universal-Fernbedienung auf Basis eines Palm-Handhelds

(Im Rahmen des Digitalen Video Projektes)

Verfasser: Ralph Dietzel
Betreuer: Dipl.-Inf. Detlef Streitferdt
Verantwortlicher Hochschullehrer: Prof. Dr.-Ing. Ilka Philippow

Ilmenau, April 2003

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

Ilmenau, den _____ _____

Ralph Dietzel

1 Einleitung

In den 30er Jahren des vergangenen Jahrhunderts trat ein neues Medium, das Fernsehen, seinen Siegeszug an. Bis heute erlebte dieses Medium eine weitreichende Entwicklung, die sich nicht nur auf den Übergang von Schwarz-Weiß- zu Farbbildern beschränkt. In späteren Jahren gelang es, über die reinen Bild- und Toninformationen hinaus weitere Daten in das Signal einzubinden. Mit der Integration der Digitaltechnik eröffneten sich auf diesem Sektor weitere neue Möglichkeiten.

Das *Digitale Video Projekt*, vor dessen Hintergrund die vorliegende Diplomarbeit entstand, hat die Entwicklung eines Systems zum Ziel, das eine neue und innovative Nutzung des Mediums „Fernsehen“ erlaubt. Diese Nutzung soll dabei weit über das „Sehen“ und „Hören“ hinausgehen. So ist die Integration eines digitalen Videorecorders ebenso angedacht, wie die Nutzung von Titel-Datenbanken und eines *Electronic Program Guide*.

Um diese Ideen realisieren zu können, ist auch ein neues Bedienkonzept erforderlich. Ziel dieser Arbeit soll es sein, ein solches Konzept zu entwickeln. Es soll die Vorteile bestehender Bedienkonzepte in sich vereinigen, sowie eventuelle Nachteile berücksichtigen und eine Lösung anbieten. Für die Bedienung ist eine möglichst hohe Variabilität und Flexibilität zu gewährleisten, d.h. das Konzept soll sich dem Nutzer anpassen und nicht der Nutzer dem Konzept.

Im Vordergrund der Entwicklung soll hierbei die Konzeption einer Fernbedienung als zentrales Element der Bedienung stehen. Diese Universal-Fernbedienung wird als Systemfamilie konzipiert. Systemfamilien spielen in der heutigen Softwareentwicklung eine immer größere Rolle, da durch sie eine gezielte Wiederverwendung von Entwicklungsleistungen möglich wird.

Bei der Entwicklung von Systemfamilien wird eine Reihe von Anwendungen, die der gleichen Domäne entstammen, auf Gemeinsamkeiten und Unterschiede untersucht. Die so gewonnenen Erkenntnisse werden in einem Merkmalmodell zusammengefasst, aus dem sich eine einheitliche Architektur für die einzelnen Familienmitglieder ableiten lässt. Auf diese Weise lässt sich eine Vielzahl von Anwendungen mit geringem zusätzlichen Aufwand erzeugen. Auch die Integration weiterer Funktionen in Form neuer Merkmale ist relativ einfach realisierbar.

Im Hinblick auf diese Verfahrensweise sollen zur Lösung der Aufgabe zunächst bestehende Bedienkonzepte untersucht und ihre Vor- und Nachteile herausgearbeitet werden. In einem weiteren Schritt erfolgt die Betrachtung existierender Fernbedienungen, um festzustellen, wie die einzelnen Konzepte umgesetzt wurden. Aus diesen Erkenntnissen wird ein Anforderungsprofil erarbeitet, welches die zukünftigen Fähigkeiten der

Universal-Fernbedienung zusammenfasst. Im Anschluss daran werden die einzelnen Anforderungen zu Merkmalen gruppiert. In dem so entstehenden Merkmalmodell werden auch Beziehungen und Abhängigkeiten zwischen den Merkmalen erfasst.

Für das zu erstellende System soll eine Architektur in Form einer Systemfamilie entworfen werden, welche die angesprochene Flexibilität garantiert. Dabei sollen aus den einzelnen Merkmalen Pakete modelliert werden. Die Architektur des Systems und der einzelnen Pakete soll so ausgelegt sein, dass auf Basis von ausgewählten Merkmalen ein kundenspezifisches Produkt konfiguriert und anschließend eine entsprechende Anwendung generiert werden kann.

Zur Abrundung der Arbeit soll ein erster Prototyp entwickelt werden, der die Funktionsfähigkeit der Architektur demonstriert.

2 Grundlagen

Bevor die Lösung der eigentlichen Aufgabe in Angriff genommen wird, ist es erforderlich, den Rahmen, in dem sich diese Arbeit bewegt, abzustecken.

Beginnen soll dies mit Erläuterungen zu Systemfamilien im Allgemeinen, ihrem Sinn und Zweck sowie den grundlegenden Charakteristika. Der zweite Abschnitt beschäftigt sich mit der Frage, welche Konzepte für die Bedienung von Geräten aus dem Bereich der Unterhaltungselektronik existieren. Verschiedene Lösungsvarianten werden an dieser Stelle betrachtet. Daran anschließend werden existierende Fernbedienungen vorgestellt und im Hinblick auf ihre Flexibilität untersucht. Nicht vergessen werden darf an dieser Stelle auch eine kurze Vorstellung des *Digitalen Video Projektes*, ein Teil dessen diese Arbeit ist. Den Abschluss bildet die Formulierung der konkreten Aufgabenstellung und eines Lösungsansatzes, mit denen sich dann die weiteren Teile der Arbeit befassen werden.

2.1 Systemfamilien

Die Art und Weise heutiger Softwareentwicklung verlangt einen hohen Aufwand, welcher in Pflege, Anpassung und Veränderung bestehender Systeme investiert wird. Ein Teil der so entstehenden Kosten kann eingespart werden, wenn konsequent Wiederverwendung von Entwicklungsleistungen betrieben wird [3].

Eine Möglichkeit hierfür ist die Entwicklung von Systemfamilien. Sie kommt vor allem bei Softwaresystemen, die der gleichen Domäne, d.h. dem gleichen Anwendungsbereich, entstammen, zum Einsatz.

Das grundsätzliche Verfahren bei der Entwicklung einer Systemfamilie ist in *Abb. **2-1*** dargestellt.

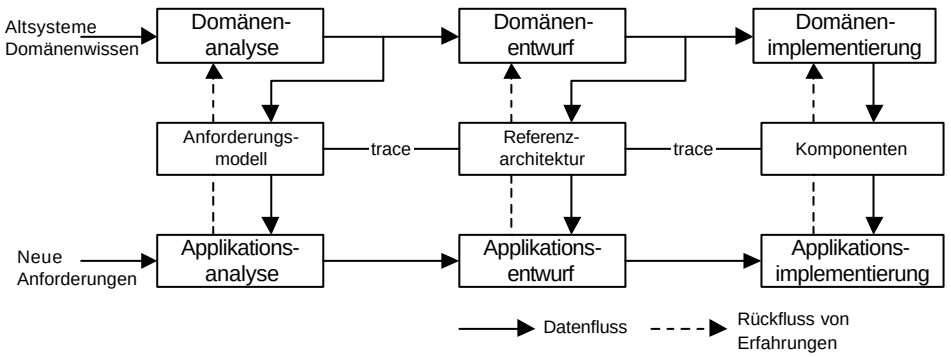

Abb. 2-1: Entwicklungsprozess für Systemfamilien

Im Rahmen der Domänenanalyse werden die Eigenschaften der einzelnen Familienmitglieder untersucht und Gemeinsamkeiten und Unterschiede werden herausgearbeitet. Die Ergebnisse dieses Arbeitsschrittes werden mit den an die Systemfamilie gestellten Anforderungen in Verbindung gebracht und in einem Merkmalmodell dokumentiert. Für das Vorgehen bei der Domänenanalyse und dem Domänenentwurf existieren verschiedene Ansätze, wie sie beispielsweise in *FODA* (*Feature-Oriented Domain Analysis*) [1] oder *FOPLE* (*Feature-Oriented Product Line Engineering*) [2] beschrieben werden.

Die im Merkmalmodell enthaltenen Merkmale (*Features*) repräsentieren Aspekte, die für den Anwender von Belang sind, und lassen sich in vier Kategorien einteilen:

- *Funktionale Merkmale* drücken ein Verhalten oder eine Interaktion mit dem Nutzer aus,
- *Schnittstellen-Merkmale* drücken die Integration von Standards oder Subsystemen aus,
- *Parameter-Merkmale* drücken darstellbare Umgebungseigenschaften und nicht-funktionale Merkmale aus,
- *Abstrakte Merkmale* drücken Konzepte aus.

Diese Merkmale werden in einem Merkmalmodell hierarchisch strukturiert dargestellt. Aus diesem Modell werden ebenfalls die verschiedenen Beziehungen, in denen die einzelnen Merkmale zueinander stehen können, ersichtlich. Diese Beziehungen lassen sich in verschiedene Kategorien unterteilen:

- *Merkmal-Untermerkmal-Beziehungen* dienen der Abbildung der Hierarchie. Der Nutzer wird mit ihrer Hilfe durch den Auswahlprozess geleitet. Die Stellung eines Merkmals in der Hierarchie spiegelt seinen Einfluss auf die Systemarchitektur wieder.
- Bei einer *Verfeinerung* handelt es sich um die Darstellung detaillierter Untermerkmale. Sie drückt eine „ist-ein“- oder „ist-Teil-von“-Beziehung aus.
- Einschränkungen werden durch *Einschluss*- bzw. *Ausschlussbeziehungen* („requires“ bzw. „excludes“) ausgedrückt.

Weiterhin können für die einzelnen Merkmale Parameter bestimmt werden, die bei der Konfiguration des Systems berücksichtigt werden müssen. Aus dem Zusammenspiel verschiedener Parameter können sich weitere Beschränkungen für das System ergeben.

Ist die Merkmalmodellierung abgeschlossen, wird die prinzipielle Architektur der Systemfamilie entworfen. Hier finden auch die Variabilitäten und Kombinationsmöglichkeiten der einzelnen Merkmale Berücksichtigung.

Ergebnis dieses Entwurfes ist eine generische Architektur, welche durch einen Generierungsschritt in die Systemarchitektur einer konkreten Anwendung überführt wird. So lässt sich nur durch die Veränderung einzelner Generierungsparameter eine Vielzahl von Anwendungen erzeugen.

Neben der großen Anzahl Anwendungen, die auf eine einfache Art und Weise generiert werden können, besteht ein weiterer Vorteil von Systemfamilien darin, dass Änderungen an existierenden Architekturteilen ohne größere Probleme vorgenommen werden können. Aufgrund des modularen Aufbaus ist es jederzeit möglich, einzelne Komponenten herauszulösen und zu verändern oder auch vollkommen neue Komponenten in das System einzufügen.

2.2 Bedienkonzepte

Bei der Entwicklung eines Gerätes steht neben anderen Aspekten auch die Entscheidung an, wie die Bedienung dieses Gerätes erfolgen soll. Hierzu gibt es verschiedene Ansätze, welche sich in Art und Umfang der Realisierung sowie ihren Vor- und Nachteilen unterscheiden. In *Abb. 2-2* sind die verschiedenen Konzepte grafisch zusammengefasst.

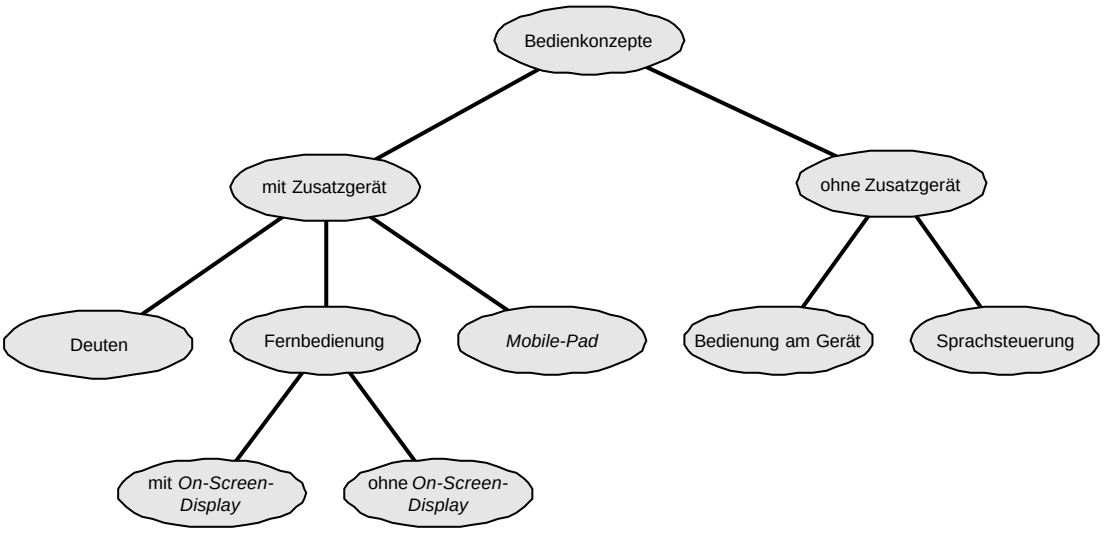

Abb. 2-2: Bedienkonzepte

Die hier dargestellten Möglichkeiten sollen im Folgenden analysiert werden.

2.2.1 Fernbedienung

Diese Art der Ansteuerung ist vor allem im Bereich der Unterhaltungselektronik die am häufigsten verwendete. Hierbei ist es oftmals so, dass einige Basisfunktionen direkt am Gerät ausgelöst werden können, der größte Teil aber über die Fernbedienung ange-

steuert wird. Es existieren jedoch auch Lösungen, bei denen die Nutzerschnittstelle redundant sowohl am Gerät selber als auch auf der entsprechenden Fernbedienung angelegt wurde.

Auch bei der Gestaltung der Fernbedienung kommen verschiedene Konzepte zum Einsatz. Zunächst besteht die Möglichkeit, für jede Funktion eine Taste bereitzustellen und diese direkt auf der Fernbedienung zu platzieren. Dies gestattet zwar eine direkte Verfügbarkeit jeder einzelnen Funktion, erzeugt aber eine enorme Unübersichtlichkeit und macht es so vor allem weniger versierten Nutzern schwer, sich in die Gerätebedienung einzuarbeiten. Dem kann man zumindest teilweise entgegenwirken, indem man mechanische Blenden installiert, welche die weniger gebräuchlichen Funktionen quasi unsichtbar machen.

Bei hinreichend großem Funktionsumfang wird sehr schnell die Oberfläche der Fernbedienung zu klein, so dass nicht alle notwendigen Tasten untergebracht werden können. Hier wird häufig auf das Prinzip der Mehrfachbelegung von Tasten zurückgegriffen. Diese Zweit- und Drittfunktionen können dann beispielsweise über einen Schalter oder eine gleichzeitig zu drückende Taste angesprochen werden, was jedoch die Handhabung erheblich komplizieren kann.

2.2.2 *On-Screen-Display*

Mit der im Laufe der Zeit immer weiter steigenden Komplexität der Geräte sahen die Hersteller sich gezwungen, nach neuen Bedienkonzepten Ausschau zu halten. Fündig wurde man im Bereich der PCs, bei deren Software das Prinzip von Menüs und der Navigation in diesen bereits seit vielen Jahren Anwendung findet.

Zum Einsatz kommt die Steuerung mittels *On-Screen-Displays* (*OSD*) in erster Linie bei Geräten, die bereits über einen Bildschirm verfügen, wie dies z.B. bei Fernsehgeräten der Fall ist. Selbstverständlich lassen sich auch andere Geräte mit einem *OSD* ausstatten, jedoch ist hier ein erhöhter technischer Aufwand abseits der primären Funktionen erforderlich.

Auch bei diesem Konzept wird mit einer Fernbedienung gearbeitet. Im Unterschied zum in Abschnitt *2.2.1* vorgestellten Verfahren muss die Fernbedienung jedoch nur noch mit Tasten zum Aufruf des Menüs bzw. zur Auswahl eines Menüpunktes sowie Tasten zur Navigation innerhalb der Menüs ausgestattet sein. Optional können darüber hinaus Tasten existieren, die eine direkte Ansteuerung einzelner Basisfunktionen erlauben.

Vorteil eines *OSD* ist unter anderem die Möglichkeit, in einem Menüeintrag mehr erklärende Informationen unterzubringen, als dies beispielsweise auf einer einzelnen Taste, welche im Normalfall nur mit einem (häufig wenig aussagekräftigen) Zeichen versehen ist, möglich ist. Weiterhin sind dem Funktionsumfang prinzipiell keinerlei

Grenzen gesetzt, wobei auch die Nutzerfreundlichkeit bei entsprechender Gestaltung der Menüstruktur erhalten bleibt.

Die Nachteile dieser Lösung sind in der Tatsache zu suchen, dass die Realisierung nur für Systeme mit einem Bildschirm interessant ist, sowie darin, dass während der Arbeit mit den Menüs der eigentliche Bildinhalt zumindest teilweise überblendet wird.

2.2.3 *Mobile-Pad*

Die Bedienung eines Gerätes mit Hilfe eines so genannten *Mobile-Pads* oder eines *PDA* ist eine noch nicht weit verbreitete aber dennoch zukunftsträchtige Variante.

Aufgrund der Tatsache, dass auf solchen Geräten auch komplexe Software zum Einsatz gebracht werden kann, ist es möglich, ein Konzept zu realisieren, welches eine hohe Flexibilität der Bedienung erlaubt.

So ist es möglich, Funktionen, die weit über die eigentliche Bedienung des Gerätes hinausgehen, zu integrieren. Denkbar wäre an dieser Stelle beispielsweise die Darstellung und Verwendung von *EPG-Daten* oder die Nutzung verschiedener Datenbanken, auf welche über eine Internetverbindung zugegriffen werden kann. Daneben lässt sich auch die Art des Auslösens von Funktionen variabel gestalten. An dieser Stelle können z.B. Konzepte aus dem PC-Bereich verwendet werden.

Nachteil dieser Lösung sind die hohen Kosten, welche ein solches Gerät verursacht.

2.2.4 Andere Varianten

Neben den bisher genannten Bedienkonzepten existieren auch noch weitere Möglichkeiten, verschiedene Geräte anzusteuern. Hierzu zählt auch die ursprünglichste Variante, bei der die Bedienelemente an der Vorderseite des Gerätes angebracht sind und die Bedienung hierüber erfolgt.

Daneben gibt es jedoch eine Reihe innovativer Methoden, die an dieser Stelle ebenfalls nicht unerwähnt bleiben sollen. Allerdings haben sie teilweise den Nachteil, dass sie (noch) nicht ausgereift sind und sich noch in der Entwicklung befinden.

Hierzu zählt beispielsweise die Steuerung durch Sprachkommandos, wie sie heute schon u.a. bei Mobiltelefonen zum Einsatz kommt. Bei diesem Verfahren werden keine zusätzlichen Geräte zum Versenden von Steuersignalen benötigt und die Kommunikation mit dem anzusteuernden Gerät kann im Grunde nach den gleichen Prinzipien wie die Kommunikation mit einem menschlichen Gegenüber erfolgen. Der Nachteil dieser Lösung liegt darin begründet, dass heutige Spracherkennungssysteme darauf angewiesen sind, dass keine großen Variationen in der Aussprache der Befehle existieren und dass umgebende Störgeräusche möglichst gering gehalten werden.

Auch eine Steuerung von Geräten mittels Deuten wäre realisierbar. Hierbei wird beispielsweise mit einem Laserpointer auf eine Stelle auf dem Gerät oder dem Bildschirm

gezeigt, woraufhin die Funktion ausgelöst wird. Denkbar ist in diesem Zusammenhang eine Realisierung in Verbindung mit einem *On-Screen-Display.*

2.2.5 Anforderungen an ein Bedienkonzept

Da im Rahmen dieser Arbeit u.a. ein Bedienkonzept erstellt werden soll, eröffnet sich die Frage, welche Anforderungen die Nutzer an ein solches Konzept stellen. Hier lassen sich verschiedene Vorlieben erkennen:

- *Einfache Bedienung*: Vor allem ältere Anwender legen Wert auf eine möglichst einfache Bedienung. Sie beschränken sich im Allgemeinen auf die Verwendung der grundlegenden Funktionen des entsprechenden Gerätes.
- *Individuelles Konzept*: Menschen, die häufig Unterhaltungsgeräte nutzen, sind daran interessiert, ein möglichst exakt auf ihre Bedürfnisse abgestimmtes Konzept zur Verfügung zu haben.
- *Anpassungsfähigkeit*: Die Bedienung sollte auch die Möglichkeit haben, sich im Laufe der Zeit an veränderte Bedingungen anzupassen.
- *Intuitive Bedienung*: Insbesondere technisch weniger versierten Nutzern ist daran gelegen, dass sich das Prinzip der Bedienung möglichst intuitiv erschließt und keine langwierige Einarbeitungszeit notwendig ist. Dies sollte allerdings nicht zu Lasten des Funktionsumfangs gehen.
- *Integration vieler Geräte*: Auch die Tatsache, dass das Nebeneinander verschiedener Geräte meist mit einer Vielzahl von Fernbedienungen einhergeht, wird als Nachteil angesehen. Es besteht der Wunsch nach einem Konzept, das die Integration möglichst vieler Geräte erlaubt.
- *Umfassende Funktionalität*: Neben der unmittelbaren Ansteuerung von Geräten soll die Fernbedienung eine Reihe von darüber hinaus gehenden Funktionen aufweisen.

2.3 Flexibilität der Bedienung - Stand der Technik

In diesem Abschnitt werden auf dem Markt verfügbare Fernbedienungen daraufhin untersucht, inwieweit die dort realisierten Konzepte in der Lage sind, Aufgaben, die jenseits der Ansteuerung des Gerätes liegen, zu dem sie gehören, zu erfüllen.

So ist es beispielsweise wünschenswert, dass Fernbedienungen in der Lage sind, mehrere unterschiedliche Geräte anzusteuern. Daneben sollten auch individuelle Gestal-

tungsmöglichkeiten realisiert werden und es sollte die Möglichkeit gegeben sein, weiterführende Daten in die Fernbedienung zu integrieren.

2.3.1 Fernbedienungen für die Ansteuerung mehrerer Geräte

Nahezu alle Geräte aus dem Bereich der Unterhaltungselektronik sind heutzutage mit einer eigenen Fernbedienung ausgerüstet. Dies ist nicht zwingend ungünstig, da die Komplexität der Bedienung in Grenzen gehalten werden kann und somit der Bedienkomfort steigt.

Die meisten Haushalte sind jedoch mit mehreren Geräten ausgestattet. Aufgrund dessen entwickelt sich schnell eine stattliche Sammlung von Fernbedienungen, die untereinander inkompatibel sind. Um ein solches Chaos zu vermeiden, gibt es bereits verschiedene Ansätze.

Zum besseren Verständnis soll an dieser Stelle zunächst auf die grundsätzliche Funktionsweise von Infrarot-Fernbedienungen eingegangen werden, bevor dann einige Realisierungsvarianten betrachtet werden.

2.3.1.1 Prinzipielle Funktionsweise von Fernbedienungen

Fernbedienungen verwenden für gewöhnlich Infrarotdioden, welche bei einer Wellenlänge von 950 nm arbeiten [5]. Auf eine Trägerfrequenz, die zwischen 30 und 40 kHz liegt, werden die eigentlichen Daten aufmoduliert. Das entsprechende Empfängermodul hat die Aufgabe, die verwendete Trägerfrequenz zu erkennen und das gesendete Datensignal zu extrahieren.

Die oben erwähnte Inkompatibilität entsteht durch die Tatsache, dass die Wellenlänge und die Trägerfrequenz die einzigen vergleichbaren Faktoren bei den verschiedenen Herstellern sind. Schon bei der notwendigen Kodierung der Daten existieren erhebliche Unterschiede. Am gebräuchlichsten sind hier die Standards *RC-5*, *RECS 80* und *NEC*.

RC-5 bedient sich zur Datenübertragung der so genannten *Biphase*-Kodierung. Hierbei wird die logische „0“ durch einen Lichtimpuls gefolgt durch eine Pause und die logische „1“ durch eine Pause gefolgt durch einen Lichtimpuls kodiert.

Im *RECS 80*-Standard werden die Bits durch unterschiedlich lange Pausen, die Lichtimpulsen einer konstanten Länge von 140,8 µs folgen, kodiert. Die Länge der Pausen beträgt hierbei für die logische „0“ 5,06 ms und für die logische „1“ 7,59 ms.

NEC funktioniert ähnlich dem *RECS 80*-Standard, wobei sich die Länge der Pausen unterscheidet.

2.3.1.2 Fernbedienungen für Geräte des gleichen Herstellers

Bei manchen Herstellern ist es üblich, verschiedene Geräte mit einer Fernbedienung anzusteuern. Ein Nachteil dieser Lösung ist, dass man beim Kauf zusätzlicher Geräte an

einen bestimmten Hersteller gebunden ist. Weiterhin ist es häufig so, dass bestimmte Funktionen nur mit Hilfe der Originalfernbedienung nutzbar sind.

Die Firma *Bang & Olufsen* [6] hat das Konzept in hohem Maße perfektioniert und ausgeweitet. So sind die Fernbedienungen verschiedener Geräte dieser Firma ohne weitere Vorbereitungen in der Lage, das gesamte heimische Unterhaltungssystem anzusteuern. Diese Geräteintegration endet allerdings nicht bei der Fernbedienung an sich, sondern schlägt sich ebenso in der Architektur des gesamten Systems nieder. So können sämtliche Geräte miteinander vernetzt werden, und man ist in der Lage, über den Infrarotsensor eines Gerätes Befehle an ein anderes zu senden.

Möchte man allerdings die Beschränkung auf einen einzigen Hersteller nicht akzeptieren, sieht man sich gezwungen, auf andere Lösungsvarianten zurückzugreifen. Welche Möglichkeiten für die Bedienung mehrerer Geräte unterschiedlicher Hersteller bestehen, beschreiben die folgenden Abschnitte.

2.3.1.3 Einfache Universalfernbedienungen

Diese Lösung ist verhältnismäßig preiswert aber in ihrem Funktionsumfang entsprechend eingeschränkt. Meist sind diese Fernbedienungen lediglich in der Lage, eine Anzahl von Standardkommandos zu versenden, die von den meisten Geräten verstanden und akzeptiert werden. Bei Verwendung einer solchen Fernbedienung ist es häufig der Fall, dass ein Großteil der Funktionen des entsprechenden Gerätes nicht nutzbar ist.

Erheblich flexibler und somit in der Lage, höheren Ansprüchen gerecht zu werden, sind lernfähige Universalfernbedienungen.

2.3.1.4 Lernfähige Universalfernbedienungen

Diese Fernbedienungen sind mit einem Fundus an Codes sowie der Möglichkeit, den einzelnen Tasten beliebige Funktionen zuzuordnen, ausgestattet. Darüber hinaus besitzen sie jedoch auch das Potenzial, weitere Geräte zu integrieren.

Hierfür gibt es verschiedene Realisierungsvarianten. Zunächst besteht die Möglichkeit, Kommandos von der Originalfernbedienung zu übernehmen, also sozusagen von ihr zu lernen. Hierzu muss die Universalfernbedienung in der Lage sein, Befehle in Form von Infrarotsignalen aufzuzeichnen. Die Programmierung erfolgt, indem die einzelnen Kommandos auf der Originalbedienung ausgelöst, in der Universalbedienung gespeichert und einer beliebigen Taste zugeordnet werden.

Durch dieses Verfahren wird zwar eine sehr hohe Flexibilität der Fernbedienung gewährleistet, jedoch muss die Programmierung zumeist per Hand durchgeführt werden. Hieraus entsteht ein hoher Aufwand, der die Nutzerfreundlichkeit senkt. Eine Realisierung dieses Konzeptes existiert u.a. auch für *Palm-Handhelds*. Es handelt sich hierbei um das Projekt *OmniRemote* [8].

