View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Diposit Digital de Documents de la UAB

Projecte Fi de Carrera

Enginyeria de Telecomunicació

Automatic DVB signal analyser

Rosa Mª González Luque

Director: Eduardo César Galobares

Arquitectura de Computadors i Sistemes Operatius

Escola Tècnica Superior d'Enginyeria (ETSE)
Universitat Autònoma de Barcelona (UAB)

Setembre 2009

El tribunal d'avaluació d'aquest Projecte Fi de Carrera, reunit el dia ....., ha acordat concedir la següent qualificació:

President: Nom del president del tribunal

Vocal: Nom del vocal del tribunal

Secretari: Nom del secretari del tribunal

El sotasignant, Eduardo César Galobares, Professor de l'Escola Tècnica Superior d'Enginyeria (ETSE) de la Universitat Autònoma de Barcelona (UAB),

CERTIFICA:

Que el projecte presentat en aquesta memòria de Projecte Fi de Carrera ha estat realitzat sota la seva direcció per l'alumna Rosa Mª González Luque.

I, perquè consti a tots els efectes, signa el present certificat.

Bellaterra, 7 de Setembre de 2009.

Signatura: Eduardo César Galobares

Index

Introduction

This document describes the development and features of an application based in C++ that analyses, manages, and records digital television signals. It creates reports from Digital Video Broadcasting (DVB) information by an automatic analysis of digital television signal files.

As a complementary functionality, the program can also record air signal by means of an extern demodulator and can be executed by command line; this last point will be useful to automate the process of analysis, control, storage and search of all the files with the objective of controlling all European digital television broadcastings. Finally, this application tries to approach the DVB information analysis to those users that are not experts in this subject.

From digital television introduction, controlling all European digital television broadcastings became a necessity to avoid, prevent or correct problems in receivers, to find and detect errors in broadcastings or to have situations out of the scope of the standard. Consequently, the necessity of an automatic analysis and control process of these signals appears. It is worth noticing that this task is very thought if no automation is provided. Mainly due to the amount of available information which is in the range of terabytes.

An application development is a mental, creative and human process from the need to solve a problem. Software development is a process of knowledge transformation.

For those reasons, this document wants to show this transformation and it begins explaining several concepts that are needed to understand the environment of the application. First, basic concepts of digital television are introduced; this information is focused in the information that this project covers. This introduction also includes detailed information from the scope of the application to knowledge and availability for those that will use the system or want to know more details.

Following this introduction, the essay focuses at explaining all the process of software development with each stage done and the information related to the implemented application. This application has been created as an internal project inside FTVE Software Group of Sony in Viladecavalls. Consequently, in some cases some of the points that describe a project could seem forgotten or avoided but it should be noticed that some stages of a generic project development are not needed when the project is an internal one.

Finally, we conclude with software development achievement conclusions, possible improvements, and future prospects for this project that could lead to new applications with improved management of digital signals capabilities.

Chapter 1: Theoretical foundations

This essay deals with Digital Television (DTV) and it is focused on the information that digital signal carries in order to make feasible the successful decodification in receivers. For this reason, this first chapter tries to define the basics of DTV and its standards in order to facilitate the explanation and understanding of the rest of this essay.

1.1. Digital Television

1.1.1. Why Digital Television?

Television (TV) is one of the most important media and it is nearly indispensable in our homes. TV is an entertainment media and it also informs us about news. Many efforts have been done in TV to make it more available, attractive and addictive by means of improved contents, audience participation, delayed emissions, etc.

Consequently, as for other devices such as mobiles, the change that TV could not avoid was digitalization.

Analogue TV became too limited and this made visible the necessity of TV evolution to the Digital TV. Above all Analogue TV problems users could find transmission problems; the degradation of analogue signal was very high due to propagation and obstacles. Another handicap of Analogue TV was the low data transmission capacity, it doesn't allow sending better applications like enhanced teletext or more programs inside the same bandwidth like Digital TV does.

Digital TV solves some of the problems of Analogue TV such as broadcasting problems. It also optimizes spectrum band by means of multiplexing programs in the same bandwidth. This fact will allow creating Single Frequency Networks (SFN)[1], a big improvement in data transmissions by providers' side.

Finally, it is necessary to say that DTV is having several difficulties and troubles in its introduction and implementation. Mainly there are economical implications; DTV is a new opportunity in audiovisual world. Producer manufacturers, broadcasters, public or private ones, receivers' manufacturers can take profit of this new technology that will affect nearly all users in the world or has affected already. Users, on the other hand, shall adapt their homes to be able to receive DTV.

[1] SFN are used to cover all the territory (country) with the same frequency to one service. On the other hand, we found Multiple Frequency Network (MFN) that are using different frequencies inside the territory, combined to avoid overlap of frequencies in one zone.

1.1.2. DVB standard

Nowadays, Digital TV is a fact in many countries. But it would be useful to know that not all countries used the same standards. There are four main standards in the whole world: ATSC, DVB, ISDB and DMB. There is a summary of the last status of Digital TV in Figure 1.

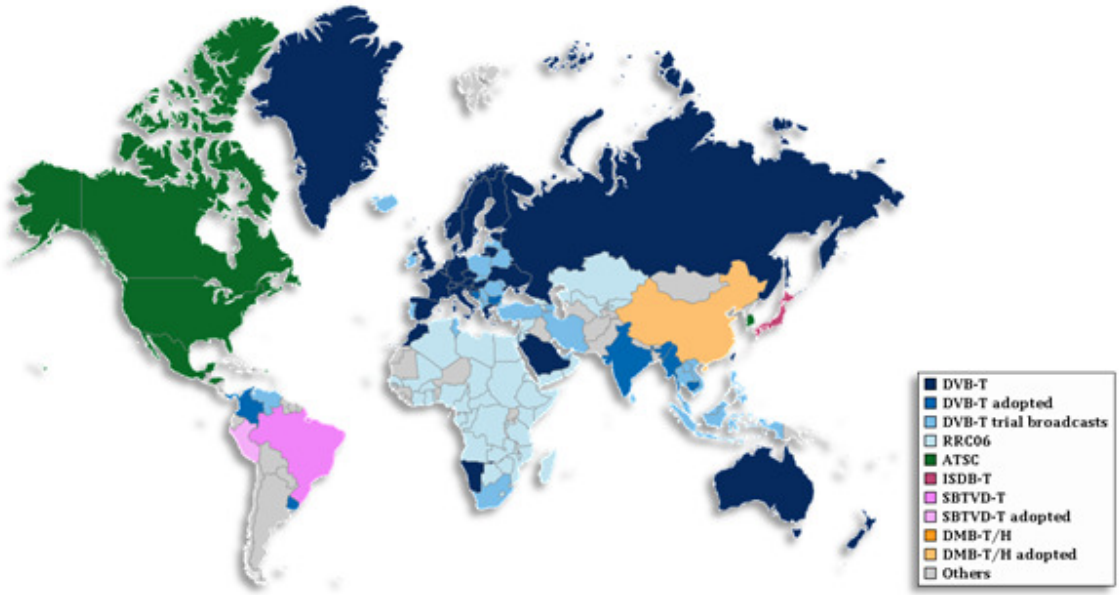


Figure 1 – Standards adaptation around the world[2]

Advanced Television Systems Committee (ATSC) is used mainly in the USA. This standard is focused in HDTV (High Definition Television) and without portable receivers.

Digital Video Broadcasting (DVB) is adopted in more than 50 countries like Australia, Malaysia, New Zealand or most of the countries in the European Union. This standard takes HDTV and SDTV (Standard Definition Television) and there are different platforms like Cable, Satellite, Terrestrial or Handheld.

Integrated Services Digital Broadcasting (ISDB) is established in Japan and it tries to combine the advantages of ATSC and DVB.

DMB (Digital Multimedia Broadcasting) was developed in South Korea and it will be used by the countries near to South Korea, like China. This standard is focused in handheld transmission though it is used for terrestrial broadcast too.

It is necessary to say that this document will focus in DVB standard because it is used in nearly the whole Europe. For this reason, more details about this standard will be introduced in the following sections.

[2] Illustration and information from http://www.dtvstatus.net

Digital Video Broadcasting (DVB) is a suite of international standards for digital television. DVB standards are maintained by the DVB Project (http://www.dvb.org/), an industry-led consortium of over 270 broadcasters, manufacturers, network operators, software developers, regulatory bodies and others in over 35 countries committed to designing open technical standards for the global delivery of digital television and data services.

The standards are published by a Joint Technical Committee (JTC) of European Telecommunications Standards Institute (ETSI), European Committee for Electrotechnical Standardization (CENELEC) and European Broadcasting Union (EBU). Many aspects of DVB are patented, including elements of the MPEG (Moving Pictures Experts Group) video coding and audio coding.

1.2. Technical details

However we have previously seen that DVB has several broadcast systems (satellite, cable, terrestrial...). Those differences in this essay are meaningless due to it is focused in the System Information and all systems are based in the source codification adapted from MPEG-2 which generates Transport Stream (TS).

As we have mentioned before, the best advantage of Digital TV is the opportunity of compressing the signal in a big rate compared to Analogue TV to obtain nearly the same quality. This operation is done inside "Compression layer" which is not deeply explained in this essay because it has no useful application to this introduction.

In System layer, MPEG-2 system proportionate packets with compressed data. If this data is from a unique program, a system clock is included and then a Program Stream is created. If the packets are from several programs, the correct Program Specific Information (PSI) is included and then a Transport Stream is obtained. The summary of this system is in Figure 2.

1.2.1. Compression layer

The Compression layer is in charge of doing pure MPEG operations. Like reducing video/audio information that is not useful for human eye/ear, taking profit of human eye/ear faults for colour detection in front of intensity receivers or audio sensitivity. With this action video/audio load is reduced to a half of the information that would be sent otherwise. This point is useful to reduce the necessary memory for the systems involved in this process too.

After this, Compression layer for video also uses spacial/timing (intraframe/interframe) redundancy. Intraframe compression is done by means of Discrete Cosine Transform (DCT) and with the results methods like (quantification, zigzag scanning, entropy codification...) are applied to minimize even more the load to be sent. Interframe compression is applied between consecutive images, the different

compression level between frames and no-linear transmission of images in the broadcast helps in this process too.

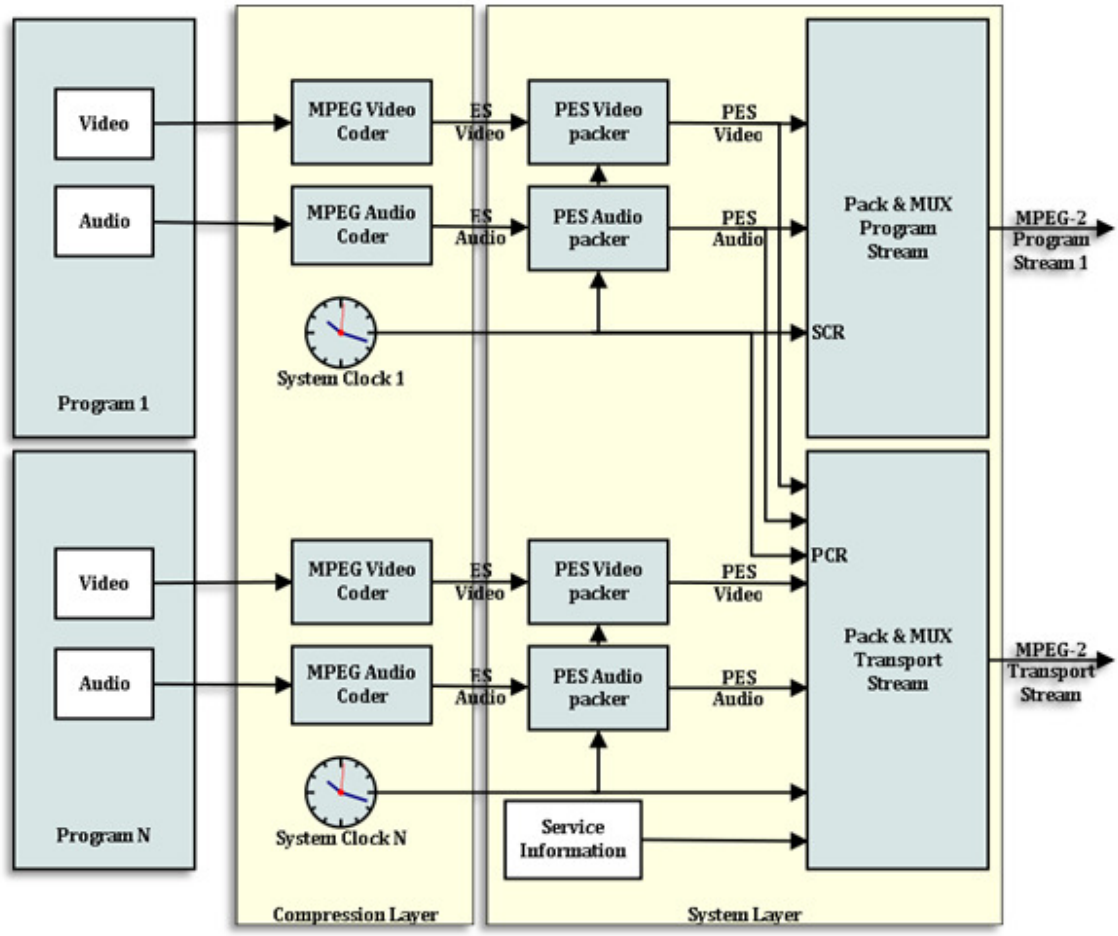


Figure 2 – Program and Transport Streams creation

1.2.2. System layer

System layer is in charge of including information from TS, synchronization and decoding data. In this layer, the system MPEG-2 defines how the packets are made and PSI/SI information that includes information about how to decode these packets to recover the original data.

1.2.2.1. MPEG-2 system

First, it is necessary to define some concepts that will be used in this explanation. Elementary Streams (ES) are the result of consecutive frames compression and codification. Those ES are made up of Access units (AU) that are coded Presentation Units (PU), video and audio frames.

ES don't have timing nor synchronization information to help in the sequence decodification. For this reason, Packet Elementary Stream (PES) is created. PES is an

ES with a header of 32 bits, with information such as the type of data (video, audio or data), the number of ES that corresponds in case that this ES was greater than the PES payload size, ES length and the necessary intermediary memory for decodification. It also can contain a Decoding time Stamp (DTS) to the first packet AU and/or or a Presentation Time Stamp (PTS)… After having PES, they are packed with headers that contain time, synchronization, and byte flow information by means of System Clock Reference (SCR).

TS are made up of PES of the different programs that will be sent in the same bandwidth (multiplex). TS are not only from exclusively one TV program, TS combine several programs inside its streaming. But TS is also packetized and inside each TS packet there must be only one PES which first byte shall be the first byte of the TS packet payload, with stuffing bytes in case it is necessary.

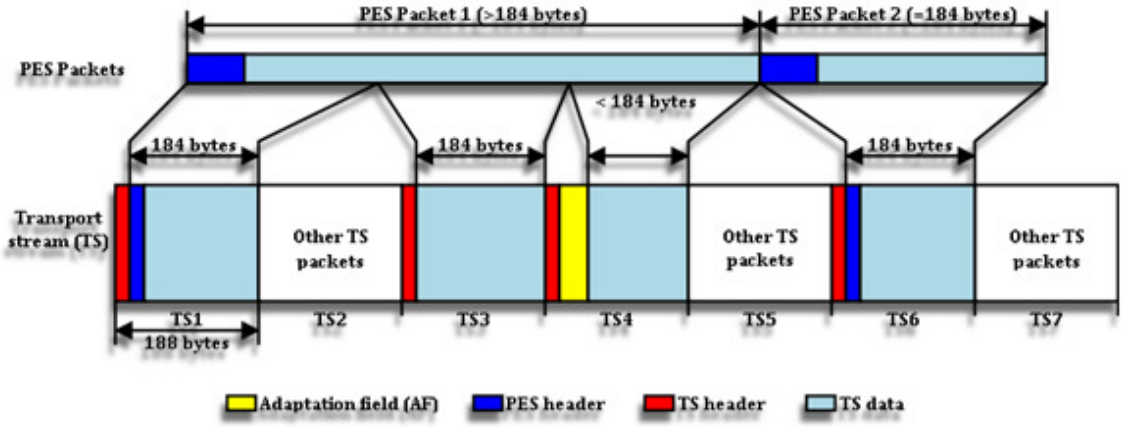


Figure 3 – Transport stream organization

TS packet length will be of 188 bytes, 4 of them are defined as header and the rest, as payload. It is also possible to add an adaptation field if it is necessary. Each TS packet is defined by a Packet Identifier (PID) that indicates which ES is the transported in each TS packet. PID is a field of 13 bits and it shall be unique to allow the correct demultiplexation, because from this identification it is possible to know which TS packets are from the selected program in the receiver. An example of TS organization is described in Figure 3.

