Technischen Universität Ilmenau
Fachgebiet Prozessinformatik

Diplomarbeit

zum Thema

Entwurf einer Bibliothek zur Auswertung der Service Informationen in DVB Systemen

vorgelegt von

Andreas Schrankel

am

22.10.2003

Technischen Universität Ilmenau
Fachgebiet Prozessinformatik

Diplomarbeit

Entwurf einer Bibliothek zur Auswertung der Service Informationen in DVB Systemen

Vorgelegt von	:	Andreas Schrankel
Beginn der Arbeit	:	22.04.2003
Abgabe der Arbeit	:	22.10.2003
Registriernummer	:	
Verantwortlicher Hochschullehrer	:	Prof. Dr.-Ing. habil. Ilka Philippow
Betreuer	:	Dipl.-Inf. Detlef Streitferdt
		Technische Universität Ilmenau Fakultät für Informatik und Automatisierung Fachgebiet Prozessinformatik D-98693 Ilmenau
In Zusammenarbeit mit	:	Dipl.-Inf. Guido Richardt Dipl.-Ing. Jörg Reinholdt
		TechnoTrend AG Research and Development Melchior-Bauer-Strasse 5 D-99092 Erfurt

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

BAT	Bouquet Association Table
CA	Conditional Access
CAT	Conditional Access Table
CRC	Cyclic Redundancy Check
DIT	Disconuity Information Table
EBU	European Broadcasting Union
EIT	Event Information Table
EPG	Electronic Program Guide
ETSI	European Telecommunications Standarts Institute
IRD	Integrated Receiver Decoder
ISO	International Organisation for Standardization
NIT	Network Information Table
NVOD	Near Video On Demand
MPEG	Moving Pictures Expert Group
PAT	Program Association Table
PID	Packet IDentifier
PMT	Program Map Table
PSI	Program Specific Information
RST	Running Status Table
SDT	Service Description Table
SIT	Selection Information Table
ST	Stuffing Table
TDT	Time Date Table
TOT	Time Offset Table
TS	Transport Stream
TSDT	Transport Stream Description Table
UTC	Universal Time, Co-ordinated

Kapitel 1

Einleitung

Jedes Empfangsgerät für DVB Datenströme muss wenigstens rudimentär die Service Informationen nutzen. Eine Implementierung der Verarbeitung ist also grundsätzlich nichts neues. Allerdings nutzen diese Implementierungen die SI nicht vollständig aus, sondern nur Teile daraus, um eine bestimmte Aufgabe erfüllen zu können. Es gibt Beispiele von ganz einfachen Anwendungen, um nach vorhandenen Sendern zu suchen oder um eine elektronische Programmzeitung (EPG) zu generieren. Sehr komplexe Applikationen, die weite Teile der SI nutzen, sind meist für Set-Top-Boxen entworfen oder Teil einer TV- oder Datenapplikation für den PC. Dazu kommt, daß die meisten Implementierungen auf geschlossenem Quelltext basieren. Einzige Ausnahme sind Implementierungen für Linux.

Ziel dieser Arbeit ist es eine Bibliothek zu entwerfen, die sich ausschliesslich auf die Verarbeitung der SI konzentriert und diese möglichst vollständig nutzen kann. Dazu soll sie flexibel zukünftigen Erweiterungen und Veränderungen der Spezifikationen anpassbar sein.

Die TechnoTrend AG

Die TechneTrend AG ist eine im Jahre 1990 gegründetes mittelständiges Unternehmen, welches heute etwa 50 Mitarbeiter beschäftigt. Behaimatet ist es in Efurt Bindersleben.

Hauptbeschäftigungsfeld der Firma inst die Entwicklung von Hardwarelosungen zum Empfang von digitalem Fernsehen (DVB). Dabei werden für alle Arten der digitalen Sendetechnik Geräte entwickelt (Breitbandkabel, Satellit und terrestrische Antenne). Es werden sowohl Set-Top-Boxen (Beistellgeräte zum Fernseher) als auch PC-Karten zum Empfang geschaffen.

Man kann die Firma in mehrere Bereiche aufteilen: Den Größten und Wichtigsten stellt die Hardwareentwicklung mit der dazugehörigen hardwarenahen Softwareentwicklung dar. Dann folgt u.a. die Softwarenetwicklung für den PC (Applikationen und Treiber für MS Windows). Dazu kommt ein Testlabor zur Verifikation der Produkte, ein Hardwarelabor zum Erstellen Reparieren der Prototypen und die kaufmännische Abteilung.

Weitere Informationen über das Unternehmen und seine Produkte sind im Internet unter www.technotrend.de zu finden.

Kapitel 2

Stand der Technik

Dieses Kapitel erklärt die Grundlagen, auf denen diese Arbeit basiert. 

2.1 Das DVB-System

DVB wurde als Standart für die Verbreitung multimedialer Dienste in Breitbandnetzen entwickelt. Hauptaugenmerk liegt dabei auf dem digitalen Fernsehen, also auf Video- und Audiodatenströmen. Dazu kommen eine Reihe verschiedener Datendienste.

Zuordnung von Namen zu Programmen und Providern, was die Auswahl für den Nutzer erleichtert. Beschreibungen der Sendungen mit Start- und Endezeitpunkt informieren den Nutzer über das aktuelle Programm und in Form des Electronic Program Guides (EPG) sogar über mehrere Tage hinweg. Für Betreiber von Conditional-Access (CA) Diensten, sogenannten Pay-TV, bieten sich erweiterte Möglichkeiten, verschiedene Verschlüsselungssysteme nutzen zu können. Meist von diesen Providern werden auch andere Möglichkeiten von DVB genutzt wie Near-Video-On-Demand (NVOD) und Mosaic Dienste. Bei Near-Video-On-Demand wird zeitversetzt um beispielsweise 15 Minuten das gleiche Programm mehrmals ausgestrahlt. Der Nutzer kann innerhalb der vorgegebenen Rasters frei entscheiden, zu welchem Zeitpunkt er das Programm empfangen möchte. Bei Mosaic Dienste wird das Videobild in 9 Bereiche im Raster 3 x 3 unterteilt. In diese Teilbereiche werden dann verschiedene empfangene Videodatenströme gelegt und dann zu einem Gesamtbild gemischt. Dabei können benachbarte Teilbereiche zu einem Bereich vereinigt werden und einen Videostrom zeigen.

2.2 Spezifikation der SI

2.2.1 Offizielle Dokumente

DVB basiert auf der MPEG-2 Spezifikation ISO 13818-1 von 1994.

Standadisiert wurde DVB von der European Broadcast Union (EBU) bei der European Telecommunications Standarts Institute (ETSI). Die ersten Standarts waren für DVB-S und DVB-C im Dezember 1994. 1997 folgte die Normierung für DVB-T.

Standart für DVB-SI ist EN 300 468 „Specification for Service Information in DVB Systems“ (ursprünglich Oktober 1995)und TR 101 211 „Guideline on implementation and usage of Service Information“ (ursprünglich Juli 2000) bei ETSI. Der Standart TR 101 162 „Allocation of Service Information and Data Broadcasting Codes codes for DVB Systems“ liegt momentan noch als Entwurf (Draft) vor.

2.2.2 Beschreibung der Begriffe:

delivery system (Übertragungssystem):
: Ein physikalisches Medium, das einen oder mehrere Multiplex-Datenströme überträgt. Momentan werden 3 Wege zur Übertragung genutzt: kabelgebunden in den Breitbandverteilkabeln der Telekom und kabellose Funkübertragung per Satellit (Astra, Eutelsat) und von Funkanlagen auf der Erde (terrestrische Übertragung). Letztere löst derzeit im Raum Berlin die bisherige analoge Technik ab.

network (Netzwerk):
: Ein Bündel von Multiplex MPEG-2-Transportströmen, die über ein Medium gemeinsam übertragen werden.

multiplex:
: Im Zeitmultiplexverfahren zusammengefasste Transportströme. Ein Kanal eines Übertragunssystems enthält einen Multiplex.

transport stream (Transportstrom):
: Datenstruktur zur paketorientierten Übertragung von Daten. Es stellt die Basis der DVB-Standarts dar.

broadcaster oder *(service) provider*:
: Organisation, die eine zeitliche Abfolge von Ereignissen oder Sendungen erstellt und diese zum Zuschauer überträgt.

service (Dienst, Programm):
: Eine Abfolge von Ereignissen und Sendungen unter der Kontrolle eines Providers.

event (Ereignis, Sendung):
: Zusammenfassung von Elementardatenströmen mit einem definierten Start- und Endzeitpunkt. Die Elementardatenströme gehören alle zu einem gemeinsamen Service.

Die o.g. Begriffe lassen sich auf 2 Arten miteinander verbinden: anhand der Struktur der Datenströme und auf logische Art.

Wenn man die Struktur der Datenströme betrachtet, enthält jedes der vielen existierenden Übertragunssysteme einige Kanäle für Nutzdaten. Diese können pro Kanal entweder ein analoges Programm oder ein digitaler Multiplex sein.