Die zweite Variante lernfähiger Fernbedienungen verlangt vom Nutzer zwar weniger Vorbereitungsarbeit, ist aber technisch aufwändiger zu realisieren und somit teurer. Hierbei werden die jeweiligen Befehlscodes für zu integrierende Geräte nicht von der Originalfernbedienung übernommen, sondern mit Hilfe eines Modems, welches eine Telefon- oder Internetverbindung errichten kann, herunter geladen. Es gibt auch bereits einige Hersteller, die Fernbedienungen anbieten, bei denen diese Lösungsvariante realisiert wurde [7].

Unabhängig von der Art und Weise der Integration neuer Geräte bieten nahezu alle Produkte die Möglichkeit, Makros aufzuzeichnen. Eine Taste kann so programmiert werden, dass beim Betätigen eine Reihe von Funktionen nacheinander ausgeführt wird. Dies ist vor allem bei häufig verwendeten Funktionskombinationen sinnvoll.

Alle hier angesprochenen Konzepte haben jedoch einen Nachteil: Die Anzahl, Form und Beschriftung der Tasten ist von vornherein festgelegt und durch den Nutzer nicht mehr modifizierbar. Ob und in welcher Form Konzepte und Lösungen existieren, die diesem Missstand Abhilfe leisten, soll im nächsten Abschnitt untersucht werden.

2.3.2 Flexibilität jenseits der Multi-Geräte-Steuerung

Bisher verstehen die meisten Hersteller unter „Flexibilität“ lediglich die Fähigkeit, möglichst viele Geräte bedienen zu können. Selbstverständlich gibt es aber darüber hinaus noch eine Reihe von Funktionen, die man sinnvoll in eine Fernbedienung integrieren kann.

An erster Stelle steht hier natürlich eine Erhöhung des Bedienkomforts durch individuelle Gestaltungsmöglichkeiten der Bedienoberfläche. Ansatzweise wird dies auch schon bei Fernbedienungen, wie sie im Abschnitt *2.3.1.4* beschrieben werden, umgesetzt.

Um jedoch diese Gestaltungsmöglichkeiten konsequent umzusetzen, ist es erforderlich, dass man sich vom herkömmlichen Konzept des Designs von Fernbedienungen verabschiedet. Die Realisierung der Tasten als „Hardware“ ist hier nicht mehr effizient. Vielmehr sieht man sich gezwungen, die Fernbedienung mit einem *Touchscreen* auszustatten, auf dem dann die Schaltflächen eingeblendet werden.

Ein offensichtlicher Nachteil einer derartigen Lösung besteht in den vergleichsweise hohen Kosten, die hier entstehen. Diese resultieren zum einen aus der erheblich teureren Hardware, die eingesetzt werden muss. Zum anderen ist man hier aber auch gezwungen, eine komplexe Software zu entwerfen, die nicht nur die Ansteuerung des *Touchscreens* und das Auslösen der Befehle realisiert, sondern in gleichem Maße Möglichkeiten bietet, neue Tasten zu generieren und ihnen Funktionen zuzuweisen.

An dieser Stelle muss auch wieder auf [7] verwiesen werden. Diese Firma bietet Fernbedienungen an, welche bereits die beschriebenen Fähigkeiten besitzen.

Ein weiterer Aspekt bei der Betrachtung der Flexibilität von Fernbedienungen ist die Beantwortung der Frage, ob die Fernbedienung neben der Steuerung von Geräten und den verschiedenen Gestaltungsmöglichkeiten weitere Fähigkeiten besitzt. Verwiesen werden soll in diesem Zusammenhang auf *iPronto* [9], ein Produkt der Firma *Philips*, welches neben umfassenden Kontrollmöglichkeiten für verschiedene Geräte noch weitere Funktionalitäten bietet. So ist bei diesem Produkt u.a bereits ein *Electronic Program Guide* integriert. Weiterhin besteht hier die Möglichkeit, das Gerät als *Internet-Browser* einzusetzen.

Darüber hinaus wäre es ebenfalls denkbar, eine lernende Fernbedienung zu realisieren, welche das Verhalten des Nutzers protokolliert und sich entsprechend anpasst bzw. eigenständig Vorschläge für diese Anpassung unterbreitet.

2.4 Das Digitale Video Projekt

Die vorliegende Arbeit beschäftigt sich mit der Entwicklung von Software für eine Universal-Fernbedienung. Eingebettet ist sie in das *Digitale Video Projekt* [10].

Das *Digitale Video Projekt* hat zum Ziel, ein System zu entwickeln, das eine neue und innovative Nutzung des Mediums „Fernsehen“ erlaubt. Ein Grundgedanke hierbei war die ausschließliche Verwendung von Standardkomponenten.

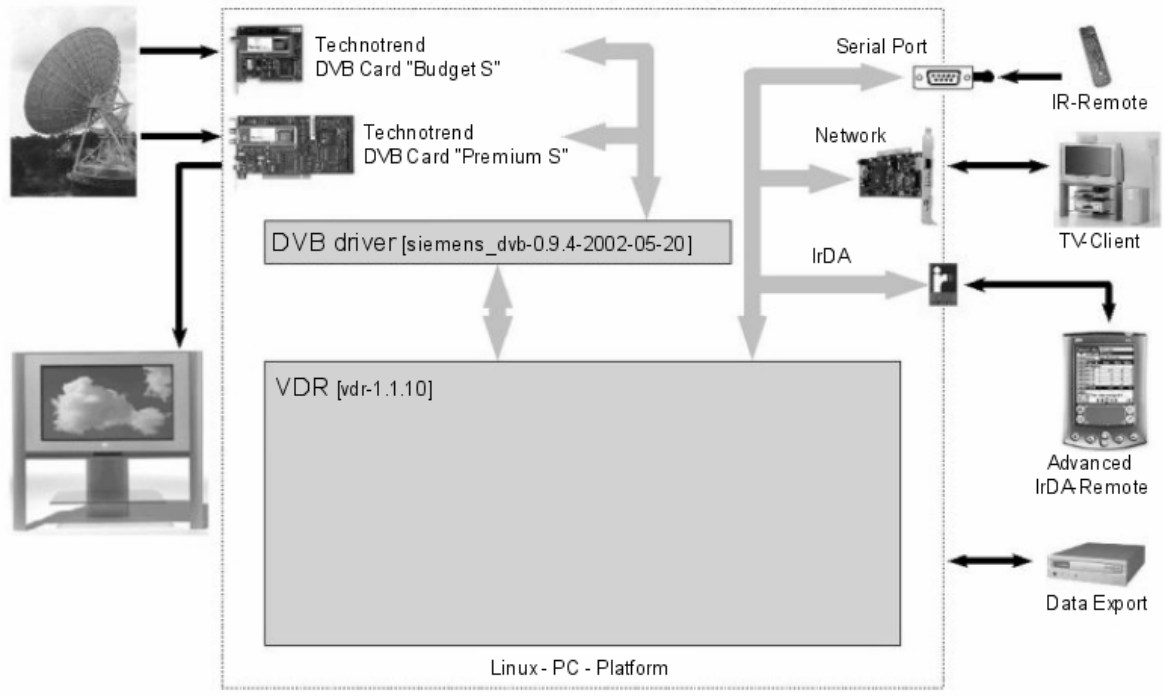

Abb. 2-3: Schema des Digitalen Video Projektes

Im Endergebnis soll ein System entstehen, welches nicht nur den Empfang und die Wiedergabe von digitalen Fernsehsignalen gewährleistet, sondern gleichzeitig Funktionalitäten bietet, die über das „gewöhnliche“ Fernsehen hinausgehen.

So soll es beispielsweise möglich sein, Sendungen sowohl auf Festplatte aufzuzeichnen, als auch diese auf andere Medien (DVD, CD etc.) zu exportieren. Das System kann nicht nur als Einzelplatz-Lösung eingesetzt werden, es existiert vielmehr auch die Variante, über eine Netzwerkverbindung beliebig viele *TV-Clients* zu bedienen. Neben dem Empfang der reinen Bild- und Tondaten ist es ebenfalls möglich, dem Nutzer programmbegleitende Informationen (*Electronic Program Guide*) zur Verfügung zu stellen.

Die Steuerung des Gerätes kann über eine herkömmliche Infrarot-Fernbedienung erfolgen, es ist aber ebenfalls die Verwendung einer *Advanced IrDA-Remote Control* unter Ausnutzung einer *IrDA*-Schnittstelle möglich. Die Realisierung der zweiten Variante ist Gegenstand der vorliegenden Arbeit.

Um eine möglichst hohe Flexibilität zu erreichen und sich effizient den Wünschen und Bedürfnissen der Nutzer anpassen zu können, wurde das Konzept als Systemfamilie realisiert.

Zur Vervollständigung sollen die (derzeitigen) technischen Daten des *Digitalen Video Projektes* nicht unerwähnt bleiben. Es werden momentan u.a. folgende Hardwarekomponenten eingesetzt:

- *Technotrend DVB Card „Budget S“*
- *Technotrend DVB Card „Premium S“*
- *Sony® Clié PEG-SL10*

Neben diesen Hardwarekomponenten wird folgende Software verwendet:

- *Linux* (Kernel-Version 2.4)
- *DVB*-Treiber *Siemens* 0.9.4
- *VDR* 1.1.10

Die letztgenannte Komponente bildet das zentrale Element des Systems und soll daher noch etwas näher betrachtet werden.

2.4.1 Das Projekt *Video Disk Recorder*

Mit dem Aufkommen von Festplatten mit großer Kapazität und der Entwicklung von *MPEG-Encodern* und *–Decodern* entstand auch die Idee eines vollständig digitalen Videorekorders. Mittlerweile existiert eine Reihe von kommerziellen Produkten, welche eine derartige Funktionalität bieten. Allerdings sind die meisten dieser Produkte mit Kosten verbunden, die sich nicht auf die Anschaffung der jeweiligen Geräte beschränken, vielmehr wird eine Gebühr für die Inanspruchnahme verschiedener Dienste erhoben.

Das Projekt *Video Disk Recorder* [11] bietet nun eine Möglichkeit, sich seinen eigenen digitalen Satellitenreceiver und *Video Disk Recorder* zu erstellen. Hauptaugenmerk wird hierbei auf die Entwicklung einer Software gelegt, die in der Lage ist, in Zusammenarbeit mit den entsprechenden Treibern eine oder mehrere DVB-Karten anzusteuern. Daneben realisiert diese Software ein Bedienkonzept, welches dem Anwender vielfältige Nutzungsmöglichkeiten bietet:

- Steuerung via Infrarot-Fernbedienung (*LIRC*) oder Tastatur
- Unterstützung mehrerer *DVB-S*-Karten in einem System (bis zu vier Karten)
- Darstellung der *EPG-Daten* in verschiedenen Modi
- Rekorderprogrammierung via *Electronic Program Guide* oder manuell
- Speicherung der Aufnahmen auf Festplatte: automatische Aufteilung der Aufnahmen in Dateien (<2GB); unterstützt Verteilung der Aufnahmen auf verschiedene Verzeichnisse
- Unterstützung multipler Tonspuren und des *Dolby Digital*-Systems
- Wiedergabemodi: Normal, Pause, schneller Vor- und Rücklauf, Sprung an definierte Stelle, Sprung um 60 Sekunden
- Unterstützung der Editierung von Aufnahmen
- Netzwerkunterstützung (*SVDRP*): Verwaltung der Aufnahmen via *Telnet*
- Unterstützung der automatischen Ausführung von Kommandos zu Beginn oder Ende einer Aufnahme
- Unterstützung von *Mp3, DVD, (S)VCD, DivX* durch *Patches*.
- …

Beim *Video Disk Recorder*-Projekt handelt es sich um ein *Open-Source*-Projekt, d.h. jeder der Interesse hat, kann an der Weiterentwicklung der Software arbeiten, die Quellen sind allen frei zugänglich. Diese Eigenschaft des Systems war für die Verwendung im *Digitalen Video Projekt* enorm wichtig, da es sich früher oder später als unumgänglich erweisen wird, einige Modifikationen an der Software vorzunehmen.

2.5 Problemstellung & Lösungsansatz

Wie in den vorangegangenen Abschnitten bereits festgestellt wurde, existieren zu diesem Zeitpunkt eine Reihe verschiedener Bedienkonzepte und Fernbedienungsarten nebeneinander.

Diese sind für sich betrachtet in vielen Bereichen durchaus ausgereift. Das gilt vor allem im Sektor der Geräteintegration, hier gibt es eine große Auswahl an Produkten, die vielfältige Möglichkeiten in dieser Hinsicht bieten.

Darüber hinaus existieren jedoch kaum Anstrengungen, die Fernbedienung über die Verwendung zur Steuerung von Geräten hinaus einzusetzen. Im Bezug auf Mehrnutzerverwaltung und individuelle Gestaltung der Fernbedienung beispielsweise gibt es heutzutage nur sehr wenige und unzureichende Lösungen.

Davon abgesehen berücksichtigt keines der angesprochenen Bedienkonzepte alle möglichen Fähigkeiten und Eventualitäten. Des Weiteren sind die bestehenden Konzepte statisch organisiert und besitzen nicht die Möglichkeit, auf sich verändernde Anforderungen zu reagieren. Hierdurch wird der Funktionsumfang der entsprechenden Fernbedienungen auf seinen aktuellen Stand beschränkt und kann nicht bzw. nur unter hohem Aufwand erweitert werden.

Im Rahmen dieser Arbeit soll nun eine Lösung der oben beschriebenen Probleme entwickelt werden. Diese Aufgabenstellung gliedert sich in folgende Bereiche:

- *Entwicklung eines neuen Bedienkonzeptes*: Das Konzept soll die angesprochenen Probleme berücksichtigen und eine Lösung anbieten. Weiterhin ist eine hohe Variabilität des Konzeptes zu gewährleisten, so dass der Nutzer die Möglichkeit hat, sich eine seinen Wünschen angepasste Fernbedienung zusammenstellen zu können.

- *Entwicklung eines Systems als Umsetzung des Konzeptes*: Das entwickelte Bedienkonzept soll in einem weiteren Schritt in einer konkreten Systemarchitektur umgesetzt werden.

- *Entwicklung eines Prototypen*: Um die Funktionsfähigkeit des entworfenen Systems zu demonstrieren, soll ein erster Prototyp entwickelt werden.

Zur Lösung der Aufgabe soll eine Architektur in Form einer Systemfamilie entwickelt werden. Diese soll es erlauben, auf Basis der ausgewählten Merkmale ein kundenspezifisches Produkt zu konfigurieren und anschließend eine entsprechende Applikation zu generieren. Die Entwicklung einer derartigen Architektur gliedert sich in folgende Abschnitte:

- *Erstellen eines Merkmalmodells*: Die potenziellen Fähigkeiten des Systems sind in Form von Merkmalen zu erfassen und in einem Merkmalmodell hierarchisch darzustellen.

- *Umsetzung der Merkmale in Pakete*: Die einzelnen Merkmale sind in einem weiteren Schritt geeignet zusammenzufassen und in Pakete, welche separat entwickelt werden können, umzusetzen.

- *Modellierung der Pakete*: Die einzelnen Pakete werden näher definiert und es wird eine grundlegende Architektur für sie entworfen.
- *Entwicklung des Prototypen*: Zur Abrundung der Arbeit soll ein erster Prototyp entworfen werden, in dessen Rahmen grundlegende Strukturen und ausgewählte Pakete implementiert werden.

Ebenfalls im Vordergrund steht, wie auch beim *Digitalen Video Projekt*, die Verwendung existierender Techniken. Dies gilt vor allem im Bereich der verwendeten Hardware. Bei der Realisierung der Universal-Fernbedienung soll ein *PDA* zum Einsatz kommen. Dieser arbeitet mit dem Betriebsystem *Palm OS®*, auf dessen Basis die Software der Fernbedienung entwickelt wird.

3 Anforderungsanalyse & Merkmalerfassung

Aus den im vorangegangenen Kapitel gewonnenen Erkenntnissen sollen nun die zu realisierenden Fähigkeiten des zu erstellenden Systems erarbeitet werden. In einem weiteren Schritt sollen dann aus diesen Anforderungen Merkmale modelliert werden. Weiterhin wird festgestellt, welche Abhängigkeiten zwischen diesen Merkmalen bestehen und welche Einschränkungen sich hieraus für das System ergeben.

3.1 Anforderungsanalyse

In Kapitel *2.3* wurde bereits darauf eingegangen, in welcher Form heutzutage flexible Fernbedienungen existieren und wo die Vor- und Nachteile dieser Lösungen liegen Es lässt sich feststellen, dass die meisten Fernbedienungen die möglichen Anwendungsgebiete jeweils nur zu einem kleinen Teil unterstützen. Deswegen sieht sich der Nutzer bei der Auswahl seiner Fernbedienung gezwungen, Abstriche bei seinen Ansprüchen zu machen.

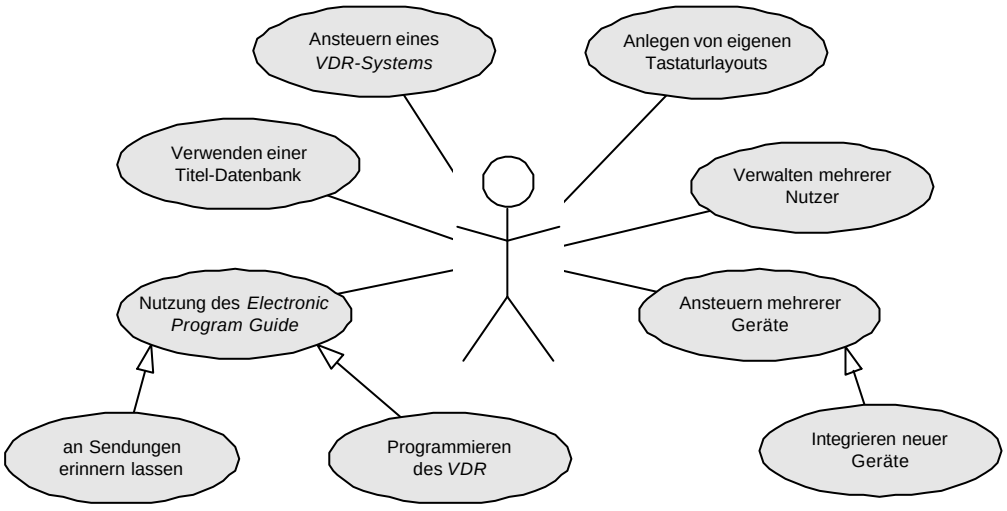

Abb. 3-1: Anwendungsmöglichkeiten der zu entwickelnden Universal-Fernbedienung

Es gilt nun, in dem zu entwickelnden System die Vielzahl der Anwendungsmöglichkeiten zu integrieren und so potenziellen Nutzern eine möglichst umfassende Lösung zur Verfügung zu stellen. Im weiteren Verlauf dieses Abschnittes sollen die einzelnen Anforderungen näher definiert werden, wobei nicht darauf eingegangen wird, welche dieser geforderten Fähigkeiten in allen Mitgliedern der Systemfamilie zur Verfügung

stehen sollen und welche optional sind. Diese Entscheidungen werden im Abschnitt *3.2* getroffen.

Die in *Abb. 3-1* dargestellten Anwendungsmöglichkeiten für die Universal-Fernbedienung sollen im Weiteren näher erläutert werden.

3.1.1 Fähigkeiten im Bereich des *VDR-Projektes*

Die vorliegende Arbeit ist im Kontext des *Digitalen Video Projektes* zu betrachten, zu dessen grundlegenden Merkmalen die Integration des *Video Disk Recorders* zählt. Daher soll u.a. im Ergebnis des Entwicklungsprozesses die Möglichkeit gegeben sein, ein *VDR-System* anzusteuern.

Hierbei ist es in einer Basisversion zunächst nur erforderlich, grundlegende Funktionen, welche die Ausnutzung aller Fähigkeiten des *Video Disk Recorders* gewährleisten, zu integrieren. Daher ist die Ansteuerung folgender Funktionalitäten zu modellieren:

- Senderanwahl (direkt über einen Zifferblock oder indirekt über [+/-])
- Regulierung der Lautstärke (sofern vom *VDR* angeboten)
- Regulierung der Bildattribute (Helligkeit, Kontrast,…) (sofern vom *VDR* angeboten)
- Aufruf des *On-Screen-Menüs* des *Video Disk Recorders*
- Navigation zwischen den einzelnen Menüpunkten
- Auswahl eines Menüpunktes
- Videoschnitt

Darüber hinaus lässt sich bei Betrachtung der im *VDR* realisierten Fähigkeiten feststellen, dass einige dieser Funktionalitäten durchaus sinnvoll in die zu entwickelnde Fernbedienung integriert werden können. Besonderes Augenmerk soll in diesem Zusammenhang auf die Nutzung des *Electronic Program Guide* gelegt werden.

Bisher werden die Programminformationen mittels *On-Screen-Display* über das eigentliche Fernsehbild gelegt. Wenn nun diese Informationen auf der Fernbedienung verfügbar gemacht werden, ist es möglich, sie abzurufen, ohne sich in Reichweite des *VDR-Systems* zu befinden. Es ist also zu realisieren, dass die *EPG-Daten* auf die Fernbedienung übertragen und dort angezeigt werden können. Analog zum ursprünglichen Verfahren im *Video Disk Recorder* sollen auch hier verschiedene Modi zur Anzeige der Informationen zur Verfügung stehen:

- Anzeige der Sendungen, die zum aktuellen Zeitpunkt laufen, für alle Sender
- Anzeige aller Sendungen des aktuellen Senders für den aktuellen Tag

- Anzeige aller Sendungen ausgewählter Sender über einen ausgewählten Zeitraum
- …

Um nicht jedes Mal die *EPG-Daten* neu übertragen zu müssen, muss das System in der Lage sein, die Informationen für einen bestimmten Zeitraum in der Fernbedienung zu speichern. Die Länge dieses Zeitraums hängt sowohl von den Wünschen des Anwenders als auch von der Speicherkapazität der Fernbedienung ab.

Die Programminformationen sollen selbstverständlich über die reine Anzeige hinaus genutzt werden können. Zum einen soll die Möglichkeit bestehen, sich eine Notiz zu machen, so dass die Fernbedienung in der Lage ist, zum entsprechenden Zeitpunkt auf eine beginnende Sendung hinzuweisen. Auf der anderen Seite ist ebenfalls sicherzustellen, dass über die Auswahl einer Sendung der Rekorder programmiert werden kann.

Wenn die Möglichkeit besteht, den Rekorder zu programmieren, sollten natürlich auch Funktionalitäten realisiert werden, die es erlauben, die Programmierungsdaten zu modifizieren. Konkret bedeutet dies, dass das System die Fähigkeit besitzen muss, die Informationen bezüglich der Rekorderprogrammierung vom *Video Disk Recorder* auf die Fernbedienung zu übertragen, dort die gewünschten Veränderungen vorzunehmen und die modifizierten Daten wieder an den *Video Disk Recorder* zu senden.

Beschränkt man sich bei der erwähnten Übertragung der Daten nicht nur auf die Nutzung der Infrarotschnittstelle, sondern greift vielmehr auf andere Medien, wie Telefon- oder Internetverbindungen zurück, lässt sich auch eine Fernabfrage bzw. –programmierung bewerkstelligen.

Da die Daten, die im Rahmen des *Electronic Program Guide* erhalten werden, nicht sonderlich umfassend sind, sollte die Möglichkeit bestehen, auf anderen Wegen weitergehende Informationen zu einer Sendung zu erhalten. Hierbei soll auf die Nutzung einer (im Internet verfügbaren) Titel-Datenbank, wie sie auch im *Digitalen Video Projekt* bereits vorgesehen ist, zurückgegriffen werden.

3.1.2 Fähigkeiten im Bereich anderer Geräte

Die wenigsten potenziellen Nutzer werden sich auf die Verwendung eines *VDR-Systems* beschränken, sondern vielmehr noch eine Anzahl anderer Geräte betreiben. Daher ist es erforderlich, dass auch in diesem Bereich entsprechende Fähigkeiten modelliert werden.

Es ist also sicherzustellen, dass neben einem *Video Disk Recorder* noch weitere Geräte angesteuert werden können. Dabei soll die Art des Gerätes keine Rolle spielen, es ist vielmehr zu gewährleisten, dass Videorekorder, Fernseher, CD-Player und ähnliches bedient werden können.

Da das System nicht darauf beschränkt werden soll, nur „ab Werk“ integrierte Geräte ansteuern zu können (dennoch sollte auch diese Möglichkeit gegeben sein), muss dafür gesorgt werden, dass der Nutzer die Möglichkeit hat, die Liste der unterstützten Geräte im Laufe der Zeit selbstständig zu erweitern. Diese Erweiterung der Bedienmöglichkeiten soll auf verschiedene Art und Weise geschehen können:

- *Lernen von anderen Fernbedienungen*: Das System soll in der Lage sein, von fremden Fernbedienungen gesendete Infrarotsignale zu empfangen und diese dann einer bestimmten Funktion zuzuordnen.
- *Download von Codes*: Das System soll in der Lage sein, von Dritten bereitgestellte Steuercodes für verschiedene Geräte per Telefonverbindung oder über eine Internetverbindung zu erhalten.

An dieser Stelle soll auch eine Verknüpfung der beiden Möglichkeiten berücksichtigt werden, so dass eine Art „intelligentes Lernen“ realisiert wird, bei dem nur ein Teil der Funktionen von der anderen Fernbedienung direkt übernommen wird und die Vervollständigung des *Code-Satzes* mittels eines Downloads vorgenommen wird.

Zu beachten ist, dass Art und Umfang der Funktionen der jeweiligen Geräte sehr unterschiedlich sein können. Es ist daher sicherzustellen, dass die entsprechenden Tastenfelder vom Nutzer individuell generiert und angepasst werden können.

3.1.3 Fähigkeiten aus anderen Bereichen

Aufgrund der Tatsache, dass ein Großteil der Nutzer verschiedene Ansprüche an den Bedienkomfort seiner Universal-Fernbedienung stellt, soll auch diesen Bedürfnissen Rechnung getragen werden.

So sollte dem Nutzer die Möglichkeit gegeben sein, die individuelle Darstellung der Tastatur anzupassen und nach seinen eigenen Präferenzen zu gestalten. Dies schließt sowohl die Position, Größe und Beschriftung einer Taste ein, als auch die Anordnung verschiedener Tasten auf einer Bildschirmseite.

Ebenfalls ist es in den meisten Haushalten so, dass mehrere Personen die gleichen Unterhaltungsgeräte und somit auch die entsprechenden Fernbedienungen nutzen. Dabei kann jedoch nicht davon ausgegangen werden, dass alle Nutzer die gleichen Vorlieben und Bedürfnisse haben. Die Universal-Fernbedienung sollte also die Möglichkeit bieten, mehrere Nutzerprofile anzulegen und diese unabhängig voneinander zu gestalten. Diese Gestaltung wird sich zunächst in erster Linie auf die Definition eigener Tastaturlayouts und das Setzen von *Remindern*, um sich an bestimmte Sendungen erinnern zu lassen, konzentrieren.

3.2 Merkmalerfassung

Im vorangegangenen Abschnitt wurde dargelegt, welche Fähigkeiten die Systemfamilie prinzipiell besitzen soll. Diese Anforderungen sind nun in Merkmale und Submerkmale zu fassen, und es ist festzulegen, welche dieser Merkmale als obligatorisch bzw. optional gelten sollen. Das Merkmalmodell, das sich aus den folgenden Erläuterungen ergibt, ist in *Abb. 3-2* dargestellt.

3.2.1 Das Merkmal ANSTEUERUNG DES VDR

Zunächst einmal war gefordert, dass die zu entwickelnde Universal-Fernbedienung in der Lage sein soll, ein *VDR-System* anzusteuern. Hierbei soll zwischen Grundfunktionalitäten und weitergehenden Fähigkeiten unterschieden werden.

Als erstes Merkmal ist daher die grundlegende Möglichkeit, einen *Video Disk Recorder* anzusteuern, festzuhalten. Da es sich hierbei sozusagen um das Kernelement der Systemfamilie handelt, ist das Merkmal GRUNDFUNKTIONEN als obligatorisch zu kennzeichnen.