The TS packet header has some other fields which give more information about transport and multiplexation. They are the transport priority, the continuity counter (that indicates the order of TS packets of the same PID), and adaptation field control flag (to know if there is adaptation field or not).

Finally, TS packets created of the different PES (of different data types) are multiplexed to generate a TS. From this point, bit rate is fixed and the last step is the adaptation of TS to the network and the transmission channel to be received at each home.

1.2.2.2. PSI/SI information

There are two types of tables inside a TS:

- Those specified by MPEG PSI to bind all the elements of a transport stream together.
- Those specified by DVB SI (Service Information) to bind a number of services and transport streams together in order to provide a multi-channel broadcasting environment.

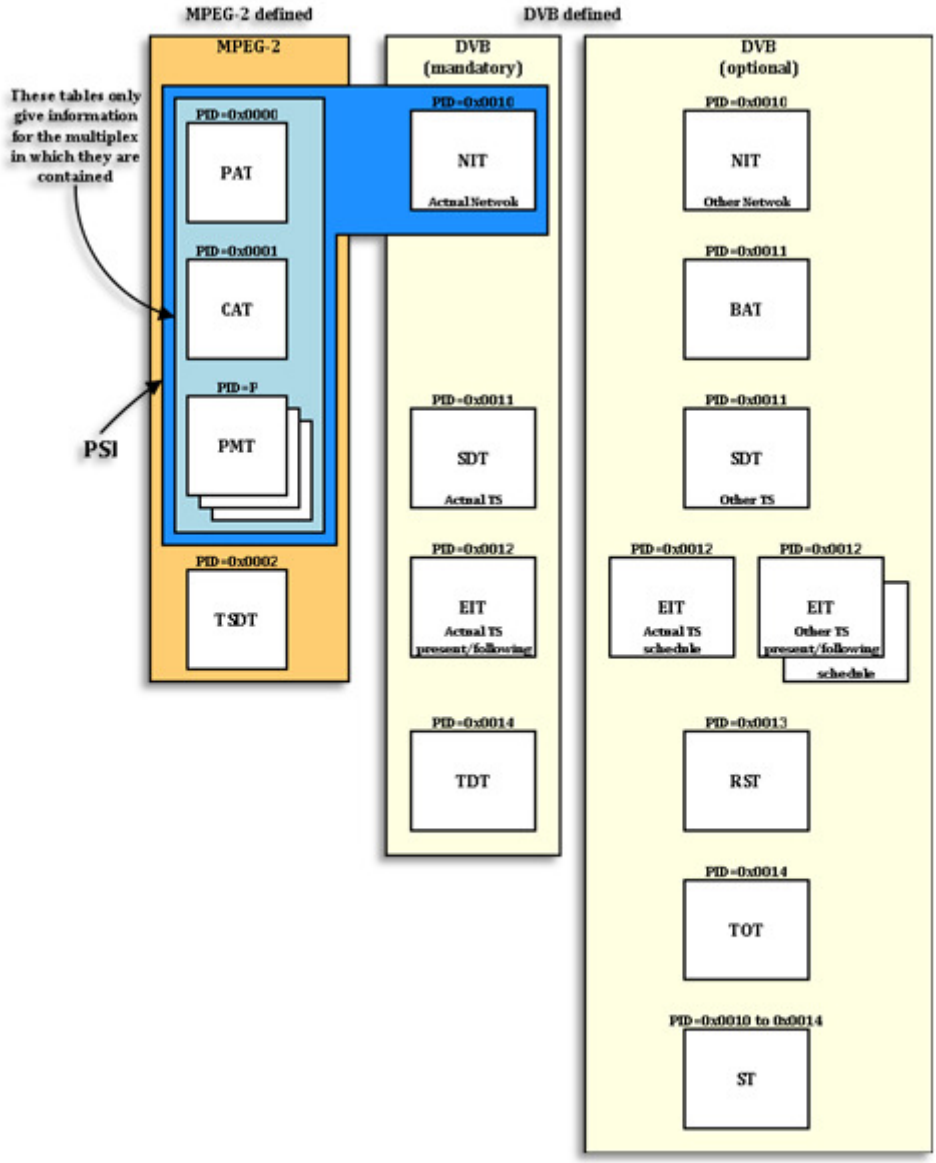


Figure 4 – General organization of the Service Information (SI)

SI data for that the receivers can automatically configure itself for the selected service is mostly specified within ISO/IEC 13818-1 as Program Specific Information (PSI). The document EN 300 468 specifies the Service Information (SI) data which forms a part of DVB bit streams, in order that the user can be provided with information to assist in selection of services and/or events within the bit stream.

SI tables' classification is summarised in Figure 4.

Program Specific Information

PSI data provides information to enable automatic configuration of the receiver to demultiplex and decode the various streams of programs within the multiplex. The PSI data (specified in ISO/IEC 13818-1) is structured as four types of table. From those tables, three (PAT, PMT and CAT) are specified by MPEG-2 and one (NIT) by DVB-SI.

- Program Association Table (PAT): PAT has a list of each service in the multiplex; the PAT indicates the location (PID) values of the TS packets of the corresponding Program Map Table (PMT) for each service. It also gives the location of the Network Information Table (NIT). Additionally, the PAT is always on PID 0, and there is only one inside TS.
- Conditional Access Table (CAT): CAT provides information on the CA systems used in the multiplex; the information is private and dependent on the CA system. It controls the scrambling of a service by associating one or more CA systems with their EMM (Entitlement Management Message) stream and any other extra data that may be required.
- Program Map Table (PMT): PMT identifies and indicates the locations of the Elementary Streams that make up each service and the location of the Program Clock Reference (PCR) fields for a service. There is one PMT per service, but there may be more that one PMT on the same PID.
- Network Information Table (NIT): location of the NIT is defined in EN 300 468 standard. It is intended to provide information about the physical network.

Service information

The document EN 300 468 specifies additional data which complements the PSI by providing data to aid automatic tuning of receivers, and additional information intended for display to the user. Rules of operation for the implementation of the document EN 300 468 are specified in TR 101 211.

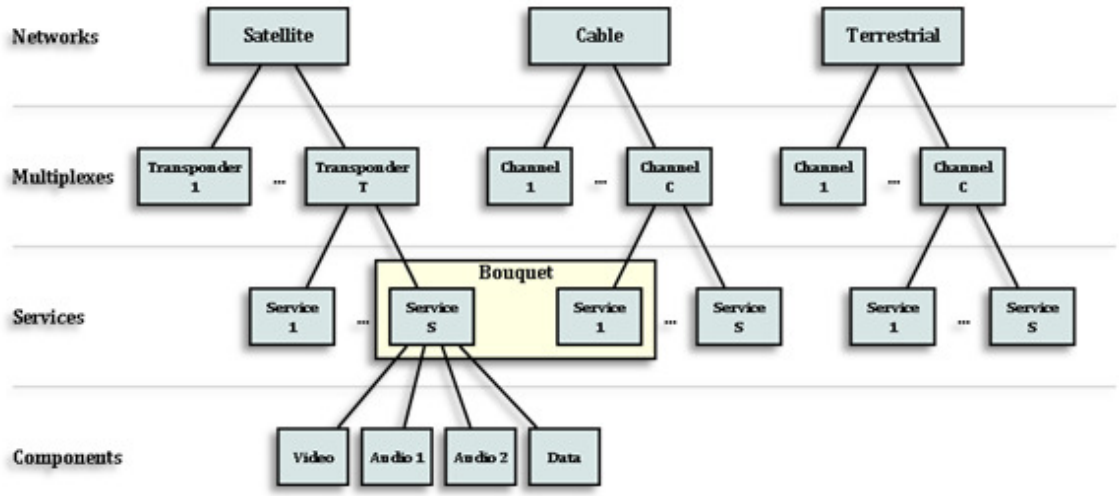


Figure 5 – Digital broadcasting, service delivery model

Figure 5 illustrates some of the concepts used in this standard.

In addition to the PSI, data is needed to provide identification of services and events for the user. In contrast with the PAT, CAT, and PMT of the PSI, which give information only for the multiplex in which they are contained (the actual multiplex), the additional information defined within the document EN 300 468 can also provide information on services and events carried by different multiplexes, and even on other networks. This data is structured as nine tables:

- Bouquet Association Table (BAT): BAT provides information regarding bouquets. As well as giving the name of the bouquet, it provides a list of services for each bouquet.
- Service Description Table (SDT): SDT contains data describing the services in the system e.g. names of services, the service provider, etc.
- Event Information Table (EIT): EIT contains data concerning events or programs such as event name, start time, duration, etc.; the use of different descriptors allows the transmission of different kinds of event information e.g. for different service types.
- Running Status Table (RST): RST gives the status of an event (running/not running). The RST updates this information and it allows timely automatic switching to events.
- Time and Date Table (TDT): TDT gives information relating to the present time and date. This information is given in a separate table due to the frequent updating of this information.
- Time Offset Table (TOT): TOT gives information relating to the present time and date and local time offset. This information is given in a separate table due to the frequent updating of the time information.
- Stuffing Table (ST): ST is used to invalidate existing sections, for example at delivery system boundaries.
- Selection Information Table (SIT): SIT is used only in "partial" (i.e. recorded) bit streams. It carries a summary of the SI information required to describe the streams in the partial bit stream.
- Discontinuity Information Table (DIT): DIT is used only in "partial" (i.e. recorded) bit streams. It is inserted where the SI information in the partial bit stream may be discontinuous.

Table	PID value
PAT	0x0000
CAT	0x0001
TSDT	0x0002
reserved	0x0003 to 0x000F
NIT, ST	0x0010
SDT, BAT, ST	0x0011
EIT, ST CIT (TS 102 323)	0x0012
RST, ST	0x0013
TDT, TOT, ST	0x0014
network synchronization	0x0015
RNT (TS 102 323)	0x0016
reserved for future use	0x0017 to 0x001B
in band signalling	0x001C
measurement	0x001D
DIT	0x001E
SIT	0x001F

Table 1 – PID allocation for SI

IEC 61883 describes methods for delivering TS over the IEEE 1394.1 to receivers. One likely source for this data is a digital storage device. In this certain cases TSs can be "incomplete", thus not conforming to the normal broadcast specifications. These "partial" TS represent a subset of the data streams in the original TS. They may also be "discontinuous" - that is there may be changes in the TS or the subset of the TS presented and there may be temporal discontinuities. Storage Media Interoperability (SMI) describes the SI and PSI required in the delivered data in these cases.

The SMI tables are encoded using the private section syntax defined in ISO/IEC 13818-1. The SIT may be up to 4 096 bytes long.

Partial TS shall not carry any SI tables other than the Selection Information Table (SIT) and Discontinuity Information Table (DIT). The PSI shall be restricted to the PAT and PMT instances required to correctly describe the streams within the partial TS.

The presence of the SIT in a bit stream identifies the bit stream as a partial TS coming from a digital interface. In this case the receiver should not expect the SI information required in a broadcast TS and should instead rely on that carried by the SIT. The SIT contains a summary of all relevant SI information contained in the broadcast stream. The DIT shall be inserted at transition points where SI information is discontinuous. The use of the SIT and DIT is restricted to partial TSs; they shall not be used in broadcasts.

Tables and descriptors definition

Once all tables are explained, it is useful to have their definition and to know how they are organized inside TS for this project. Tables' definition can be found in Annexes.

As it is defined in all tables, they can carry descriptors that give more information about TS. For this reason the important ones in this essay are detailed in this section. Also general information is remarked. Most descriptors are defined in EN 300 468 but some of them are not; it is indicated in the description.

All descriptors transport these two semantics:

- Descriptor_tag: identifies each descriptor. Those are values are described in ISO/IEC 13818-1. The values of descriptor_tag are defined in the next table.
- Descriptor_length: specifies the total number of bytes of data portion of the descriptor.

Identification and location of them is described in following table. The definitions of important ones are in Annexes.

Descriptor	Tag value	NIT	BAT	SDT	EIT	TOT	PMT	SIT (see note 1)
network_name_descriptor	0x40	*	-	-	-	-	-	-
service_list_descriptor	0x41	*	*	-	-	-	-	-
stuffing_descriptor	0x42	*	*	*	*	-	-	*
satellite_delivery_system_descriptor	0x43	*	-	-	-	-	-	-
cable_delivery_system_descriptor	0x44	*	-	-	-	-	-	-
VBI_data_descriptor	0x45	-	-	-	-	-	*	-
VBI_teletext_descriptor	0x46	-	-	-	-	-	*	-
bouquet_name_descriptor	0x47	-	*	*	-	-	-	*
service_descriptor	0x48	-	-	*	-	-	-	*
country_availability_descriptor	0x49	-	*	*	-	-	-	*
linkage_descriptor	0x4A	*	*	*	*	-	-	*
NVOD_reference_descriptor	0x4B	-	-	*	-	-	-	*
time_shifted_service_descriptor	0x4C	-	-	*	-	-	-	*
short_event_descriptor	0x4D	-	-	-	*	-	-	*
extended_event_descriptor	0x4E	-	-	-	*	-	-	*
time_shifted_event_descriptor	0x4F	-	-	-	*	-	-	*
component_descriptor	0x50	-	-	*	*	-	-	*
mosaic_descriptor	0x51	-	-	*	-	-	*	*
stream_identifier_descriptor	0x52	-	-	-	-	-	*	-
CA_identifier_descriptor	0x53	-	*	*	*	-	-	*
content_descriptor	0x54	-	-	-	*	-	-	*
parental_rating_descriptor	0x55	-	-	-	*	-	-	*
teletext_descriptor	0x56	-	-	-	-	-	*	-
telephone_descriptor	0x57	-	-	*	*	-	-	*
local_time_offset_descriptor	0x58	-	-	-	-	*	-	-
subtitling_descriptor	0x59	-	-	-	-	-	*	-
terrestrial_delivery_system_descriptor	0x5A	*	-	-	-	-	-	-
multilingual_network_name_descriptor	0x5B	*	-	-	-	-	-	-
multilingual_bouquet_name_descriptor	0x5C	-	*	-	-	-	-	-
multilingual_service_name_descriptor	0x5D	-	-	*	-	-	-	*
multilingual_component_descriptor	0x5E	-	-	-	*	-	-	*
private_data_specifier_descriptor	0x5F	*	*	*	*	-	*	*
service_move_descriptor	0x60	-	-	-	-	-	*	-
short_smoothing_buffer_descriptor	0x61	-	-	-	*	-	-	*
frequency_list_descriptor	0x62	*	-	-	-	-	-	-
partial_transport_stream_descriptor (see note 1)	0x63	-	-	-	-	-	-	*
data_broadcast_descriptor	0x64	-	-	*	*	-	-	*
scrambling_descriptor	0x65	-	-	-	-	-	*	-
data_broadcast_id_descriptor	0x66	-	-	-	-	-	*	-
transport_stream_descriptor (see note 2)	0x67	-	-	-	-	-	-	-

Descriptor	Tag value	NIT	BAT	SDT	EIT	TOT	PMT	SIT (see note 1)
DSNG_descriptor (see note 2)	0x68	-	-	-	-	-	-	-
PDC_descriptor	0x69	-	-	-	*	-	-	-
AC-3_descriptor	0x6A	-	-	-	-	-	*	-
ancillary_data_descriptor	0x6B	-	-	-	-	-	*	-
cell_list_descriptor	0x6C	*	-	-	-	-	-	-
cell_frequency_link_descriptor	0x6D	*	-	-	-	-	-	-
announcement_support_descriptor	0x6E	-	-	*	-	-	-	-
application_signalling_descriptor	0x6F	-	-	-	-	-	*	-
adaptation_field_data_descriptor	0x70	-	-	-	-	-	*	-
service_identifier_descriptor	0x71	-	-	*	-	-	-	-
service_availability_descriptor	0x72	-	-	*	-	-	-	-
default_authority_descriptor	0x73	*	*	*	-	-	-	-
related_content_descriptor	0x74	-	-	-	-	-	*	-
TVA_id_descriptor	0x75	-	-	-	*	-	-	-
content_identifier_descriptor	0x76	-	-	-	*	-	-	-
time_slice_fec_identifier_descriptor (see note 3)	0x77	*	-	-	-	-	-	-
ECM_repetition_rate_descriptor	0x78	-	-	-	-	-	*	-
S2_satellite_delivery_system_descriptor	0x79	*	-	-	-	-	-	-
enhanced_AC-3_descriptor	0x7A	-	-	-	-	-	*	-
DTS descriptor	0x7B	-	-	-	-	-	*	-
AAC descriptor	0x7C	-	-	-	-	-	*	-
XAIT location descriptor	0x7D	*	*	*	*	*	*	*
FTA_content_management_descriptor	0x7E	*	-	*	*	-	-	-
extension descriptor	0x7F	*	*	*	*	*	*	*
user defined	0x80 to 0xFE							
Forbidden	0xFF							

NOTE 1: Only found in Partial Transport Streams.
NOTE 2: Only in the TSDT (Transport Streams Description Table).
NOTE 3: May also be located in the CAT (ISO/IEC 13818-1) and INT (TS 102 006).
NOTE 4: * Possible location.