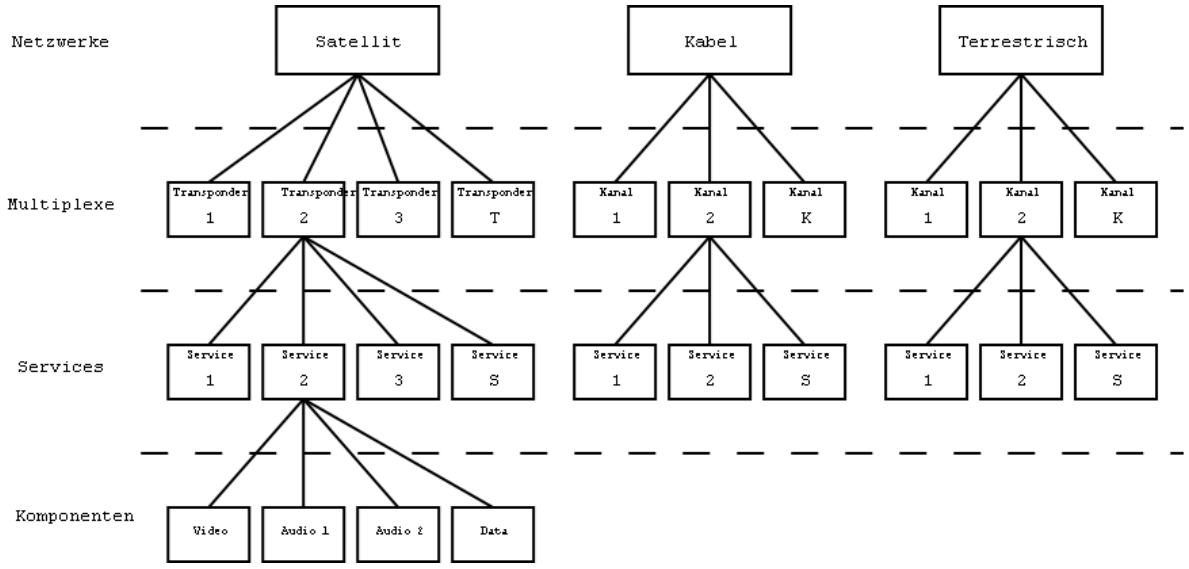

Abbildung 2.1: Hierahie der Datenströme

Die o.g. Hierchien werden wie folgt auf Tabellen abgebildet:

2.2.3 Beschreibung der Tabellen

Tabellen der SI:

Network Information Table (NIT) Informationen zu einem Netzwerk (Namen) und die darin enthaltenen Transportströme. Für jeden Transportstrom wird je nach Art des zugrundeliegenden Übertragungsmediums die nötigen Daten für den Empfang wie Frequenzen, Symbolraten, Position des Satelliten usw. angegeben. Dazu wird auf die enthaltenen Service verwiesen.

Bouquet Association Table (BAT) In der Tabelle wird der Bouquetname und Verweise auf die zugehörigen Services übertragen.

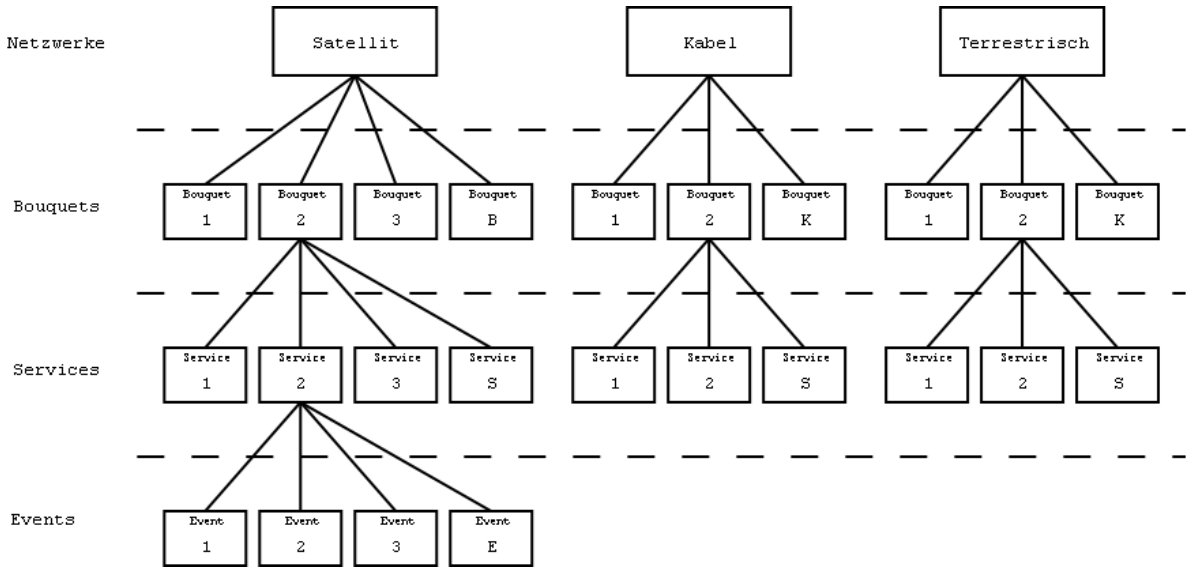

Abbildung 2.2: Hierahie der Tabellen

Service Description Table (SDT) Zum Namen eines jeden Services werden auch angaben über Verschlüsselung gesendet.

Event Information Table (EIT) Zu jedem Service werden hier Informationen zu den enthaltenen Events gesendet, z.B. Startzeit und Dauer, Status, Verschlüsselung, Bezeichner.

Time Date Table (TDT) Die Tabelle enthält die aktuelle Zeit und das Datum.

Time Offset Table (TOT) Diese Tabelle enthält wie die TDT die aktuelle Zeit und das Datum, zusätzlich auch Angaben zur Zeitverschiebung in der aktuellen Zeitzone.

Running Status Table (RST) Mit der RST werden aktuell auftretende Änderungen im Status eines oder mehrerer Events angezeigt. So kann der zu frühe oder verspätete Beginn einer Sendung gemeldet werden.

Stuffing Table (ST) Mit einer ST werden auf Senderseite ungültig gewordene Sections überschrieben, so daß die Nummerierung der Sections im Datenstrom konsistent zu halten. Der Inhalt der Tabelle wird nicht ausgewertet.

Discontinuity Information Table (DIT) Normalerweise wird ein Transportstrom kontinuierlich, also quasi endlos gesendet. Es gibt Fälle, in denen der Transportstrom aber nur

stückweise, also unvollständig übertragen werden kann oder ein Teil auf einem Speichermedium abgelegt werden soll.
Die DIT signalisiert, daß der Transportstrom unterbrochen und unvollständig ist.

Selection Information Table (SIT) Die SIT ersetzt in einem unvollständigen Transportstrom die anderen SI-Tabellen und liefert Informationen über die enthaltenen Services und Events.

IP Notification Table (INT) Die INT enthält Informationen für Datendienste.

MPEG-2 Tabellen:

Program Association Table (PAT) Die PAT liefert die Verknüpfung zwischen den Programmnummern und den Nummern der Transportstrompaketen (PIDs). Die Programmnummern dienen lediglich als numerische Bezeichner für Programme.

Program Map Table (PMT) Die PMT verknüpft die Programmnummern mit den im Programm enthaltenen Elementen.

Conditional Access Table (CAT)

Transport Stream Description Table (TSDT)

2.2.4 Verarbeitung der Tabellen

Prizipielle Beschreibung des Aufbaus der Tabellen:
Übertragung der Tabellen in Sections und prinzipieller Aufbau der Sections:
Alle Tabellen haben einen gemeinsamen Header mit den Feldern table_id, section_syntax_indicator und section_length. Jede Section beginnt mit diesen Feldern. Ein Teil der Tabellen besitzt auch einen erweiterten Header mit den Feldern table_id_extension, version_number, current_next_indicator, section_number und last_section_number. Diese Felder werden nur von einem Teil der Tabellentypen implementiert. Über diese Felder können Subtables unterschieden werden (table_id_extension), über die Versionsnummer veraltete Tabellen durch aktuellere ersetzt werden und lange Tabellen auf mehrere Sections verteilt werden (section_number und last_section_number). Eine Tabelle kann eindeutig über die Felder table_id und table_id_extension idendifiziert werden.
Tabelle 2.1 erläutert diese und weitere Felder näher. Die Abbildungen 2.3 und 2.4 zeigen als Beispiele die Struktur einer Section einer sehr kurzen und einer langen Tabelle.

Danach folgen tabellenspezifische Datenfelder, die sehr unterschiedlich genutzt werden. Manche Tabellen wie TDT oder DIT nutzen nur wenige Felder festern Länge. Die meisten enthalten eine oder 2 Descriptorloops.

Bis auf TDT, RST, DIT und ST sichern alle Tabellen ihre Daten durch eine Prüfsumme vom Typ CRC32. Die Ausnahmen bestehen deswegen, weil die 4 Bytes der Prüfsumme bei den kleinen Tabellen die Länge der Nutzdaten übersteigen würde oder im Fall der ST werden gar keine schützenswerten Nutzdaten übertragen werden.

Feld	**Erklärung**
table_id	Dieses 8-Bit Feld gibt den Typ der Tabelle an
section_syntax_indicator	Ist der Indicator 0 dann folgen die Datenbytes nach dem section_length Feld. Sonst folgen 5 weitere Felder und am Ende der Section ist ein CRC-Wert zur Fehlerkontrolle enthalten.
section_length	Das Feld gibt die Anzahl der Bytes an, die direkt auf diesess Feld folgen. Die Gesamtlänge ist auf 4096 Bytes begrenzt, bei einigen Tabellentypen sogar auf 1024 Bytes.
table_id_extension	Mit diesem Feld werden verschiedene Tabellen gleichen Typs unterschieden. Je nach Tabellentyp hat das Feld unterschiedliche Bedeutung, z.B. bei der NIT enthält es die network_id
version_number	Dieses 5-Bit Feld gibt die Versionsnummer der Tabelle an.
current_next_indicator	Wenn dieses Flag den Wert 1 hat, sind die aktuellen Daten gültig. Der Wert 0 kündigt einen Wechsel der Daten und damit der Version an.
section_number	Wenn eine Tabelle über mehrere Sections verteilt gesendet wird, werden die Sections über diese Nummer unterschieden.
last_section_number	Dieses Feld gibt die höchste Nummer aller Sections dieser Tabelle an.