Die in GRUNDFUNKTIONEN zusammengefassten Funktionen stellen lediglich einen Teilaspekt bei der Ansteuerung eines *Video Disk Recorders* dar, daher erscheint es angebracht, eine übergeordnete Ebene zu schaffen, welche dann durch die verschiedenen Funktionen des *Video Disk Recorders* verfeinert wird.

Als zweites Submerkmal der eben erwähnten Ebene, welche im Weiteren den Namen ANSTEUERUNG DES VDR erhalten soll, wird das Merkmal ELECTRONIC PROGRAM GUIDE modelliert. Aus dem Anforderungsprofil ist erkennbar, dass sich die Funktionen im Bereich dieses Merkmals in drei Teilbereiche gliedern:

- Ansehen und Speichern der Programminformationen
- Setzen eines *Reminders* auf Basis der *EPG-Daten*
- Programmieren des Videorekorders auf Basis der *EPG-Daten*

Sinnvollerweise sollten diese drei Bereiche auch als separate Merkmale modelliert werden. Dabei ist zu beachten, dass für das Empfangen der *EPG-Daten* sowie das Senden bzw. Empfangen der Programmierungsdaten verschiedene Übertragungsvarianten zur Verfügung stehen sollen. Unter Berücksichtigung dieser Randbedingungen erhält ELECTRONIC PROGRAM GUIDE folgende Submerkmale:

- DATENABGLEICH VIA INFRAROT als obligatorisches Merkmal,
- DATENABGLEICH VIA TELEFON,

- **REMINDER**,
- **PROGRAMMIEREN VIA TELEFON** sowie
- **PROGRAMMIEREN VIA INFRAROT** als optionale Merkmale.

Bei der Auswahl von **PROGRAMMIEREN VIA TELEFON** wäre es durchaus zweckmäßig, auch das Merkmal **DATENABGLEICH VIA TELEFON** auszuwählen. Es handelt sich hierbei jedoch nicht um eine zwingende Notwendigkeit.

Im Anforderungsprofil wird darüber hinaus noch die Fähigkeit, die Programmierungsdaten für den Rekorder modifizieren zu können, verlangt. Diese ist sinnvollerweise in den Merkmalen **PROGRAMMIEREN VIA INFRAROT** bzw. **PROGRAMMIEREN VIA TELEFON** zu integrieren.

Bei der Implementierung des Merkmals **ELECTRONIC PROGRAM GUIDE** ist zu beachten, dass die *EPG-Daten* für einen bestimmten Zeitraum gespeichert werden können.

Neben der Verwendung des *Electronic Program Guide* zur Erlangung von Informationen über bestimmte Sendungen soll das System auch die Option bieten, eine Titel-Datenbank zu nutzen. Für die Realisierung dieser Forderung ist der Aufbau einer Internetverbindung vonnöten. An dieser Stelle muss darauf hingewiesen werden, dass im *Digitalen Video Projekt* ebenfalls Merkmale erfasst sind, die eine solche Verbindung ermöglichen. Es ist dem Nutzer die Wahl zu lassen, ob die Internetfähigkeiten des *Digitalen Video Projektes* genutzt werden sollen oder ob das System in die Lage versetzt werden soll, über die Fernbedienung eine eigenständige Internetverbindung aufzubauen.

Das zu modellierende Merkmal **TITEL-DATENBANK** unterteilt sich also in zwei Submerkmale

- **EIGENE VERBINDUNG** und
- **VERBINDUNG ÜBER VDR**.

Soll die Internetverbindung allerdings über den *Video Disk Recorder* hergestellt werden, ist es erforderlich, das Merkmal **WEB** im *Digitalen Video Projekt* auszuwählen.

Aufgrund des Umstandes, dass die Suche in der Titel-Datenbank auf Basis von *EPG-Daten* erfolgt, bedingt die Auswahl des Merkmals **TITEL-DATENBANK** auch die Auswahl des Merkmals **ELECTRONIC PROGRAM GUIDE**.

3.2.2 Das Merkmal ANSTEUERUNG ANDERER GERÄTE

Als zweiten großen Abschnitt der gewünschten Fähigkeiten des Systems lässt sich die Forderung nach der Möglichkeit, andere Geräte ansteuern zu können, erkennen. Hierbei soll neben dem Senden der korrekten Infrarotsignale auch die Integration weiterer Geräte im Vordergrund stehen.

Diese Integration soll laut Anforderungsprofil auf drei verschiedene Arten vollzogen werden können. Zum einen soll die Möglichkeit bestehen, *Codes* von anderen Fernbedienungen zu empfangen, zum anderen ist zu realisieren, dass *Codes* per Telefon oder Internet bezogen werden können. Letztlich können natürlich auch bereits „ab Werk“ einige Geräte integriert sein.

Hier sind also zunächst drei Merkmale zu definieren:

- **LERNEN VON FERNBEDIENUNGEN**: Dieses Merkmal beinhaltet die Übernahme der Funktionen von anderen Fernbedienungen.
- **DOWNLOAD VON CODES**: Hier werden Funktionen zum Download von *Code-Sätzen* aus dem Internet bzw. über eine Telefonverbindung bereitgestellt.
- **PROGRAMMIERT**: Dieses Merkmal beinhaltet„ab Werk“ zur Verfügung stehende *Code-Sätze*.

Bei der ersten Variante, dem Lernen von anderen Fernbedienungen, werden zwei verschiedene Methoden realisiert:

- **EINFACH**: Das „Lernen“ erfolgt dadurch, dass auf der Original-Fernbedienung sämtliche Funktionen ausgelöst und die entsprechenden Infrarot-Sequenzen auf der Fernbedienung gespeichert werden.
- **INTELLIGENT**: Hierbei werden nur einige Infrarotsequenzen übermittelt und das System versucht auf Basis dieser Daten den Fernbedienungs-Typ zu ermitteln und die entsprechenden *Codes* per Download zu erhalten.

Die Auswahl des Merkmals **INTELLIGENT** bedingt jedoch auch die Auswahl des Merkmals **DOWNLOAD VON CODES** bzw. eines seiner Submerkmale.

Diese Submerkmale beschreiben die verschiedenen Weisen, den Download zu bewerkstelligen:

- **DOWNLOAD VON CODES VIA TELEFON**: Für den Download kann eine Telefonverbindung genutzt werden.
- **DOWNLOAD VON CODES VIA INTERNET**: Es besteht die Möglichkeit, die *Codes* über eine Internetverbindung zu erhalten.

Bei letzterem Submerkmal gibt es, äquivalent zum Merkmal **TITEL-DATENBANK**, die beiden Varianten, eine eigene Internetverbindung aufzubauen oder aber eine durch das *VDR-System* zur Verfügung gestellte Verbindung zu nutzen.

3.2.3 Das Merkmal NUTZERPROFIL

Unter diesem Merkmal sollen Funktionen und Fähigkeiten, wie sie im Abschnitt *3.1.3* beschrieben wurden, erfasst werden.

NUTZERPROFIL umfasst zwei Submerkmale:

- **TASTATUREDITOR**: Dieses Merkmal beinhaltet Funktionen, die es dem Nutzer erlauben, Tastenfelder in beliebigen Variationen zu erstellen und so die Fähigkeiten, die durch die anderen Merkmale zur Verfügung gestellt werden, optimal zu nutzen.
- **MEHRNUTZERVERWALTUNG**: Das Merkmal versetzt das System in die Lage, verschiedene Nutzer zu verwalten und die Profile unabhängig voneinander zu gestalten. Dies kann allerdings nur in dem Rahmen geschehen, der von der Konfiguration des Gesamtsystems gesteckt wird.

Wichtig ist es, an dieser Stelle anzumerken, dass eine Auswahl des Merkmals **ANSTEUERUNG ANDERER GERÄTE** die Selektion von **TASTATUREDITOR** zwingend erforderlich macht. Dies ergibt sich aus der Tatsache, dass die Art und Anzahl der Funktionen zur Bedienung eines Fremdgerätes nicht vorhersagbar sind und der Nutzer die Möglichkeit haben muss, diesen Funktionen auch entsprechende Tasten zuordnen zu können.

So man sich für das Merkmal **NUTZERPROFIL**, welches selbst optional ist, entscheidet, ist die Selektion von **TASTATUREDITOR** obligatorisch.

Das hier dargestellte Merkmalmodell ist als Bestandteil des Merkmalmodells des *Digitalen Video Projektes* zu sehen. Für die weiteren Teile der Arbeit ist allerdings nur dieser Teil des Modells relevant.

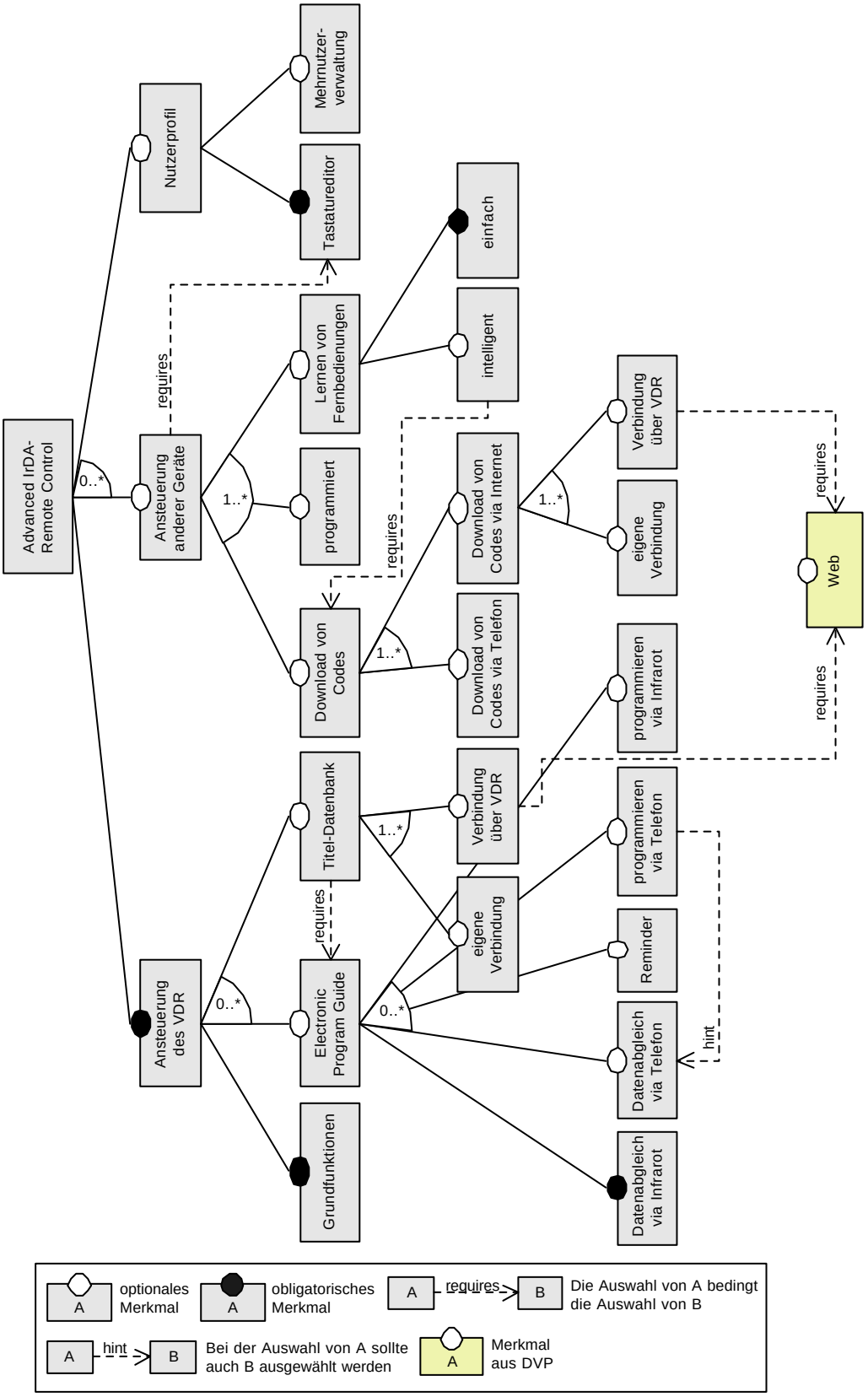

Abb. 3-2: Merkmaldiagramm der zu entwickelnden Universal-Fernbedienung

3.2.4 Parameter und Beschränkungen

Um ein funktionsfähiges System zu erhalten, ist es notwendig, sowohl für die gesamte Systemfamilie als auch für einzelne Merkmale Parameter festzulegen. Ebenfalls soll an dieser Stelle betrachtet werden, welche Beziehungen zwischen diesen Parametern bestehen und welche Beschränkungen für die Systemkonfiguration sich hieraus ergeben.

Ein wichtiger Parameter bei der Erstellung eines Systems ist der in der Fernbedienung verfügbare Arbeitsspeicher. *Abb. 3-3* zeigt die sich aus diesem Parameter ergebenden Beschränkungen.

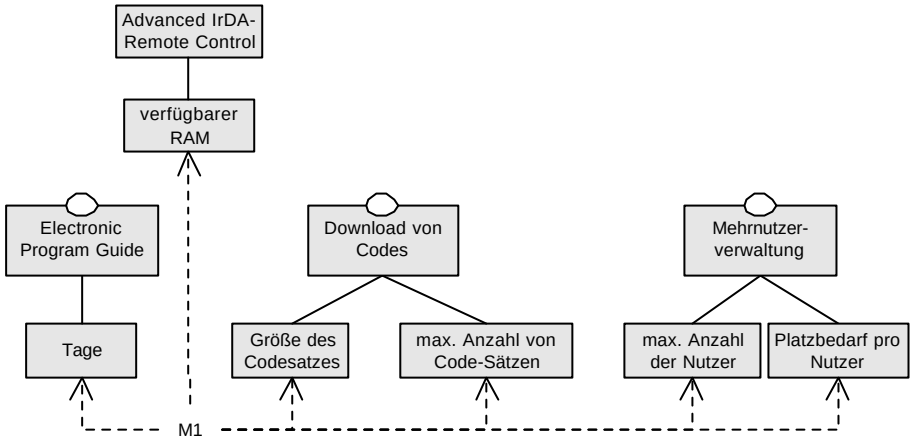

Abb. 3-3: Beschränkung M1

Es ist leicht zu erkennen, dass der von den Merkmalen beanspruchte Speicherplatz nicht größer sein darf als der maximal verfügbare Arbeitsspeicher. Geht man davon aus, dass die Parameter `verfügbarer RAM`, `Größe des Codesatzes` und `Platzbedarf pro Nutzer` in Kilobyte berechnet werden, ergibt sich für die Beschränkung `M1` folgende Formel:

> `Verfügbarer RAM` >= `Tage*400KB` + `Größe des Codesatzes` * `max. Anzahl von Code-Sätzen` + `Platzbedarf pro Nutzer` * `max. Anzahl der Nutzer`.

Anzumerken wäre an dieser Stelle, dass der Platzbedarf eines *EPG-Eintrags* nicht exakt bestimmt werden kann, da die einzelnen Sender hier keinen Restriktionen unterworfen sind und die Einträge individuell gestalten können. Man kann jedoch von einem ungefähren Platzbedarf von 8 Kilobyte pro Tag und Sender ausgehen. Legt man nun eine Senderzahl von 50 zugrunde, ergibt sich der hier verwendete Platzbedarf für die *EPG-Daten* von 400 Kilobyte.

Ebenfalls von großer Wichtigkeit für die Konfiguration des Systems ist die Verfügbarkeit verschiedener Schnittstellen. Ebenfalls von großer Wichtigkeit für die Konfiguration des Systems ist die Verfügbarkeit verschiedener Schnittstellen. Die sich hieraus ergebenden Beschränkungen sind in *Abb. 3-4* dargestellt.

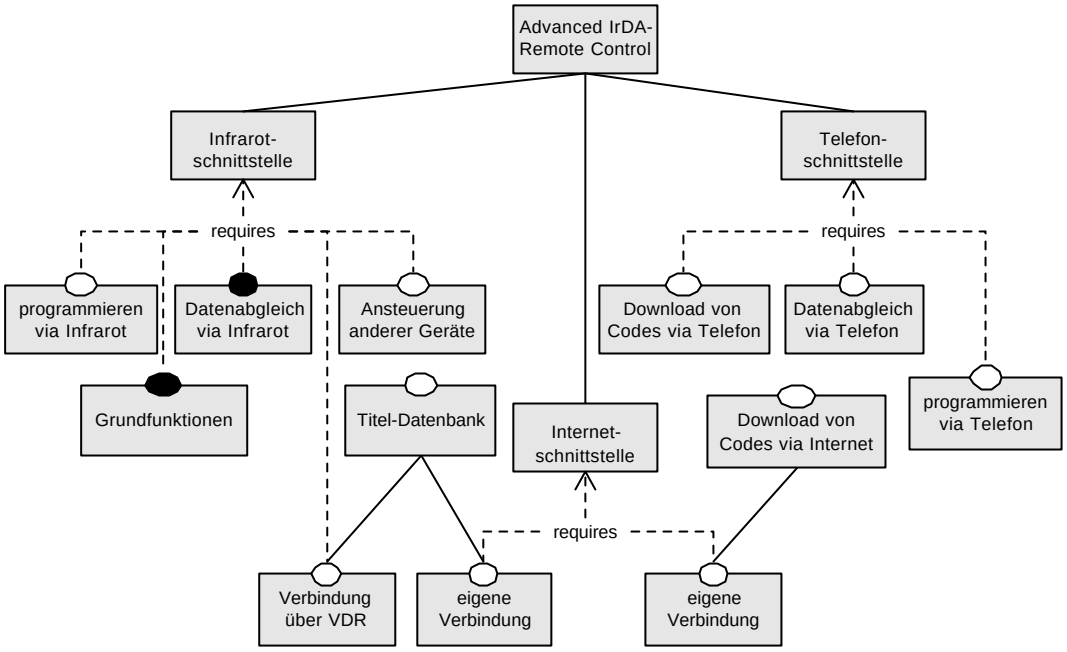

Abb. 3-4: Benötigte Schnittstellen

Wie leicht zu erkennen ist, ist die Aktivierung verschiedener Merkmale abhängig vom Vorhandensein bestimmter Schnittstellen. So machen die Merkmale **Grundfunktionen**, **programmieren via Infrarot**, **Datenabgleich via Infrarot** und **Ansteuerung anderer Geräte** die Integration einer Infrarotschnittstelle unabdingbar.

Weiterhin ist ein Internetzugang für die Aktivierung des Submerkmals **eigene Verbindung** der Merkmale **Download von Codes via Internet** bzw. **Titel-Datenbank** notwendig.

Eine Möglichkeit, eine Telefonverbindung aufzubauen, muss für die Merkmale **Download von Codes via Telefon**, **Datenabgleich via Telefon** und **programmieren via Telefon** vorhanden sein.

4 Lösungsansatz

Nachdem im Kapitel *3* die Anforderungen erfasst und die Merkmale der Systemfamilie herausgearbeitet wurden, ist es nun erforderlich, eine Architektur zu entwerfen, die es erlaubt, die dort formulierten Fähigkeiten zu realisieren.

Der Entwurf dieser Architektur wird in drei Schritten geschehen. Zunächst soll festgestellt werden, welche Komponenten das System beinhalten soll und wie die einzelnen Merkmale darauf abgebildet werden. Anschließend wird der funktionale Aspekt der zu entwerfenden Architektur beleuchtet, d.h. es wird untersucht, wie die Komponenten miteinander kommunizieren und welche Datenflüsse zwischen ihnen auftreten.. Im abschließenden dritten Teil folgt eine Betrachtung der innerhalb der Komponenten stattfindenden Prozesse und Aktivitäten.

Die konkrete Definition von Klassen für die einzelnen Komponenten kann an dieser Stelle nicht erfolgen, da hierfür die Beachtung der spezifischen Eigenschaften der für den Prototyp verwendeten Hardware notwendig ist.

4.1 Abbildung der Merkmale auf Pakete

Kapitel *3* beschäftigt sich mit der Frage, über welche Merkmale und Möglichkeiten das System verfügen soll. Es gilt nun, einen Weg zu finden, diese Merkmale derart auf Pakete bzw. Subsysteme abzubilden, dass eine möglichst einfache Generierung von Familienmitgliedern gewährleistet werden kann. Diese Generierung wird mit Hilfe von *Compiler-Direktiven* erfolgen.

Bei genauerer Betrachtung wird schnell klar, dass einige Merkmale sinnvoll in einem Paket zusammengefasst werden können, wohingegen andere wiederum ein separates Paket bilden werden. Darüber hinaus werden allerdings auch Elemente entstehen, die nicht direkt einem Merkmal zugeordnet werden, sondern vielmehr grundlegende Strukturen und Funktionen zur Verfügung stellen.

Zunächst sollen jedoch diejenigen Pakete definiert werden, die unmittelbar Merkmale repräsentieren. Das erste Merkmal, **Ansteuerung des VDR**, besteht aus drei Submerkmalen, von denen zwei direkt in einem Paket umgesetzt werden:

- `Grundfunktionen` (repräsentiert das Merkmal **Grundfunktionen**),
- `TDB` (repräsentiert das Merkmal **Titel-Datenbank**).

Das dritte Submerkmal **Electronic Program Guide** erfährt eine geringfügig andere Behandlung, da es selbst eine Reihe von Submerkmalen besitzt, deren Fähigkei-

ten für sich betrachtet bereits eine relativ hohe Komplexität aufweisen. So werden aus **Electronic Program Guide** und den fünf Submerkmalen drei Pakete gebildet:

- `EPG` (realisiert die Darstellung der *EPG-Daten* sowie die Submerkmale **Datenabgleich via Infrarot** und **Datenabgleich via Telefon**),
- `RecProg` (realisiert die Submerkmale **programmieren via Infrarot** und **programmieren via Telefon**),
- `Reminder` (realisiert das Submerkmal **Reminder**).

Das Merkmal **Ansteuerung anderer Geräte** beinhaltet drei Submerkmale, von denen zwei, nämlich **Download von Codes** und **Lernen von Fernbedienungen**, jeweils ein eigenes Paket erhalten werden. Zusätzlich dazu soll noch ein drittes Paket entstehen, welches die schlichte Ansteuerung von Geräten realisiert. **Ansteuerung anderer Geräte** wird also folgende Pakete umfassen:

- `AndereGeräte` (für die einfache Ansteuerung und grundlegende Strukturen),
- `Lernen` (repräsentiert das Merkmal **Lernen von Fernbedienungen** sowie dessen Submerkmale),
- `Download` (repräsentiert das Merkmal **Download von Codes** sowie dessen Submerkmale).

Die Umsetzung des Merkmals **Nutzerprofil** erfolgt analog zur Struktur seiner Submerkmale. Es entstehen demzufolge zwei Pakete:

- `Mehrnutzer` (Umsetzung des Merkmals **Mehrnutzerverwaltung**),
- `Tastatureditor` (Umsetzung des gleichnamigen Merkmals).

Es lässt sich feststellen, dass die einzelnen Merkmale eine Reihe von einander ähnelnden Funktionen aufweisen., z.B. senden bzw. empfangen viele Merkmale Infrarotsignale. Es ist daher durchaus sinnvoll, diese Fähigkeiten in eigenen Paketen zu realisieren, derer sich dann die anderen Pakete bedienen können. Es wird sich hierbei um folgende Elemente handeln:

- `Nutzerschnittstelle` (Funktionalitäten für die Darstellung von Daten (bspw. *EPG-Daten*) und Steuerelementen (Tasten, Menüs etc.) auf der Fernbedienung),
- `Datenbank` (Funktionalitäten für den Zugriff auf eventuell zu nutzende Datenbanken),

- `Kommunikation` (Funktionalitäten für den Datenaustausch via Infrarot, Telefon oder Internet).

Um eine einheitliche Architektur der einzelnen Merkmale bzw. der jeweiligen Pakete zu gewährleisten, ist die Festlegung von grundlegenden Strukturen und Richtlinien, an die sich beim Design neuer Merkmale gehalten werden soll, notwendig. Diese Vorgaben werden im Paket `Familie` definiert.

Die folgende Tabelle fasst die Abbildung der Merkmale auf die jeweiligen Pakete zusammen:

Merkmal	**Paket**
Grundfunktionen	`Grundfunktionen`
Electronic Program Guide	`EPG`
Datenabgleich via Telefon	`EPG`
Datenabgleich via Infrarot	`EPG`
Reminder	`Reminder`
Programmieren via Telefon	`RecProg`
Programmieren via Infrarot	`RecProg`
Titeldatenbank (eigene Verbindung)	`TDB`
Titeldatenbank (Verbindung über VDR)	`TDB`
Ansteuerung anderer Geräte	`AndereGeräte`
Programmiert	`AndereGeräte`
Download von Codes via Telefon	`Download`
Download von Codes via Internet (eigene Verbindung)	`Download`
Download von Codes via Internet (Verbindung über VDR)	`Download`
Lernen (Einfach)	`Lernen`
Lernen (Intelligent)	`Lernen`
Tastatureditor	`Tastatureditor`
Mehrnutzerverwaltung	`Mehrnutzer`

Ein Teil der Funktionen macht es erforderlich, dass einige Pakete sowohl auf der Fernbedienung als auch auf dem *VDR-System* präsent sind.

Es entsteht auf diese Weise ein System, welches den in *Abb. 4-1* dargestellten prinzipiellen Aufbau besitzt. Aus dieser Illustration sind ebenfalls die Zusammenhänge und Nutzungsbeziehungen zwischen den einzelnen Pakete zu erkennen.

Wie aus der Darstellung ersichtlich ist, existieren zwei verschiedene Arten von Beziehungen zwischen den Paketen. Zum einen entstehen Datenflüsse, wie sie beispielsweise zwischen dem Paket `Kommunikation` und den Merkmalpaketen bestehen, zum anderen gibt es auch strukturelle Abhängigkeiten der Pakete untereinander. Wie sie ent-

stehen und welche Funktion sie im Speziellen haben, soll in den folgenden Abschnitten erläutert werden.