Table 2 – Possible locations of descriptors

Tables organization inside TS

At the time of packing the tables, they are treated like PES inside TS. There are little differences, for example, table sections shall not begin and finish in a TS mandatorily. In case a section is started inside a packet, the Payload Unit Start Indicator (PUSI) will change to 1. The first byte of payload will be the pointer field that will say how many bytes there are until the new section.

Following, it is possible to see some rules/steps that it is necessary to perform to find the correspondent packets of each stream to tune correctly each service:

- Firstly, filter PID 0 to get all the packets of PAT
- Build PAT from its sections
- In its information is visible each available program that can be tuned
- Once a service is selected, then filter PIDs from the chosen program
- Build Program map table from its sections
- Filter the packet from PCR_PID field to get PCR information and synchronize with system clock

- If there are several PID of audio, video or data, the user should choose between them
- Filter each chosen PID and begin to decode

User can only choose all this steps from Electronic Program Guide (EPG) that is the list of all services proportionate from DVB-SI information that allow the user to navigate easily between the different services found.

Chapter 2: Doctor TS

In this section, it is intended to explain all the steps of the application development and explain what is important of the application in each step. For this reason, first, each step has an introduction on what is important and what shall be done in the stage. Next, all the theory is focused on the application developed and is detailed for this step.

2.1. Description of the problem

In this step, it is needed to describe the current situation and detect the exact problem. If this is done correctly, a good solution will be found, solving the real problem without unnecessary requirements or implementations.

With DVB introduction in television emissions, control and analysis of the broadcasted signal is needed for all European countries in this case because is the company focus. In Integrated Digital Television (IDTV) design, it is important to know the real conditions of air emission to avoid or to solve problems in TVs.

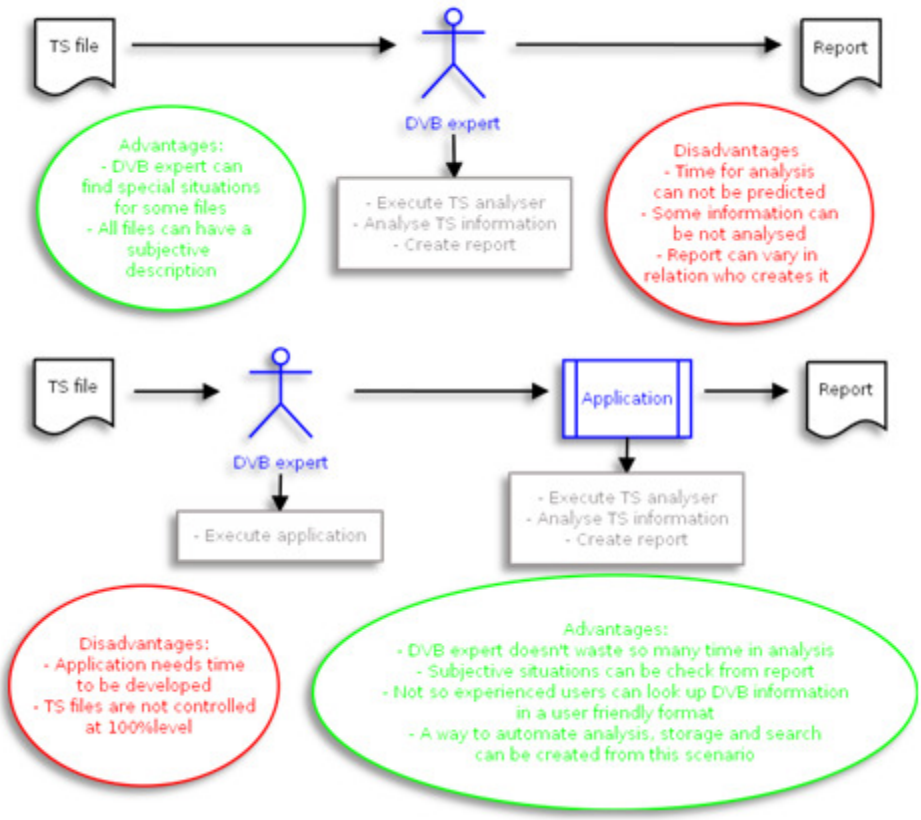


Figure 6 – Main problem and its solution representation

Controlling all DVB information and properties of all signals is a hard work and it wastes a lot of knowledge and, above all, time. For this reason an automatic tool is

needed with the intention of making faster and easier this task. In Figure 6, there is a simple diagram explaining the main problem and the solution.

In the one hand, a tool that reports automatically specified information among all the rest of the information that can be found in a Transport Stream is the first and most important objective.

On the other hand, obtain a recorder of DVB signals in order to help in activities like Field Test, market problems follow-up and broadcasters changes tracking is also desirable. Moreover, it is also useful to show DVB information in a friendly way to allow users to know signal properties without having deep SI knowledge.

Finally, it wanted to control all this information to be stored and to make it easier to search between the hundreds files and terabytes of information. So, it is required that this tool could be used from command line or scripts to make the process to control DTV information more automatic.

2.2. Requirements

Requirements capture is necessary to understand and specify what the system should do from a black box perspective. The purposes of requirements are:

- Determine exactly what the customers and users want
- Develop a contract with the customers
- Provide system developers with a better understanding of the system requirements
- Define the boundaries of (delimit) the system
- Provide a basis for planning the project
- Specify what the software product is to do
- Establish and maintain agreement with the customers and other stakeholders on what the system should do

From the point of view of this project, most of those requirements purposes are unnecessary because this project is an internal project inside the company where some costumers are users or the own developer is a user and costumer. Consequently, the requirements should specify what the software must do, delimit the system (or at least delimit what this deployment should do), and provide a basis for planning the project.

In this step it is mandatory avoid the difficulty of doing specifications that are ambiguous, inconsistent or incomplete.

2.2.1. Preliminary requirements

Several applications were used to record and analyse DVB signal so far. But the most used and able to do both operations is TSReader.

TSReader is a transport stream analyzer, decoder, recorder and stream manipulator for MPEG-2 systems. It supports DVB extensions to the base MPEG-2 specification. TSReader gives the user the overview of what is being carried inside MPEG-2 transport streams and can be very useful for finding errors or inefficiencies.

TSReader can work by command line and scripts; in this case, the solution to the problem seems easy by doing several scripts with the desired functionalities. But the problems of TSReader in the current environment are several, for example:

- The analysed information cannot be customised for the desired information; information that is the most used or looked up.
- The analysed information is huge in many cases.
- The analysed information varies in some conditions in terms of Service Information.

So, in most of the cases the problem is that analysed information needs to be analysed by an "expert" that knows what is important in SI or needs to be noticed and create a new report. This doesn't allow making this system automatic.

Other solutions can be thought, like make a SI analyser from recorded streams but this option will waste too much time; or create a parser of the information, but this option will not complete the system. For this reason the easiest way to automate the system is to develop an application that will automated the commands to TSReader to record and analyse, and then parse its information to give a report with established information and a friendly report to occasional or inexpert users.

So, the solution thought is described in Figure 7. This application is named Doctor TS (DrTS) and must do three basic functions, record (if it is needed), demultiplex, and analyse.

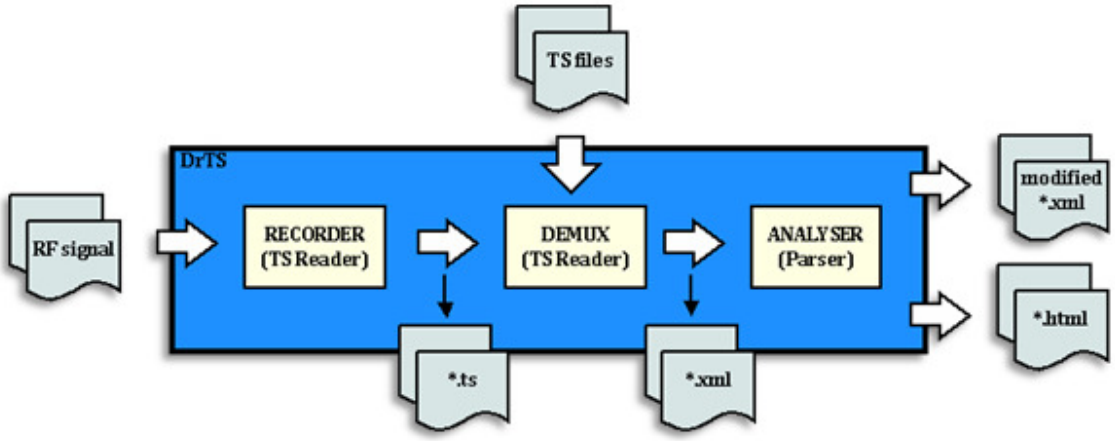


Figure 7 – Detailed functionality of the system

Other requirements are that the application will control the TSReader and parse its information in a standalone automatic test environment, should be friendly for occasional users which need it, and also compatible in a portable environment.

For occasional or inexpert users, the application should give the opportunity of obtaining a summarised report. So, the results of the execution of this tool should be reported in HTML (HyperText Markup Language) files. HTML files shall include more or less information depending on the needs of the user. Moreover, a mandatory and most important requirement, report files shall be also in XML (eXtensible Markup Language) format.

There will be the option of analyzing just one TS file or frequency, or all available DVB signals, making a scan, or an individual analysis of each multiplex.

TS file or RF signal should be demultiplexed, one or several "at the same time", controlled and managed by our application and almost transparent to the user. This part is done by TSReader that creates an XML file with the results of demultiplexing. There are two xml profiles, one with SI information except descriptors and event information and another with all the information. It is worth mentioning that DrTS is based in XML reported by TSReader, so it will be dependent of TSReader version and its changes. So, for example, EIT information is not reported by TSReader in recorded streams in some situations, this limitation will affect reports of the application that will be created.

Once the XML file is created it should be parsed and analyzed. This is done by the application. From this analysis, two files are obtained, one with friendly appearance in HTML as a report and a XML with the useful information extracted for further analysis. These results would have two profiles, basic profile, with the most important information of DVB Tables, and another complete profile, containing descriptors information in addition to basic profile.

In the next figure, a summary of the proposed requirements is shown.

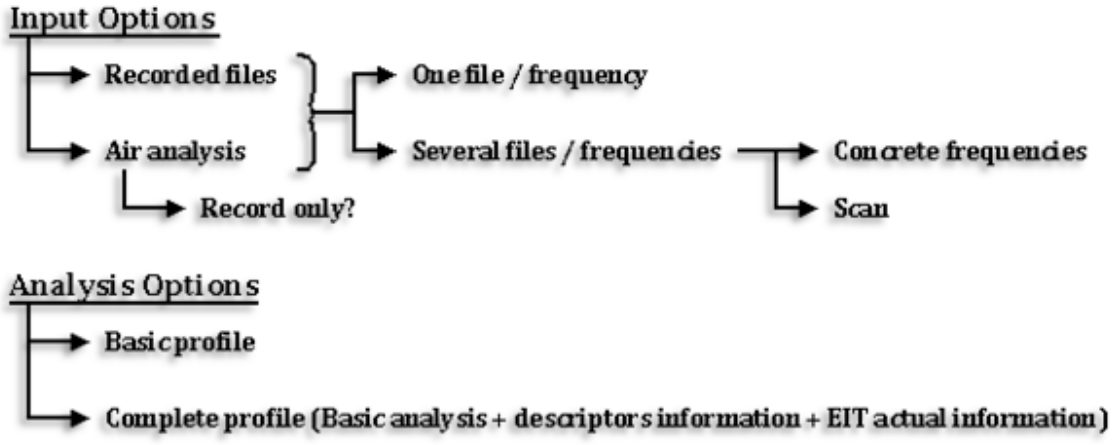


Figure 8 – Scheme of input and analysis requirements

On the other hand, the user interface shall be specified. Application control shall be possible from command line to allow the automatic analysis for other applications by scripts or other methods. In addition, a Windows Graphical User Interface (GUI) will be created to provide a user-friendly environment.

Finally, the application shall be configurable. The application configuration depends on the desired use and environment. This application is aimed at several situations and users. Consequently, the user interface shall allow configuring all the possibilities explained above.

2.2.2. Delimiting the system

In order to delimit the system in this section, the information required, different report files and, command line commands will be specified.

2.2.2.1. Service Information

The service information that is important to report in that moment is basically what has been defined in Chapter 1, with the exception of some tables. Next, there is the list of tables and descriptors that shall be parsed and reported by DrTS. There are a lot of descriptors in SI, most of them are not used frequently or are not significant enough to be completely described.

- Program Association Table
- Program Map Table
- Network Information Table
- Service Description Table
- Event Information Table
- Time and Date Table
- Time Offset Table

- 0x02– Video stream descriptor
- 0x03 – Audio stream descriptor
- 0x0A – ISO 639 Language descriptor
- 0x40 – Network Name descriptor
- 0x41 – Service List descriptor
- 0x48 – Service descriptor
- 0x4A – Linkage descriptor
- 0x56 – Teletext descriptor
- 0x58 – Local Time Offset descriptor
- 0x59 – Subtitling descriptor
- 0x65 – Scrambling descriptor
- 0x6F – Application Signalling descriptor
- 0x83 – Logical Channel Number descriptor
- 0x55 – Parental Rating descriptor

Other information is considered out of scope and could be included in the future.

2.2.2.2. Reports

Each report will be delimited in a separated way.

XML report

XML report will contain all the parameters shown in a complete analysis (although the user selects a basic analysis) in XML format. All the information specified in the previous section should appear even if it doesn't appear in TSReader analysis; these parameters shall appear with no information.

Basic HTML

This report would be created when basic analysis is selected. The information that shall appear is the parameters related with all tables except neither EIT nor descriptors.

Complete HTML

From basic profile, EIT information is added and descriptors indicated in the section 2.2.2.1. in the corresponding table section in HTML.

2.2.2.3. Command line

So far, for command line, the only desired function is to analyse recorded files indicating the path of the source file and the path of the report must be stored, only XML report from DrTS, HTML and XML from TSReader are unnecessary. More functions are considered out of scope and shall be defined in the future.

2.2.3. Project Planning

For the schedule estimation of the application development, it will be considered an engineer in nearly full time (some days, full time, and others part time, in average 6 hours per day)

- Design and analysis of requirements – 20 days
- Implementation – 95 days
- Test – 25 days
- Deployment – 5 days

This means about 6 or 7 months of estimation (without holidays), once all requirements are studied, this estimation will be adjusted to ensure the final schedule. However, the application is estimated to be finished before August.

2.3. Analysis and Design

The purpose of the step of Analysis and Design is to transform the requirements into the design of the system-to-be. It is supposed that it specifies how the software product

would perform its tasks by means of evolving a robust architecture for the system, decomposing software into modules with interfaces. It is necessary to adapt the design to match the implementation environment with the objective of getting the best possible performance.

It is also useful to maintain a record of design decisions for traceability, because if we have a good traceability, each transformation can be found in an easy and fast way. This traceability is used to find the source of future problems and to avoid miscommunication between module designers or specification inconsistencies.