Tabelle 2.1: Felder einer Tabelle

2.2.5 Descriptoren und Descriptorloops

Ein Descriptor ist Container für Nutzdaten innerhalb einer Tabelle. Jeder Descriptor beginnt mit 2 festen 8-Bit Feldern: descriptor_tag gibt den Typ an und descriptor_length die Anzhal der Bytes, die auf das Feld folgen.

```
time_data_section
{
    table_id                          8
    section_syntax_indicator          1
    reserved_future_use               1
    reserved                          2
    section_length                   12
    UTC_time                         40
}
```

Abbildung 2.3: Struktur der TDT als Beispiel einer kurzen Tabelle

Die MPEG-2 Spezifikation listet 17 verschiedene Descriptortypen auf, die SI Spezifikation weitere 46. Dazu kommt eine große Zahl privater, also von Drittanbietern definierte Typen. Durch den Einsatz von Descriptoren können Tabellen flexibel Nutzdaten verschiedenen Typs und Länge aufnehmen. Descriptoren sind nicht an einen Tabellentyp gebunden, manche Tabellen nutzen auch einige Descriptortypen geminsam.

Die Abbildungen 2.5 und 2.6 zeigen 2 Descriptoren als Beispiele.

Sogenannte Descriptorloops (siehe Abb. 2.4) enthalten einen oder mehrere Descriptoren in direkter Folge. Durch die Felder descriptor_tag und descriptor_length können diese unterschieden werden. Die Gesamtlänge einer solchen Loop steht entweder explizit als Datenfeld vor der Loop oder ergibt sich implizit aus der Gesamtlänge der Section.

2.2.6 Filterung und Dekodierung

- Zugriff auf Informationen in den Tabellen, feste Datenfelder, Loops in der Tabelle, Descriptorloops, Abbildung auf geschachtelte Listen

- Eigenschaften der Codierung wie Bandbreitenoptimierung durch platzsparende Codierung und verschiedene Wiederholraten

Eindeutige Idendifizierung einer Tabelle über die Felder table_id, table_id_extension und version_number, eine Teilsection zusätlich über das Feld section_number. Das Feld last_section_number gibt die höchte vorkommende Nummer der Teilsections an, nicht die Anzahl der Teilsections. Wenn eine Tabelle nur in einer Section übertragen wird, ist dieses Feld daher 0. Alle Felder in den Kopf der Sections sind bis auf die Nummer und Länge gleich. Überlicherweise werden alle Sections einer Tabelle direkt hintereinander in der Reihenfolge der Sectionnummern gesendet. Die Teilsections gehören zwar alle zu einer gemeinsamen Tabelle, aber die Daten-

```
network_information_section
{
    table_id                                8
    section_syntax_indicator                1
    reserved_future_use                     1
    reserved                                2
    section_length                         12
    network_id                             16
    reserved                                2
    version_number                          5
    current_next_indicator                  1
    section_number                          8
    last_section_number                     8
    reserved_future_use                     4
    network_descriptors_length             12
    for(i=0;i<N;i++)
    {
        first_descriptor_loop
    }
    reserved_future_use                     4
    transport_stream_length                12
    for(i=0;i<N;i++)
    {
        transport_stream_id                16
        original_network_id                16
        reserved_future_use                 4
        transport_descriptors_length       12
        for(j=0;j<M;j++)
        {
            second_descriptor_loop
        }
    }
    CRC_32                                 32
}
```

Abbildung 2.4: Struktur der NIT als Beispiel einer großen Tabelle

```
satellite_delivery_system_descriptor
{
    descriptor_tag                    8
    descriptor_length                 8
    frequency                        32
    orbital_position                 16
    west_east_flag                    1
    polarization                      2
    modulation                        5
    symbol_rate                      28
    FEC_inner                         4
}
```

Abbildung 2.5: Struktur eines Satellite Delivery System Descriptors als Beispiel eines Descriptors mit fester Läge

```
network_name_descriptor
{
    descriptor_tag                    8
    descriptor_length                 8
    for(i=0;i<N;i++)
    {
        char                          8
    }
}
```

Abbildung 2.6: Struktur eines Network Name Descriptors als Beispiel eines Descriptors mit variabler Läge

strukturen der Nutzdaten sind jede für sich eine vollständige Tabelle. Jede Teilsection kann auch für sich genommen dekodiert werden, aber sie beinhalten eben nur einen Teil der Daten.

Mit Hilfe des Feldes version_number wird eine Versionierung der Tabellendaten möglich. Das Feld speichert 5 Bit, also sind Versionsnummern von 0 bis 31 möglich. Jede Tabelle hat eine Versionsnummer zugeordnet, für eine neuere Version wird die Versionsnummer um 1 erhöht. Bei Versionsnummer 31 erfolgt ein Rücksprung auf den Wert 0.

2.2.7 Zugriff auf Informationen

Aus Sicht des Nutzers (hier vor allem eine Applikation für TV oder Datendienste) interessiert nicht die Aufteilung in Sections oder Tabellen. Er hat konkrete Fragen, die mit den Informationen, die in den Tabellen stecken beantwortet werden sollen.

Zum einen gibt es einfache Anfragen zu Informationen, die in einer Tabelle allein stecken. Dazu sind keine oder wenige Parameter notwendig, z.B. Ausgabe der Nummern aller Service auf einem Transponder oder den Namen eines Services zu einer vorgegebenen Nummer.

Dann gibt es auch komplexe Anfragen, für deren Beantwortung mehrere Tabellen verknüpft werden müssen. Ein Beispiel wäre die Frage, alle Servicenamen zu einen gegebenen Bouquetnamen auszugeben. Dazu muss anhand der BAT zum Bouquetnamen die Bouquet-ID und alle dazugehörenden Service-IDs. Mit diesen kann dann in den SDTs die Servicenamen ermittelt werden.

In beiden Fällen die Tabellendaten eines oder verschiedener Tabellentypen gebraucht, um sie nach den benötigten Informationen zu durchsuchen.

2.3 Nutzung in der Praxis

Abweichungen bzw. Verletzungen der Spezifikation:

Fehlerhafte Implementierungen:
Beispielsweise falsche Längenangaben (besonders bei ungenutzten Descriptorloops), Dadurch Probleme bei der Verarbeitung eintreffender Daten, Eine mögliche Folge wäre ein Absturz der Software, der sich aber durch geeignete Fehlerbehandlungen abfangen läßt. Schlimmer als das ist aber der entstehende Datenverlust, der dazu führt, daß Informationen zu Netzwerken, Services oder Events verloren gehen oder falsch zugeordnet würden. Für den Nutzer zeigen sich solche Fehler darin, daß einzelne Programme „verschwunden“ sind oder daß beispielsweise die Videodaten des einen Programms den Audiodaten eines anderen zuordnet werden.

Unvollständige Implementierungen:
Hierbei sind notwendige Informationen nicht im Datenstrom enthalten. Diese Informationen betreffen nicht die Kernfunktionen Audio und Video sondern Zusatzdienste wie Teletext, EPG oder einfach Sender- und Bouquetnamen. Diese Abweichungen stören die Verarbeitung der Tabellendaten nicht und stellen aus technischer Sicht kein Problem dar.

Senden von Zusatzinformationen:
Nutzung von Private-Tables, Private-Descriptors oder eines der vielen aktuell nicht genutzten Datenfeldern in den Tabellen und Descriptoren („resered“, „reserved_future_use“ oder „private“), um Informationen nach selbst definierten Formaten zu übertragen. Diese Felder stören die Verarbeitung der übrigen, allgemein bekannten Felder nicht, aber es werden Informationen übersehen. Das Problem besteht darin, vom Sender genaue Angaben über die Lage der Daten und deren Formate zu bekommen. Auch ändern sich die herstellerspezifischen Spezifikationen öfters als die internationalen und man hat einen höheren Aufwand, um immer aktuell zu bleiben.

- Beispiel: Premiere World

2.4 Plattformen für DVB-Empfänger

- Beschreibung der verschiedenen Plattformen: PC und Set-Top-Box
  - Unterschiede in der Hardware: Prozessoren: Arbeitsspeicher: STB: im kB Bereich, PC: meist mind. 128MB eher mehr, einige MB nutzbar Festspeicher: STB: Flash als beschreibbarer Festspeicher, nur wenige kB nutzbar, ansonsten EEPROM (reed-only), PC: Festplatte, einige MB nutzbar Bedienung: STB: nur über On-Screen-Display und Fernbedienung, PC: auch über Monitor, Maus, Tastatur
  - daraus resultierende Unterschiede in der Software: STB: ANSI-C, wenige Bibliotheken, PC: OO-Sprachen wie C++ um vollem Umfang nutzbar, Kein aufwendiges Haushalten mit Arbeitsspeicher notwendig, Anlegen umfangreicher Datenstrukturen möglich,
  - PC Plattform als Grundlage für diese Arbeit

2.5 Implementierung von TechnoTrend

- Vorstellung von API und App
  - Funktionsumfang, Nutzung der SI
  - Grenzen des Funktionsumfanges

2.6 LibDVBSI

- Vorstellung
  - Funktionsumfang
  - Nutzung in dieser Arbeit, Abgrenzung der Arbeiten

2.7 BDA als neue Basis

input/output pins, Baukasten, Module, Filterkette
Vorteil: Hardwareunabhängigkeit, Module können (frei) ausgetauscht werden

2.8 Präzisierte Aufgabenstellung

- Geforderter Eigenschaften:
  - Auswertung aller Tabellentypen aller Descriptoren, flexible Erweiterbarkeit auf weitere Tabellen und Descriptoren
  - Konfigurierbarkeit zur Laufzeit
  - Konfigurierbarkeit um Libs mit verschiedenem Funktionsumfang bauen zu können, Beschränkung auf bestimmte Tabellen-, Descriptor- ud Applikationsmodule ermöglichen
  - Geforderte Umgebung:

  Plattform: PC, Unterstützung aller TT PCline Produkte (Premium, Budget-PCI, Budget-USB, SetTopBox)
  Softwarebasis: API und BDA
  Applikationen: zuerst eigene Testapp
  Entwicklungsumgebung: MS Visiual C++, weitgehend ohne MFC

Kapitel 3

Lösungsansatz

In diesem Kapitel wird für die gestellte Aufgabe eine geeignete Lösung erarbeitet. 