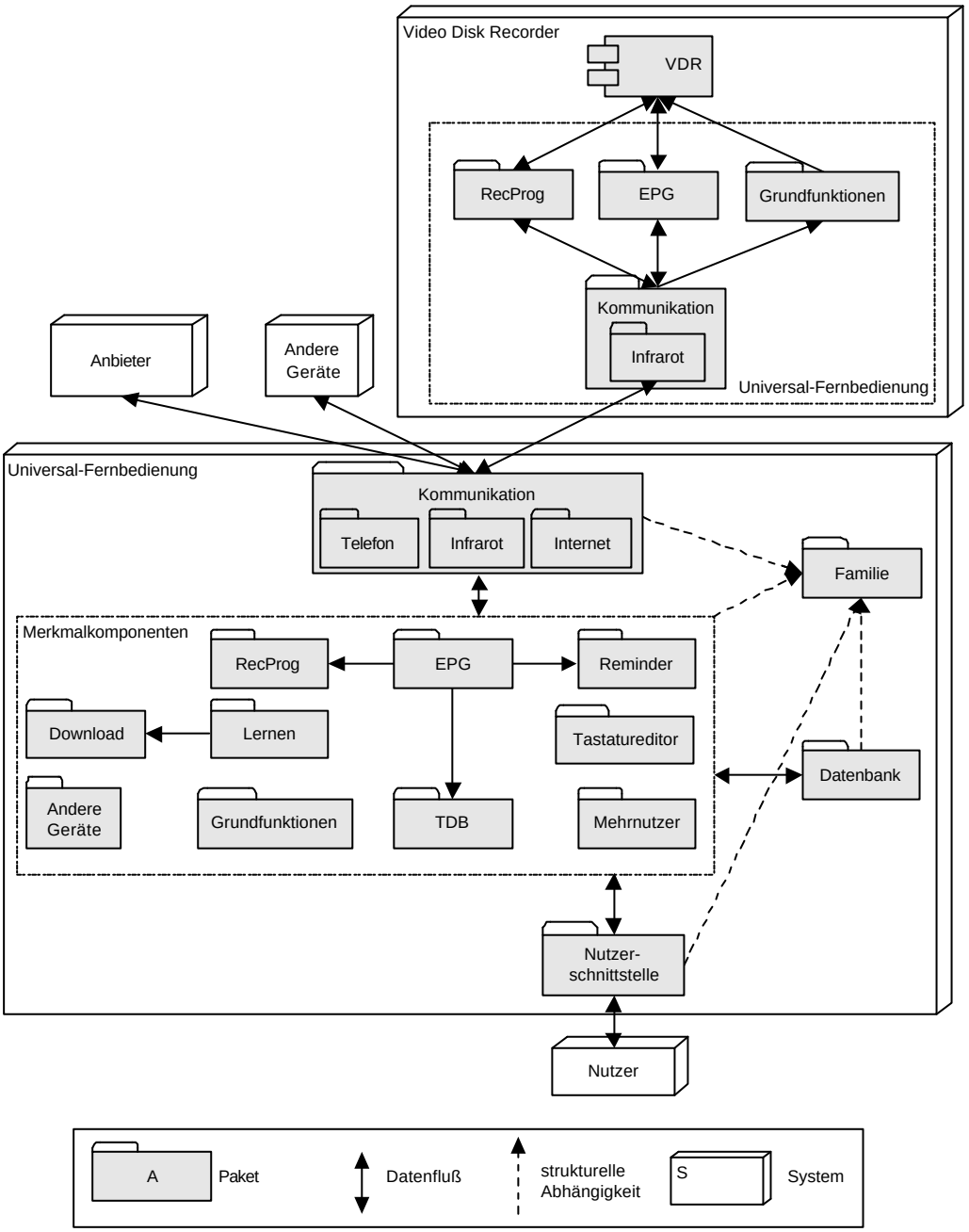

Abb. 4-1: Prinzipielle Architektur des Systems

4.2 Funktionale Architektur

Nachdem im vorhergehenden Abschnitt festgehalten wurde, welche Pakete das System beinhalten soll und in welcher Form die einzelnen Merkmale durch sie modelliert

werden, beschäftigt sich der vorliegende Teil der Arbeit mit der Frage, welche Datenflüsse zwischen den einzelnen Paketen auftreten können.

Um einen besseren Überblick zu gewährleisten, werden die Pakete, welche unmittelbar Merkmale repräsentieren, jeweils separat betrachtet und die Datenflüsse zwischen ihnen und anderen Paketen dargestellt.

Die untenstehende Legende soll das Verständnis der nachfolgenden Abbildungen etwas erleichtern:

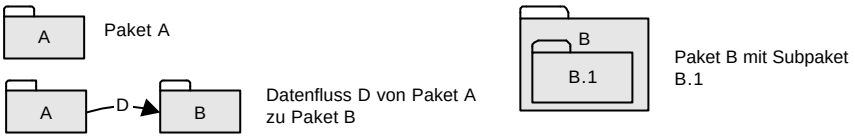

Abb. 4-2: Legende Funktionale Architektur

4.2.1 Funktionale Architektur des Paketes `Grundfunktionen`

Dieses Paket modelliert das Merkmal **GRUNDFUNKTIONEN**. *Abb. 4-3* zeigt die Datenflüsse, die mit `Grundfunktionen` im Zusammenhang stehen. Es ist zu erkennen, dass sich das Paket in zwei Teile gliedert.

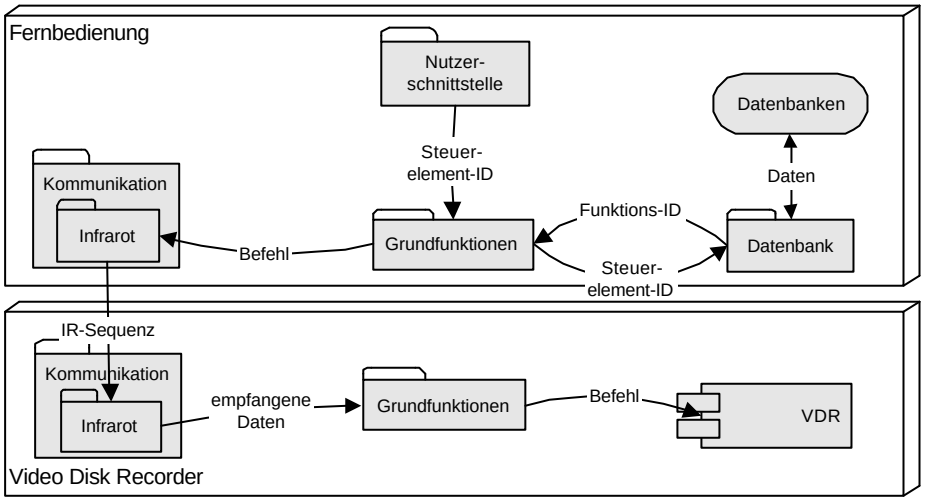

Abb. 4-3: Paket `Grundfunktionen` (Funktionale Architektur)

Das erste Teilpaket befindet sich auf der Fernbedienung und ist dafür zuständig, auf Basis der ermittelten *Funktions-ID* die entsprechende Infrarotsequenz zusammenzustellen. Ebenfalls sorgt `Grundfunktionen` dafür, dass diese Sequenz mit Hilfe des Paketes `Kommunikation` an das *VDR-System* gesendet wird. Auf der Seite des *Video Disk Recorders* wird diese empfangen und `Grundfunktionen` übergibt an die Software des *VDR* den korrekten Befehl.

Auf Grund dieser Verfahrensweise ist es notwendig, dass auch `Kommunikation` in zwei Teilpakete zerlegt wird. Ebenfalls erkennbar ist, dass dieses Paket eine Subsystem

`Infrarot` enthält. Im weiteren Verlauf dieses Abschnittes werden dem Paket noch weitere Subsysteme hinzugefügt.

Das Paket `Nutzerschnittstelle` ist, wie der Name schon impliziert, zuständig für die Interaktion mit dem Nutzer. Es wird also verschiedene Steuerelemente (Tasten, Menüs etc.) verwalten und vom Nutzer angewählte Elemente ermitteln.

Aus Gründen einer möglichst hohen Flexibilität und im Hinblick auf die zu realisierenden Fähigkeiten des Merkmals **NUTZERPROFIL** wird darauf verzichtet, den einzelnen Steuerelementen statisch eine Funktion zuzuordnen.

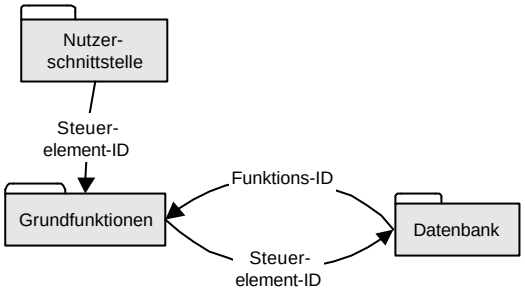

Abb. 4-4: Funktion ermitteln (Funktionale Architektur)

Vielmehr wird diese Zuordnung in einer Datenbank festgehalten, so dass jederzeit einem Steuerelement eine neue Funktion zugewiesen werden kann. Die einzelnen Pakete haben dann die Möglichkeit, auf Basis einer *Tasten-ID* die entsprechende Funktion zu ermitteln (↗*Abb. 4-4*).

4.2.2 Funktionale Architektur des Paketes `EPG`

`EPG` ist das erste von drei Paketen, welche das Merkmal **ELECTRONIC PROGRAM GUIDE** sowie dessen Submerkmale repräsentieren. Das Paket ist zuständig für das Anfordern und Empfangen der *EPG-Daten* sowie deren Darstellung. *Abb. 4-5* zeigt die Daten- und Kontrollflüsse, welche mit diesem Paket in Zusammenhang stehen.

Äquivalent zu `Grundfunktionen` gibt es auch bei dem Paket `EPG` zwei Teilpakete, welche auf dem *VDR-System* bzw. der Fernbedienung präsent sind.

`EPG` ist in der Lage, eine Datenanforderung zu erstellen, welche unter Verwendung des Paketes `Kommunikation` an das *VDR-System* gesendet werden kann. Auf gleichem Wege gelangen die entsprechenden *EPG-Daten* auf die Fernbedienung. Hier werden sie in einer Datenbank gespeichert und an `Nutzerschnittstelle` zur Darstellung übergeben. Weiterhin wird von `EPG` das Speichern und Löschen der *EPG-Daten*, d.h. die Pflege der Datenbank, nach den Vorgaben des Nutzers übernommen.

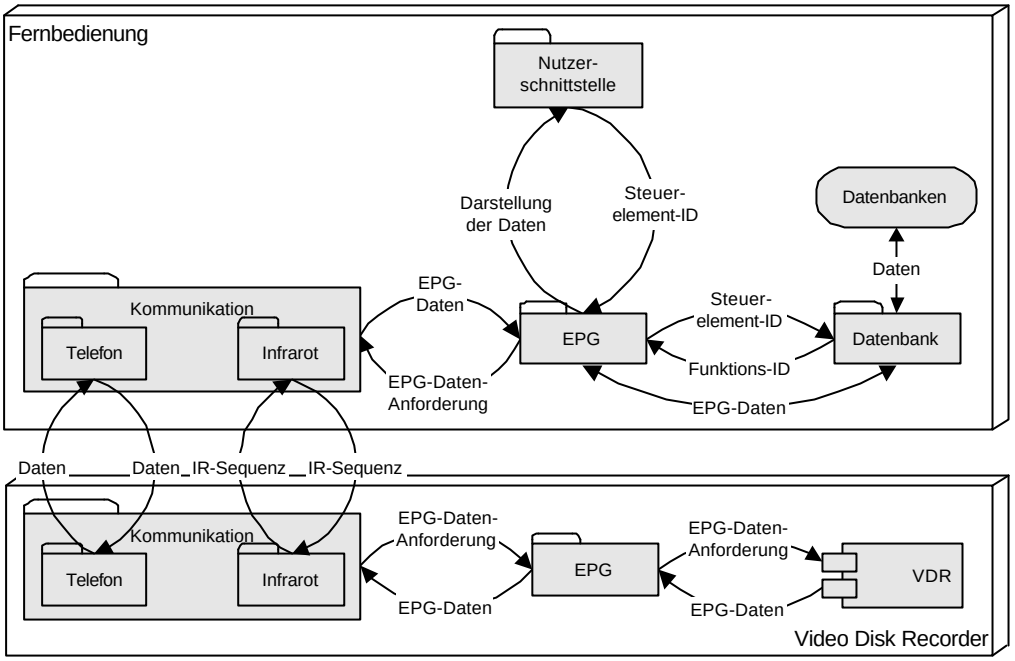

Abb. 4-5: Paket `EPG` (Funktionale Architektur)

In *Abb. 4-5* stehen die Datenflüsse `EPG-Datenanforderung` bzw. `EPG-Daten` direkt mit dem Paket `Kommunikation` in Verbindung. In der Realität ist es jedoch selbstverständlich so, dass die entsprechenden Daten von `EPG` direkt an `Telefon` oder `Infrarot` übergeben werden. Das Verfahren ist jedoch bei beiden Teilpaketen identisch. Die vereinfachte Darstellung wurde gewählt, um eine bessere Übersicht zu gewährleisten. Die nachstehende Abbildung dokumentiert diesen Zusammenhang:

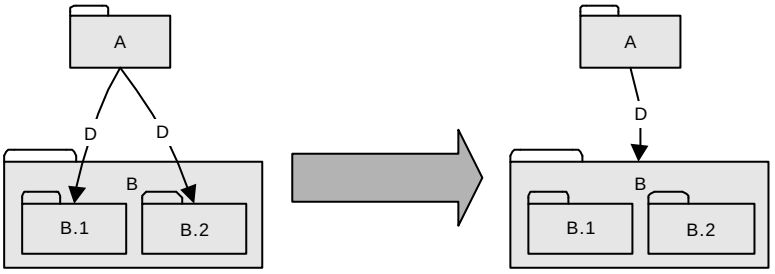

Abb. 4-6: Vereinfachung von Datenflüssen

Neben der Anforderung und Darstellung der *EPG-Daten* erfüllt das Paket noch eine weitere Funktion, wie aus den nachfolgenden Abschnitten erkenntlich wird.

4.2.3 Funktionale Architektur des Paketes `RecProg`

`RecProg` wird Funktionalitäten beinhalten, die es erlauben, den Recorder eines *VDR-Systems* zu programmieren. Mit anderen Worten, es handelt sich hierbei um die Realisierung der Merkmale **PROGRAMMIEREN VIA TELEFON** und **PROGRAMMIEREN VIA INTERNET**. Die entsprechenden Datenflüsse sind in *Abb. 4-7* illustriert.

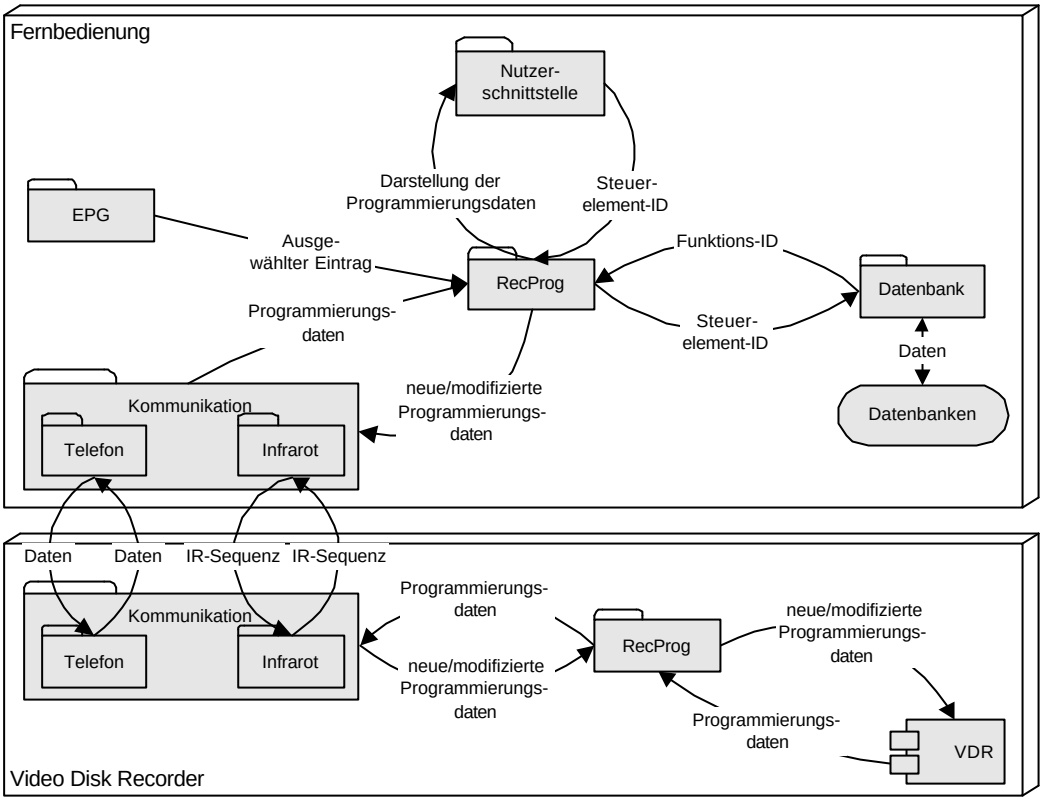

Abb. 4-7: Paket `RecProg` (Funktionale Architektur)

`RecProg` kann neue bzw. modifizierte Programmierungsdaten an das *VDR-System* senden. Dies geschieht unter Verwendung des Paketes `Kommunikation`. Das im *VDR-System* präsente Teilpaket von `RecProg` kann die empfangenen Daten auswerten und den Rekorder entsprechend programmieren. Auf eine analoge Art und Weise gelangen auch bereits existierende Programmierungen vom *VDR* zur Fernbedienung.

Die Informationen für neue Programmierungsdatensätze werden dem aktuell ausgewählten *EPG-Eintrag* entnommen, welchen das Paket `EPG` zur Verfügung stellt.

4.2.4 Funktionale Architektur des Paketes `Reminder`

Den Abschluss der Pakete, die das Merkmal **Electronic Program Guide** repräsentieren, bildet das Paket `Reminder`.

Im Gegensatz zu den anderen Paketen gibt es hier jedoch kein Teilpaket innerhalb des *VDR-Systems.*

Die benötigten *EPG-Daten* werden, wie auch schon beim Paket `RecProg`, von `EPG` zur Verfügung gestellt. `Reminder` extrahiert hieraus die erforderlichen Daten und speichert sie in einer entsprechenden Datenbank. Gleichzeitig sorgt `Reminder` dafür, dass der Anwender benachrichtigt wird, wenn einer der von ihm gesetzten Termine eintritt. Die Verwaltung und Pflege der *Reminder*-Datenbank wird ebenfalls von `Reminder` übernommen.

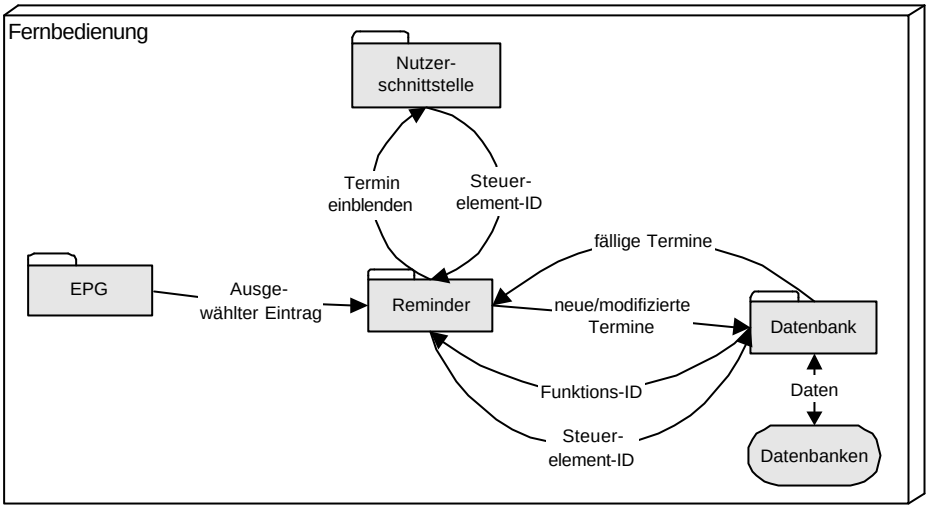

Abb. 4-8: Paket `Reminder` (Funktionale Architektur)

4.2.5 Funktionale Architektur des Paketes `TDB`

Das letzte Submerkmal von **ANSTEUERUNG DES VDR** wird ebenfalls durch ein eigenes Paket repräsentiert. Die zu `TDB` gehörigen Datenflüsse sind *in Abb. 4-9* dargestellt.

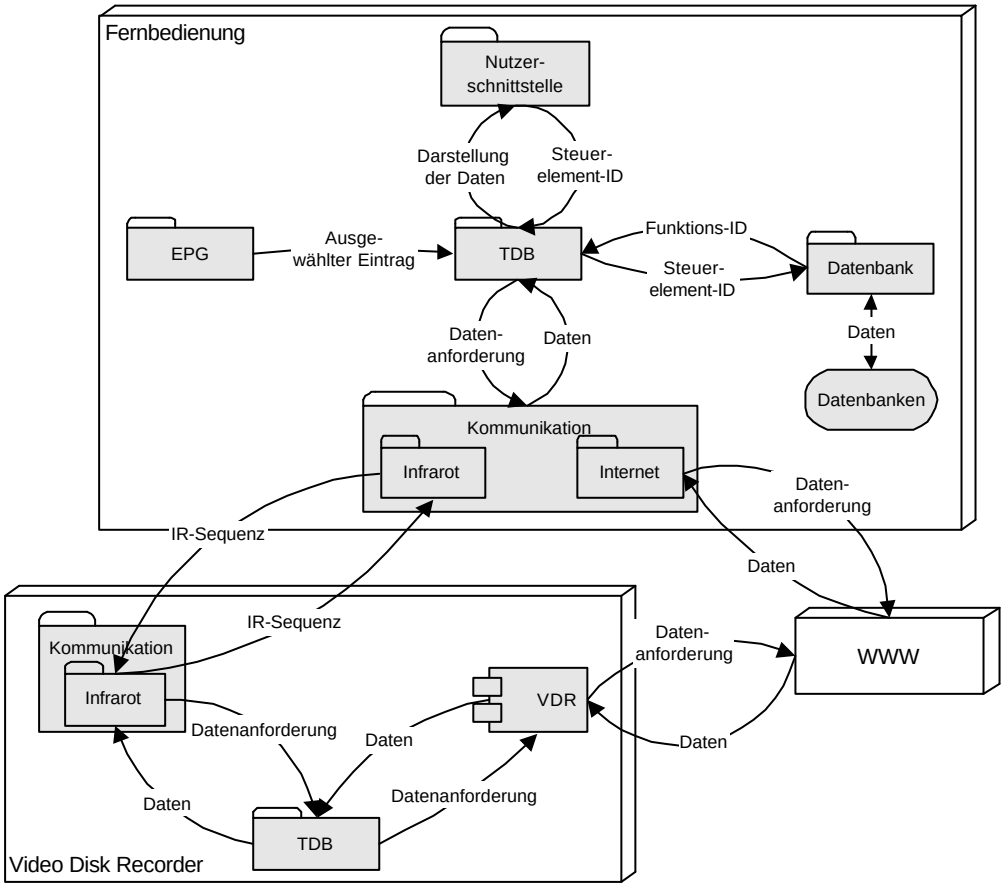

Abb. 4-9: Paket `TDB` (Funktionale Architektur)

TDB ist dafür zuständig, über eine Internetverbindung Informationen zu Sendungen, die über die Daten des *Electronic Program Guide* hinausgehen, zu ermitteln. Hierzu steht ein weiteres Subsystem des Pakets Kommunikation, Internet, zur Verfügung.

Die Datenflüsse an dieser Stelle sind ähnlich denen im Umfeld des Paketes EPG. Eine auf dem aktuellen *EPG-Eintrag* basierende und den Wünschen des Nutzers angepasste Datenanforderung wird von TDB erstellt und von Internet übertragen. Auf gleichem Wege gelangen die Informationen aus der Titel-Datenbank auf die Fernbedienung und werden dort dargestellt.

Eine Besonderheit dieses Paketes bzw. des entsprechenden Merkmals ist die Möglichkeit, neben dem Aufbau einer eigenen Internetverbindung auch die Fähigkeiten des *VDR-Systems* zu nutzen und die Internetverbindung über dieses zu erzeugen. Die Übertragung der Anforderung zum *VDR-System* und der Programminformationen auf die Fernbedienung erfolgt dann über die Teilkomponente Infrarot.

4.2.6 Funktionale Architektur des Paketes AndereGeräte

Wie bereits erwähnt, wird das Merkmal **Ansteuerung anderer Geräte** durch drei Pakete repräsentiert. Zu Beginn soll das Paket AndereGeräte näher vorgestellt werden.

Es ist zuständig für das Senden von Infrarotsignalen an die anzusteuernden Geräte. Die aus dieser Anforderung resultierenden Datenflüsse zeigt *Abb. 4-10*.

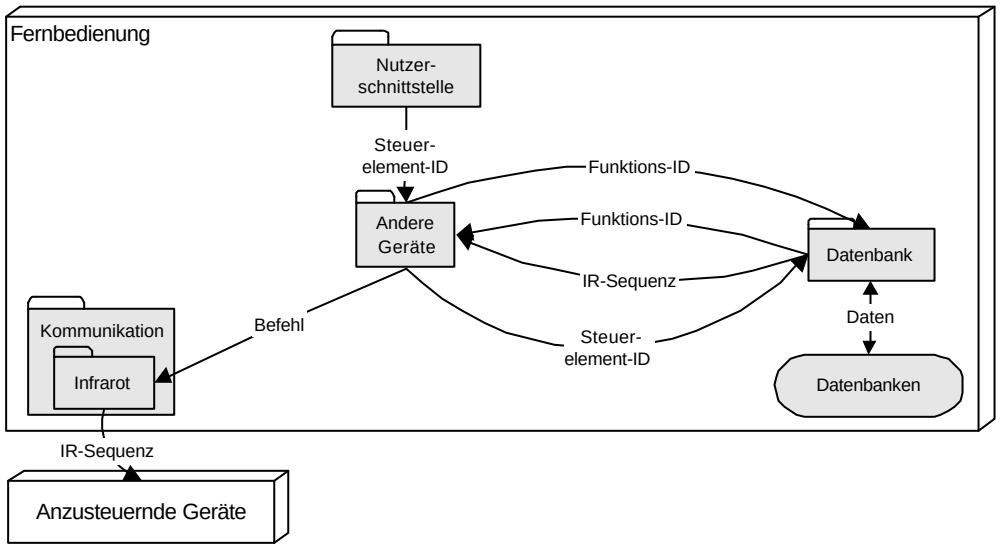

Abb. 4-10: Paket AndereGeräte (Funktionale Architektur)

Die Funktion, welche dieses Paket erfüllt, ist ähnlich der von Grundfunktionen. Im Gegensatz zu dieser ist es jedoch so, dass die zu sendenden Befehle nicht im Vorfeld bekannt sind und es daher erforderlich ist, die entsprechenden Infrarotsequenzen in einer Datenbank zu hinterlegen. Diese Notwendigkeit ergibt sich auch aus den Aufgaben-

bereichen und Funktionsweisen der im Anschluss beschriebenen Pakete `Lernen` und `Download`.

Die jeweiligen Befehle werden über das Paket `Kommunikation` versendet.

4.2.7 Funktionale Architektur des Paketes `Lernen`

Um mit Hilfe des Paketes `AndereGeräte` verschiedene Geräte ansteuern zu können, ist es erforderlich, die entsprechenden Infrarotsequenzen in Erfahrung zu bringen. Eine Methode, eine solche Geräteintegration zu realisieren, stellt das Paket `Lernen` zur Verfügung.

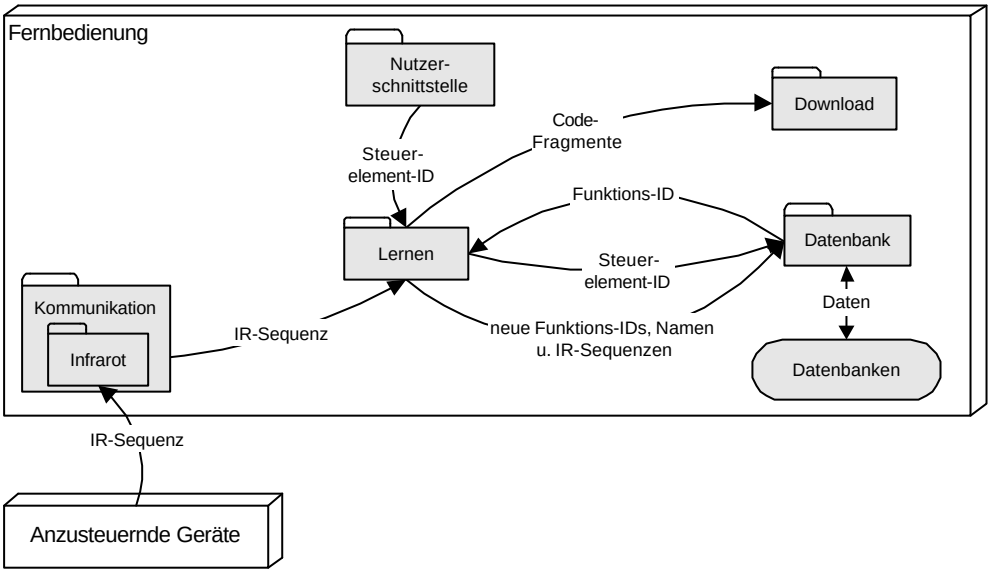

Abb. 4-11: Paket `Lernen` (Funktionale Architektur)

Die von anderen Fernbedienungen gesendeten Signale werden vom Paket `Kommunikation`, d.h. dessen Subsystem `Infrarot`, empfangen und an `Lernen` weitergeleitet. Dort wird ihnen eine *Funktions-ID* sowie ein Name zugeordnet und dieses Tupel in einer Datenbank gespeichert.