Summarising, the objective is to avoid the design to be inconsistent, incomplete, or ambiguous by understanding and defining how the system performs its tasks from a white box perspective. Consequently, all details shall be specified inside this section.

For DrTS application, we will start by defining the application and design environment that will be used for its implementation.

2.3.1. Detailed requirements

2.3.1.1. Environment

From the point of view of a simple TS file analyzer, any personal computer (PC) is the only hardware required to run TSReader because analysis is made by means of an XML file created by the program TSReader. To run TSReader successfully is needed Windows 2000 or Windows XP, 800 MHz or faster processor to ensure packets aren't lost with SD (standard definition) streams and 2.5 GHz or faster for HD (high definition).

In case of radio frequencies (RF) analysis a demodulator mechanism supported by TSReader is needed. Then, computer requirements can depend on this hardware. For example in this case, Hauppauge WinTV-NOVA-T Stick is used and it needs Windows XP Service Pack 2, a 1GHz processor and a USB port 2.0.

Nevertheless, besides the software (DrTS and TSReader – Standard Edition), it is only needed a method to demodulate RF signals and a PC that should be adequate at least to TSReader requirements.

DrTS is implemented in Visual C++ and it's compatible with most PCs, with a nearly complete portability to any computer, as any executable application. So, it will be a standalone application like it was required.

2.3.1.2. TSReader

TSReader can be controlled through command line, so, DrTS shall manage user options from GUI to achieve all the information from TSReader. It has been studied how it works and it will be derived from the following TSReader control.

Typing the next commands in TSReader through command line it is possible to analyse air signal from our demodulator or from a TS file.

Air signal:

```
> C:\_path_\TSreader.exe -s Sources\TSReader_HCWUSB70xxx.dll –M –x
_xml_output_file_ _freq_ _inversion_ _bandwidth_
> Example: "C:\TSReader\TSReader.exe" -s Sources\TSReader_File.dll -M -x
"C:\xml_document.xml" 794000 0 2
```

Transport Stream:

```
> C:\_path_\TSreader.exe –s Sources\TSReader_File.dll –M –x _xml_output_file_
_transport_stream_file_
> Example: “C:\_path_\TSreader.exe” –s Sources\TSReader_HCWUSB70xxx.dll –M –x
“C:\xml_document.xml” “C:\transport_stream.ts”
```

Note: -M argument minimizes TS Reader execution.

2.3.1.3. Settings

Some settings must be defined and also how they will be saved in case of default settings to have a backup of them.

As default settings, TSReader configuration is mainly the most important setting, because this setting is external to DrTS and it should be set by user. Besides TSReader configuration itself, the configuration of HW (demodulator DLL file) in case of recording situation can be also stored, to avoid its configuration every time recording is executed.

In addition to these two parameters also other settings can be stored to those users that will often execute DrTS. This type of parameters will be available to be change in each situation without change the default value, unless they were saved another time. These parameters will be:

- Report profile preferred: which report basic or complete report is the option to the user → default profile: full
- Analysis option: the option of analyse the live TS together when they are recorded or not → default option: analysis enabled
- Recording time: default time in seconds for TS recordings → default time: 60 sec

To save these settings, a configuration text file will be used. It will be located in “C:” with the name of “Settings-DrTS.txt” and will be parsed every time that application is opened and executed to check the information.

There are other settings to be indicated from user:

For recording:

- Band of frequencies: Ultra High Frequencies (UHF) or Very High Frequencies (VHF) bands → default band: UHF

- Inversion modulation parameter → default value: No activated
- Bandwidth of channel: 8MHz, 7MHz or 6MHz → default bandwidth: 8MHz
- Frequencies or scan bandwidth → default values are set to scan from 474MHz to 858 MHz (single analysis has set value 474MHz)
- Where to save recorded TS → No default option

For analysis:

- TS files list to be analysed → Empty by default (files format accepted are those supported from TSReader, i.e. MPEG, MPG, TS... XML can also be accepted)

2.3.1.4. Service information

The service information that should be parsed is clear by the specification from the requirements, in Chapter 1, all the detailed information can be looked up. In relation to tables, only the specified parameters shall appear. But in relation to descriptors, all of them should be copied to the XML file, but in a different way, so, it is necessary to establish the method for descriptors.

Descriptors specification

Some descriptors are needed to make the basic analysis, because these descriptors are included in TSReader. Deeper descriptors analysis would be available only in complete analysis.

Among all descriptors only the specified in the requirements would be completely described in reports and the rest of them would be only detected and identified. Next, descriptors are divided in relation to its type of implementation inside the application.

Consequently, there are four types of descriptors in DrTS report. They differ in the description point of view from TSReader and DrTS. All descriptors of the specified tables are shown in a defined structure or/and raw structure. So far, most of them have only specified the name of the descriptor.

Type 1: Descriptors defined in TSReader and included in basic profile

There are descriptors already defined in TSReader as tags but they can be found also in raw data. These descriptors are not implemented directly because they are implemented by means of tags directly in XML file. These descriptors are:

- 0x02 – Video stream descriptor
- 0x03 – Audio stream descriptor
- 0x0A – ISO 639 Language descriptor
- 0x40 – Network Name descriptor
- 0x48 – Service descriptor

These descriptors are copied like in TSReader report to DrTS XML with the tag and raw data

Example descriptor type 1:

```
TSReader:
…
<NETWORK-NAME>_TELEVISIO DE
CATALUNYA</NETWORK-NAME>
…
<DESCRIPTOR>
<TAG>0x40</TAG>
<LENGTH>23</LENGTH>
<DATA>0x05 0x54 0x45 0x4c 0x45
0x56 0x49 0x53 0x49 0x4f 0x20
0x44 0x45 0x20 0x43 0x41 0x54
0x41 0x4c 0x55 0x4e 0x59
0x41</DATA>
</DESCRIPTOR>
```

```
DrTS:
…
<NETWORK-NAME>&#x05;TELEVISIO DE
CATALUNYA</NETWORK-NAME>
…
<DESCRIPTOR-RAW>
<DESCNAME>Network name
descriptor</DESCNAME>
<TAG>0x40</TAG>
<LENGTH>23</LENGTH>
<DATA>0x05 0x54 0x45 0x4c 0x45
0x56 0x49 0x53 0x49 0x4f 0x20
0x44 0x45 0x20 0x43 0x41 0x54
0x41 0x4c 0x55 0x4e 0x59
0x41</DATA>
</DESCRIPTOR-RAW>
```

Type 2: Descriptors that shall be described

The following descriptors would be fully implemented as described in Chapter 1.

- 0x41- Service List descriptor
- 0x4a- Linkage descriptor
- 0x56- Teletext descriptor
- 0x58- Local Time Offset descriptor
- 0x59- Subtitling descriptor
- 0x65- Scrambling descriptor
- 0x6f- Application Signalling descriptor
- 0x83- Logical Channel Number descriptor
- 0x55- Parental Rating descriptor

Example descriptor type 2:

```
TSReader:
- <DESCRIPTOR>
<TAG>0x55</TAG>
<LENGTH>4</LENGTH>
<DATA>0x45 0x53 0x50 0x00</DATA>
</DESCRIPTOR>
```

```
DrTS:
- <DESCRIPTOR>
<DESCNAME>Parental rating
descriptor</DESCNAME>
<COUNTRY>Spain</COUNTRY>
<RATE>0</RATE>
<YEARS>0</YEARS>
</DESCRIPTOR>

- <DESCRIPTOR-RAW>
<DESCNAME>Parental rating
descriptor</DESCNAME>
<TAG>0x55</TAG>
<LENGTH>4</LENGTH>
<DATA>0x45 0x53 0x50 0x00</DATA>
</DESCRIPTOR-RAW>
```

Type 3: Descriptors that are only commented

The rest of descriptors are only detected, but the possibility to include any of them in the future is completely open.

All the descriptors listed below and not described by TSReader or DrTS, are shown with the name, as only descriptors-raw tags:

0x02, 0x03, 0x0a, 0x40, 0x48, 0x41, 0x4a, 0x56, 0x58, 0x59, 0x65, 0x6f, 0x83, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x17, 0x18, 0x19, 0x1a, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x49, 0x4b, 0x4c, 0x50, 0x51, 0x52, 0x53, 0x57, 0x5a, 0x5b, 0x5c, 0x5d, 0x5e, 0x5f, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x67, 0x68, 0x69, 0x6a, 0x6b, 0x6c, 0x6d, 0x6e, 0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78, 0x79, 0x7a, 0x7b, 0x7c, 0x7f, 0x85, 0x86, 0x80, 0x81, 0x82, 0x83, 0x84, 0x90, 0x91, 0x92, 0xc0, 0x4d, 0x4e, 0x4f, 0x54, 0x55

Example descriptor type 3:

```
TSReader:
- <DESCRIPTOR>
<TAG>0x45</TAG>
<LENGTH>34</LENGTH>
<DATA>0x01 0x1a 0xe7 0xe8 0xe9
0xea 0xeb 0xec 0xed 0xee 0xef
0xf3 0xf4 0xf5 0xf6 0xc7 0xc8
0xc9 0xca 0xcb 0xcc 0xcd 0xce
0xcf 0xd3 0xd4 0xd5 0xd6 0x04
0x01 0xf0 0x05 0x01 0xf7</DATA>
</DESCRIPTOR>
```

```
DrTS:
- <DESCRIPTOR-RAW>
<DESCNAME>VBI data
descriptor</DESCNAME>
<TAG>0x45</TAG>
<LENGTH>34</LENGTH>
<DATA>0x01 0x1a 0xe7 0xe8 0xe9
0xea 0xeb 0xec 0xed 0xee 0xef
0xf3 0xf4 0xf5 0xf6 0xc7 0xc8
0xc9 0xca 0xcb 0xcc 0xcd 0xce
0xcf 0xd3 0xd4 0xd5 0xd6 0x04
0x01 0xf0 0x05 0x01 0xf7</DATA>
</DESCRIPTOR-RAW>
```

If the descriptor is not in the list, the structure is the same in this way:

Example of descriptors not defined previously:

```
TSReader:
- <DESCRIPTOR>
<TAG>0x66</TAG>
<LENGTH>6</LENGTH>
<DATA>0x01 0x06 0x01 0x01 0x00
0x00</DATA>
</DESCRIPTOR >
```

```
DrTS:
- <DESCRIPTOR-RAW>
<DESCNAME>Descriptor
0x66</DESCNAME>
<TAG>0x66</TAG>
<LENGTH>6</LENGTH>
<DATA>0x01 0x06 0x01 0x01 0x00
0x00</DATA>
</DESCRIPTOR-RAW>
```

2.3.1.5. XML parser

DrTS is used to create and parse XML files. TS Reader application is called by DrTS in order to create and XML file including all relevant information about the transport stream.

DrTS parses this XML and generates an HTML file readable by the final users and generates a new XML with the information specified. In this case, it will be useful to use a library that already parses XML files to make this task easier.

For this task, TinyXML library has been chosen. This library is a simple, small, C++ XML parser that can be easily integrated into other programs. As its author says "TinyXML has evolved from community feedback and become the work of many contributors. It is a simple, stable, basic XML parser used by many open source and commercial products".

Then, from this work tinyxml.lib library will be used to parse xml files inside DrTS.

2.3.1.6. Reports

Some details to reports specification from requirements shall be described.

The reports will have the same name and location than source file (TS), but changing the extension (in exception for analysis done from command line that maybe specifies a different destination path), for example:

```
_path_\test_file.ts      (is our TS source file)

- the result from TSReader will be named _path_\test_file.xml
- the result from DrTS parsing will be named _path_\test_file-DrTS.xml
- the result from creating HTML will be named _path_\test_file.html
```

If the file will be recorded (and maybe analysed), the example follows same rules in addition to the naming for new TS files.

TS shall be created with the name:

```
_path_indicated_by_user_\YYYYMMDD-FFFMHz-HHMM.ts
```

where:

- YYYYMMDD is the date recording with four digits for the year, two digits for the month and two digits for the day
- FFF is the frequency in MHz of signal with three digits
- HHMM is the time of recording with two digits for hour and two more digits for the minute

With this rule, the example of a TS recording is:

```
- the result of the TSReader recording is _path_indicated_by_user_\YYYYMMDD-
FFFMHz-HHMM.ts    (as TS source file)

In case of enable the analysis option:
- the result from TSReader will be named _path_indicated_by_user_\YYYYMMDD-
FFFMHz-HHMM.xml
- the result from DrTS parsing will be named _path_indicated_by_user_\YYYYMMDD-
FFFMHz-HHMM -DrTS.xml
- the result from creating HTML will be named
_path_indicated_by_user_\YYYYMMDD-FFFMHz-HHMM.html
```

The final report appearance will be decided by developer on final implementation due to this is an internal project and this point is not a closed requirement from user point of view, the only requirement is to contain all the specified information.

2.3.1.7. Graphical User Interface

All the options shall appear in a considered order inside GUI, for this reason here there are the examples of GUI design for this application.