3.1 Konzept für die Verarbeitung der SI

3.1.1 Definition der benötigten Funktionalität

Auf der einen Seite soll die Bibliothek den Empfang der Sections steuern können und die Rohdaten annehmen können. Dem Nutzer soll sie eine erweiterbare Schnittstelle für Anfragen bieten, die dessen Aufgaben angepaßt sein soll. Über diese Schnittstelle soll der Nutzer bereits aufgearbeitet Informationen erhalten. Die Struktur der Tabellen und Sections soll für den Nutzer dabei keine Rolle spielen und versteckt sein. Intern muss die Bibliothek empfangene Sections zwischenspeichern, um Anfragen des Nutzers schnell beantworten zu können.

3.1.2 Aufbau des Schichtenmodells

Im wesentlichen besteht die Klassenstruktur aus 3 Schichten: SectionLayer, TableLayer, ApplicationLayer. Dazu kommt ein Hilfspaket Descriptors und eine globale Controllerklasse.

SectionLayer:
Empfang der Sections (bei API: eines Tabellentyps, bei BDA: für alle Typen), Dekodierung des Sectionheaders, Vereinigung von Tabellen, die über mehrere Sections erteilt gesendet werden, Aussortieren von bereits empfangenen Sections

Kommuniziert nach unten mit API- bzw. BDA-Klassen. Benutzt darin nur 2 Funktionen: SetFilter und ResetFilter um einen Filter auf einen Tabellentyp zu starten bzw. zu beenden. Um die Daten zu empfangen bietet die SectionLayer eine Callback-Funktion an.

Nach oben werden die bearbeiteten Daten an den Controller weitergereicht, der sie an die passende Klasse in der TableLayer weiterreicht.

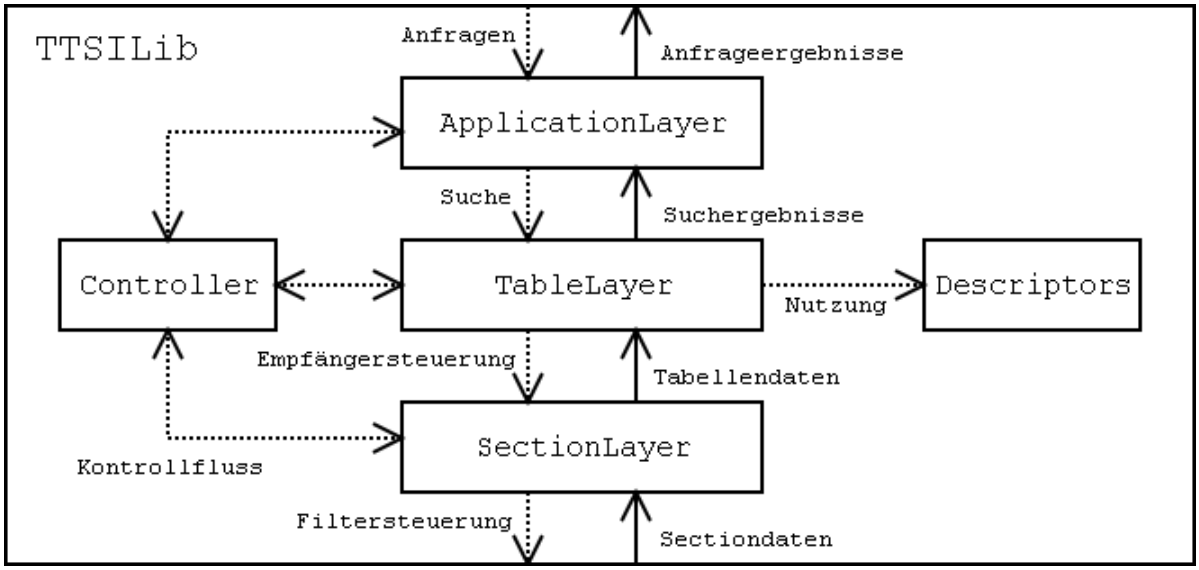

Abbildung 3.1: Aufbau des Schichtenmodells

TableLayer:
Funktion: Speicherung der empfangenen Tabellen (nicht nur Sections), Dekodierung der Tabellen, auch der Nutzdaten (Descriptoren)

ApplicationLayer:
Enthält Klassen, die jeweils einen Anwendungsfall implementieren. Nach aussen bieten sie die für den AF benötigte Schnittstelle, nach innen greifen sie auf die Daten der TableLayer zu. Diese Klassen beinhalten die eigendlcihe Anwendungslogik und hier werden die Daten der Tabellen verknüpft.

Descriptors:
Hilfsklassen, die die Datenstrukturen der verschiedenen Descriptoren und Funktionen für deren Dekodierung implementieren. Die TableLayer greift auf die zu. Daher sind diese Klassen stark mit denen der TableLayer verknüpft. Eine Vereinigung der beiden Bereich ist aber nicht sinnvoll, da keine Ein-Eindeutige Zuordnung von Tabellentyp zu Descriptortypen möglich ist.

Controller:
Zentrale Steuerung der Bibliothek. Ausserhalb der Lib nicht sichtbar. Beinhaltet globale Funktionen und Daten sowie die Instanzen der Klassen aus der Sectio- und TableLayer.

3.2 Anwendungsfälle

Die Anwendungsfälle sollen die Klassen der ApplicationLayer näher beschreiben. Jeder Fall wird in Form einer oder mehrerer Klassen innerhalb der ApplicationLayer implementiert. Das heißt auch, daß jeder davon Zugriff auf die internen Speicherstrukturen hat und nach aussen hin in Form eines einer Schnittstelle seine Funktionalität veröffentlicht.

Natürlich sind die hier gezeigten 5 Fälle nur ein Anfang. Die Bibliothhek kann und soll nach Bedarf erweitert werden.

3.2.1 Senderlisten und Suchlauf

Nachfolgend wird der Ablauf eines Sendersuchlaufes beschrieben:

Nacheinander wird der Empfänger (das Forntend) auf die verschiedenen Frequenzen eingestellt. In diesem Punkt unterscheiden sich die Netzwerke etwas: Bei kabelgebundenen und terrestrischem Empfang gibt es zwar länderspezifische aber feste Kanäle. Jeder dieser Kanäle hat feste und bekannte Merkmale wie Frequenz und Signalschrittrate. Bei diesen beiden Netzwerken wird bei einem Suchlauf nacheinander jeder dieser Kanäle am Frontend eingestellt und geprüft, ob dort ein digitaler Datenstrom empfangen wird. Bei Satellitenempfang werden die Daten nicht in festen Kanälen ausgestrahlt. Die notwendigen Daten wie Frequenzen, Polarisation, Signalschrittrate usw. muss einer PC-Applikation oder SEt-Top-Box vom Hersteller in Form von Transponderlisten mitgegeben werden. Dieser wiederum bezieht die Daten aus Veröffentlichungen im Internet. Wenn ein Empfänger einmal mindestens einen digitalen Sender empfängt, können die Listen online aktualisiert werden. Der genaue Mechanismus des Aktualisierens beschreibt der Anwenungsfall 3.2.2 „Generierung von Transponderlisten“.

Das weitere Vorgehen beim Suchlauf ist wieder für alle Netzwerke identisch. Für jeden empfangenen Kanal wird geprüft, ob es sich um einen analogen oder digitalen Kanal handelt. D.h. ob ein MPEG-2 Datenstrom vorhanden ist oder nicht. Aus diesem können dann Tabellen gefiltert werden, um Informationen über die enthaltenen Dienste zu bekommen. Die Tabellen PAT und PMT enthalten allgemeine Informationen, welche Dienste es gibt, welchen Typ diese haben und auf welchen PIDs sie empfangen werden können. Die SDT beschreibt die Dienste näher, sie enthält beispielsweise die Namen des Dienstes und des Providers. Optional kann der BAT die Zuordnung der Dienste zu den Bouquets und die Bouquetnamen entnommen werden. All diese Informationen können von einer TV-Applikation genutzt werden, um dem Nutzer eine detailierte Übersicht über die Fernseh-, Radio-, Datendienste zu präsentieren und auch eine komfortable Auswahl der gewünschten Dienste zu ermöglichen.

3.2.2 Generierung von Transponderlisten

Wie im Abschnitt 3.2.1 beschrieben, brauchen Satellitenempänger die Transponderlisten, um die im Netzwerk enthaltenen Datenströme zu finden. Konkret bestehen die Transponderlisten aus je einer Datei für jedes Satellitennetzwerk. Die Dateien enthalten im ersten Teil die ID, die aus der Orbitalposition des Satelliten gebildet wird (siehe Abb. 3.2). Darauf folgt der Name des Netzwerkes. Im zweiten Teil enthält der Schlüssel „0“ die Anzahl der Transponder. Danach folgen für jeden Transponder ein eigener Schlüssel mit Frequenz, Polarisation, Symbolrate und Modulation.

```
[SATTYPE]
1=0192
2=Astra 1B,1C,1E,1F,1G,1H,2C
[DVB]
0=57
1=10773,H,22000,56
2=10832,H,22000,56
3=11720,H,27500,34
4=11758,H,27500,34
5=11798,H,27500,34
(...)
56=12640,V,22000,56
57=12670,V,22000,56
```

Abbildung 3.2: Struktur einer Transponderliste, gekürztes Beispiel für die Astra Satelliten

Die Schnittstelle zum Nutzer hin ist hier sehr kompakt: sie beinhaltet lediglich eine Methode, um den Pfad, wohin die Dateien gespeichert werden solln. Dazu eine parameterlose Methode, um den Speichervorgang zu starten. Die eigendliche Arbeit erfolgt intern und die Ausgabe bilden die geschriebene Dateien.