Die Zuordnung einer solchen Funktion zu einer Taste wird innerhalb des Paketes `Tastatureditor` realisiert.

Bei der Integration des Submerkmals **INTELLIGENT** kommt noch ein weiterer Datenfluss hinzu. Es ist dann möglich, nur einen Teil der Infrarotsequenzen auf dem beschriebenen Weg zu ermitteln, der Rest des *Code-Satzes* wird dann über das Paket `Download` in Erfahrung gebracht. Bei dieser Methode erfolgt die Sicherung der *Code-Sätze* in der Datenbank über das Paket `Download`.

4.2.8 Funktionale Architektur des Paketes `Download`

Das abschließende Element im Rahmen des Merkmals **ANSTEUERUNG ANDERER GERÄTE** bildet das Paket `Download`. In ihren Zuständigkeitsbereich fällt die Integration von Geräten durch *Code-Sätze*, welche beispielsweise von Herstellern anderer Geräte zur Verfügung gestellt werden. Welche Datenflüsse hierbei auftreten können, zeigt *Abb. 4-12*.

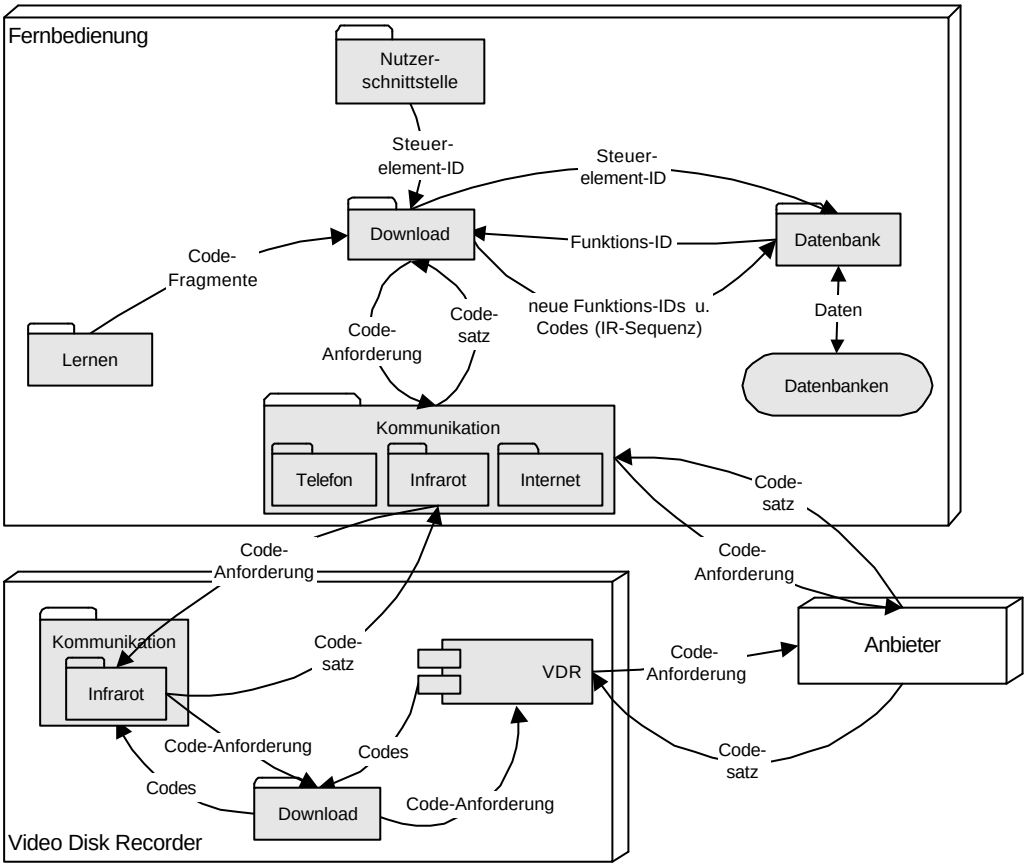

Abb. 4-12: Paket `Download` (Funktionale Architektur)

`Download` sendet mit Hilfe des Paketes `Kommunikation` eine Anforderung und empfängt dann die entsprechenden Codes. Diesen wird eine *Funktions-ID* zugewiesen, welche gemeinsam mit der jeweiligen Infrarotsequenz in einer Datenbank gespeichert wird.

Die erwähnte Code-Anforderung kann auch auf Basis von *Code-Fragmenten* erfolgen, welche vom Paket `Lernen` (↗*4.2.7*) zur Verfügung gestellt werden.

Gemäß den Anforderungen können verschiedene Medien zur Übertragung der Code-Anforderungen bzw. der *Code-Sätze* genutzt werden. Beim Download via Internet steht, äquivalent zum Paket `TDB`, auch der Umweg über den *Video Disk Recorder* zur Verfügung. Auch hier erfolgt die Übertragung der Daten zwischen Fernbedienung und *Video Disk Recorder* über die Infrarotschnittstelle der jeweiligen Geräte.

4.2.9 Funktionale Architektur des Paketes `Tastatureditor`

Um dem Nutzer die Möglichkeit zu geben, die Fernbedienung seinen Wünschen und Vorstellungen individuell anpassen zu können, wurde das Merkmal **TASTATUREDITOR** eingeführt. Das Paket, in welcher die entsprechenden Funktionalitäten realisiert werden, ist `Tastatureditor`.

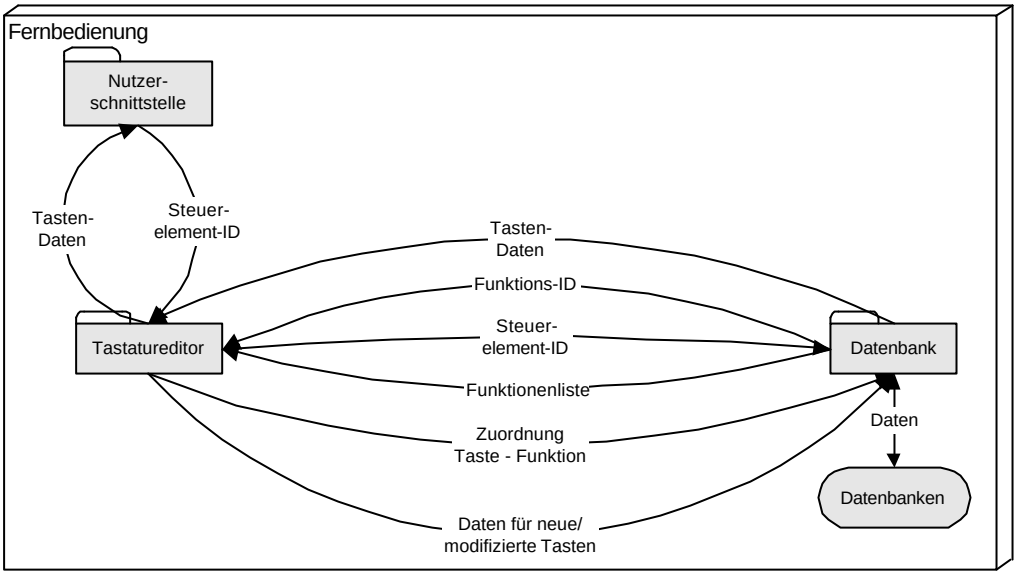

Abb. 4-13: Paket `Tastatureditor` (Funktionale Architektur)

Mit `Tastatureditor` wird dem Nutzer die Möglichkeit gegeben, eigene Schaltflächen zu erstellen, d.h. ihre Größe und Position sowie ihre Beschriftung zu bestimmen. Darüber hinaus wird dem Anwender eine Liste aller ausführbaren Funktionen zur Verfügung gestellt, die er dann einer Taste seiner Wahl zuordnen kann. Es ist natürlich nicht nur möglich, neue Schaltflächen zu erstellen, sondern auch bestehende zu modifizieren oder zu löschen. Die jeweils zu den Schaltflächen gehörenden Daten werden in den entsprechenden Datenbanken gespeichert.

4.2.10 Funktionale Architektur des Paketes `Mehrnutzer`

`Mehrnutzer` ist ein Paket, welches auch Einfluss auf andere Pakete nimmt. Besonders im Bereich der jeweiligen Datenbanken kommt es zu Modifikationen, wenn das Merkmal **MEHRNUTZERVERWALTUNG**, das dieses Paket repräsentiert, ausgewählt wird. Auf Art und Umfang dieser Änderungen soll an anderer Stelle eingegangen werden.

In *Abb. 4-14* werden die Datenflüsse, an denen `Mehrnutzer` beteiligt ist, dargestellt.

Der Anwender hat die Möglichkeit, beliebig viele Nutzer anzulegen, aus denen bei jedem Start der Software einer ausgewählt werden muss. Alle Änderungen, die während des Betriebs vorgenommen werden, z.B. mit Hilfe des Paketes `Tastatureditor`, wir-

ken sich ausschließlich auf den aktuellen Nutzer aus. Die Daten der einzelnen Nutzer können selbstverständlich auch modifiziert oder gelöscht werden.

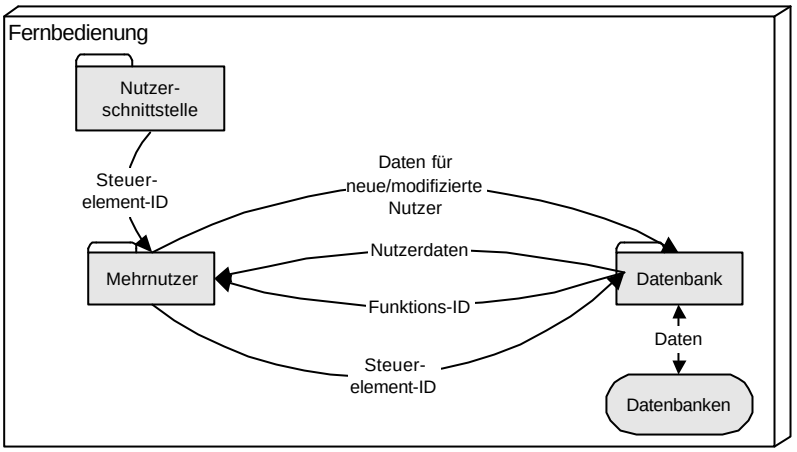

Abb. 4-14: Paket `Mehrnutzer` (Funktionale Architektur)

4.3 Prozessarchitektur

Ausgehend von den im vorhergehenden Abschnitt dargestellten Datenflüssen soll nun geklärt werden, wie diese zustande kommen. Aus diesen Erkenntnissen lassen sich dann in einem weiteren Schritt Klassen sowie deren Methoden entwickeln, so dass die hier beschriebenen Aktivitäten realisiert werden können. Auch hier soll zunächst eine Legende die verwendeten Zeichnungselemente näher erläutern:

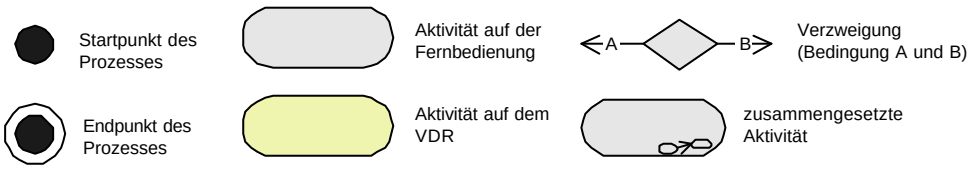

Abb. 4-15: Legende Prozessarchitektur

Zugunsten einer höheren Übersichtlichkeit werden die Aktivitäten vereinfacht dargestellt. Zusammengesetzte Aktivitäten sind wie in der Legende ersichtlich gekennzeichnet und werden im Anhang näher definiert.

4.3.1 Aktivitäten des Paketes `Grundfunktionen`

`Grundfunktionen` weist eine im Vergleich zu anderen Paketen verhältnismäßig geringe Komplexität auf. Sie ist zwar in der Lage, eine Vielzahl von Funktionen des *VDR-Systems* auszulösen, das Schema, nach dem hierbei verfahren wird, ist jedoch immer identisch. *Abb. 4-16* zeigt dieses Schema.

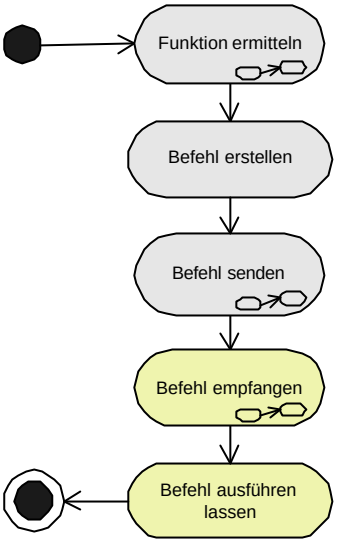

Abb. 4-16: Auslösen der Grundfunktionen (Prozessarchitektur)

Zunächst muss auf Basis der gedrückten Taste die auszuführende Funktion sowie deren eventuelle Parameter ermittelt werden. In einem weiteren Schritt wird der entsprechende Befehl, d.h. die Infrarotsequenz, zusammengestellt. Anschließend wird diese Sequenz gesendet, welche dann im *VDR-System* empfangen werden kann. Das hier residierende Teilpaket von `Grundfunktionen` führt dann den gewünschten Befehl aus bzw. gibt ihn an die Software des *VDR-Systems* weiter.

4.3.2 Aktivitäten des Paketes `EPG`

Das Paket ist zuständig für das Anfordern, Empfangen und Darstellen der *EPG-Daten* auf der Fernbedienung. Eine Illustration der Aktivitäten erfolgt in *Abb. 4-17*.

Es lassen sich drei verschiedene Teilprozesse erkennen. Zum einen können die gespeicherten *EPG-Daten* auf dem Display dargestellt werden. Hierzu müssen sie aus der Datenbank ausgelesen werden und können dann unter Nutzung des Paketes `Nutzerschnittstelle` eingeblendet werden.

Der zweite Teilprozess beschäftigt sich mit dem Datenabgleich. Vom Paket `EPG` wird eine Datenanforderung erstellt, welche dann über das Paket `Kommunikation` an den *VDR* gesendet wird. Dort werden die *EPG-Daten* übernommen und an die Fernbedienung gesendet. Nachdem sie hier empfangen wurden, werden sie sinnvollerweise auch eingeblendet.

Der *EPG-Daten*-Bestand muss natürlich auch gewartet werden, d.h. es muss die Aktualität der Daten überprüft werden. Obsolete Datensätze sind gegebenenfalls zu löschen. Falls der Anwender dies wünscht, kann in diesem Zusammenhang auch ein neuer Datenabgleich erfolgen.

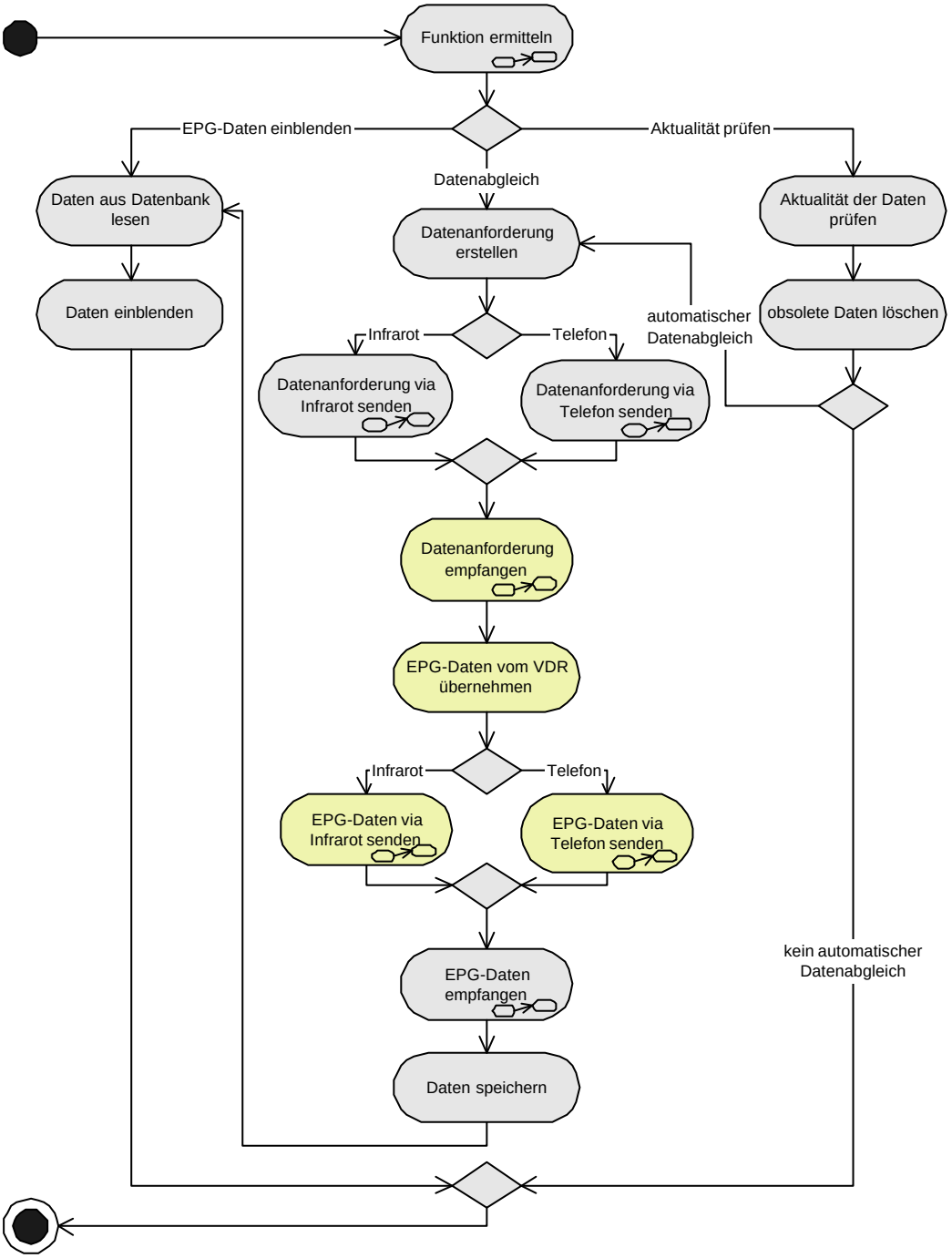

Abb. 4-17: Verwaltung der *EPG-Daten* (Prozessarchitektur)

4.3.3 Aktivitäten des Paketes `RecProg`

Dieses Paket umfasst im Wesentlichen zwei große Prozesse:

- Erzeugen von neuen Programmierungen aus gegebenen *EPG-Daten*
- Modifizieren von vorhandenen Programmierungen.

Die Teilprozesse, in die sich der erstgenannte Prozess gliedert, sind in *Abb. 4-18* dargestellt.

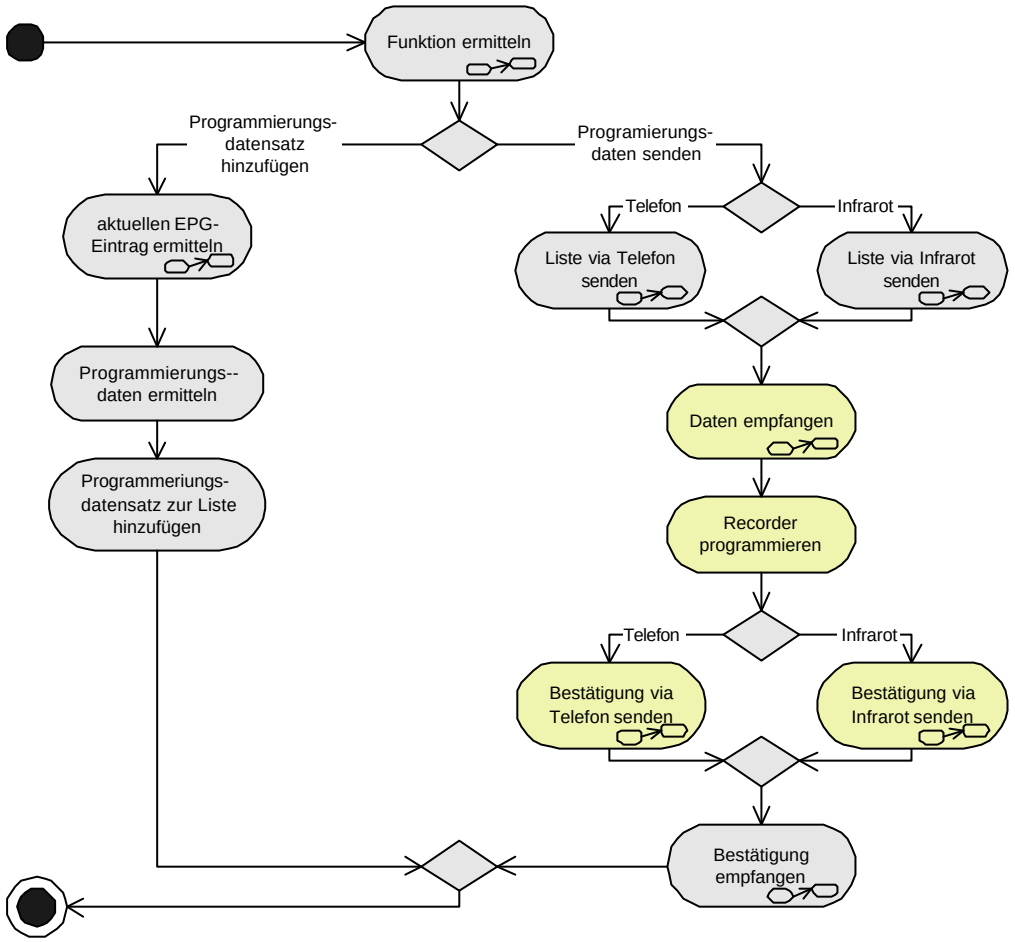

Abb. 4-18: Erzeugen von neuen Programmierungsdaten (Prozessarchitektur)

Wie zu erkennen ist, können im Rahmen dieses Prozesses zwei unterschiedliche Funktionen ausgelöst werden.

Zum einen kann ein neuer Datensatz der Programmierungsdaten-Liste hinzugefügt werden. Hierzu ist es zunächst einmal notwendig, den aktuell ausgewählten *EPG-Eintrag* zu ermitteln, auf dessen Basis die neue Programmierung entstehen soll. Diese Information wird, wie auch schon aus *Abb. 4-5* erkenntlich, vom Paket `EPG` zur Verfügung gestellt. Aus dem Eintrag werden die für die Programmierung benötigten Daten extrahiert, und ein neuer Programmierungsdatensatz wird erstellt und der Liste hinzugefügt.

Der zweite Teilprozess beschäftigt sich mit dem Senden der Programmierungsdaten-Liste und der Programmierung der Recorderfunktion des *VDR-Systems.* Vor dem Versenden ist es erforderlich, die Art und Weise der Übertragung festzulegen. Hier besteht die Wahl zwischen einer Infrarot- und einer Telefonübermittlung, sofern die entsprechenden Merkmale aktiviert sind. Die Programmierungsdaten werden entsprechend gesendet und auf dem *Video Disk Recorder* empfangen und extrahiert. Nach erfolgter

Programmierung der Recorderfunktion wird noch eine Bestätigung (oder Fehlermeldung) generiert und an die Fernbedienung gesendet. Dort kann sie bei Bedarf dem Nutzer sichtbar gemacht werden.

Das Modifizieren vorhandener Programmierungsdaten erfolgt auf eine ähnliche Art und Weise, jedoch ist es hier naturgemäß zunächst einmal erforderlich, die bereits bestehenden Daten anzufordern. Wie dies geschieht und welche Aktionen darüber hinaus noch möglich sind, zeigt *Abb. 4-19*.

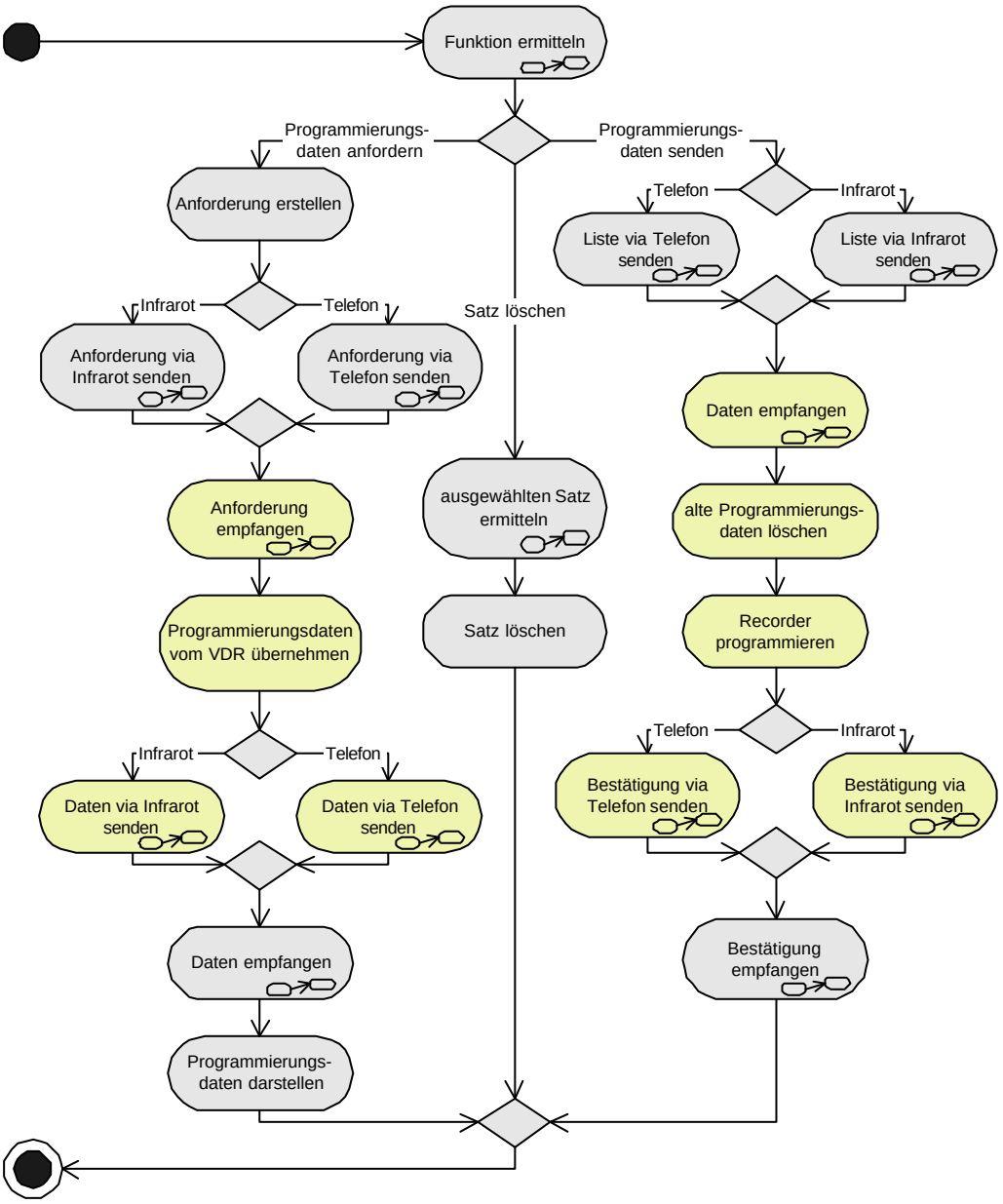

Abb. 4-19: Ändern von Programmierungsdaten (Prozessarchitektur)

Wie zu erkennen ist, erfolgt die Anforderung der Programmierungsdaten äquivalent zur Anforderung der *EPG-Daten* im Abschnitt *4.3.2*.