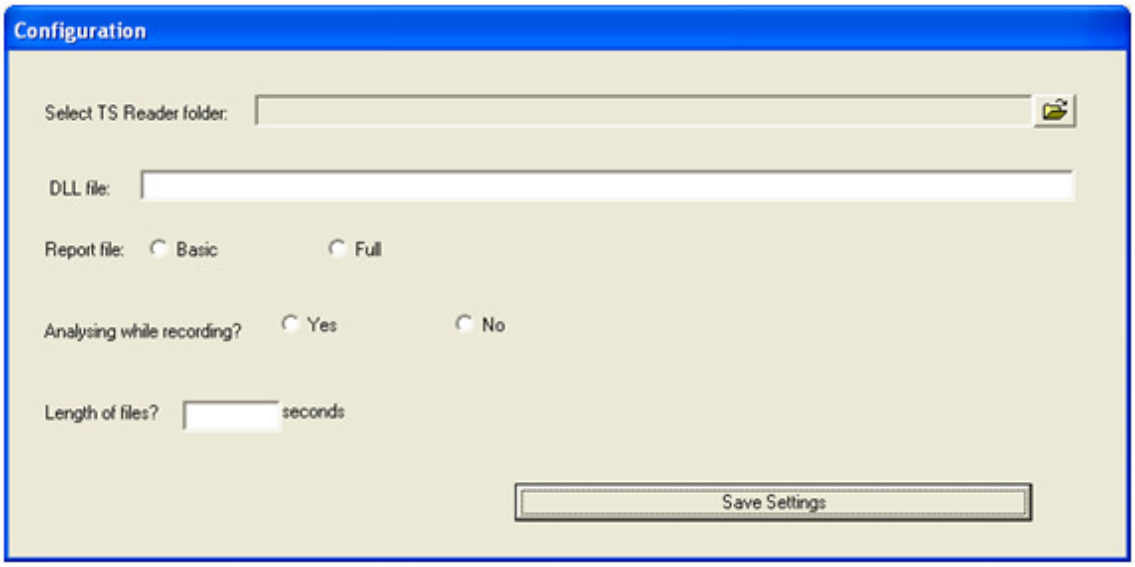


Figure 9 – Example of Configuration screen

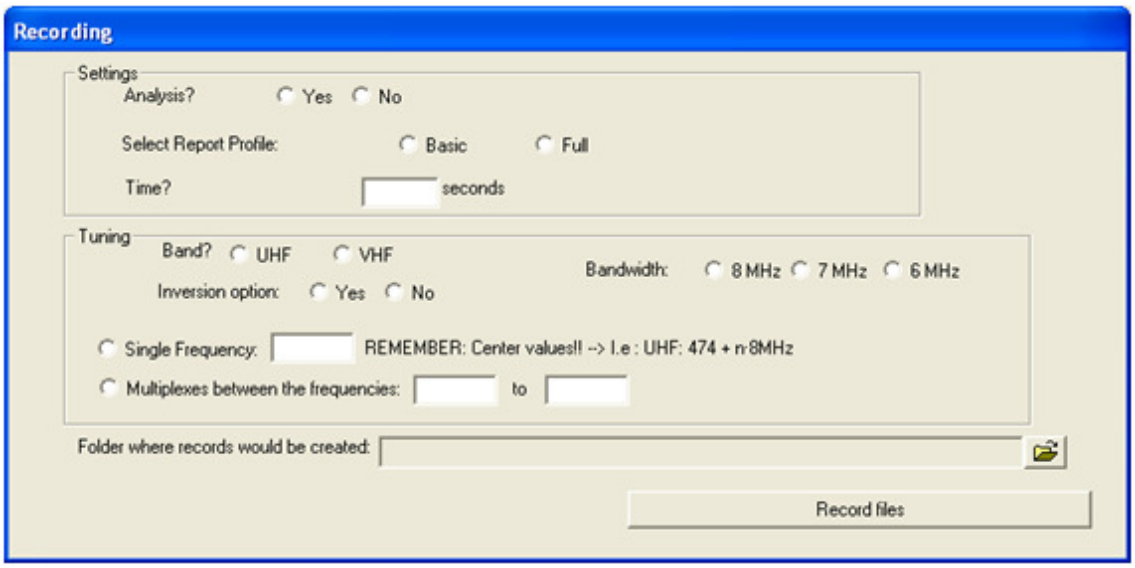


Figure 10 – Example of Recording screen

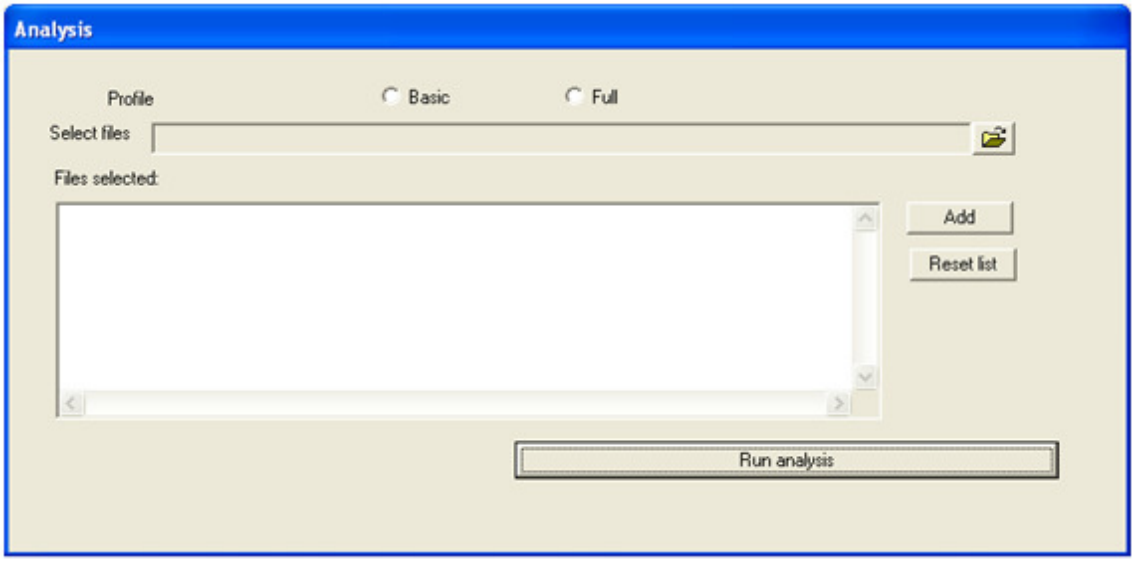


Figure 11 – Example of Analysis screen

Details of GUI will depend on the final implementation, due to, this is an internal project and GUI is not a closed requirement from the user point of view. So, the developer, in the implementation step, will adapt the design in order to maximize the relation usability and requirements.

2.3.1.8. Command line

The only purpose for command line so far is to include the possibility of control of XML DrTS analysis by Command Prompt.

This requirement shall work once it is ensured that TSReader is already installed in the computer. Moreover to facilitate the configuration, the execution of the user interface must be done the first time that DrTS is executed (not needed if it was used previously). This step will be necessary to save TSReader path to Doctor TS use.

The command format can be:

```
    - _path_\DoctorTS.exe -TSServer "_filepath_" "_directorypath_"
    - _path_\DoctorTS.exe -TSServer "_filepath_"

    _filepath_: indicates which file shall be analysed
    _directorypath_: indicates where the -DrTS.xml report will be saved
In case of not having _directorypath_, the report will be saved in the same
directory than _filepath_
```

2.3.2. Use cases

Use cases are descriptions of the system and its scope. It is a description of the system functionality (functional requirements) in a schematic representation. This practice helps to identify actors and their role in the system. Each use case describes the final intention of the user.

Use cases model the functional requirements describing the scenarios of the application where an actor (external agent) uses the system to do a process that will give him any value.

The scenarios are defined by a specific sequence of actions and interactions between the actor and the system. On the other hand, the actor is an external entity to the system that participates in the use case (history). The actor can be a person, a group of people, another system, etc and can be the Initiator (if it stimulates the process) or a Participant (if it takes part in the process).

DrTS system is described by the use cases diagram in Figure 12.

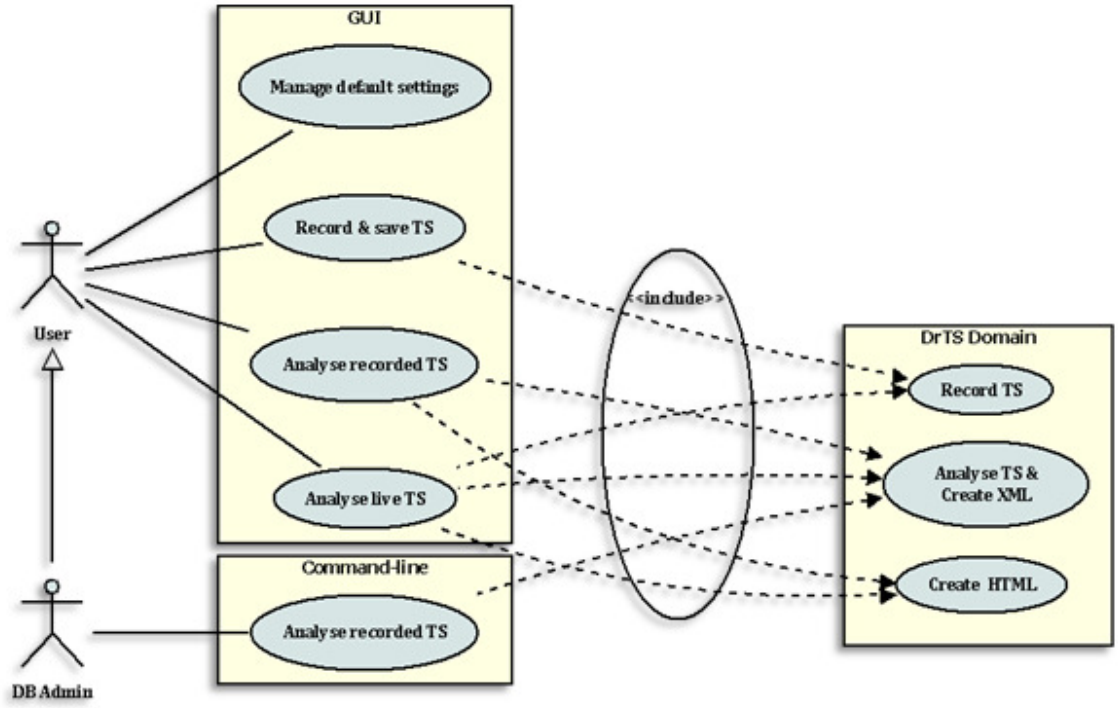


Figure 12 – Use cases of DrTS

Here there is a description of each representation in the previous use cases diagram.

Actors:

- User: This actor is the initiator of the system. This actor interacts with GUI system for the operations described below.
- Data Base (DB) admin: This actor is a specialization of *User*. *DB admin* inherits *User* properties. This actor also interacts by command line.

Use cases:

- Manage default settings: This use case is defined to manage (modify and save) the settings of the application.
- Record and save TS: Use case executed when the analysis option in the recording action is disabled. In this case, a signal is recorded and saved. Options of this recording depend on options selected by user in the GUI

(multiple frequencies, full scan, single recording, etc). Consequently, this use case includes *Record TS* use case of DrTS domain.

- Analyse recorded TS (GUI domain): This use case invokes *Analyse TS & Create XML* use case in DrTS domain with the defined parameters filled in the Analysis GUI. For this use case, unlike the same use case but in command line domain, *Create HTML* use case is invoked too.
- Analyse live TS: This use case is executed when analysis option in recording action is enabled. In this use case, *Record TS, Analyse TS & Create XML* and *Create HTML* use cases are invoked.
- Analyse recorded TS (Command Line domain): Use case that invokes *Analyse TS & Create XML* use case by command line.
- Record TS: This use case is in charge of control TSReader for recording stream with the parameters that users proportionate in use case *Record and save TS* or *Analyse live TS*.
- Analyse TS & Create XML: This use case controlling TSReader to create XML report and parse it while the desired XML is created.
- Create HTML: When this use case is invoked by use cases, *Analyse recorded TS (GUI domain)* and *Analyse live TS*, the task for parsing DrTS XML file and create HTML report is executed.

2.3.2.1. Use Case Flow-of-Events

The use case diagram is important for visualizing a system. However, a textual description of the sequence of transactions of a use case is also needed for understanding what really happens in a use case. The flow of events is written in terms of what the system should do, not how the system does it.

A flow of events is used to textually describe the possible events in the use case. It is used to describe the actors by means of the interactions with the system described in the use case.

- First, preconditions indicate what state the system should be before the use case will be started.
- Secondly, the main flow of events is listed in the order they should be implemented. The main flow is the interaction between initiator and the system that perform the use case objective in a successful way.
- Third, subflows are listed. Subflows are used to break down the main flow into smaller pieces, alternative or complementary situations that complete the objective in a different way or directly don't achieve the objective (errors).
- Last, postconditions, extension points of use case and scenarios are described. Postconditions are the system state after use case execution. Extension points summarise extensions to other use cases and the points that could be invoked. Scenarios description is the environment description where users will use the system.

All possible flow of events should be described by any combination of basic flux and/or any or several alternative flows.

For DrTS, flows of events are attached in Annexes. Only "Analyse live TS" is included below in Figure 13 as example of flow of events.

Use case	Analyse live TS
Actors	User
Preconditions	This use case is performed when "Manage default settings" use case has been done at least once. For executing this use case, "Recording" tab should be selected. Data in configuration file shall be correct (Settings-DrTS.txt), if not ERROR message appears. "Analysis" option in this tab shall be enabled.

Main flow

Actor	**System**
1- Input desired parameters to tune or leave default parameters	2- Save parameters to be used in the execution
3- Press "Record files" button	4- Show INFO message asking to wait for results
5- Accept the information of the message	6- Start "Record TS" use case
	7- Start "Analyse TS & Create XML" use case
	8- Start "Create HTML" use case
	9- Show INFO message reporting the end of process and the results of analysis (number of TS analysed)
10- Accept the information of the message	

Subflows

A- Change default settings (except "analysis" option) for the actual execution

Actor	**System**
1A / 3A- Change default setting (except "analysis" option) for the actual execution	2A / 4A- Apply settings to be used without saving them in configuration file
In case of calling this subflow as step 1A, continue in main flow, step 1. In case of doing this subflow as step 3A, continue with step 3.	

B- Change default "analysis" option for the actual execution

Actor	**System**

1B / 3B- Change default "analysis" option for the actual execution	
This use case is finalised and shall start "Record and Save TS" use case in following step: In case of calling this subflow as step 1B, continue in main flow, step 1. In case of doing this subflow as step 3B, continue with step 3.	
Postconditions	Application is ready for executing any other use case.
Extension points	1- Application/use case can be closed/cancelled/stopped at any point by pressing [x] close button of application. 2- At any point, user can change tab and start other user case.
Scenarios	User is in front his/her computer, executing DrTS.

Figure 13 – Flow of events example for Analyse live TS use case

2.3.3. Architecture

Once the relation of actors with the system has been described with detail, the design of the system shall be described.

Design is the step between Analysis and Implementation. Analysis specifies "what" should be implemented and Design specifies "how" should be implemented by means of different tools that are describe below.

2.3.3.1. Sequence diagrams

One of the tools used is the sequence diagrams. They show the interaction between objects of the system. The interactions are a collection of messages between instances with an order in time in the lifetime of objects. These diagrams are use in external behaviour actor-system modelling.

DrTS sequence diagrams are includes in the Annexes, but one of them is shown in the following figure.

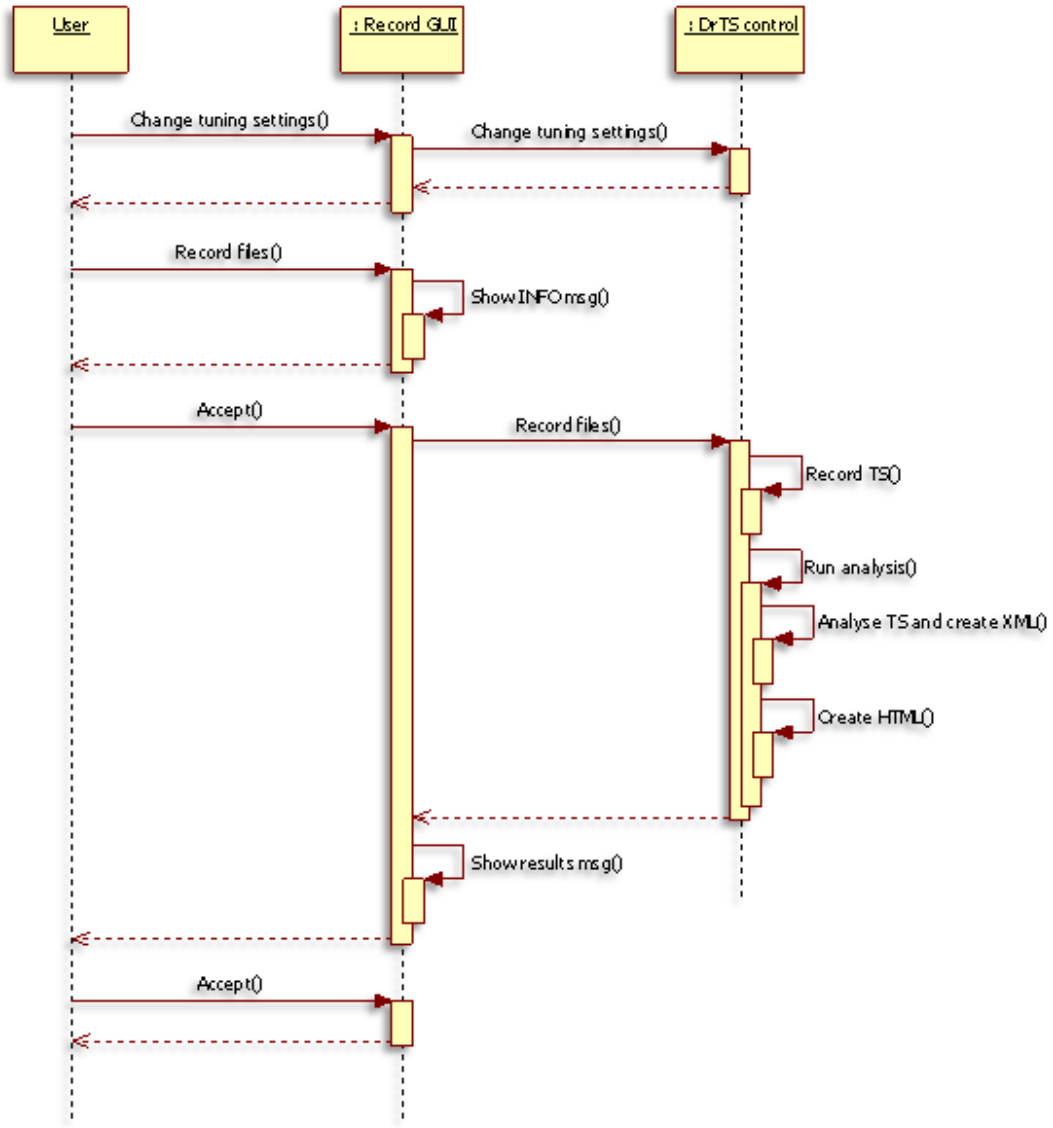


Figure 14 – Diagram of sequence of Analyse live TS use case

2.3.3.2. Class diagram

On the other hand, there are class diagrams that are a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes of the domain modelling by means of obtain participants classes. Class diagrams are typically used to:

- Explore domain concepts in the form of a domain model
- Analyze requirements in the form of a conceptual/analysis model
- Depict the detailed design of object-oriented or object-based software

A class model is includes one or more class diagrams and the supporting specifications that describe model elements. The class diagram provides the basic

building blocks for all other structure diagrams, such as the component or object diagrams (just to name a few).