3.2.3 Dynamische Reaktion auf Events

Mit den Service Information wird zusätzlich zu den statische Daten auch dynamische Events ausgestrahlt.

Beispielsweise wird der Beginn einer Sendung als Event signalisiert. So kann auch auf eine Abweichung vom eigendlichen Sendetermin vom Empfänger richtig reagiert werden und eine

geplante Aufzeichnung der Sendung pünklich zum tatsächlichen Sendebeginn starten.

Als ein weiteres Beispiel dienen die Regionalprogramme vieler Sender. Hierbei „teilt“ sich ein Sender für einige Zeit in mehrere Regionalsender auf. Diese strahlen dann jeweils ein eigenständiges Programm aus. Am Ende der Zeitspanne „vereinigen“ sich die Regionalprogamme wieder zu ihrem gemeinsamen Hauptprogramm. Ein konkretes Beispiel ist der MDR, der sich in den MDR Thüringen, MDR Sachsen und MDR Sachsen-Anhalt untergliedert. Allabendlich von 19:00 Uhr bis 19:30 Uhr werden die Regionalprogramme gesendet, in der übrigen Zeit gibt es ein gemeinsames Programm. Jeder der 3 Regionalsender ist einzeln in der Programmliste aufgeführt und kann ausgewählt werden, obwohl meistens das gleiche Programm zu sehen ist. Um nicht wertvolle Netzwerkkapazitäten zu verschwenden, wird nicht 3 mal identische Video- und Audiodatenströme gesendet, sondern genau einer, auf den alle 3 Programme verweisen. Lediglich in den Zeiträumen mit eigenen Regionalprogrammen wird auch jeder Sender über einen eigenen Datenstrom versendet. Der Sprung von gemeinsamen zu regionalen und der Rücksprung zum gemeinsamen Programm wird dynamisch mit Events signalisiert. Konkret hat der Sprung einen Wechsel der Video- und Audio-PIDs innerhalb des gleichen Multiplexes zur Folge.

Realisiert werden kann die Benachrichtigung über neue Events durch blibliotheksinternen Weitergabe von Nachrichten an alle verarbeitenden Komponenten. Diese Nachrichten können dann ausgewertet und bei Bedarf auch an komponenten ausserhalb der Bibliothek wietergeleitet werden. So brauch an keiner Stelle durch wiederholte Abfragen systemressourcen verbraucht werden.

3.2.4 Datendienste

DVB wurde nicht nur für den Fernseh- und Radioempfang entworfen. Auch Datendienste sollten zum Funktionsumpfang gehören. Damit sind primär die Übertragung von IP-Paketen gemeint.

3.2.5 Conditional-Access-Dienste

3.3 Datenstruktur für die interne Speicherung der SI

3.3.1 Zweck der Speicherung

Die Daten der SI werden zyklisch wiederkehrend gesendet, so daß ein Empfänger sich zu jedem beliebigen Zeitpunkt in den Datenstrom einklinken kann. Nach einer Wartezeit stehen dem Empfänger dann alle bnötigten Daten zur Verfügung. Um die SI zu nutzen, ist also

grundsätzlich eine Zwischenspeicherung der empfangenen Daten nicht zwingend notwendig.

Tabelle 3.1 gibt für einige Tabellen die minimalen Wiederhohlraten an. Diese minimal geforderten Raten werden von den meisten Providern als tatsächlichen zeitlichen Abstand bis zum wiederholten Senden der gleichen Tabelle genutzt. Beispielsweise wird eine NIT alle 10s neu gesendet, mit anderen Worten muss man durchschnittlich 5s warten, bis man eine NIT empfängt.

Tabellentyp	**Wiederholrate**
NIT	10s
BAT	10s
SDT	2s (für aktuellen Transportstrom)
SDT	10s (für alle anderen Transportströme)
EIT (present/following)	2s (für aktuellen Transporstrom)
EIT (present/following)	10s (für alle anderen Transportströme)
EIT (other)	10s (für aktuellen Transporstrom)
EIT (other)	30s (für alle anderen Transportströme)
TDT und TOT	30s

Tabelle 3.1: Minimale Wiederholraten für Tabellen in Satelliten- und Kabelsystemen

Ohne jede Zwischenspeicherung müßte der Empfänger also bei jedem Zugriff auf eine NIT durchschnittlich 5s warten, bis die benötigten Daten zur Verfügung stehen. Diese Wartezeit würde die Verarbeitung einer Applikation viel zu stark verlangsamen. Wenn dagegen einmal empfangene Daten gepuffert werden, stehen sie zur Laufzeit des Programms schnell zur Verfügung. Die Wartezeit auf Daten wird also nur beim Start der Applikation und nach einem Transponderwechsel notwendig.

Natürlich muss eine Applikation Mechanismen implementieren, um auf die gepufferten Daten aktuell zu halten.

3.3.2 Varianten der Speicherung

Im wesentlichen gibt es 2 Möglichkeiten, die Daten der Tabellen abzuspeichern: zum einen die Speicherung der empfangenen Rohdaten und zum anderen durch Speicherung vorverarbeiteter Daten.

Im ersten Fall werden von den eintreffenden Sections nur die Kopfdaten analysiert, die Nutzdaten werden nicht weiter verarbeitet. Die Kopfdaten sind notwendig, um die Sections richtig zuordnen zu können und neue von bereits empfangenen unterscheiden zu können. Für diese Aufgaben stecken in den Nutzdaten keine notwendigen Informationen. Wenn der Verar-

beitungsschritt der Analyse der Nutzdaten an dieser Stelle eingespart wird, wird damit die Verarbeitung etwas schneller. Auch wird die Datenstruktur, in der die Sections abgelegt werden, stark vereinfacht. Die Nutzdaten werden darin lediglich durch ein Feld aus Bytes dargestellt, das nicht weiter aufgeschlüsselt wird.

Der Aufwand der Analyse der Nutzdaten ist damit allerdings nur verschoben. Er wird hier bei jedem Zugriff auf die Informationen notwendig. Bei Sections, die sich selten ändern, aber oft gelesen werden, wird der Mehraufwand unnötig hoch. Diese Möglichkeit ist nur interessant für Sections, die sehr oft aktualisiert gesendet und damit abgespeichert werden, aber auf die nur selten zugegriffen wird.

Besonders wenn die Daten mehrerer Sections durchsucht werden müssen, um eine Information zu finden, wird dieser Nachteil deutlich. Hier liegt der große Vorteil, wenn die Nutzdaten der Sections bereits direkt nach dem Empfang analysiert werden. Die Datenstruktur enthält dann alle Informationenn bereits fertig aufgeschlüsselt und der Zugriff wird damit schneller und auch einfacher. Natürlich wird dadurch die Struktur auch komplizierter. Auch wird mehr Platz im Speicher belegt, was aber beim PC als zugrundeliegende Plattform kein Problem darstellt. Insgesamt wiegt der Vorteil der direkten Zugriffsmöglichkeit auf Einzelinformationen die Nachteile.

Eine Kompromislösung wäre es, die empfangenen Daten erstmal als Rohdaten abzulegen, beim ersten Zugriff zu analysieren und dann die Rohdaten durch Einzelinformationen zu ersetzen. Dies wäre aber unnötig aufwändig und lohnt sich daher nicht.

3.3.3 Entwurf der Datenstruktur

- Pro Tabellentyp eine Klasse. Alle von einer Basisklasse abgeleitet, um gemeinsame Datenfelder dort zu definieren.

- Pro Tabelle eine Instanz der entsprechenden Tabellenklasse

- Pro Tabellentyp gibt es eine verkettete Liste. Die Instanzen der Wurzelelemente liegen im Controller. In der Basisklasse sind die Zeiger für nächstes und voriges Listenelement definiert.

Die Tabellendaten selsbt werden in den Klassen durch Membervariablen passenden Typs repräsendtiert. In der Basisklasse sind dafür die Kopfdaten definiert, die typspezifischen Daten in den abgeleiteten Klassen. Einige Tabellen beinhalten intern Schleifen mit verschiedenen Datenfeldern. Diese lassen sich am einfachsten als interne Klassen abbilden, die als verkettete Liste an die Tabellenklasse angehängt wird.

Weitere verkette Listen werden genutzt, um die Descriptorloops der Tabellen darzustellen. Jeder Descriptor wird, ähnlich wie die Tabellen, durch eine eigene Klasse dargestellt die alle von einer gemeinsamen Basisklasse erben. Die Datenstrukturen der Descriptoren sind allerdings wesentlich einfacher als die der Tabellen. So gibt es keine Schachtelung von Descriptoren in Descriptoren. Für die Darstellung der Schleifen werden keine internen Klassen benötigt.

Felder aus einfachen Datentypen genügen.

3.4 Definition der Schnittstellen

3.5 Verarbeitung der Sections

Der Empfang, die Dekodierung und Speicherung der Tabellen gliedert sich in 2 Schritte: in der SectionLayer liegen die Daten noch als einzelne Sections vor. Erst wenn die Daten einer Tabelle vollständig sind, werden sie an den Controller übergeben, von dort aus in die Speicherstruktur eingeordnet und die Dekodierung der tabellenspezifischen Daten gestartet.

3.5.1 Ablauf des Empfangs

Die Verarbeitung startet mit dem Empfang einer Section als Bytefeld vom Transportstromfilter. Als erster Schritt wird die Prüfsumme (CRC32) berechnet und mit den empfangenen Daten verglichen. Bei einer Verletzung wird hier sofort abgebrochen.