Aus den Daten, welche nun in einer Liste dargestellt werden, kann der Nutzer einzelne Einträge auswählen und die zugehörigen Programmierungsdatensätze löschen. Wenn

die Modifikationen abgeschlossen sind, werden die veränderten Programmierungsdaten an das *VDR-System* gesendet und der Recorder kann wieder programmiert werden. Allerdings müssen an dieser Stelle zuvor die alten Programmierungen gelöscht werden, um eventuelle Inkonsistenzen zu vermeiden.

4.3.4 Aktivitäten des Paketes `Reminder`

Mit Hilfe von `Reminder` ist dem Nutzer die Möglichkeit gegeben, sich an bestimmte Sendungen erinnern zu lassen. Das Paket besitzt Fähigkeiten, diese Gedächtnisstützen zu verwalten. Im Wesentlichen lassen sich die hier stattfindenden Prozesse in drei Bereiche gliedern:

- Erzeugen neuer *Reminder*
- Modifizieren bestehender *Reminder*
- Einblenden fälliger Termine.

Illustriert werden diese Aktivitäten in *Abb. 4-20*.

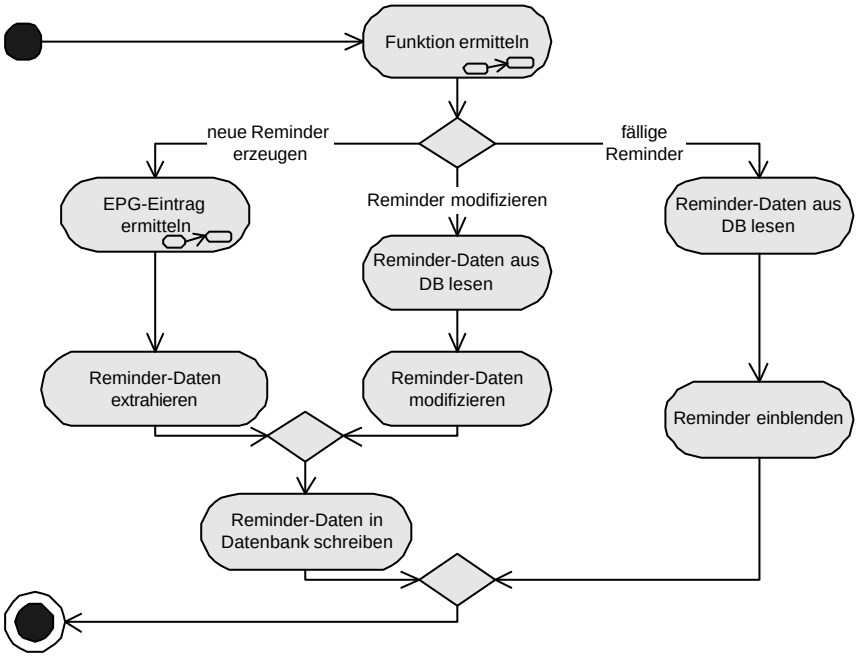

Abb. 4-20: Verwalten von *Remindern* (Prozessarchitektur)

Ein neuer *Reminder* wird erzeugt, indem aus der Liste der *EPG-Daten* ein Eintrag ausgewählt wird. Aus diesem werden dann die für den *Reminder* relevanten Informationen (Startzeit und –datum, Sender, Titel der Sendung) extrahiert, und der daraus entstehende Datensatz wird in die Datenbank geschrieben.

Ist ein *Reminder* fällig, werden die entsprechenden Informationen aus der Datenbank gelesen und eingeblendet. Dies kann durchaus auch ausgeweitet werden. So ist es vorstellbar, die Einblendung mit der Auslösung eines akustischen Signals zu koppeln.

Ebenfalls möglich ist die Modifikation bestehender *Reminder*. Hier können z.B. einzelne Einträge gelöscht werden oder es kann die Art und Weise der Erinnerung geändert werden. Notwendig hierfür ist es, zunächst den jeweiligen Datensatz aus der Datenbank zu lesen und nach erfolgter Modifikation wieder in die Datenbank zu schreiben bzw. aus der Datenbank zu löschen.

4.3.5 Aktivitäten des Paketes `AndereGeräte`

Bei `AndereGeräte` handelt es sich um ein Paket, das in Struktur und Funktionsumfang mit `Grundfunktionen` vergleichbar ist. Auch dieses Paket ist ausschließlich dafür zuständig, einfache Steuerbefehle via Infrarotschnittstelle zu versenden. Im Gegensatz zu `Grundfunktionen` ist hier jedoch kein Teilpaket im *VDR-System* notwendig.

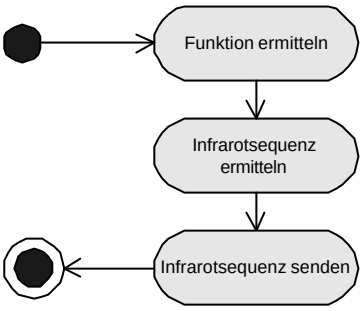

Abb. 4-21: Ansteuern anderer Geräte (Prozessarchitektur)

Nachdem die auszuführende Funktion ermittelt wurde, wird die zur *Funktions-ID* gehörende Infrarotsequenz aus der Datenbank gelesen. Unter Nutzung des Paketes `Kommunikation` wird abschließend diese Sequenz gesendet.

4.3.6 Aktivitäten des Paketes `Lernen`

Auch diese Komponente zeichnet sich nicht durch eine übermäßige Komplexität der hier realisierten Prozesse aus.

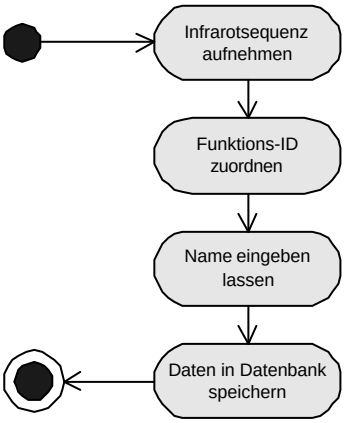

Abb. 4-22: Lernen von Infrarotsequenzen (Prozessarchitektur)

Wenn eine Sequenz von der Infrarotschnittstelle aufgenommen wird, ordnet `Lernen` dieser eine *Funktions-ID* zu. Anschließend erhält die Funktion noch einen Namen, welcher vom Nutzer zu vergeben ist. Dieses Tupel, bestehend aus Infrarotsequenz, *Funktions-ID* und Funktionsname, wird zum Schluss in der entsprechenden Datenbank gespeichert.

Der hier dargestellte Prozess gilt nur für die Aufnahme einer Funktion der jeweiligen Fernbedienung.

4.3.7 Aktivitäten des Paketes `Download`

Eine weitere Möglichkeit, neue Geräte zu integrieren, besteht im Download von *Code-Sätzen*, welche von verschiedenen Anbietern zur Verfügung gestellt werden. Der Ablauf, wie er sich bei der Integration eines neuen Gerätes gestaltet, ist in *Abb. 4-23* dargestellt.

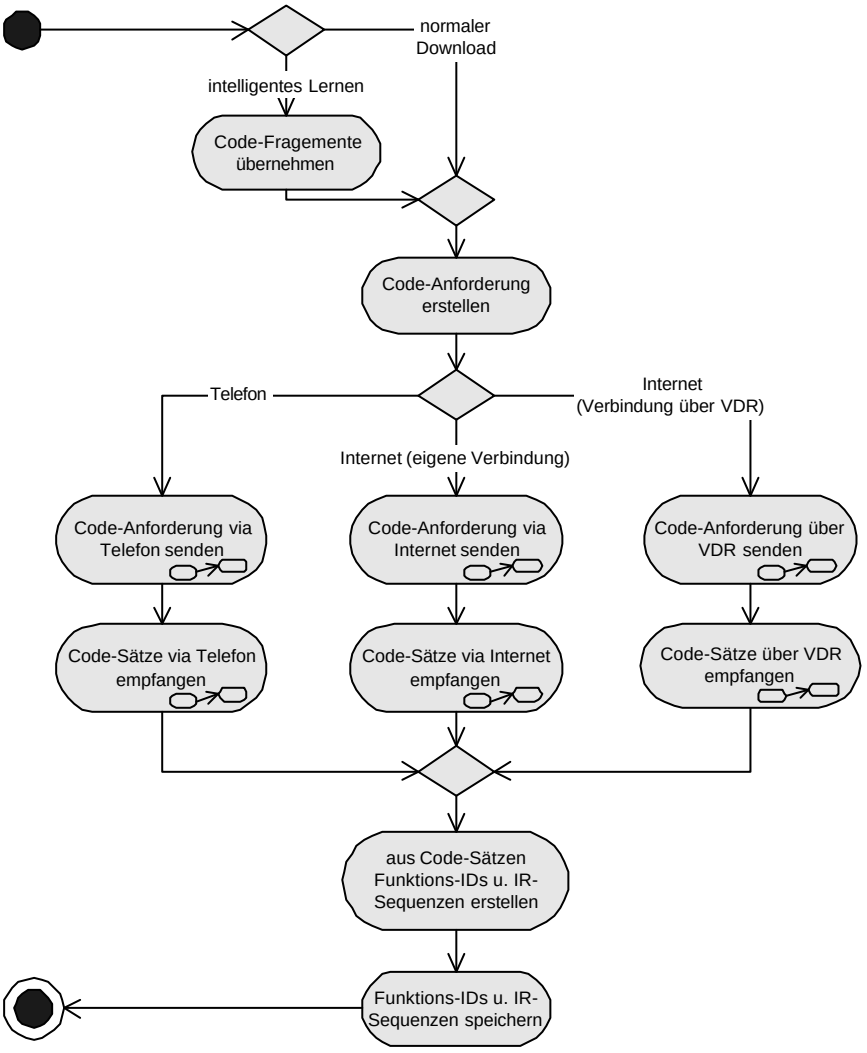

Abb. 4-23: Download von Code-Sätzen

Um neue *Code-Sätze* empfangen zu können, ist es zunächst notwendig, eine entsprechende Anforderung zu erstellen. Dies kann, falls das Submerkmal INTELLIGENT des Merkmals LERNEN aktiviert ist, auch mit Hilfe von *Code-Fragmenten*, welche vom Paket `Lernen` übernommen werden, geschehen.

Anschließend wird die erstellte Anforderung versendet. Wie im Merkmalmodell schon zu erkennen ist, existieren hier verschiedene Varianten der Datenübertragung: Zum einen kann die Anforderung via Telefon übermittelt werden, zum anderen kann die Übertragung auch via Internet erfolgen. Bei dieser zweiten Variante besteht wiederum die Wahl zwischen dem Aufbau einer eigenen Verbindung und der Nutzung der Fähigkeiten des *VDR-Systems.*

Die angeforderten *Code-Sätze* werden anschließend auf identische Weise wie die Anforderung übermittelt und empfangen.

Aus den *Code-Sätzen* werden die jeweiligen Infrarotsequenzen extrahiert. Diesen wird dann eine *Funktions-ID* zugeordnet, und das so entstandene Tupel wird in der entsprechenden Datenbank gespeichert.

4.3.8 Aktivitäten des Paketes `Tastatureditor`

`Tastatureditor` dient dazu, dem Nutzer die Möglichkeit zu geben, die einzelnen Schaltflächen seinen Wünschen und Anforderungen anzupassen.

Das Paket besitzt Fähigkeiten, die es erlauben, neue Steuerelemente und Seiten zu erstellen und bereits existierende zu modifizieren oder zu löschen. Die Aktivitäten, die hierbei auftreten können, sind in *Abb. 4-24* dargestellt.

Beim Erzeugen und Löschen von Elementen besteht die Wahl zwischen dem Generieren und Entfernen eines Steuerelementes oder einer Seite.

Wird ein Steuerelement erzeugt, müssen zunächst Position, Größe und Beschriftung des Elementes ermittelt werden. Im Anschluss wird dem Steuerelement eine Funktion und damit ein zuständiges Merkmal zugeordnet. Weiterhin muss die Seite, auf der das Element platziert werden soll, bestimmt werden. Diese Daten werden in den entsprechenden Datenbanken gespeichert und die grafische Ausprägung des Steuerelementes wird erzeugt.

Das Erzeugen einer Seite erfolgt auf eine ähnliche Weise. Auch hier werden zunächst die Daten (Position, Größe, Name) der Seite ermittelt und in der entsprechenden Datenbank gespeichert. Anschließend wird die Seite, d.h. ihre grafische Ausprägung, erzeugt. Im Gegensatz zum Erstellen eines Steuerelementes muss hier jedoch noch ein Menüeintrag generiert werden, so dass die Seite später auch aufgerufen werden kann.

Existierende Steuerelemente können selbstverständlich auch wieder entfernt werden. Beim Löschen eines Steuerelementes werden die zugehörigen Daten aus den Datenbanken entfernt, und die grafische Ausprägung des Elementes wird gelöscht.

Das Entfernen von Seiten gestaltet sich geringfügig aufwändiger. Hier müssen neben den Daten und der grafischen Ausprägung der Seite auch die auf der Seite platzierten Steuerelemente gelöscht werden.

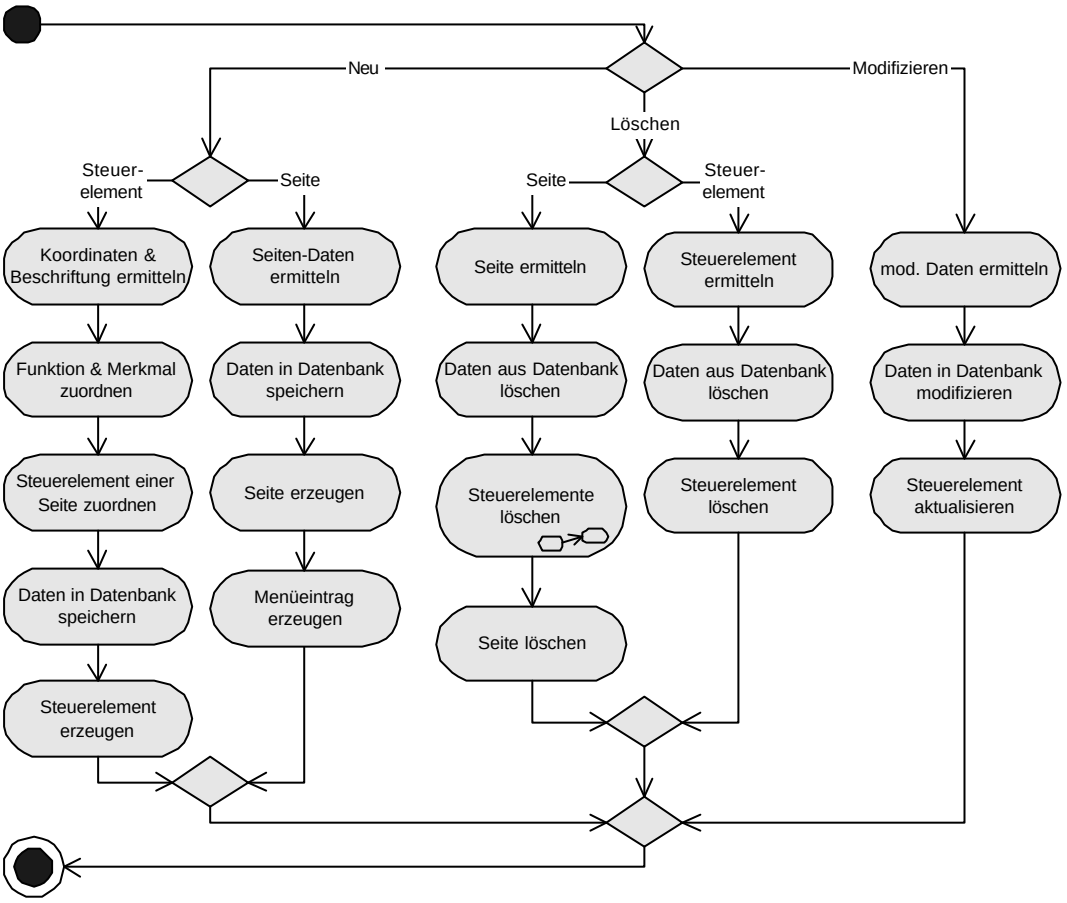

Abb. 4-24: Aktivitäten des Paketes `Tastatureditor`

4.4 Bewertung der Architektur

In diesem Kapitel wurde das in Abschnitt *3* definierte Merkmalmodell auf eine Anzahl von Paketen abgebildet. Darüber hinaus wurden die Datenflüsse zwischen den Paketen sowie die Prozesse, an denen die Pakete beteiligt sind, herausgearbeitet. Auf dieser Basis lassen sich in einem weiteren Schritt die Architektur der einzelnen Pakete und die Funktionen, die dort realisiert werden müssen, modellieren.

Ein Vorteil der hier realisierten Lösung besteht in der Trennung von grundlegenden Strukturen und Strukturen, die unmittelbar der Funktionalität eines Merkmals zugeordnet werden können. Dies wurde erreicht, indem neben den merkmalmodellierenden Paketen auch Basispakete definiert wurden, welche diese grundlegenden Funktionen und Strukturen beinhalten. Weiterhin wurde mit dieser Trennung der Funktionen gleichzei-

tig eine einheitliche Richtlinie geschaffen, die bei der Entwicklung der Architektur der einzelnen Pakete beachtet werden muss.

Bei der Modellierung der Architektur wurde darauf geachtet, die Konzepte, die bei der Entwicklung einer Systemfamilie zum Tragen kommen, umzusetzen. So wurde durch die direkte Abbildung der Merkmale auf die einzelnen Pakete ein hoher Grad der Modularisierung erreicht. Aus diesem Umstand resultieren ebenfalls gute Möglichkeiten, die Pakete zu warten. Da die einzelnen Pakete unabhängig voneinander modelliert werden, ist es möglich, Veränderungen vorzunehmen, ohne dass eine Beeinflussung der Pakete untereinander stattfindet. Durch die Schaffung von Basispaketen wird die Modellierung neuer Merkmalpakete erleichtert, es ist jedoch auch hier darauf zu achten, dass die Unabhängigkeit der Pakete bestehen bleibt. Die einzelnen Pakete sind im Rahmen der Beschränkungen, die im Merkmalmodell definiert wurden, frei kombinierbar. Auf diese Weise kann eine große Vielfalt der Anwendungen erzielt werden.

Bei der Konfiguration der einzelnen Anwendungen werden zunächst die Basispakete `Familie`, `Datenbank`, `Nutzerschnittstelle` und `Kommunikation` dem zu erstellenden System hinzugefügt. Dies ist, unabhängig von den ausgewählten Merkmalen, bei allen zu konfigurierenden Anwendungen der Fall. Allerdings gibt es hier eine Einschränkung: Das Paket `Kommunikation` besteht aus drei Subpaketen. Es ist nicht erforderlich, zu jeder Anwendung alle drei Subpakete `Infrarot`, `Internet` und `Telefon` hinzuzufügen, hier können je nach Bedarf einzelne dieser Subpakete entfernt werden. Abschließend werden abhängig von den aktivierten Merkmalen die jeweiligen merkmalmodellierenden Pakete dem System hinzugefügt.

5 Prototypische Implementierung der Systemfamilie

Nachdem die Architektur des Systems entworfen ist, soll nun ein erster Prototyp entwickelt werden, der sich dieser Architektur bedient.

Zur Verwirklichung wird ein *Sony Clié Handheld* verwendet. Dieser arbeitet mit dem Betriebssystem *Palm OS®*. Aufgrund einiger Besonderheiten, die sich in der Entwicklung von Anwendungen für *Palm OS®* ergeben, kann der Entwurf von Klassen und Strukturen für die einzelnen Pakete erst an dieser Stelle erfolgen.

Um die erwähnten Besonderheiten zu verdeutlichen, soll zunächst ein kleiner Überblick über die Anwendungsentwicklung für *Palm OS®* gegeben werden. Im Anschluss daran werden für ausgewählte Pakete Klassen und Objekte definiert sowie das Verhalten derselben modelliert.

5.1 Entwicklung von Anwendungen für das *Palm OS®*

Das Erstellen von Anwendungen für einen *Palm-Handheld* unterscheidet sich vom Erstellen von Anwendungen für normale PCs. Zum einen muss aufgrund der geringen Bildschirmgröße die Nutzerschnittstelle entsprechend angepasst werden, zum anderen muss auch die vergleichsweise geringe Speicherkapazität beachtet werden.

Anwendungen für das Palm OS® sind *single-threaded* und ereignisgesteuert.

Jede Anwendung verfügt über eine Funktion `PilotMain`, die den Einstiegspunkt bildet. Beim Start wird diese Funktion vom System aufgerufen, und ihr wird ein so genannter *Launch-Code* übergeben. Dieser bestimmt die Art und Weise des Starts der Applikation. So lässt sich beispielsweise bestimmen, ob eine Applikation im Hintergrund arbeiten soll oder ob ein vollständiger Start inklusive der Einblendung der Nutzerschnittstelle durchgeführt wird.

Für gewöhnlich besteht eine *Palm OS®*-Anwendung aus drei Teilen:

- einer Start-Routine,
- dem *Event-Loop*,
- einer Stopp-Routine.

In der Start-Routine werden benötigte globale Variablen geladen und es erfolgt die Initialisierung der Applikation.

Hauptbestandteil einer Anwendung für das *Palm OS*® ist der *Event-Loop* (Ereignisschleife). Im *Event-Loop* werden die Ereignisse vom *Event-Queue* genommen und verarbeitet. Die Anwendung verbleibt in dieser Schleife, bis das Ereignis `appStopEvent` eintritt. Innerhalb des *Event-Loop* werden verschiedene Ereignisbehandlungsroutinen aufgerufen, die jeweils unterschiedliche Arten von Ereignissen behandeln. Einer dieser *Event-Handler*, `ApplicationHandleEvent`, muss von der Anwendung zur Verfügung gestellt werden. Des Weiteren wird jedem Formular ein *Event-Handler* zugeordnet, welcher ebenfalls von der Anwendung zur Verfügung zu stellen ist. Von den einzelnen Ereignisbehandlungsroutinen können weitere Ereignisse erzeugt und im *Queue* platziert werden.

Auch die dynamische Speicherverwaltung erfolgt nicht auf die herkömmliche Art und Weise. Alle Daten sind in so genannten *Chunks* gespeichert, die eine Größe von bis zu 64 KB haben können. Beim Allokieren eines solchen *Chunks* wird zwar auch ein *Pointer* auf diesen generiert, dieser wird jedoch nicht zurückgeliefert. Vielmehr wird dieser Zeiger (*Master Chunk Pointer*) in einer Tabelle gespeichert, und der Nutzer erhält nur eine Referenz auf den *Pointer*. Diese wird als *Handle* bezeichnet. Diese Verfahrensweise wurde gewählt, da es vorkommen kann, dass die *Chunks* von der Speicherverwaltung verschoben werden, wodurch die *Pointer* ungültig würden.

Daten, die dauerhaft gespeichert werden sollen, werden in Datenbanken abgelegt. Dieses Konzept bildet das Äquivalent zu einem *File-System* auf herkömmlichen PCs. In solchen normalen Dateisystemen werden die Dateien (oder Teile davon) in einen Puffer geschrieben, dort bearbeitet und anschließend wieder zurück geschrieben. Im Gegensatz dazu werden auf *Palm-Handhelds* die Daten bearbeitet ohne sie vorher zu verschieben oder zu kopieren. Die Organisation der Daten erfolgt in so genannten *Records*, welche zu Datenbanken zusammengefasst werden. Jeder dieser *Records* ist exakt einer Datenbank zugeordnet und die *Records* einer Datenbank besitzen eine konstante Größe.

5.2 Struktur & Design ausgewählter Pakete

In diesem Abschnitt werden für einige Pakete Klassen und Objekte definiert. Konzentriert werden soll sich hierbei auf die Architektur der Basispakete `Familie`, `Nutzerschnittstelle`, `Datenbank` und `Kommunikation`. Darüber hinaus wird, um die Funktionsfähigkeit der Systemarchitektur zu demonstrieren, auch die Architektur des Paketes `Grundfunktionen` entwickelt. Begonnen werden soll jedoch mit einigen Erläuterungen zur grundsätzlichen Funktionsweise des Systems, um die folgenden Entwurfsentscheidungen besser nachvollziehen zu können.

5.2.1 Funktionsprinzip des Systems

Wie bereits erwähnt, existieren zwei verschiedene Arten von Paketen:

- Merkmalpakete,
- Pakete, die grundlegende Strukturen und Funktionen beinhalten.

Zunächst wurde eine Designrichtlinie definiert, bei der im Vordergrund die Schaffung einer zentralen Klasse für jedes merkmalmodellierende Paket steht. Diese Klasse wird u.a. ein Attribut, das die eindeutige Identifizierung des Merkmals ermöglicht, sowie eine Methode, welche den Zugriff auf alle Funktionen des Merkmals erlaubt, enthalten. Diese Methode wird die Bezeichnung `doIt` tragen. Die Struktur der Merkmalpakete folgt dieser einheitlichen Designrichtlinie.

Die Steuerelemente, über die der Anwender die einzelnen Funktionen auslösen kann, erhalten neben einer identifizierenden *Steuerelement-ID* auch eine *Merkmal-ID* und eine *Funktions-ID*, über die sich das zuständige Merkmal bzw. die auszulösende Funktion ermitteln lassen. Die *Merkmal-ID* korrespondiert hierbei mit dem entsprechenden Attribut in der oben erwähnten zentralen Klasse.

Bei der Anwahl eines Steuerelementes ist es nun möglich, auf Basis der dem Steuerelement zugeordneten *Merkmal-ID* das zuständige Merkmal zu ermitteln. Im Anschluss daran kann die Methode `doIt` des jeweiligen Objektes aufgerufen werden. Im Rahmen dieser Methode kann die zur *Funktions-ID* des Steuerelementes passende Funktion ausgeführt werden.

Abb. 5-1 stellt diese Zusammenhänge noch einmal grafisch dar:

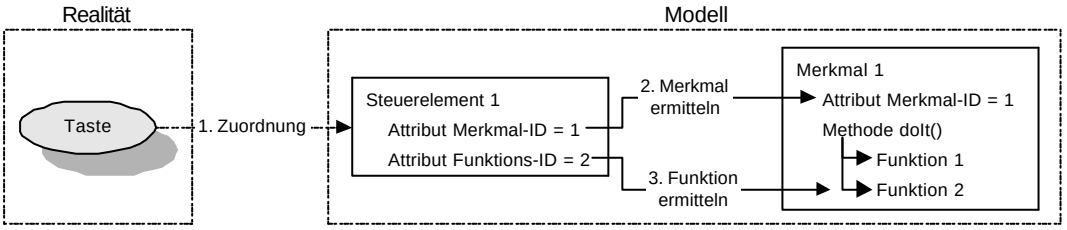

Abb. 5-1: Funktionsprinzip

Wie die hier beschriebene Idee sich in der Architektur der einzelnen Pakete widerspiegelt, soll im Folgenden erläutert werden.

5.2.2 Struktur des Paketes `Familie`

Dieses Paket bildet die Basis für die Entwicklung aller weiteren Pakete. Hier werden grundlegende Strukturen, welche von den anderen Paketen verwendet werden, definiert.

Da im Rahmen der *Palm-API*-Funktionen keine Strukturen zur Verfügung gestellt werden, die es erlauben, einfach oder doppelt verkettete Listen zu erstellen, ist es notwendig, entsprechende Klassen zu entwickeln. Dies geschieht durch die Klasse

`cListenObjekt`, welche das grundsätzliche Verhalten eines Listenelementes definiert, sowie die Klasse `cListenBasisOps` und das Template `cListe`, in denen die Fähigkeiten der Liste modelliert werden.