Some basics concepts are described below:

- Class name: UML representation of a class is a rectangle containing three compartments stacked vertically. The top compartment shows the class's name. The middle compartment lists the class's attributes. The bottom compartment lists the class's operations. When drawing a class element on a class diagram, you must use the top compartment, and the bottom two compartments are optional.
- Class attribute list: attribute section of a class (the middle compartment) lists each of the class's attributes on a separate line. The attribute section is optional.
- Class operations list: class's operations are documented in the third (lowest) compartment of the class diagram's rectangle, which again is optional. The operations of a class are displayed with each operation on its own line.
- Inheritance: a very important concept in object-oriented design, *inheritance*, refers to the ability of one class (child class) to *inherit* the identical functionality of another class, and then add new functionality of its own. To model inheritance on a class diagram, a solid line is drawn from the child class (the class inheriting the behaviour) with a closed, unfilled arrowhead pointing to the super class.
- Abstract classes and operations: italicized text in class diagram indicates that class or operation is an abstract class or operation.
- Associations: when you model a system, certain objects will be related to each other, and these relationships themselves need to be modelled for clarity.
- Packages: inevitably, if the model is a large system or a large area of a business, there will be many different classifiers in the model. Packages enable modellers to organize the model's classifiers into namespaces, which is sort of like folders in a filing system.

For DrTS, class diagram is shown in the following figure.

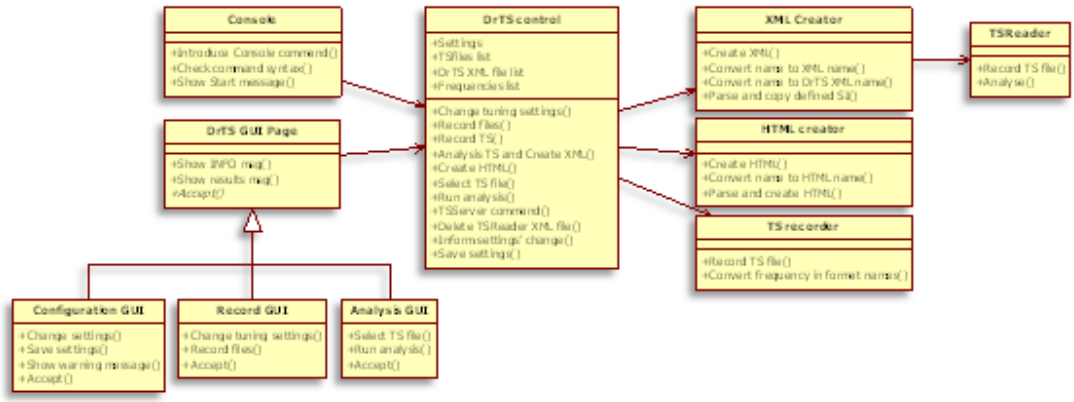


Figure 15 – DrTS class diagram

2.3.4. Task planning

After specifying all the aspects of DrTS, a detailed scheduled can be shown.

As in this project only one person is involved and analysis, design, implementation, test and deployment is done by the same person, a cyclic development cannot be applied as it is defined. It is not necessary to find a critical timing in the project, because all the steps shall be done and the same person shall do all the process, so, all the tasks are critical in this situation.

Firstly, in the schedule has been included design and analysis process to have a realistic estimation of the time of study and documentation.

Secondly, implementation has been split in several tasks that are very limited and well defined. Each task needs a timing for unit testing (in next section we explain its importance), so, timing of each task includes an estimation of unit testing and debugging in case it is necessary. Next, there is the description of each task of the implementation step.

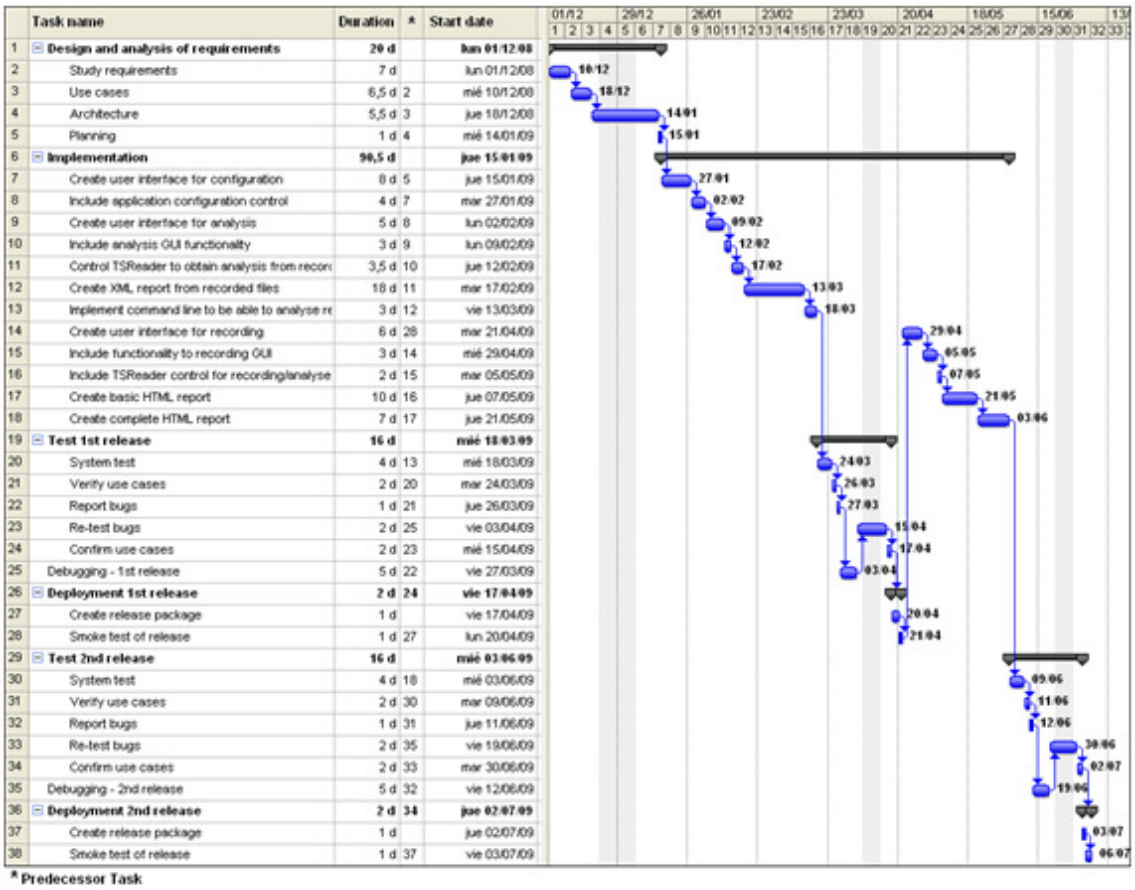
- Create user interface for configuration: as application will be based in GUI for most of users, this step will decide the first appearance of user interface including the controls for set default settings.
- Include application configuration control: this task will implement the functionality of the controls in the previous step. These two steps will proportionate the information of TSReader to allow its control and inform about some settings to the analysis and recording process.
- Create user interface for analysis: the part of analysis in GUI will be created to allow getting information for next implementation steps.
- Include analysis GUI functionality: functionality to the controls in GUI is implemented in this task.
- Control TSReader to obtain analysis from recorded files: the control of TSReader from application to analyse files shall be included.
- Create XML report from recorded files: once TSReader report is obtained, parse it and extract the desired information to create the new XML shall be done.
- Implement command line to be able to analyse recorded files: when analysis is enabled from GUI, command line will be enabled to be capable of automatic analysis by external control.
- Create user interface for recording: GUI for recording and analysis of live signal shall be modelled.
- Include functionality to recording GUI: functionality to recording interface shall be implemented.
- Include TSReader control for recording/analyse RF signal: TSReader control for modulated signal is done in this task.
- Create basic HTML report: this task will create the report in HTML. It is necessary to decide the design too.
- Create complete HTML report: from the design of previous task, complete information in HTML shall be included.

Due to the dependency to other applications of this program, such as the creation of a DB based in the information given by DrTS that only needs the analysis use case by command line, it has been decided to create two releases of the application during the development. This action is to avoid in some way the problems of a waterfall development.

Testing cycle means a group of tasks that will be defined in section 2.5. but that will alternate with a task of debugging to solve the possible problems that can be found in the first testing tasks. After debugging, more testing tasks are done to check the changes of this task and to check possible side effects of these modifications.

Finally, the deployment of a release (after the complete testing, debugging and re-testing cycle) means the task of release the executable and test itself to avoid problems of third parties. This step is also described deeply in section 2.5.

Those tasks are scheduled in a Gantt diagram in Figure 16. The ending of the project is estimated for begin of July with a duration of 128.5 days with holidays included (99.5 days of real job), nearly 600 hours of work.



	Task name	Duration	*	Start date
1	**Design and analysis of requirements**	**20 d**		**lun 01/12/08**
2	Study requirements	7 d		lun 01/12/08
3	Use cases	6,5 d	2	mié 10/12/08
4	Architecture	5,5 d	3	jue 18/12/08
5	Planning	1 d	4	mié 14/01/09
6	**Implementation**	**90,5 d**		**jue 15/01/09**
7	Create user interface for configuration	8 d	5	jue 15/01/09
8	Include application configuration control	4 d	7	mar 27/01/09
9	Create user interface for analysis	5 d	8	lun 02/02/09
10	Include analysis GUI functionality	3 d	9	lun 09/02/09
11	Control TSReader to obtain analysis from recor	3,5 d	10	jue 12/02/09
12	Create XML report from recorded files	18 d	11	mar 17/02/09
13	Implement command line to be able to analyse re	3 d	12	vie 13/03/09
14	Create user interface for recording	6 d	28	mar 21/04/09
15	Include functionality to recording GUI	3 d	14	mié 29/04/09
16	Include TSReader control for recording/analyse	2 d	15	mar 05/05/09
17	Create basic HTML report	10 d	16	jue 07/05/09
18	Create complete HTML report	7 d	17	jue 21/05/09
19	**Test 1st release**	**16 d**		**mié 18/03/09**
20	System test	4 d	13	mié 18/03/09
21	Verify use cases	2 d	20	mar 24/03/09
22	Report bugs	1 d	21	jue 26/03/09
23	Re-test bugs	2 d	25	vie 03/04/09
24	Confirm use cases	2 d	23	mié 15/04/09
25	Debugging - 1st release	5 d	22	vie 27/03/09
26	**Deployment 1st release**	**2 d**	**24**	**vie 17/04/09**
27	Create release package	1 d		vie 17/04/09
28	Smoke test of release	1 d	27	lun 20/04/09
29	**Test 2nd release**	**16 d**		**mié 03/06/09**
30	System test	4 d	18	mié 03/06/09
31	Verify use cases	2 d	30	mar 09/06/09
32	Report bugs	1 d	31	jue 11/06/09
33	Re-test bugs	2 d	35	vie 19/06/09
34	Confirm use cases	2 d	33	mar 30/06/09
35	Debugging - 2nd release	5 d	32	vie 12/06/09
36	**Deployment 2nd release**	**2 d**	**34**	**jue 02/07/09**
37	Create release package	1 d		jue 02/07/09
38	Smoke test of release	1 d	37	vie 03/07/09

* Predecessor Task

Figure 16 – Gantt diagram for DrTS project

2.4. Implementation

Implementation task has the purpose of defining the organization of the code, in terms of subsystems organized in layers and in terms of implementation elements (source files, binaries, executables, and others). This task also includes testing the developed components as units (unit testing) and integrate the results produced by individual implementers (or teams), into an executable system.

The difficulties that can be found during the implementation are related to component interaction errors, or can be related to the order of integration that may influence quality and productivity. The most important problem that can have a software system (that happens many times, more times than can be desired) is that the system implements a task not desired for the system. In other words, the system works but it doesn't do what is required to do.

This last problem can be avoided with a good requirements capture and after this, a good analysis and design. If those steps are correct, project success is nearly sure (in functionality terms). It means that the system will do what it shall do.

Unit testing of DrTS in this stage will be based in checking use cases of each module affected and in the specific functionalities of each one.

2.4.1. Code description

After all design and analysis description, the implementation is the step which applies all the previous indications and specifications. To give a general vision of how code has been created, its architecture is described in Figure 17.

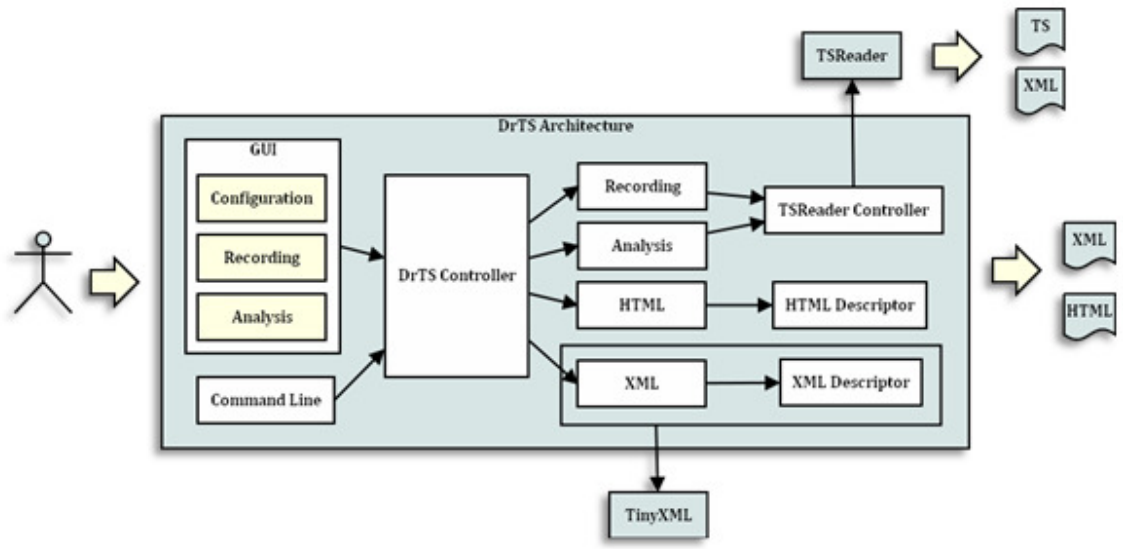


Figure 17 – SW Architecture

It is possible to identify each part of the architecture with a file in the code and with the relations of class diagram in Figure 15.

Following some of the remarkable points of the code generation are explained. For example, there is the creation of HTML and XML report that is separated in a main part of code with the management of tables and generic points and another part that control descriptors. This separation is done to make easier the expansion of descriptors definition inside DrTS.

The control of TinyXML is done from XML part exclusively and examples of how this library is used are inside code.

Example of a table parsing function using TinyXML library

```
static void xml_ParsePAT( TiXmlDocument xml_TSReaderXMLFile, TiXmlElement *
xml_root )
{
    TiXmlElement* xml_tag_pat;
    TiXmlElement* xml_subtag;
    TiXmlElement* xml_element;

    TiXmlHandle docHandle( &xml_TSReaderXMLFile );

    //PAT
    xml_tag_pat = new TiXmlElement( "PAT" );
    xml_root->LinkEndChild( xml_tag_pat );

    //version
    xml_subtag = new TiXmlElement( "VERSION" );
    xml_element = docHandle.FirstChild( "MPEG-TABLES" ).FirstChild( "PAT"
).Child( "VERSION", 0 ).ToElement();
    xml_tag_pat->LinkEndChild( xml_CheckElement ( xml_subtag, xml_element ) );

    //transport_id
    xml_subtag = new TiXmlElement( "TRANSPORT-STREAM-ID" );
    xml_element = docHandle.FirstChild( "MPEG-TABLES" ).FirstChild( "PAT"
).Child( "TRANSPORT-STREAM-ID", 0 ).ToElement();
    xml_tag_pat->LinkEndChild( xml_CheckElement ( xml_subtag, xml_element ) );
}
```

Example of the function that checks the elements to be inserted in new file

```
TiXmlElement* xml_CheckElement ( TiXmlElement* xml_subtag, TiXmlElement*
xml_element )
{
    if ( xml_element && xml_element->GetText() )
    {
           xml_subtag->LinkEndChild( new TiXmlText( xml_element->GetText() ) );
    }
    return xml_subtag;
}
```

A significant point in GUI implementation (that is not explained in detail in SW architecture), is that MFC classes that Microsoft Visual C++ distributes have been used for interface generation. BrowseCtrl[3] is an external library has been used to include file system navigation in the several points that is used.