Nun folgt die Dekodierung der Felder des gemeinsamen Headers (Common Header). Anhand der table_id kann der Tabellentyp bestimmt werden und je nach Typ auch der erweitertete Header (Extended Header) dekodiert werden.

In einer einfach verketteten Liste werden die Kopfdaten aller bisher empfangenen Sections zwischengespeichert. Durch Vergleich der gerade dekodierten Kopfdaten mit den gespeicherten kann entschieden werden, ob die empfangene Section neu ist oder nicht. Auch die Nutzdaten werden solange in der Liste abgelegt, bis alle Sections, die eine gemeinsame Tabelle bilden, eingetroffen sind. Wenn die letzte fehlende Section einer Tabelle eintrifft, können so alle Teile nach der section_number sortiert an den Controller übergeben werden. Die Nutzdaten der vollständigen Tabellen können in der Liste gelöscht werden.

3.5.2 Analyse der Nutzdaten

Der Controller prüft, ob in der Speicherstruktur schon die gleiche Tabelle mit einer älteren Versionsnummer liegt. Falls ja, wird diese gelöscht. Dann wird die neue Tabelle an die Speicherstruktur angehängt.

Nach dem Einordnung in die Datenstruktur wird von dieser die Analyse der Nutzdaten gestartet. Jede Tabellenklasse implementiert eine abstrakte Funktion, um die eigenen tabellenspezifischen Daten zu dekodieren und in die interne Datenstruktur zu schreiben. Die Datenstrukturen eventuell vorhandener Schleifen werden durch interne Klassen repräsentiert.

Zur Dekodierung einer Deskriptorloop werden die Daten der Loop als Bytefeld an ersten Deskriptor in der Liste über die Funktion übergeben. Dieser Entnimmt die Bytes aus dem Feld, die zum ersten Deskriptor gehören (erkennbar am Feld desc_length) und dekodiert sie. Wenn Daten weiterer Deskriptoren übrig sind, wird ein neues Listenelement angehängt und ihm die restlichen Daten. So wird rekursiv die komplette Deskriptorloop als verkettete Liste aufgebaut.

3.6 Zugriff auf Tabellendaten

Die Tabellen werden in verketteten Listen gespeichert, jeder Tabellentyp in einer eigenen Liste. Die Zeiger auf die Wurzelelemente sind Member des Controllers und damit für alle Klassen innerhalb der Bibliothek nutzbar. Innerhalb ist ein besonderer Zugriffsschutz nicht notwendig. Lediglich das Schreiben in die Datenstruktur muss gegen konkurierende Zugriffe gesichert werden. Ausserhalb der Bibliothek gibt es keinen direkten Zugriff auf die internen Speicherstrukturen.

Hauptsächlich nutzen nur die Klassen der ApplicationLayer die gespeicherten Informationen. Die Ausnahmen bilden lediglich Zugriffe des Controllers auf die PAT, weil in dieser Tabelle die PIDs stehen, auf denen die PMTs empfangen werden können.

3.7 Konfiguration

Eine Hauptanforderung an die Bibliothek ist die Erweiterbarkeit für zukünftige Entwicklungen und Anpaßbarkeit an verschiedene Einsatzszenarien. Dazu gehören im wesentlichen 3 Fälle:

1. Die Bibliothek soll eine andere Hardware- oder Treiberarchitektur als Basis nutzen.
2. Es sollen Tabellen- oder Descriptortypen hinzugefügt oder entfernt werden.
3. Die Zahl der Anwendungsfälle und damit der Applikationsklassen wird geändert.

Gemeinsam haben diese Fälle, daß die Änderungen nicht dynamisch zur Laufzeit geschehen können, sondern eine neue Übersetzung des Quellcodes erfordern.

Bei der Anpassung an eine neue Softwarebasis ändert sich im wesentlichen der Empfang von Sections. Dafür sind die Klassen der SectionLayer zuständig. Diese Klassen arbeiten unabhängig von der Art und Anzahl der genutzten Tabellen oder Applikationen. Auch besteht diese Schnittstelle nur aus 3 Funktionen: jeweils eine zum Setzen und zum Rücksetzen von Filtern und eine, die die Rohdaten einer empfangenen Section liefert. Anpassungen würden sich also auf diese 3 Funktionen beschränken. Die meiste plattformabhängige Funktionalität liegt sowieso bei der Applikation, die die Bibliothek benutzt.

Da die Weiterentwicklung von DVB nicht stehen bleibt, ist auch anzunehmen, daß zukünftig neue Tabellen- und Descriptortypen definiert werden. Diese sollen natürlich so einfach wie möglich in die Bibliothek integriert werden können. Daß jede Tabelle und Descriptor durch eine eigene Klasse und damit auch Datei beschrieben werden, erleichtert dies. Allerdings müssen auch die darauf zugreifenden Klassen manuel angepaßt werden, besonders die Klassen der SectionLayer und der Controller. Hier betrifft es die Funktionalität, um neu empfangene Tabellen richtig in die interne Datenstruktur einzuordnen.

Die Klassen der ApplicationLayer greifen auf die gesammelten Tabellendaten zu und verarbeiten sie weiter. Diese Module sind voneinander unabhänging, daher können sie ohne Seiteneffekte hinzugefügt oder entfernt werden. Sie benötigen für ihre Arbeit lediglich Zugriff auf die Tabellenlisten. Alle diese Klassen erben von einer gemeinsamen Basisklasse, in der einige Attibute und Methoden bereits implementiert sind und einige abstrakte noch durch die Kindklassen implementiert werden müssen. Auf diese Weise ist sichergestellt, daß ein Zugriff möglich ist.

Eine Konfiguration zur Laufzeit ist nur in ganz begrenztem Maße notwendig. Dazu gehört die Möglichkeit, die internen Datenstrukturen und Puffer löschen zu können und die Anzahl der gleichzeitig offenen Filter auf die PMT beeinflussen zu können. Dazu kommen einige Statusabfragen, um zum Beispiel die Anzahl offener Filter zu erfahren.