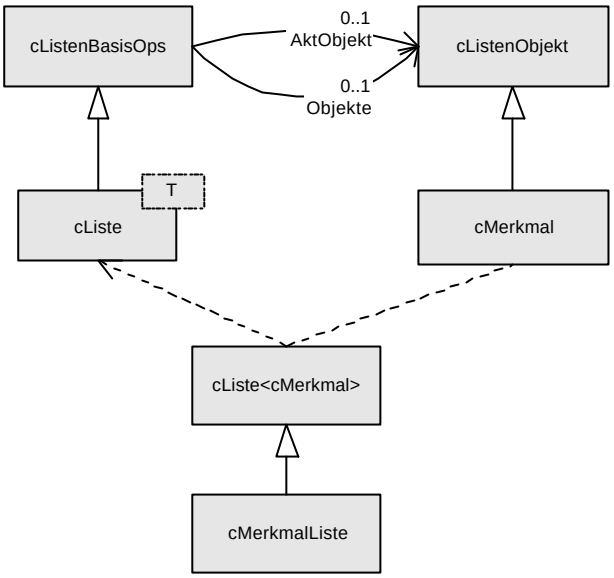

Abb. 5-2: Klassen des Paketes `Familie`

Die Klasse `cMerkmal` bildet die Basisklasse für die in *5.2.1* bereits erwähnte zentrale Klasse eines jeden merkmalmodellierenden Paketes. In ihr werden das Attribut `MID`, welches die *Merkmal-ID* beinhalten wird, sowie ein virtueller Prototyp der Methode `doIt` definiert. Die Implementierung dieser Methode erfolgt dann in den von `cMerkmal` abgeleiteten Klassen. Weiterhin definiert `cMerkmal` die Methoden `getFID` und `getFParam`, mit deren Hilfe sich die zu einer *Steuerelement-ID* gehörende Funktion, d.h. die jeweilige *Funktions-ID* sowie die jeweiligen Parameter ermitteln lassen.

Die Klasse `cMerkmalListe` schließlich ist eine Instanz von `cListe` und stellt eine Liste der implementierten Merkmale bzw. der entsprechenden Objekte zur Verfügung. Mit Hilfe dieser Lösungsvariante ist es möglich, auf alle Merkmale zuzugreifen und die Art und Anzahl der aktivierten Merkmale beliebig zu gestalten.

5.2.3 Struktur des Paketes `Nutzerschnittstelle`

`Nutzerschnittstelle` ist zuständig für die Interaktion mit dem Anwender. Das Paket stellt verschiedene Steuerelemente zur Verfügung und verwaltet diese.

Innerhalb des Paketes wird nach Betätigung einer Schaltfläche seitens des Anwenders ermittelt, um welche Taste es sich handelt und welches Merkmal angesprochen wurde. Weiterhin wird hier die Erzeugung und Positionierung von Steuerelementen und Fenstern sowie die Darstellung derselben veranlasst.

Die Struktur des Paketes ist aus *Abb. 5-3* ersichtlich. Die Klasse `cSteuerelement` bildet die Basisklasse für alle Steuerelemente. Sie beinhaltet als identifizierendes Attri-

but `SeID` sowie einige weitere Attribute, welche Größe, Position, Typ und die jeweils zugehörige *Merkmal-ID* festlegen.

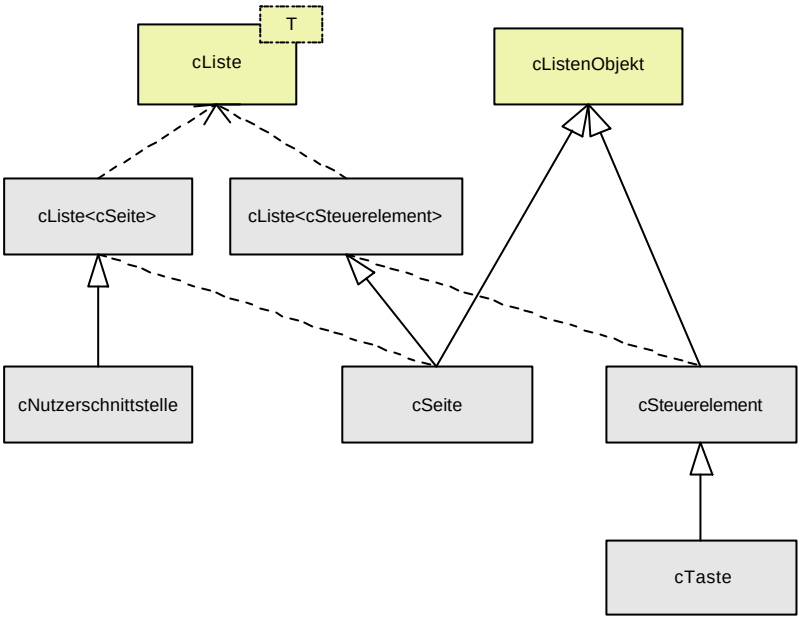

Abb. 5-3: Klassen des Paketes `Nutzerschnittstelle`

Zunächst existiert nur eine konkrete Ausprägung von Steuerelementen. Hierbei handelt es sich um Schaltflächen, welche durch die Klasse `cTaste` modelliert werden. In dieser Klasse wird auch die grafische Darstellung der Tasten unter Ausnutzung der *Palm-API*-Funktionen realisiert.

Die einzelnen Steuerelemente werden in Formularen oder Fenstern dargestellt. Die Klasse `cSeite`, die eine Liste von Steuerelementen darstellt, modelliert ein solches Verhalten. Neben Attributen, welche die *Seiten-ID*, die Größe und die Position des Fensters beinhalten, ist für diese Klasse auch das Attribut `Form` vom Typ `FormType*` definiert, das den Zugriff auf die grafische Ausprägung des Formulars gewährleistet. `FormType` ist ein vom *Palm OS*® zur Verfügung gestellter Datentyp, welcher alle notwendigen Informationen beinhaltet, um ein Formular (Fenster) zu erzeugen.

Die Klasse `cNutzerschnittstelle` schließlich repräsentiert die oberste Ebene des Paketes `Nutzerschnittstelle`. Sie ist definiert als eine Liste von Seiten und verwaltet sämtliche Fenster und damit auch die entsprechenden Steuerelemente.

`cNutzerschnittstelle` enthält u.a. eine Methode `erzeugeSeiten`, die dafür sorgt, dass die Objekte für die einzelnen Seiten und die jeweiligen Steuerelemente erzeugt und die entsprechenden grafischen Objekte generiert werden.

Eine weitere wichtige Methode ist `handleEvent`. Sie ist verantwortlich für die Behandlung verschiedener Ereignisse. In dieser Methode wird z.B. die Betätigung von Schaltflächen ausgewertet, und die *Steuerelement-ID* sowie die *Merkmal-ID* werden ermittelt. Des Weiteren wird hier die Notwendigkeit der Aktualisierung eines Fensters registriert, und die entsprechenden Aktionen werden veranlasst.

5.2.4 Struktur des Paketes `Datenbank`

Bei vielen Paketen der Systemfamilie ist es erforderlich, Daten über die Laufzeit der Software hinaus zu speichern und später wieder darauf zugreifen zu können. Es ist daher sinnvoll, einheitliche Mechanismen für die Speicherung der Daten und den Zugriff auf diese Daten zu schaffen. Derartige Mechanismen werden im Paket `Datenbank` modelliert, dessen Struktur in *Abb. 5-4* illustriert wird.

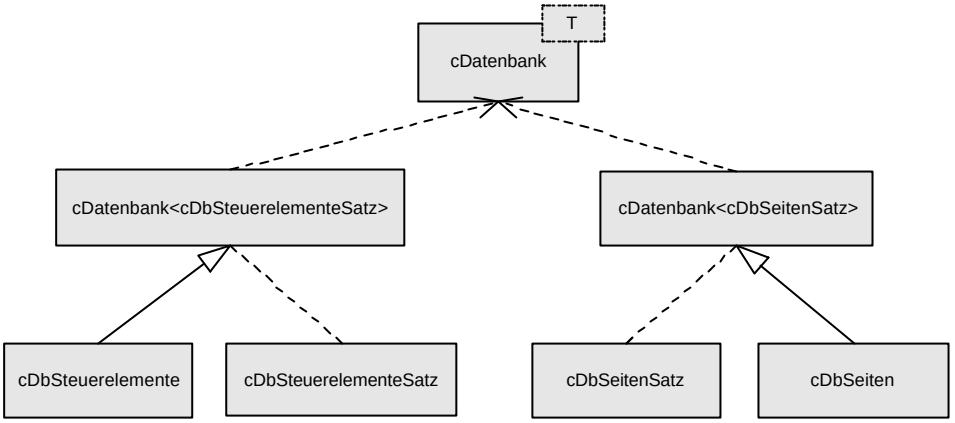

Abb. 5-4: Klassen des Paketes `Datenbank`

Es wird zunächst ein Template `cDatenbank` definiert, das grundlegende Operationen implementiert. Dazu gehören u.a. das Erzeugen (Methode `erzeuge`), Öffnen (`oeffne`) und Schließen (`schliesse`) von Datenbanken sowie das Laden eines Datensatzes, welches in der Methode `get` realisiert wird.

Die Klassen `cDbSteuerelementeSatz` und `cDbSeitenSatz` beinhalten Attribute, die für die Generierung eines Steuerelementes bzw. einer Seite, d.h. eines Formulars, notwendig sind. Sie bilden jeweils den Parameter für eine Instanz des Templates `cDatenbank`.

Von diesen Instanzen ist zum einen die Klasse `cDbSteuerelemente` und zum anderen die Klasse `cDbSeiten` abgleitet. In diesen Klassen werden Methoden definiert, die es erlauben, die Datensätze nach bestimmten Kriterien zu durchsuchen oder neue Datensätze zu erzeugen und zu speichern.

Zu beachten ist, dass diese Datenbanken nicht auf einem herkömmlichen Datenbankkonzept basieren, sondern vielmehr das *Palm OS*®-spezifische Konzept der Datensicherung repräsentieren.

5.2.5 Struktur des Paketes `Kommunikation`

Dieses Paket ist zuständig für das Senden und Empfangen von Daten unter Zuhilfenahme verschiedener Übertragungsmedien. Im Rahmen der Entwicklung dieses Prototyps werden jedoch nur die Strukturen für die Datenübertragung via Infrarot implementiert.

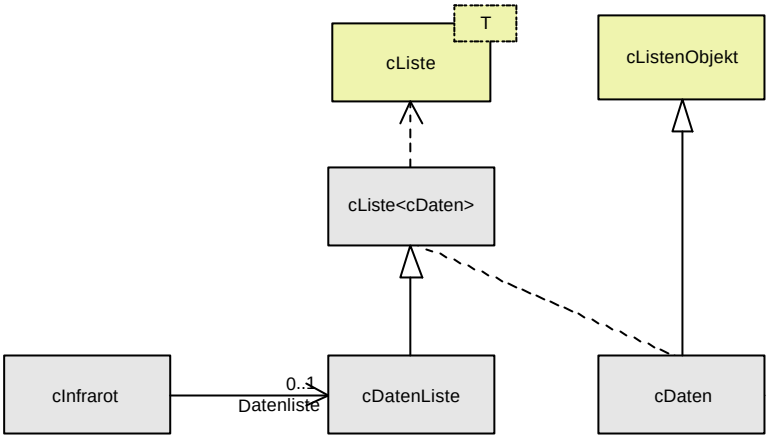

Abb. 5-5: Klassen des Paketes `Kommunikation`

Die Klasse `cInfrarot` stellt Methoden zur Verfügung, die es erlauben, eine Verbindung über die Infrarotschnittstelle aufzubauen oder zu beenden. Weitere Methoden dieser Klasse ermöglichen das Senden und Empfangen von Daten über diese Verbindung.

In `cInfrarot` wird die Liste der zu sendenden Datenpakete in dem Attribut `Datenliste` gespeichert. Dieses ist ein Objekt der Klasse `cDatenListe`. Ein Element dieser Liste wird durch ein Objekt der Klasse `cDaten` repräsentiert. Diese Klasse stellt Strukturen zur Verfügung, welche die Modellierung der jeweiligen Datenpakete ermöglichen.

Da die zu sendenden Datenpakete sich von Merkmal zu Merkmal unterscheiden, werden an geeigneter Stelle von `cDaten` abgeleitete Klassen definiert, welche die jeweils benötigten Strukturen implementieren.

5.2.6 Struktur des Paketes `Grundfunktionen`

`Grundfunktionen` ist eines der Pakete, die unmittelbar ein Merkmal repräsentieren. Aufgrund der wenig komplexen Funktionen, welche dieses Paket erfüllt, ist auch ihre Struktur, welche in *Abb. 5-6* dargestellt ist, einfach.

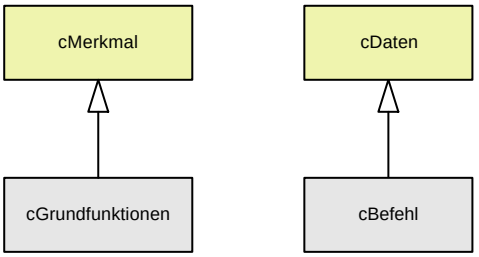

Abb. 5-6: Klassen des Paketes `Grundfunktionen`

Sie beinhaltet zum einen die Klasse `cGrundfunktionen`, welche die zentrale Klasse dieses Paketes bildet. Innerhalb dieser Klasse wird u.a. die Methode `doIt`, welche in `cMerkmal` virtuell deklariert wurde, implementiert. Darüber hinaus existieren Metho-

den, die das Senden der Infrarotsequenzen für die einzelnen Befehle veranlassen und die entsprechenden Datenpakete zusammenstellen.

Um den jeweiligen Befehl an das *VDR-System* senden zu können, wird in `Grundfunktionen` zusätzlich die von `cDaten` abgeleitete Klasse `cBefehl` definiert. Sie modelliert Strukturen, die es erlauben, ein Datenpaket zu erstellen, welches die *Merkmal-ID*, die *Funktions-ID* sowie den zur Funktion gehörenden Parameter enthält.

5.3 Implementierung ausgewählter Pakete

Um dem Leser die Funktionsweise der Software näher zu bringen, sollen in diesem Abschnitt einige Funktionen bestimmter Pakete einer näheren Betrachtung unterzogen werden. Weiterhin wird die Struktur der Datenbanken bestimmt, die für die hier implementierten Teile der Architektur notwendig sind. Begonnen werden soll jedoch mit der Beantwortung der Frage, in welcher Weise die einzelnen Merkmale aktiviert bzw. deaktiviert werden.

5.3.1 Aktivierung & Deaktivierung von Merkmalen

Für die Aktivierung und Deaktivierung von Merkmalen, d.h. die Entscheidung, ob die entsprechenden Komponenten eingebunden werden, gibt es verschiedene Lösungsvarianten. Einerseits können die einzelnen Merkmale bzw. die entsprechenden Pakete in Form von Laufzeitbibliotheken realisiert werden. Dieses Konzept wird jedoch von *Palm OS®* nicht unterstützt.

Andererseits wäre die Einrichtung eines Konfigurationsskriptes möglich, das beim Programmstart abgearbeitet wird, um so die einzelnen Merkmale zu aktivieren. Vorteil dieser Variante ist der Umstand, dass zu bei einer zusätzlichen Aktivierung existierender Merkmale nur das Skript geändert werden muss. Der Nachteil besteht darin, dass die Software auch bei wenigen aktivierten Merkmalen einen hohen Platzbedarf hat.

Die dritte Variante überlässt dem Nutzer die Aktivierung und Deaktivierung von Merkmalen mit Hilfe eines Optionsmenüs. Dies hat den Vorteil, dass der Anwender jederzeit selbst entscheiden kann, welche Funktionen ihm zur Verfügung stehen sollen. Nachteilig wirkt sich hier der ebenfalls hohe Platzbedarf auch bei wenigen aktivierten Merkmalen sowie der hohe Einarbeitungs- und Verständnisaufwand aus.

Die im Rahmen dieser Arbeit favorisierte Lösungsmöglichkeit realisiert die Aktivierung der Merkmale mit Hilfe von *Compiler-Direktiven*. Hierbei erfolgt das Einbinden der Pakete zum Zeitpunkt der Generierung der Anwendung. Der Vorteil dieser Variante besteht darin, dass der Platzbedarf der Anwendung minimiert werden kann, da nur die wirklich benötigten Quelltextsegmente berücksichtigt werden. Dies kommt der beschränkten Speicherkapazität eines *Palm-Handhelds* entgegen.

Es wird eine *Header*-Datei `MerkmalEinAus.h` erzeugt, in der mit Hilfe von `#define`-Anweisungen verschiedene Makros definiert werden, die jeweils die Aktivierung oder Deaktivierung eines Merkmals realisieren. Für das Merkmal **GRUNDFUNKTIONEN** hat die entsprechende `#define`-Anweisung z.B. folgendes Aussehen:

```
#define _Grundfunktionen
```

Soll das entsprechende Merkmal nicht aktiviert werden, wird die jeweilige Anweisung aus der *Header*-Datei entfernt.

Innerhalb des Quelltextes wird das definierte Makro dazu benutzt, einzelne Quelltext-Segmente beim Compilieren einzubinden oder zu entfernen. Hierzu werden die zu einem Merkmal gehörenden Anweisungen von einem `#if defined`-Konstrukt eingerahmt:

```
#if defined(_Grundfunktionen)
  cGrundfunktionen Grundfunktionen;
  cGrundfunktionen* gf;
            .
            .
            .
  MerkmalListe->anhaengen(gf);
  gf->unlock();
#endif
```

Innerhalb von `MerkmalEinAus.h` werden auch die Einschränkungen und Bedingungen, die im Merkmalmodell definiert wurden, umgesetzt. So findet im Anschluss an die Definition der Makros die Überprüfung statt, ob die verschiedenen Einschlussbeziehungen berücksichtigt und alle obligatorischen Merkmale aktiviert wurden. Auch dies soll am Beispiel des Merkmals **GRUNDFUNKTIONEN** verdeutlicht werden:

```
#if defined(_AnsteuerungVDR) && !defined(_Grundfunktionen)
  #define _Grundfunktionen
#endif
```

Bekanntermaßen handelt es sich bei **GRUNDFUNKTIONEN** um ein obligatorisches Merkmal. Dies bedeutet, dass bei Auswahl des Merkmals **ANSTEUERUNG DES VDR** automatisch **GRUNDFUNKTIONEN** aktiviert wird. Daher wird überprüft, ob `_AnsteuerungVDR` definiert ist. Ist dies der Fall, wird `_Grundfunktionen` ebenfalls definiert, unabhängig von der Wahl des Anwenders.

5.3.2 Struktur der verwendeten Datenbanken

Wie bereits erwähnt wurde, benötigen einige Pakete Daten, welche in Datenbanken gespeichert werden. Die Struktur dieser Datenbanken soll an dieser Stelle dargestellt werden.

Die Informationen zu den Seiten, die von `Nutzerschnittstelle` erzeugt werden, sind in der Datenbank `Seiten` festgehalten. Diese hat folgende Struktur:

Attribut	Bedeutung
`SID`	*Seiten-ID*; dient zur eindeutigen Identifikation der Seite
`Name`	Name der Seite
`Pos`	Position der Seite (linke, obere Ecke)
`Groesse`	Größe der Seite

Die Steuerelemente, die auf den Seiten platziert werden, werden ebenfalls in einer Datenbank gespeichert. Diese trägt den Namen `Steuerelemente` und hat folgende Struktur:

Attribut	Bedeutung
`SeID`	*Steuerelement-ID*; dient der eindeutigen Identifikation
`Typ`	Typ des Steuerelementes (Taste, Textfeld, Listbox, …)
`Pos`	Position des Steuerelementes (linke, obere Ecke)
`Groesse`	Größe des Elementes
`Label`	Beschriftung
`FID`	Zugehörende *Funktions-ID*
`Param`	Parameter der Funktion
`SID`	*Seiten-ID* der zugehörenden Seite
`MID`	*Merkmal-ID* des zuständigen Merkmals

Einige Komponenten benötigen darüber hinaus noch weitere Datenbanken. Diese sind jedoch zumeist nur für das jeweilige Paket relevant, so dass auch die Struktur dieser Datenbanken erst in dem entsprechenden Abschnitt definiert wird.

Wie bereits in *4.2.10* erwähnt, führt die Integration des Merkmals **MEHRNUTZERVERWALTUNG** u.a. zu einigen Änderungen in den Datenbanken. Dies ist erforderlich, da kenntlich gemacht werden muss, welches Steuerelement bzw. welche Seite den einzelnen Nutzern zugeordnet wird. Aus diesem Grund wird den Datenbanken bei Bedarf das Attribut `NID` zugeordnet, welches die zum jeweiligen Nutzer gehörende *Nutzer-ID* beinhaltet.

5.3.3 Funktionsweise der Software

Der Aufbau der Anwendung folgt zunächst einmal den prinzipiellen Vorgaben beim Design einer Applikation für das *Palm OS®*, wie sie im Abschnitt *5.1* beschrieben werden. Auch hier existiert eine Funktion `PilotMain`, von der aus die Start-Routine `AppStart`, der *Event-Loop* `AppEventLoop` und die Stopp-Routine `AppStop` aufgerufen werden.

Im weiteren Verlauf des Abschnittes sollen einige wichtige Programmteile näher vorgestellt werden.

5.3.3.1 Start & Initialisierung

In der Funktion `AppStart` werden verschiedene, globale Variablen initialisiert. Weiterhin werden hier die Objekte für die einzelnen merkmalmodellierenden Pakete bzw. die entsprechenden Klassen generiert und in einer Liste organisiert.

Die Initialisierung der Merkmale erfolgt über den Aufruf der Funktion `InitMerkmale`. In dieser Funktion wird ebenfalls die Nutzerschnittstelle initialisiert, d.h., es werden die Seiten sowie die entsprechenden Steuerelemente geladen und erzeugt. Die prinzipielle Arbeitsweise dieser Funktion ist in *Abb. 5-7* dargestellt.

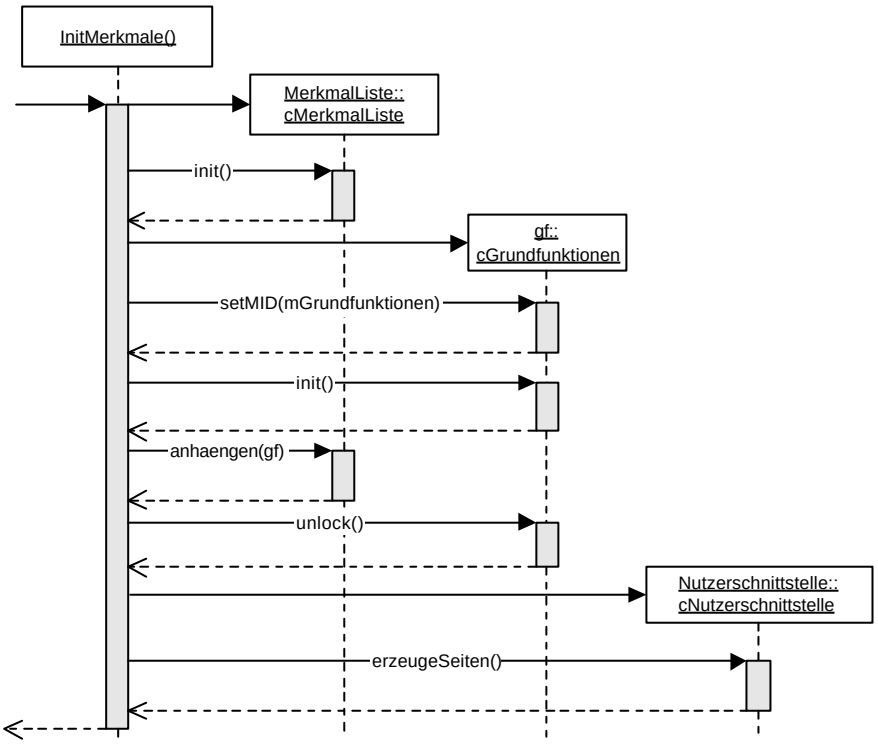

Abb. 5-7: Initialisierung der Merkmalliste und der Nutzerschnittstelle

Das Sequenzdiagramm zeigt lediglich die Initialisierung für das Merkmal **GRUNDFUNKTIONEN**. Dieses wird repräsentiert durch das Objekt `gf`, welches eine Instanz der Klasse `cGrundfunktionen` ist. `gf` wird nach dem Setzen der *Merkmal-ID* an die Merkmalliste, repräsentiert durch das Objekt `MerkmalListe` (Instanz der Klasse `cMerkmalListe`), angehängt. Das Verfahren bei anderen Merkmalen bzw. den entsprechenden Objekten ist identisch.

Im Anschluss an die Initialisierung der Merkmale folgt die Initialisierung der Nutzerschnittstelle. Der Aufruf der Methode `erzeugeSeiten` bewirkt, dass die Informationen zu den einzelnen Seiten und Steuerelementen aus den Datenbanken gelesen und die entsprechenden Objekte erzeugt werden.

5.3.3.2 Die Ereignisschleife

Nachdem der Vorgang der Initialisierung abgeschlossen ist, wird der *Event-Loop* gestartet. Hier besteht die Möglichkeit, auf Ereignisse, wie die Betätigung von Schaltflächen durch den Nutzer, zu reagieren.

Die Verarbeitung der Ereignisse erfolgt in der Funktion `MainFormHandleEvent`, in deren Rahmen die Methode `handleEvent` des Objektes `Nutzerschnittstelle` (Instanz der Klasse `cNutzerschnittstelle`) aufgerufen wird.

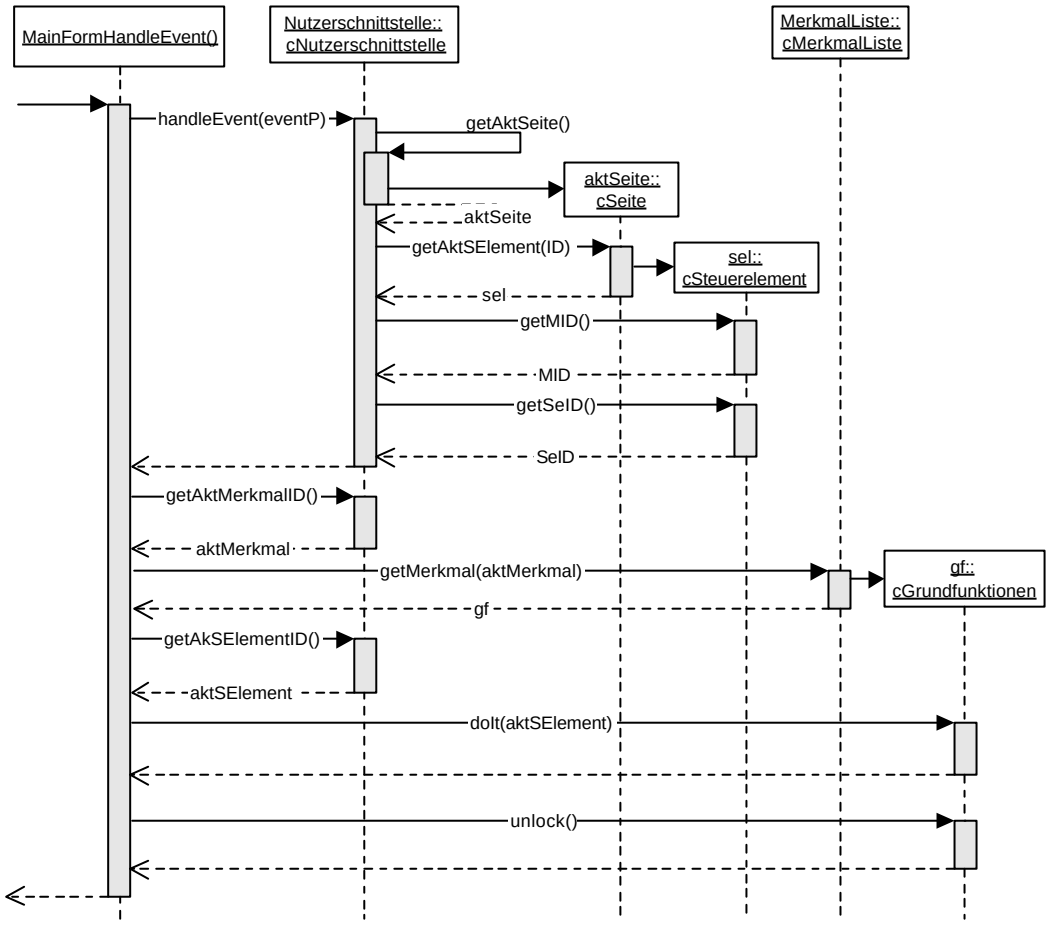

Abb. 5-8: Verarbeitung von Ereignissen

In `handleEvent` wird eine Vorverarbeitung des Ereignisses vorgenommen. Es werden die aktuell eingeblendete Seite (`aktSeite`) sowie das aktive Steuerelement (`sel`) bestimmt. Aus diesen Daten lassen sich die *Steuerelement-ID* (`SeID`) und die *Merkmal-ID* (`MID`) des zuständigen Merkmals extrahieren. Weiterhin kann in der Methode `handleEvent` bei Bedarf die Aktualisierung einer Seite oder eines Steuerelementes veranlasst werden.