A remarkable point of implementation in code is the intelligence of the report from SI analysis point. The point is that when a stream of subtitles in teletext format is sent, the identifying format of the stream is teletext. This is correct from standard point of

[3] From www.codeproject.com

view. But the information is more useful if it is identified as subtitles. To know if it is really subtitles or not, the subtitles descriptor shall be found to this stream.

DrTS is capable to report as subtitles type a stream of teletext subtitles that TSReader reports as teletext type. This intelligence inside DrTS improves the future applications to find correct information and in less time.

2.4.2. Unit testing

The stage of unit testing of DrTS will be based in checking each use cases of each module affected. But this testing is enough; there are specific functionalities of each subtask implemented that was deeply checked because use case checking was not enough at all. Following some examples are shown.

The first example is for command by console. The functionality tested in a deep manner has been how parameters are passed to DrTS. The problem is that many format names can be found (with spaces, without, with dots in the middle of the names, with inverted commas, without...). When this functionality was implemented, many tests were done in order to find all the possibilities and that those worked correctly.

Another functionality that has been deeply tested is to take correct settings from text file. Checking is done when the application is launched, when tabs are selected and when the process is initiated. This testing was necessary to know the possible problems we could found when settings were changed only for the actual execution or when TSReader was not correctly configured.

As last example, another piece of code tested with all its possibilities is at the time of XML parsing. Elements in XML file maybe don't exist, maybe are empty, maybe they are not in the expected order... So, all those possibilities have been checked by test and real files.

2.4.3. Final GUI

The following three figures are attached in order to show the final interface of the application.

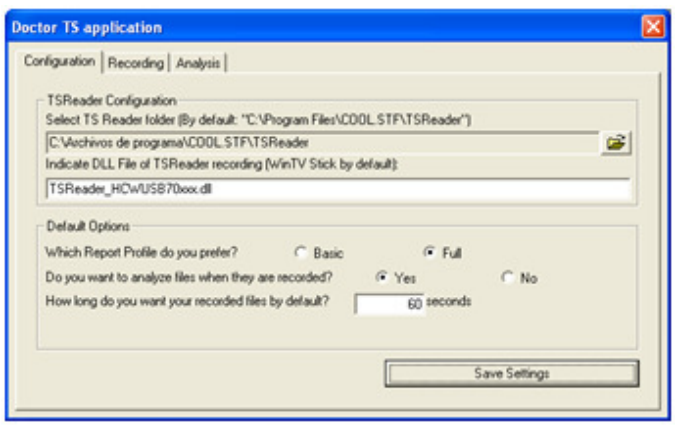


Figure 18 – Screenshot from DrTS Configuration screen

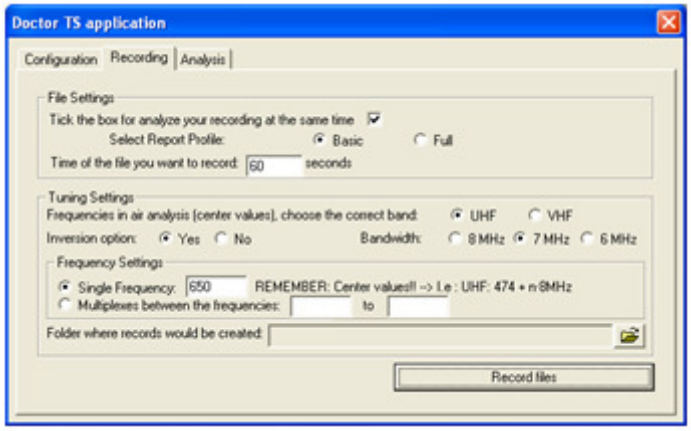


Figure 19 – Screenshot from DrTS Recording screen

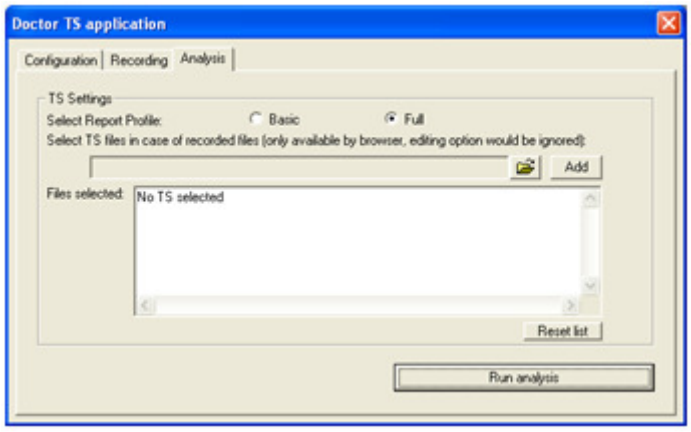


Figure 20 – Screenshot from DrTS Analysis screen

Other interfaces have been implemented, on one hand warning and information messages that are defined in use cases; on the other hand, console interface with messages and information. Messages interfaces are attached in Annexes.

Following console examples are shown:

Examples of Command Line interface

```
Example: not correct command
C:\_path_to_executable_>DoctorTS.exe - TSSServer
DOCTOR TS WARNING: No valid command

Example: not correct syntax (missing parameters)
C:\_path_to_executable_>DoctorTS.exe –TSServer
DOCTOR TS ERROR: No valid arguments

Example: not correct TSReader settings
C:\_path_to_executable_>DoctorTS.exe -TSServer
"C:\_path_to_TS_file_\example.ts" C:
DOCTOR TS: Analysis will be done, please check results in a while
Please, execute DrTS interface to set your Configuration settings. Mainly
TSReader folder.

Example: report already exists
C:\_path_to_executable_>DoctorTS.exe -TSServer
"C:\_path_to_TS_file_\example.ts"
```

```
DOCTOR TS WARNING: Directory to XML file name is missing, DrTS will use TS File
directory!
DOCTOR TS: Analysis will be done, please check results in a while

**** File already analyzed! ****

Example: missing destination path
C:\_path_to_executable_>DoctorTS.exe -TSServer
"C:\_path_to_TS_file_\example.ts"
DOCTOR TS WARNING: Directory to XML file name is missing, DrTS will use TS File
directory!
DOCTOR TS: Analysis will be done, please check results in a while

Example: successful analysis
C:\_path_to_executable_>DoctorTS.exe -TSServer
"C:\_path_to_TS_file_\example.ts" C:
DOCTOR TS: Analysis will be done, please check results in a while
```

2.4.4. Reports

Reports specification was open to developer in previous sections. Because of this, an example HTML report in basic profile is shown in this section. A full profile report is not attached because its size is about 10 pages for a simple TS file as it is the shown in the example. Only some portions are reported in the Annexes to show it as example.

XML reports neither can be attached due to its size. Also some portion will be attached in Annexes.

Sony BCNTec SW group

TS Reader - v.2.7.45h

General Information			
File:	example.ts		
Frequency (MHz):	0	Channel:	0
Date (TDT):	2007/03/14	Hour (TDT):	13:26

Network Information		NIT version:	9
Network Name:	TELEVISIO DE CATALUNYA	Network ID:	12674
Network Type:	DVB-T	Original Net ID:	8916

Stream Information	TS ID:	97	SDT version:	5

Service ID	**Service Name (short & long name)**	**Type**
801	TV3 - TVC	digital television service
802	33 - TVC	digital television service
803	3/24 - TVC	digital television service

804	K3/300 - TVC	digital television service

ES information					
Service ID:	801	**Service Name (short name):**	TV3		
EIT availability:		Scheduled:	Yes	Now/Next:	Yes
PMT PID:	110		**PCR PID:**	111	

INDEX	**PID**	**Stream Type: RAW - TYPE**	
1	111	2	VIDEO
2	112	3	AUDIO
3	114	3	AUDIO
4	115	6	AUDIO
5	116	3	AUDIO
6	113	6	TELETEXT
7	601	12	12
8	701	11	11
9	704	11	11
10	702	11	11
11	703	11	11
12	801	6	SUBTITLES
13	901	5	5

Service ID:	802	**Service Name (short name):**	33		
EIT availability:		Scheduled:	Yes	Now/Next:	Yes
PMT PID:	120		**PCR PID:**	121	

INDEX	**PID**	**Stream Type: RAW - TYPE**	
1	121	2	VIDEO
2	122	3	AUDIO
3	124	3	AUDIO
4	123	6	TELETEXT
5	601	12	12
6	701	11	11
7	704	11	11
8	702	11	11
9	703	11	11
10	802	6	SUBTITLES
11	902	5	5

Service ID:	803	**Service Name (short name):**	3/24
EIT availability:	Scheduled:	Yes	Now/Next: Yes
PMT PID:	130	**PCR PID:**	131

INDEX	**PID**	**Stream Type: RAW - TYPE**
1	131	2 VIDEO
2	132	3 AUDIO
3	601	12 12
4	701	11 11
5	704	11 11
6	702	11 11
7	703	11 11
8	903	5 5
9	123	6 TELETEXT

Service ID:	804	**Service Name (short name):**	K3/300
EIT availability:	Scheduled:	Yes	Now/Next: Yes
PMT PID:	140	**PCR PID:**	141

INDEX	**PID**	**Stream Type: RAW - TYPE**
1	141	2 VIDEO
2	142	3 AUDIO
3	144	3 AUDIO
4	601	12 12
5	701	11 11
6	704	11 11
7	702	11 11
8	703	11 11
9	904	5 5
10	804	6 SUBTITLES
11	123	6 TELETEXT

Sony BCNTec SW group

Figure 21 – Example of basic HTML report

2.5. Test and Deployment

Test task tries to evaluate or assess product quality. This task is focused in finding and document defects in the software. It is a good advisor of the perceived software quality.

The objective is to validate and prove the assumptions made in design and requirement specifications through concrete demonstration. In other words, it validates that the software product works as designed and that the requirements are implemented appropriately.

This task can have problems if it is done without a clear methodology or if the complexity of the software makes complete testing an impossible goal. Lack of automated test execution sometimes makes this task unsuccessful.

As a summary, testing verifies that the system behaves as required from black box perspective.

For DrTS, this task is difficult because the same person that develops is the testing person. This is a big problem because in some way, the person is influenced from the developing stage and at the time of testing, it is very difficult to abstract from the other role and find the problems that have been introduce in the implementation process.

Taking profit that this is an internal project for internal use, testing task can be executed by the same person because maintenance task can be done with certain ease if some problem is found. Anyway, if some problem was found during use it is desirable to be found in the first release, in this moment, this problem can have a countermeasure or solution to be included in the second release.

For DrTS project, this task has been split in several subtasks that try to cover all the system in different ways to find all the problems or errors that during implementation have not been found or detected. Those tasks are:

- System test: this test wants a general vision of the application. It tries to find problems of the use of the complete system to find the compatibility problems between different actions. This process is not defined and it depends on the person that performs it. This subtask is the most useful to find problems unseen during implementation because it is an open minded vision that usually developer doesn't have. Developer used to be focused only in specific modules without generic vision.
- Verify use cases: this stage is basic to check that no degradation of application's objectives is introduced with all implementations at the same time.
- Report bugs: this subtask is defined to have the trace of problems and to make it easier to debug by means of how to reproduce the problems and details of the problem.
- Re-test bugs: once debugging stage is done, the reported problems should be verified to confirm their solution in the modified software.

- Confirm use cases: this stage wants to check that no degradation of application's objectives is introduced during debugging stage.

Finally, deployment step intends basically to install and run the system. On the other hand, it also includes developing software collateral materials (trainings, release notes, users' documentation and manuals).

One task of this stage is packaging the software in a format that is executable for the client. After this, it is necessary to do beta or smoke tests that are in charge of ensure that the software to be deployed is working fine before the client receives the software.

Deployment can be troubled if there is no good coordination with project stakeholders or if many management interdependencies are needed and it doesn't work perfectly.

In this project this task is summarised in creating the executable file and check basic functionality to know that no elemental problems will appear during use. Tasks are described below.

- Create release package: this task shall create the executable and deployable file to users. At the same time, release note shall be created. This document should contain changes from last release, problem entries that have been resolved, a list of modified files, etc. summarising all the details of the new release regarding the previous one.
- Smoke test of release: this process wants to find possible problems introduced at the time of releasing the package that can cause execution problems in the release. To do this, basic operation are done and checked.

2.6. Operation and Maintenance

These final stages are typically forgotten during scheduling. Its purposes are to maintain software during user operation and to determine whether the product still functions correctly. These objectives are difficult if the design is rigid, if there is a lack of documentation, or if there is staff turnover.

It is necessary to remark that poorly designed and/or implemented software is a critical cost factor. For example maintenance costs are more than 60% of all development costs. Of this maintenance cost: 20% is to correct the product, 30% is to adapt it and 50% is to perfect it.

In this case, those stages will be avoided in the schedule because it is an internal project, and if maintenance is necessary, it will be evaluated in that moment and resource assignment will be decided (or not). There is no action plan in any case because if all stages are done according to this documentation, no urgent problems are foreseen.

The most probable problem (disadvantage) is that the application needs extensions to add some more service information in reports for future utilities. In this situation, another time resources will be estimated and assignment will be decided.

Chapter 3: Conclusions and results

This section summarises the conclusions of this project and results of this execution.

Firstly, in this section should be mentioned that this project is not a purely telecommunications subject itself. It is related with television but as a generic theme, but this project is focused in the software development of an application. Telecommunications degree is not focused in software as a main subject, simply as a tool. From this point of view, this essay focus in a deeper question that is the creation of a tool that makes easier a job related directly with telecommunications such as signal analysis is.

This project gives the opportunity of learning more concepts, ideas and subjects based on the tools and knowledge learned during the degree studies. Besides the application results, this experience is a good way to know all the different roles that a person can have inside the software development process. The situation is not real but as an academic work, this project is very instructive at the time of performing the complete process and taking in each moment a different role that focus in different objectives.

3.1. DrTS results

In this essay, it is described the process from the arising of an idea to the solution implementation. This solution now is real and it is working satisfactorily and improving the daily job of the author and other co-workers.

But the most important result of DrTS implementation is the time saving in analysis and collecting of files. As a reference, some statistic results are shown below.

The comparison is between the time of recording process and the time of analysis.

For recording process, analysis is excluded; this includes only recording the file. For the comparison, two cases have been done, record only a file (of 60 sec) and record each frequency (60 sec) with data of Spanish DTV full bandwidth.

For the human factor, analysis process means the study of SI characteristics of the file with a friendly analyser (not TSReader) without reporting results (that implies more time than shown in the statistic. For DrTS, the analysis process is what is described in this essay, analyse with TSReader, and parse and report in XML format and HTML format.

Then with those conditions, the comparison is in the following table.

Time (min)	Expert	DrTS	Improvement
File recording	1,50	1,08	27,78%
Scan recording	17,50	9,42	46,19%
File analysis	17,00	11,00	35,29%
Total (scan + analysis)	**102,50**	**64,42**	37,15%
		Average	**36,60%**

Table 3 – Comparison of DrTS results

As it can be seen the time saved with DrTS is about 36%, to this result is necessary to add the point that report is already done in two formats and that DrTS can work 24hours every day.

3.2. DrTS inside TS Server

The results of this project itself are the usage of the tool for different members inside the company once testing stage is completed and the tool can be deployed. But in this section an example is shown.

Now DrTS is working inside TS DB management. DB is fulfilled by means of the information that automatic analysis of DrTS provides. It is only needed to ask to DrTS for analyse the file and then the file can be stored and can be found by a searcher based in the fields of DB. The graphical sequence diagram in Figure 22 shows the role of DrTS for this application.