Kapitel 4

Implementierung

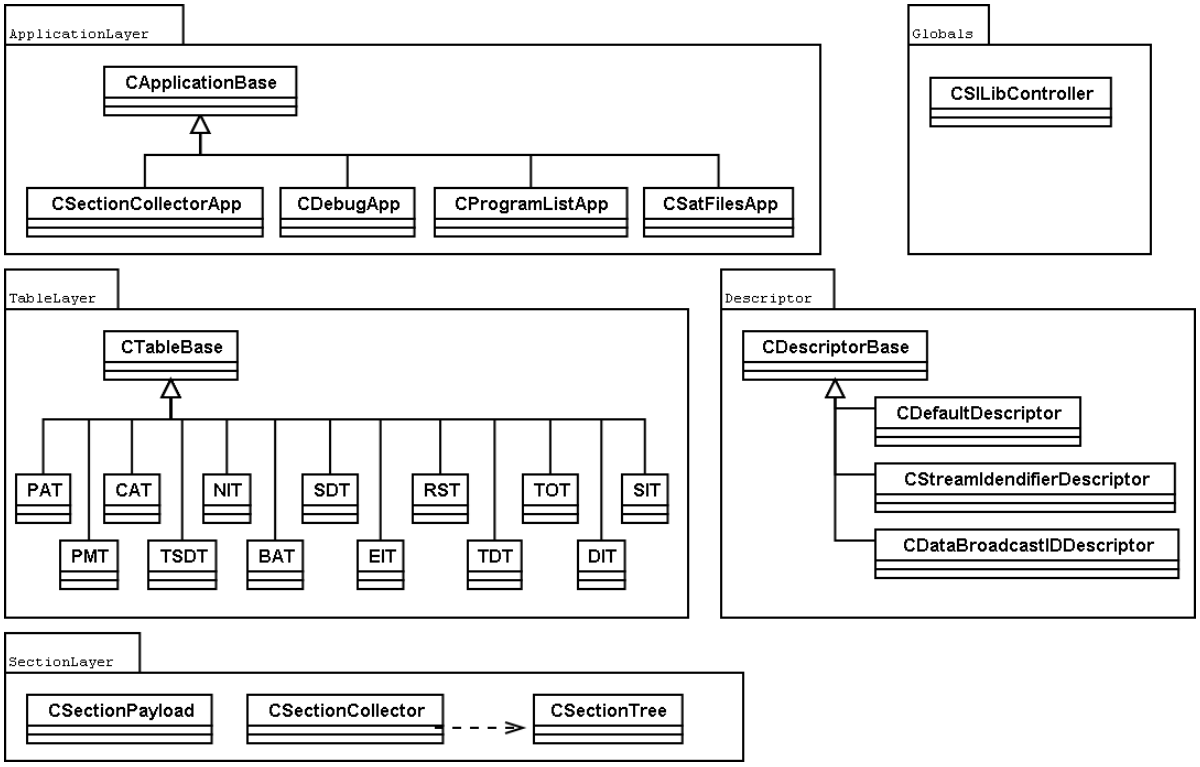

Abbildung 4.1: Klassendiagramm der Bibliothek

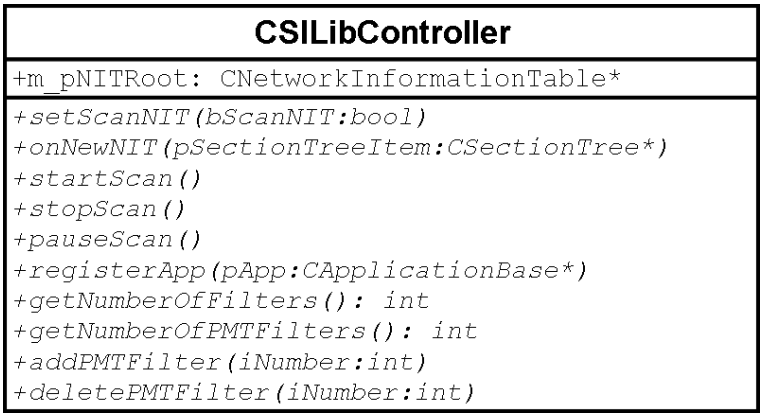

Abbildung 4.2: Öffentliche Attribute und Methoden der Klasse CSILibController

4.1 Controller

4.2 SectionLayer

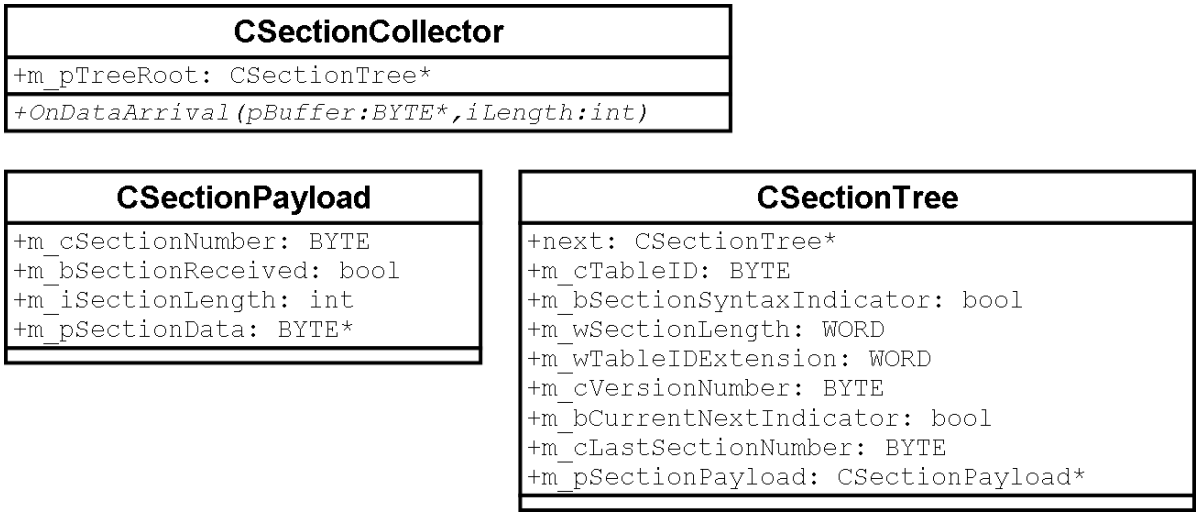

Abbildung 4.3: Detailierte Klassenstruktur der SectionLayer

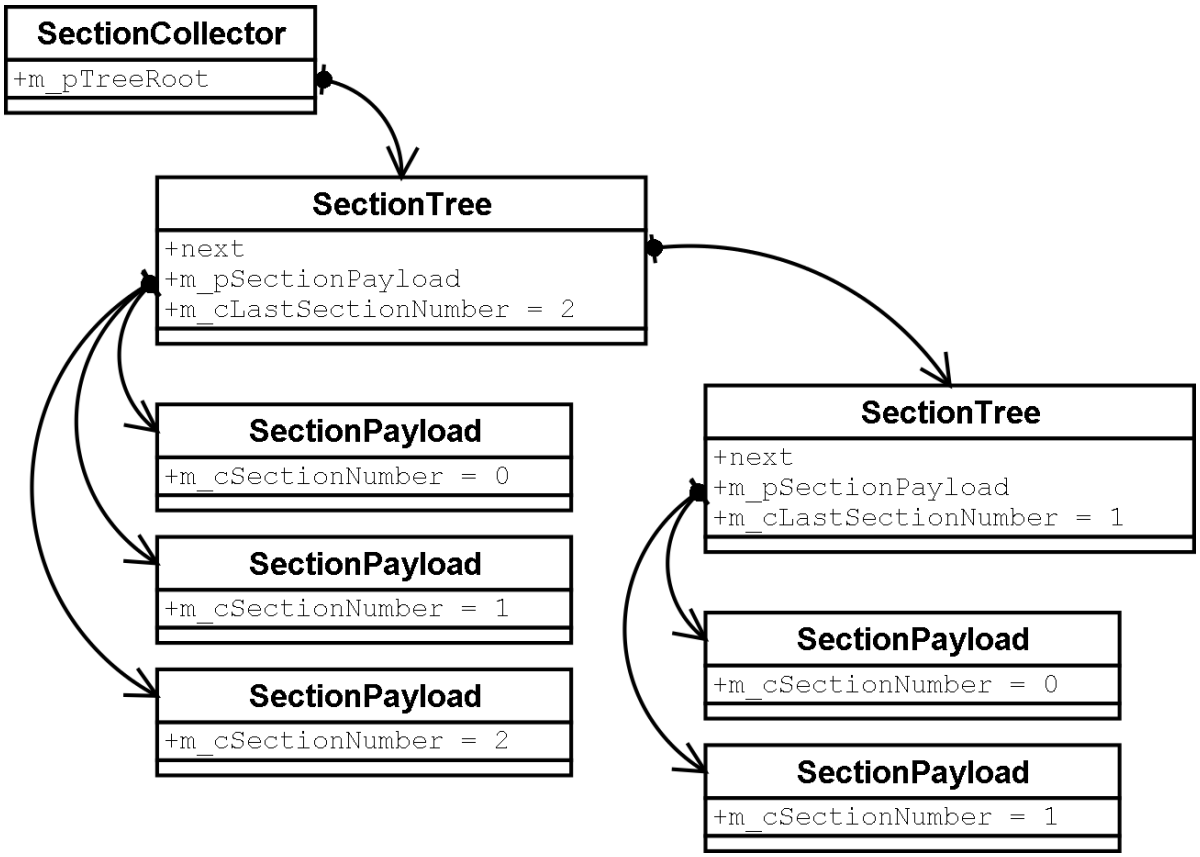

Abbildung 4.4: Detailierte Klassenstruktur der SectionLayer

4.3 TableLayer und Descriptors

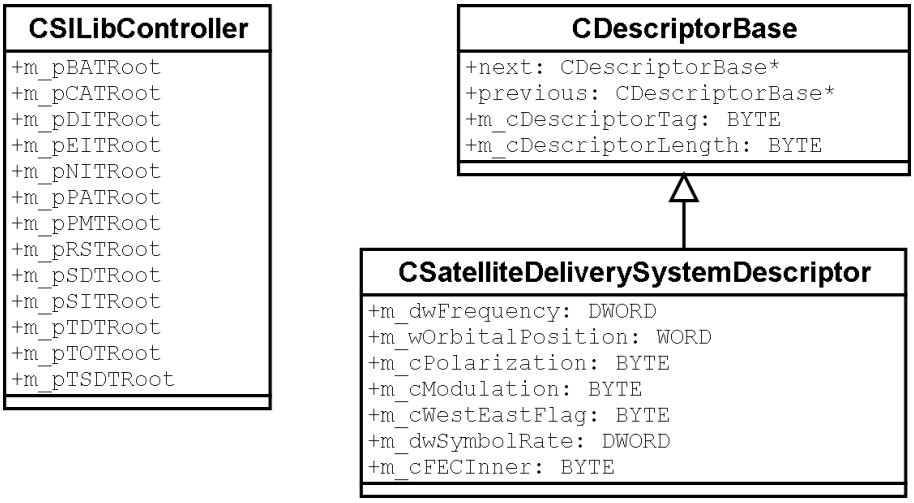

Abbildung 4.5: Detailierter Ausschnitt aus dem Klassendiagramm, Teil 1

4.4 ApplicationLayer

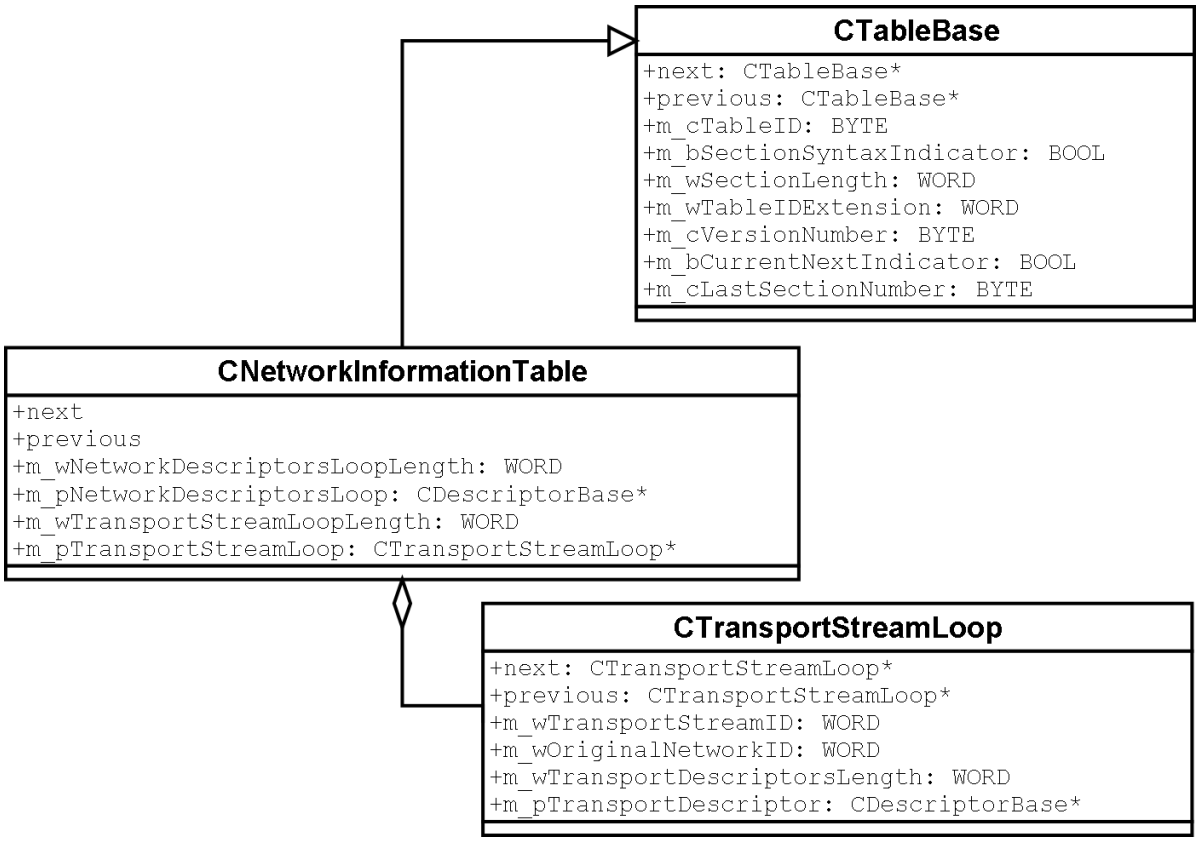

Abbildung 4.6: Detailierter Ausschnitt aus dem Klassendiagramm, Teil 2

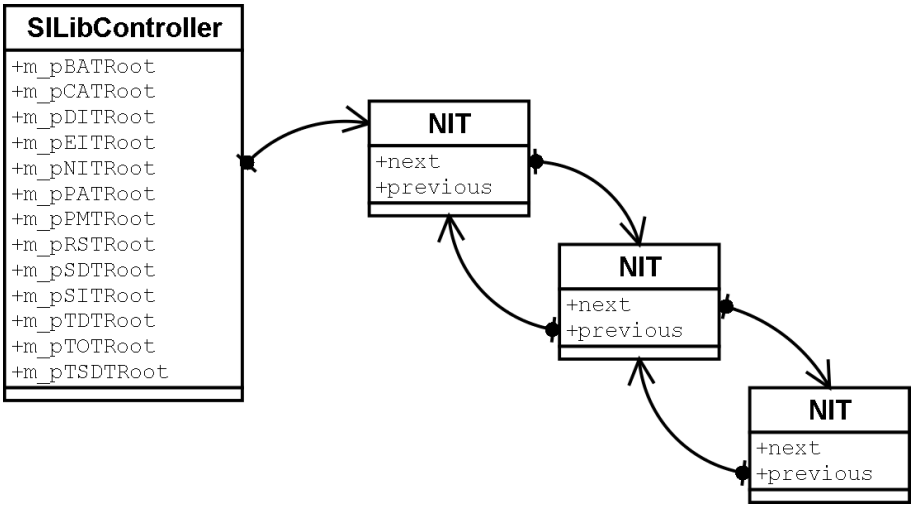

Abbildung 4.7: Liste gespeicherter Tabellen, Beispiel NIT

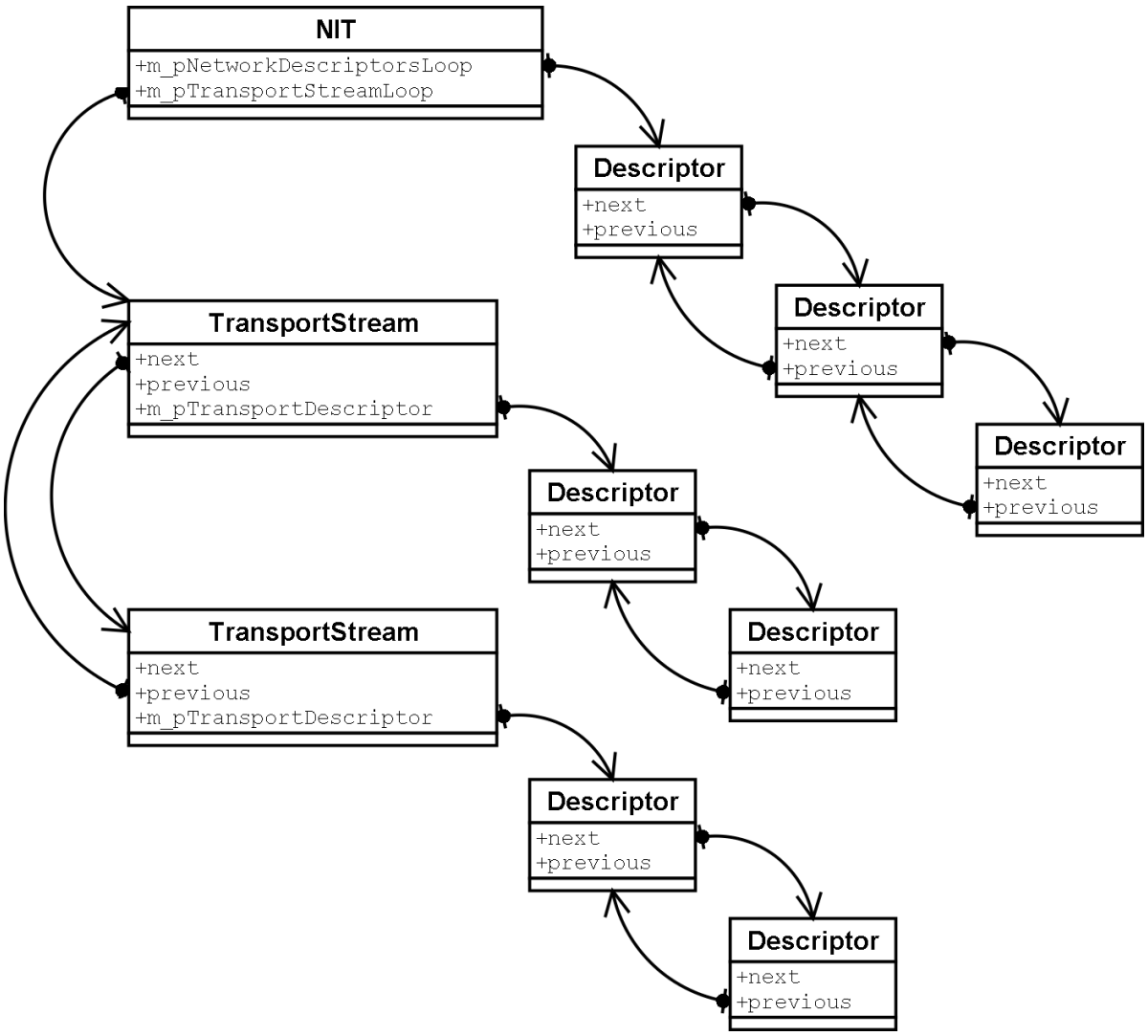

Abbildung 4.8: Innere Datenstruktur und Descriptorloop, Beispiel NIT

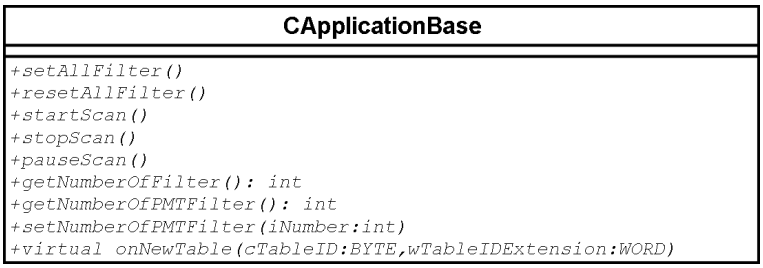

Abbildung 4.9: Öffentliche Methoden der Klasse CApplicationBase

Abbildung 4.10: Öffentliche Methoden der Klasse CProgramListApp

```
typedef struct PARAM_PROGRAM
{
TYPE_PROGRAM eType;
WORD wTransportStreamID;
WORD wPMT_PID;
WORD wNetworkID;
WORD wOrNetworkID;
WORD wServiceID;
WORD wBouquetID;
WORD wVideoPID;
WORD wAudioPID;
WORD wAC3_PID;
WORD wTeleTextPID;
WORD wMHEG5_PID;
WORD wLCN_PID;
WORD wSubTitlePID;
BOOL bCommonAccess;
char szServiceName[STR_PARAM_LEN];
char szProviderName[STR_PARAM_LEN];
char szBouquetName[STR_PARAM_LEN];
}