Im Anschluss wird innerhalb der Funktion `MainFormHandleEvent` das zuständige Merkmal auf Basis der *Merkmal-ID* aus der Merkmalliste ausgelesen. Ist dies geschehen, kann die Methode `doIt` des entsprechenden Merkmalobjektes aufgerufen werden. In der Darstellung handelt es sich hierbei um das Objekt `gf`, welches das Merkmal

GRUNDFUNKTIONEN repräsentiert. Die Arbeitsweise von `doIt` wird im folgenden Abschnitt dokumentiert.

5.3.3.3 Arbeitsweise der Methode `doIt`

Bei `doIt` handelt es sich, wie bereits erwähnt, um die Methode, die den Zugriff auf alle Fähigkeiten des Merkmals erlaubt. Die Methode kann selbstverständlich für die einzelnen Merkmale eine unterschiedliche Implementierung erfahren, die grundsätzliche Arbeitsweise wird jedoch immer ähnlich sein.

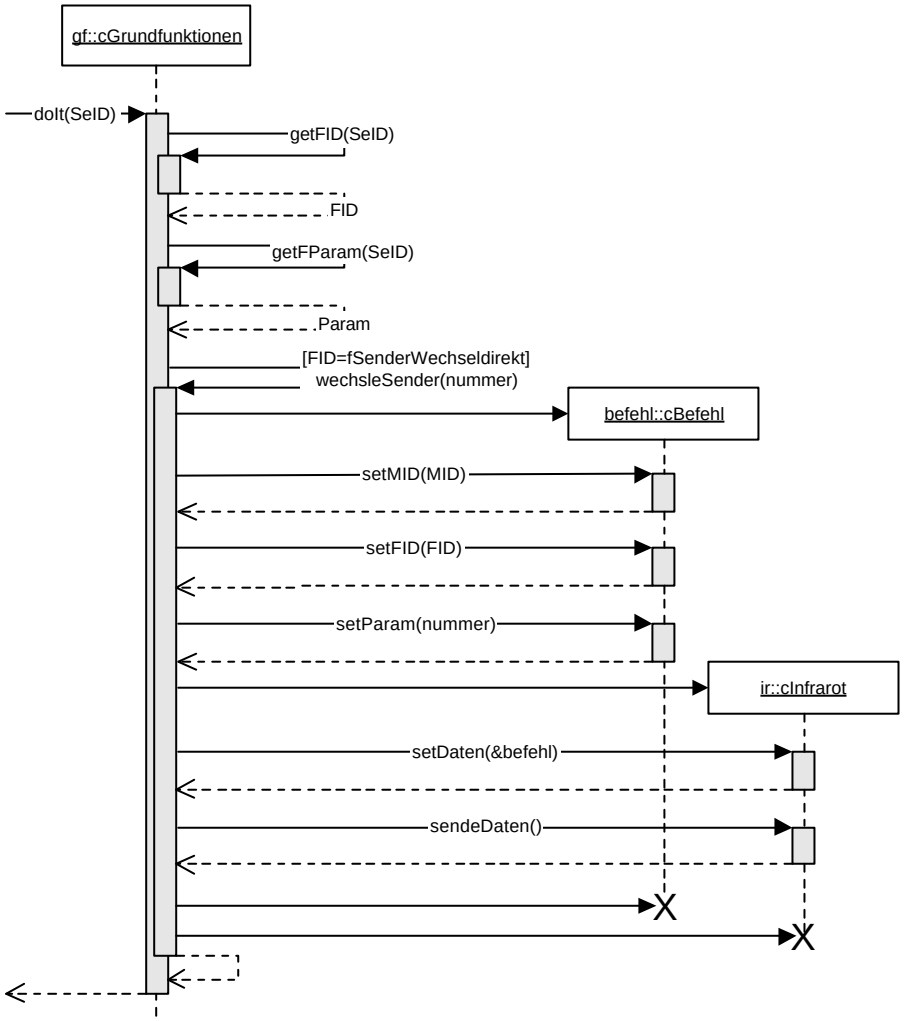

Abb. 5-9: Arbeitsweise der Methode `doIt`

Wie bereits erwähnt, wird `doIt` der Parameter `SeID`, d.h. die *ID* des betreffenden Steuerelementes, übergeben. Auf Basis dieses Parameters werden mit Hilfe der Methoden `getFID` und `getFParam`, welche von der Klasse `cMerkmal` zur Verfügung gestellt werden, die *Funktions-ID* und der jeweilige Parameter für diese Funktion ermittelt. Diese können, wie bereits aus *5.3.2* erkenntlich, aus der Datenbank `Steuerelemente` entnommen werden. Im Anschluss daran erfolgt eine Auswertung der *Funktions-ID*. Abhängig von deren Wert wird die entsprechende Funktion ausgelöst. Um die Übersicht-

lichkeit zu wahren, ist in der Abbildung nur eine einzelne Funktion dargestellt. In der Realität erfolgt hier je nach Funktionsumfang des Merkmals eine größere Verzweigung.

Das gezeigte Beispiel gewährt auch einen Einblick in die Arbeit mit dem Paket `Infrarot`. Wie zu erkennen ist, wird hier zunächst ein Objekt der Klasse `cBefehl` erzeugt. Diesem werden die *Merkmal-ID*, die *Funktions-ID* und der Funktionsparameter übergeben. Aus diesen Daten erstellt `befehl` die zu sendende Infrarotsequenz. Im Anschluss daran wird eine Instanz der Klasse `cInfrarot` erzeugt, welcher das wie beschrieben generierte Befehlsobjekt zugeordnet wird. Der Aufruf der Methode `sendeDaten` baut eine Infrarotverbindung auf und sendet die Sequenz.

6 Fazit & Ausblick

Dieses Kapitel soll sowohl einen Rückblick auf die Arbeit als auch einen Ausblick auf zukünftige Entwicklungsmöglichkeiten des Systems sein. Zunächst wird der im Rahmen dieser Arbeit entwickelte Lösungsansatz bewertet und mit der Zielsetzung der Arbeit verglichen. Darauf folgt ein Ausblick auf mögliche Richtungen der Weiterentwicklung der in Prototypform vorliegenden Universal-Fernbedienung.

6.1 Bewertung der Ergebnisse

Die grundlegenden Ziele dieser Arbeit bestanden darin,

- ein Bedienkonzept mit einer möglichst hohen Flexibilität,
- eine Systemfamilie als Umsetzung dieses Konzeptes sowie
- einen ersten Prototyp der Universal-Fernbedienung

zu entwickeln. Im Rahmen der Entwicklung entstand ein Merkmalmodell, welches mehr als 20 Merkmale umfasst. Dies gestattet es prinzipiell, ca. 6000 verschiedene Anwendungen zu konfigurieren. Von diesen ähneln sich natürlich viele enorm, sie unterscheiden sich teilweise nur in kleinen Details. Dennoch reicht die Palette der Anwendungen von der einfachen Fernbedienung, die es lediglich erlaubt, ein *VDR-System* anzusteuern, über Geräte, die zusätzlich in der Lage sind, *EPG-Daten* zu empfangen und auszuwerten, bis hin zum vollständigen System, das die Kontrolle über sämtliche Unterhaltungsgeräte eines Haushalts ermöglicht und darüber hinaus noch individuelle Gestaltungsmöglichkeiten bietet.

Die aus dem Merkmalmodell entwickelte Systemfamilienarchitektur erfüllt ebenfalls die an sie gestellten Anforderungen. Sie gibt die Möglichkeit, die einzelnen Pakete unabhängig voneinander zu entwickeln und sie entsprechend den Vorgaben durch das Merkmalmodell zu kombinieren. Um ein hohes Maß an Verständlichkeit und Wartbarkeit zu erreichen, wurden eine Richtlinie definiert, in deren Vordergrund die Schaffung einer zentralen Klasse für jedes merkmalmodellierende Paket stand. An dieser Richtlinie orientiert sich das Design der einzelnen Pakete. An dieser Stelle soll auch darauf hingewiesen werden, dass die hier entwickelte Architektur keine vollständige Lösung repräsentiert, sondern vielmehr den Ausgangspunkt für zukünftige Entwicklungen bilden soll. Die grundsätzlichen Strukturen wurden im Rahmen dieser Arbeit geschaffen, die detaillierte Ausarbeitung der einzelnen Komponenten muss an anderer Stelle erfolgen.

Einzig der entstandene Prototyp entspricht nicht vollständig den Erwartungen. Zwar wird durch ihn durchaus die grundsätzliche Funktionsfähigkeit der Systemarchitektur demonstriert, die im Vorfeld geplante, weiterreichende Implementierung konnte jedoch nicht realisiert werden. Grund hierfür sind die unerwarteten Schwierigkeiten, die bei Programmierung der modellierten Strukturen und Funktionen auftraten. Diese entstanden u.a. daraus, dass entgegen anders lautender Aussagen noch keine vollständige Umsetzung des objektorientierten Paradigmas im Rahmen der *Palm-API* erfolgt ist. Weiterhin ist für die Nutzung der *IrDA*-Schnittstelle nur eine unzureichende Dokumentation erhältlich, so dass der Zeitaufwand für die Einarbeitung in diesen Bereich unerwartet hoch ausfiel.

Zusammenfassend lässt sich sagen, dass die im Vorfeld definierten Ziele erreicht wurden, auch wenn in einigen Bereiche Abstriche gemacht werden mussten.

6.2 Weiterentwicklung der Systemfamilie und des Prototypen

Abschließend sollen verschiedene Richtungen aufgezeigt werden, in denen eine Weiterentwicklung des gesamten Systems vorstellbar ist, falls sich das zugrunde liegende Konzept bewähren sollte.

Zunächst erscheint es notwendig, den bestehenden Prototypen zu überarbeiten und zu vervollständigen. Hierzu zählt vor allem die Implementierung von umfassenden Fehlerbehandlungsroutinen sowie Untersuchungen hinsichtlich eventueller Leistungsoptimierungen der Implementierung.

Im Anschluss daran können das Design und die Implementierung der bisher definierten Pakete und Merkmale vorgenommen werden. Im Vordergrund sollte hierbei die Entwicklung der Pakete stehen, die in unmittelbarem Zusammenhang mit dem Kernelement des *Digitalen Video Projektes*, dem *Video Disk Recorder*, stehen.

Sind die in dieser Arbeit definierten Merkmale bzw. die entsprechenden Pakete vollständig implementiert, können Untersuchungen betrieben werden, in welchen Bereichen sinnvoll weitere Merkmale der Systemfamilie hinzugefügt werden können. Denkbar wäre beispielsweise eine Erweiterung des Merkmals **NUTZERPROFIL**. Hier könnten Fähigkeiten und Funktionalitäten hinzugefügt werden, die es der Fernbedienung erlauben, sich selbständig dem Nutzerverhalten anzupassen.

Ebenfalls denkbar ist eine Portierung der Architektur auf eine andere, leistungsfähigere Hardware. Besonderes Augenmerk sollte in diesem Zusammenhang auf *Mobile-Pads* gerichtet werden. Aufgrund der im Vergleich zu dem hier verwendeten *PDA* größeren Palette an Einsatzgebieten eröffnen sich hier weitere, innovative Entwicklungsmöglichkeiten.

7 Zusammenfassung

Im Rahmen dieser Arbeit wurde festgestellt, dass heutzutage viele Bedienkonzepte und Fernbedienungen nebeneinander existieren, die für sich betrachtet in vielen Bereichen durchaus ausgereift sind. Das gilt vor allem im Sektor der Geräteintegration, hier gibt es eine große Auswahl an Produkten, die vielfältige Möglichkeiten in dieser Hinsicht bieten. Darüber hinaus existieren jedoch kaum Anstrengungen, die Fernbedienung über die Verwendung zur Steuerung von Geräten hinaus einzusetzen. Im Bezug auf Mehrnutzerverwaltung und individuelle Gestaltung der Fernbedienung beispielsweise gibt es heutzutage nur sehr wenige und unzureichende Lösungen.

Es sollte nun ein Konzept entwickelt werden, welches diese Erkenntnisse berücksichtigt und eine neue und innovative Lösung des Problems anbietet. In einem ersten Schritt wurden die Anforderungen für das zu erstellende System ermittelt. Diese umfassen u.a. Fähigkeiten, welche die Ansteuerung eines *VDR-Systems*, die Integration programmbegleitender Daten (*Electronic Program Guide*), die Ansteuerung und Integration zusätzlicher Geräte sowie die Verwaltung von Nutzerprofilen erlauben.

Aus diesen Anforderungen wurde im Anschluss ein Merkmalmodell entwickelt. Hierbei entstanden drei Gruppen von Merkmalen. Die erste Gruppe umfasst Merkmale, die sowohl die Ansteuerung eines *VDR-Systems*, als auch den Empfang und die Auswertung von *EPG-Daten* erlauben. Die Ansteuerung und Integration beliebiger Geräte aus dem Bereich der Unterhaltungselektronik wird durch die zweite Merkmalgruppe realisiert. Abschließend existiert eine Gruppe von Merkmalen, welche individuelle Gestaltungsmöglichkeiten der Universal-Fernbedienung beinhalten. Gleichzeitig wurden in dieser Phase der Entwicklung Parameter und Beschränkungen, die bei der Konfiguration des Systems Beachtung finden müssen, herausgearbeitet.

Für die Entwicklung der Systemfamilie war es in einem weiteren Schritt notwendig, die beschriebenen Merkmale derart auf Pakete abzubilden, dass eine möglichst einfache Generierung von Familienmitgliedern gewährleistet werden kann. Im Verlauf des Entwurfes entstanden zwei verschiedene Arten von Paketen. Zum einen gibt es Pakete, die unmittelbar Merkmale repräsentieren. Sie folgen einer einheitlichen Designrichtlinie und beinhalten alle Strukturen und Funktionen, die notwendig sind, um die jeweiligen Fähigkeiten des Merkmals zu realisieren. Aufgrund der Erkenntnis, dass viele dieser Pakete eine Reihe von identischen Strukturen und Funktionen benötigen und nutzen, wurde eine zweite Art von Paketen geschaffen, welche diese Strukturen definieren.

Zur Abrundung der Arbeit wurde die bisher entwickelte Architektur in einem ersten Prototypen umgesetzt. Dieser umfasst neben den Basispaketen auch ein merkmalrepräsentierendes Paket, das es erlaubt, ein *VDR-System* mit seinen Grundfunktionen zu nut-

zen. Der Prototyp stellt jedoch keine uneingeschränkt nutzbare Anwendung dar, sondern soll lediglich die grundsätzliche Verwendbarkeit der Architektur belegen und die Basis für künftige Weiterentwicklungen bilden.

Die im Rahmen dieser Arbeit entwickelte Systemfamilie macht es möglich, eine Vielzahl von Anwendungen zu erzeugen. Das realisierte Konzept besitzt eine hohe Flexibilität, die es erlaubt, sich exakt den Wünschen und Vorstellungen der Nutzer anzupassen. Aufgrund des modularen Aufbaus der Systemarchitektur ist es jederzeit mit geringem Aufwand möglich, einzelne Komponenten in ihrer Struktur zu verändern oder dem System neue Komponenten hinzuzufügen.

A Ergänzende Aktivitätsdiagramme

Einige der in *4.3* dargestellten Prozesse enthalten zusammengesetzte Aktivitäten. Die entsprechenden Unteraktivitätsdiagramme werden im Folgenden dargestellt.

1. Unteraktivitätsdiagramm „Funktion ermitteln“

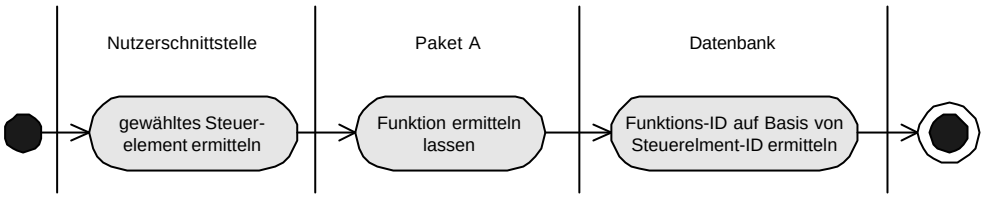

Abb. 7-1: Funktion ermitteln

Wie leicht zu erkennen ist, handelt es sich hierbei um die Umsetzung der Datenflüsse, wie sie in *Abb. 4-4* beschrieben werden. Die *ID* des betreffenden Steuerelementes wird von `Nutzerschnittstelle` ermittelt und an das jeweilige merkmalmodellierende Paket übergeben. Dieses ist dann dafür verantwortlich, aus der Datenbank die zugehörige Funktion zu lesen. Dieser Datenbankzugriff erfolgt selbstverständlich über die in `Datenbank` modellierten Strukturen.

2. Unteraktivitätsdiagramm „Senden beliebiger Daten“

Die verschiedenen Arten des Sendens von Daten unterscheiden sich zunächst nicht, unabhängig von der Art der gesendeten Daten und dem Übertragungsmedium.

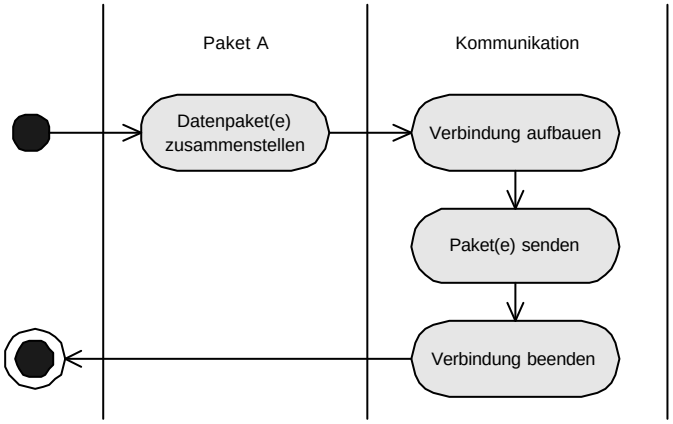

Abb. 7-2: Senden von Daten

Die einzelnen zu übermittelnden Datenpakete werden vom jeweiligen merkmalmodellierenden Paket erzeugt, anschließend wird von `Kommunikation` eine Verbindung aufgebaut. Über diese werden dann die Datenpakete übertragen, bevor die Verbindung wieder beendet wird.

3. Unteraktivitätsdiagramm „Empfangen von Daten“

Das Empfangen von Daten verläuft äquivalent zum in *Abb. 7-2* dargestellten Senden von Daten.

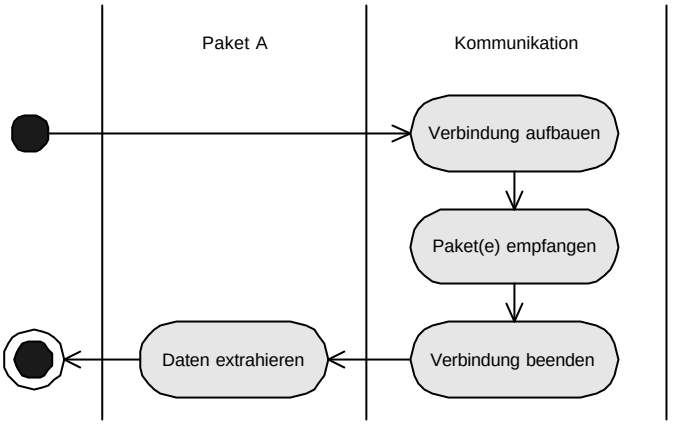

Abb. 7-3: Empfangen von Daten

4. Unteraktivitätsdiagramm „*EPG-Eintrag* ermitteln“

In den Paketen `RecProg` und `Reminder` werden Aktionen auf Grundlage eines ausgewählten *EPG-Eintrages* ausgelöst. Für die Verwaltung der *EPG-Einträge* ist das Paket `EPG` zuständig.

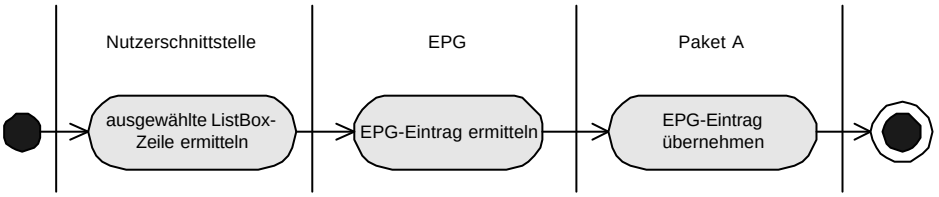

Abb. 7-4: Ermitteln des aktuellen *EPG-Eintrages*

Von `Nutzerschnittstelle` wird das momentan markierte Feld in der entsprechenden Listbox ermittelt. Auf dieser Basis kann `EPG` dann den aktuellen *EPG-Eintrag* ermitteln und an das jeweilige Paket weiterleiten.

5. Unteraktivitätsdiagramm „Senden von Daten über den *Video Disk Recorder*“

Manche Merkmale sollen die Möglichkeit zur Verfügung stellen, eine Internetverbindung entweder direkt über die Fernbedienung oder über den *Video Disk Recorder* aufzubauen. Die Abläufe eines Verbindungsaufbaus über das *VDR-System* ist *in Abb. 7-5* dargestellt.

Um die Möglichkeiten des *Video Disk Recorders* zu nutzen, ist es zunächst erforderlich, die jeweiligen Daten via Infrarotschnittstelle von der Fernbedienung auf das *VDR-System* zu übertragen. Hier kann dann mit Hilfe der `Web`-Komponente eine Internetverbindung aufgebaut werden, und die Daten können übertragen werden.

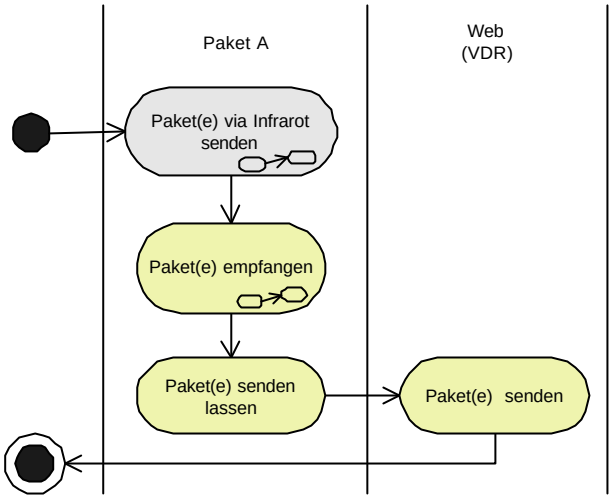

Abb. 7-5: Senden von Daten über den *VDR*

6. Unteraktivitätsdiagramm „Empfangen von Daten über den *Video Disk Recorder*“

Das Empfangen von Daten über den *Video Disk Recorder* erfolgt analog zum Senden von Daten über den *Video Disk Recorder*.

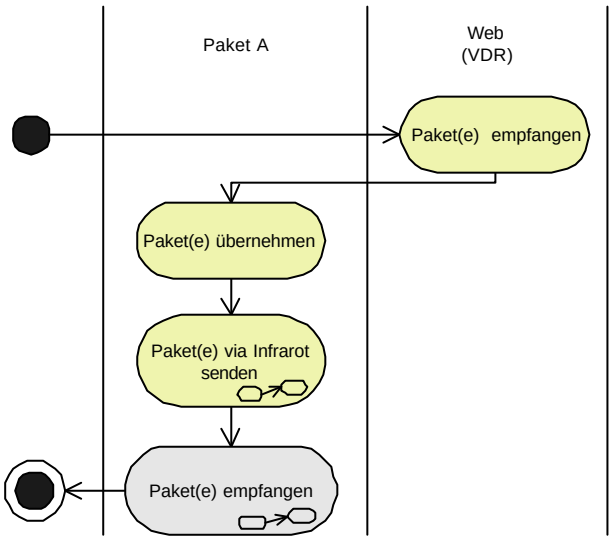

Abb. 7-6: Empfangen von Daten über den *VDR*

B Thesen zur Diplomarbeit

- Ein Bedienkonzept muss sich dem Nutzer anpassen, nicht der Nutzer dem Konzept.

- Die Flexibilität der zum aktuellen Zeitpunkt existierenden Fernbedienungen ist, vor allem im Hinblick auf die individuelle Gestaltbarkeit der Bedienung, unzureichend.

- Eine Systemfamilie ist die optimale Realisierungsvariante für eine Universal-Fernbedienung.

- Ziel der Diplomarbeit ist es, ein Merkmalmodell für die Universal-Fernbedienung zu entwickeln und dieses in einer geeigneten Systemarchitektur umzusetzen.

- Die im Rahmen dieser Diplomarbeit entwickelte Systemfamilie und deren Architektur erlauben eine einfache und flexible Generierung einer Vielzahl von Anwendungen.

- Ein *Palm-Handheld* als grundlegende Hardwarekomponente eignet sich nicht nur als Prototyp, sondern auch als praktisch anwendbare Realisierung der in dieser Arbeit konzipierten Universal-Fernbedienung.

Ilmenau, den _____ _____

Ralph Dietzel

C Quellen

Für den Bestand der in der Arbeit enthaltenen Internet-Adressen kann aufgrund der ungeheuren Dynamik, mit der sich das Informationsangebot im Internet entwickelt, keine Gewähr übernommen werden. Bei Beendigung der Arbeit waren alle Adressen intakt.

[1] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, A. Spencer Peterson. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania.

[2] Kyo C. Kang, Kwanwoo Lee, Jaejoon Lee. *Feature Oriented Product Line Software Engineering: Principles and Guidelines*. Pohang University of Science and Technology, Sajoong Kim, Korea IT Promotion Agency.

[3] Detlef Streitferdt, Kai Böllert, Matthias Riebisch. *Feature-Modell und Architektur einer Systemfamilie*. 45th International Scientific Colloquium, Ilmenau Technical University.

[4] Matthias Riebisch, Kai Böllert, Detlef Streitferdt. *Extending Feature Diagrams with UML Multiplicities*. Ilmenau Technical University

[5] Christoph Bartelmus. *Ein Pinguin sieht (infra)rot*. c't 18/2000.

[6] Seite der Firma *Bang & Olufsen*: http://www.bang-olufsen.com

[7] Seite der Firma *Universal Electronics* (Produktreihe *ONE FOR ALL®*): http://www.oneforall.de

[8] OmniRemote

[9] Seite der Firma *Philips* (Produkt *iPronto*): http://www.pronto.philips.com

[10] Seite des *Digitalen Video Projektes* : http://www.theoinf.tu-ilmenau.de/~streitdf/DVP/index.html

[11] Seite des *Video Disk Recorder*-Projektes: http://www.cadsoft.de/people/kls/vdr/index.htm

[12] Greg Wilson, Jean Ostrem. *Palm OS® Programmer's Companion*. PalmSource Inc.

[13] Greg Wilson, Jean Ostrem, Clif Liu, Doug Fulton. *Palm OS® Programmer's API Reference*. PalmSource Inc.

D Abbildungsverzeichnis