PARAM_PROGRAM , *PPARAM_PROGRAM ;
```

Abbildung 4.11: Übergabestruktur für die Daten des Suchlaufs

Kapitel 5

Zusammenfassung und Ausblick

Thesen

1. Ein System, um die SI auszuwerten muss von Anfang an flexibel und erweiterbar ausgelegt sein. Das DVB-System ist einem ständigen Wandel unterzogen, dem sich die Empfänger anpassen müssen.
2. Die Spezifikationen der ETSI und ISO beschreiben lediglich ein Grundsystem. Es wird erst durch einige firmenspezifische proprietäre Erweiterungen vollständig.
3.
4.
5.
6.

Andreas Schrankel Erfurt, 22.10.2003

Literaturverzeichnis

[1] ISO/IEC 13818-1. *Generic coding of moving pictures and associated audio systems*. International Organisation for Standardisation, 1995.

[2] ETSI TR 101 162. *Allocation of Service Information (SI) and Data Broadcasting Codes codes for DVB systems*. European Telecommunications Standards Institute, 2001.

[3] ETSI TR 101 211. *Guidelines on implementation and usage of Service Information (SI)*. European Telecommunications Standards Institute, 2000.

[4] ETSI EN 300 468. *Specification for Service Information (SI) in DVB Systems*. European Telecommunications Standards Institute, 2000.

[5] Frank Köthe. *Entwicklung einer DVB-PCI-Empfängerkarte mit Rückkanalmodem*. Diplomarbeit, Technische Universität Ilmenau, 1998.

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

Erklärung

Hiermit erkläre ich, daß diese Arbeit selbstständig durchgeführt und abgefaßt habe. Literatur und Hilfsmittel, die von mir benutzt wurden, sind als solche gekennzeichnet.

Andreas Schrankel Erfurt, 22.10.2003