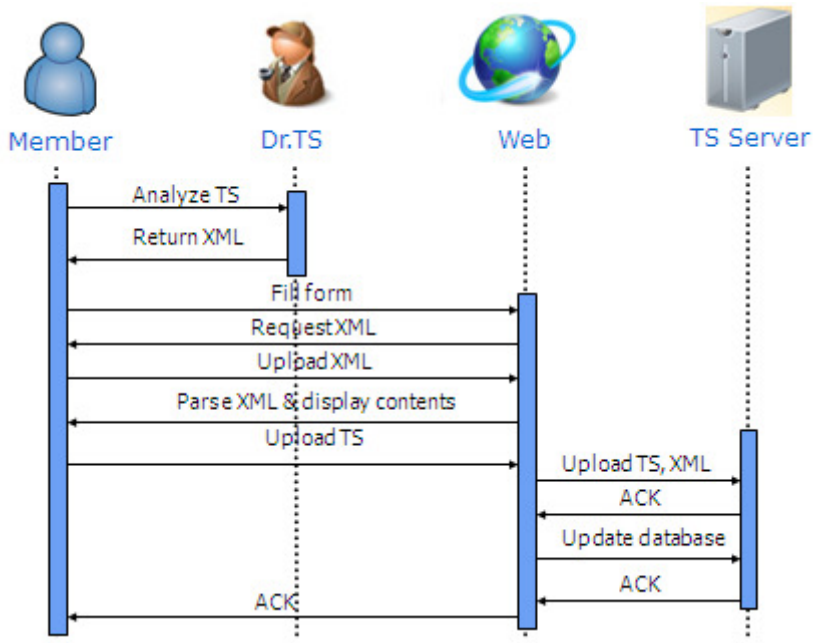


Figure 22 – Sequence diagram from web application with DrTS role

The use of DrTS or another analyser was evaluated before implementation, but it is also good to know that after the evaluation of different similar options to DrTS, the best for this objective was DrTS.

In Figure 23, there is the sequence how a searcher is used by means of DrTS analysis data (that it is stored in TS Server). From this searcher, improvements in time saving are shown in Table 4. Tasks are examples of use cases in searcher that were previously done without this tool.

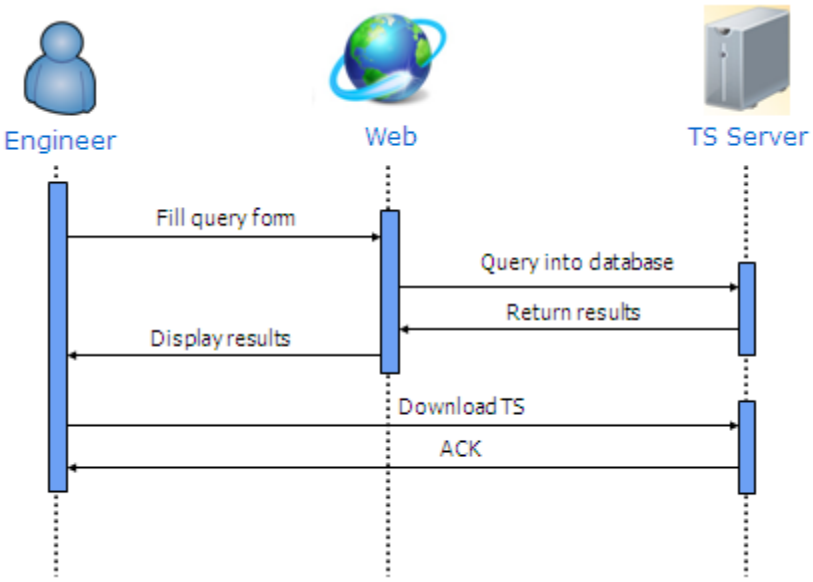


Figure 23 – Sequence diagram from web application for search TS

	Expert	Web Searcher
Provide a TS from DVB-C with Teletext	2 hours	7 min
Provide a TS with AVC and HE-AAC	3 hours	10 min
Search if there is a broadcaster in Europe transmitting in 720p	8 hours	5 min
Search for AAC – ADTS stream	8 hours	15 min
Locate different audio codecs in TS Server	19 hours	1 hour

Table 4 - Comparison of Web Searcher results

It is remarkable that in this use case, time saving is a true statement of the application.

For the time being, there is good acceptation from DrTS users and no complains nor problems found or reported. But as usage of this new tool in tasks related with this subject is still low, definitive results are not clear. Deficiencies can be found, mainly in usability terms; possible improvements are explained later.

3.3. Improvements

Application objectives have been accomplished but in this section some of the faults that have been detected and are not yet solved will be listed.

In the one hand, there are several usability issues that could be improved in new releases. Usability is defined as the ease with which people can employ a particular tool or other human-made object in order to achieve a particular goal. It can also refer to the methods of measuring usability and the study of the principles behind an object's perceived efficiency or elegance. The examples of these faults are:

- The possibility of multiple files selection in analysis GUI is not possible; in case the user wants to analyse many TS files, they should be included in the list one by one.
- Other issues, such as the application analysis speed.

On the other hand, there is another pending issue:

- The availability to work with the different packages of TSReader is not possible. TSReader has a standard and a professional version that allows the XML report in which DrTS is based. The problem is that the professional executable name is different from the standard version one, this causes that command lines used to control TSReader in DrTS are not valid (XML results are the same). This problem can be solved easily but need to be evaluated if it is necessary to be solved. Alternative options are:
    - As both versions can run in the same environment, the user can have both packages and use one with DrTS and another for other purposes.
    - If only the professional version is available, the option is to change the executable name to "TSReader.exe"

No more significant improvements have been found so far. Consequently, maybe there is no need for solving them if no more releases are done. Maybe the best option is to leave those problems reported and in case future prospects were implemented, then solve them at same time, with this methodology, we avoid the necessity of a new release process (including testing/debugging stages).

3.4. Future prospects

Finally, future prospects are described in order to show some of the possibilities that this implementation leaves open inside the digital television signal management and analysis.

First, DrTS can be expanded in service information terms. All the information that has not been included could be included in reports, and then the DB could be expanded and TS could be found with more efficiency. TS files monitoring is also possible with more information mapped, and statistics can be created too.

DrTS could be also improved if TSReader dependency is avoided. This dependency in this development was needed because the development of a tool for demultiplexing and recording could be difficult, problematic, and, above all, expensive.

Finally, improve and complete all the process of analysis and control of this information can be a very good idea that will save time in the future. An idea of a full implementation can be that this tool could control air transmissions of broadcasters in any country to detect changes in the emissions by means of periodic analysis. This could be done by remote control of HW in each country and launch analysis that could be configured from the application. Once new recordings were obtained, they could be analysed and then compared with previous ones of the same network and frequency. After this a delta report could be created (showing the changes in DVB signalization from one to another) and be sent to an administrator or responsible.

This last feature could avoid most of the market problems because situation changes could be detected in advance and then make easier and faster to solve.

More future prospects could surely be thought from use and experience. But at this moment those are the ones that have been arisen.

Annexes

Tables definition

Tables that are not mandatory or important inside this essay are not described in detail from each field, because it is not the objective of this introduction to define all the detailed DVB information. Inform about the structure and semantics is enough for the objectives' explanation.

First, tables are sent into sections. Sections can be variable in length limited to 1024 bytes to be inserted into TS packets (except EIT sections that are limited to 4096 bytes). Each section is uniquely identified by the combination of some of the following elements:

- table_id: table_id identifies to which table the section belongs. Some table_ids have been defined by ISO and others by ETSI. Other values of the table_id can be allocated by the user for private purposes.

Value	Description
0x00	program_association_section
0x01	conditional_access_section
0x02	program_map_section
0x03	transport_stream_description_section
0x04 to 0x3F	Reserved
0x40	network_information_section - actual_network
0x41	network_information_section - other_network
0x42	service_description_section - actual_transport_stream
0x43 to 0x45	reserved for future use
0x46	service_description_section - other_transport_stream
0x47 to 0x49	reserved for future use
0x4A	bouquet_association_section
0x4B to 0x4D	reserved for future use
0x4E	event_information_section - actual_transport_stream, present/following
0x4F	event_information_section - other_transport_stream, present/following
0x50 to 0x5F	event_information_section - actual_transport_stream, schedule
0x60 to 0x6F	event_information_section - other_transport_stream, schedule
0x70	time_date_section
0x71	running_status_section
0x72	stuffing_section
0x73	time_offset_section
0x74	application information section (TS 102 812)
0x75	container section (TS 102 323)
0x76	related content section (TS 102 323)
0x77	content identifier section (TS 102 323)
0x78	MPE-FEC section (EN 301 192)
0x79	resolution notification section (TS 102 323)
0x7A	MPE-IFEC section (TS 102 772)

Value	Description
0x7B to 0x7D	reserved for future use
0x7E	discontinuity_information_section
0x7F	selection_information_section
0x80 to 0xFE	user defined
0xFF	Reserved

Table 5 – Allocation of table_id values

- section_syntax_indicator: The section_syntax_indicator is a 1 bit field which shall be set to '1'.
- section_length: This is a twelve bit field, the first two bits of which shall be '00'. It specifies the number of bytes of the section, starting immediately following the section_length field, and including the CRC. The value in this field shall not exceed 1021.
- table_id_extension: table_id_extension is used for identification of a sub_table.
- section_number: section_number field allows the sections of a particular sub_table to be reassembled in their original order by the decoder. It is recommended, that sections are transmitted in numerical order, unless it is desired to transmit some sections of the sub_table more frequently than others, e.g. due to random access considerations.
- last_section_number: This 8 bit field specifies the number of the last section (that is, the section with the highest section_number) of the complete corresponding table. In case of PMT, the value of this field shall be always 0x00.
- version_number: When the characteristics of the TS described in the SI given (e.g. new events start, different composition of elementary streams for a given service), then new SI data shall be sent containing the updated information. A new version of the SI data is signalled by sending a sub_table with the same identifiers as the previous sub_table containing the relevant data, but with the next value of version_number.
- current_next_indicator: Each section shall be numbered as valid "now" (current), or as valid in the immediate future (next). This allows the transmission of a future version of the SI in advance of the change, giving the decoder the opportunity to prepare for the change. There is however, no requirement to transmit the next version of a section in advance, but if it is transmitted, then it shall be the next correct version of that section.

Sections may start at the beginning of the payload of a TS packet, but this is not a requirement. There is never more than one pointer field in a TS packet, as the start of any other section can be identified by counting the length of the first and any subsequent sections, since no gaps between sections within a TS packet are allowed by the syntax.

In systems where random access is considered, it is recommended to re-transmit even when changes do not occur in the configuration. For SI the minimum time interval between the arrival of the last byte of a section to the first byte of the next transmitted section with the same PID, table_id and table_id_extension, and with the same or different section_number shall be 25 ms. This limit applies for TS with a total data rate of up to 100 Mbit/s.

Program Association Table

Syntax	Number of bits	Identifier
program_association_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
'0'	1	bslbf
Reserved	2	bslbf
section_length	12	uimsbf
transport_stream_id	16	uimsbf
Reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
for(i=0;i<N;i++){		
program_number	16	uimsbf
Reserved	3	bslbf
if (program_number == '0'){		
network_PID	13	uimsbf
}		
else{		
program_map_PID	13	uimsbf
}		
}		
CRC_32	32	rpchof
}		

Table 6 – PAT definition

transport_stream_id: This is a 16 bit field which serves as a label to identify this Transport Stream from any other multiplex within a network. Its value is defined by the user.

program_number: Program_number is a 16 bit field. It specifies the program to which the program_map_PID is applicable. If this is set to 0x0000 then the following PID reference shall be the network PID. For all other cases the value of this field is user defined. This field shall not take any single value more than once within one version of the program association table. The program_number may be used as a designation for a broadcast channel, for example.

network_PID: network_PID is a 13 bit field specifying the PID of the Transport Stream packets which shall contain the Network Information Table. The value of the network_PID field is defined by the user. The presence of the network_PID is optional.

program_map_PID: program_map_PID is a 13 bit field specifying the PID of the Transport Stream packets which shall contain the program_map_section applicable for the program as specified by the program_number. No program_number shall have more than one program_map_PID assignment. The value of the program_map_PID is defined by the user.

Program Map Table

Syntax	Number of bits	Identifier
program_association_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
'0'	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
program_number	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
reserved	3	bslbf
PCR_PID	13	uimsbf
reserved	4	bslbf
program_info_length	12	uimsbf
for(i=0;i<N;i++){		
descriptor()		
}		
for(i=0;i<N1;i++){		
stream_type	8	uimsbf
reserved	3	bslbf
elementary_PID	13	uimsbf
reserved	4	bslbf
ES_info_length	12	uimsbf
for(j=0;j<N2;j++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

Table 7 – PMT definition

program_number: It specifies the program to which the program_map_PID is applicable. One program definition shall be carried within only one TS_program_map_section. This implies that a program definition is never longer than 1016 bytes.

PCR_PID: This is a 13 bit field indicating the PID of the Transport Stream packets which shall contain the PCR fields valid for the program specified by program_number. If no PCR is associated with a program definition for private streams then this field shall take the value of 0x1FFF.

program_info_length: This is a 12 bit field, the first two bits of which shall be '00'. It specifies the number of bytes of the descriptors immediately following the program_info_length field.

stream_type: This is an 8 bit field specifying the type of program element carried within the packets with the ID whose value is specified by the elementary_PID. The values of stream_type are specified in the following table.

Value	Description
0x00	ITU-T | ISO/IEC Reserved
0x01	ISO/IEC 11172 Video

Value	Description
0x02	ITU-T Rec. H.262 | ISO/IEC 13818-2 Video or ISO/IEC 11172-2 constrained parameter video stream
0x03	ISO/IEC 11172 Audio
0x04	ISO/IEC 13818-3 Audio
0x05	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 private sections
0x06	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 PES packets containing private data
0x07	ISO/IEC 13522 MHEG
0x08	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 - DSM CC
0x09	ITU-T Rec. H.222.1
0x0A	ISO/IEC 13818-6 type A
0x0B	ISO/IEC 13818-6 type B
0x0C	ISO/IEC 13818-6 type C
0x0D	ISO/IEC 13818-6 type D
0x0E	ISO/IEC 13818-1 auxiliary
0x0F-0x7F	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Reserved
0x80-0xFF	User Private

Table 8 – Stream_type assignments

elementary_PID: This is a 13 bit field specifying the PID of the Transport Stream packets which carry the associated program element.

ES_info_length: This is a 12 bit field, the first two bits of which shall be '00'. It specifies the number of bytes of the descriptors of the associated program element immediately following the ES_info_length field.

Conditional Access Table

Syntax	Number of bits	Identifier
CA_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
'0'	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
reserved	18	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
for(i=0;i<N;i++){		
descriptor()		
}		
CRC_32	32	rpchof
}		

Table 9 – CAT definition

Network Information Table

Syntax	Number of bits	Identifier
network_information_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
network_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
reserved_future_use	4	bslbf
network_descriptors_length	12	uimsbf
for(i=0;i<N;i++){		
descriptor()		
}		
reserved_future_use	4	bslbf
transport_stream_loop_length	12	uimsbf
for(i=0;i<N;i++){		
transport_stream_id	16	uimsbf
original_network_id	16	uimsbf
reserved_future_use	4	bslbf
transport_descriptors_length	12	uimsbf
for(j=0;j<N;j++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

Table 10 – NIT definition

network_id: This is a 16-bit field which serves as a label to identify the delivery system, about which the NIT informs, from any other delivery system. Allocations of the value of this field are found in TR 101 162.

network_descriptors_length: This 12-bit field gives the total length in bytes of the following network descriptors.

transport_stream_loop_length: This is a 12-bit field specifying the total length in bytes of the TS loops that follow, ending immediately before the first CRC-32 byte.

transport_stream_id: This is a 16-bit field which serves as a label for identification of this TS from any other multiplex within the delivery system.

original_network_id: This 16-bit field gives the label identifying the network_id of the originating delivery system.

transport_descriptors_length: This is a 12-bit field specifying the total length in bytes of TS descriptors that follow.

Bouquet Association Table

Syntax	Number of bits	Identifier
bouquet_association_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
bouquet_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
reserved_future_use	4	bslbf
bouquet_descriptors_length	12	uimsbf
for(i=0;i<N;i++){		
descriptor()		
}		
reserved_future_use	4	bslbf
transport_stream_loop_length	12	uimsbf
for(i=0;i<N;i++){		
transport_stream_id	16	uimsbf
original_network_id	16	uimsbf
reserved_future_use	4	bslbf
transport_descriptors_length	12	uimsbf
for(j=0;j<N;j++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

Table 11 – BAT definition

Service Description Table

Syntax	Number of bits	Identifier
service_description_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
transport_stream_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
original_network_id	16	uimsbf
reserved_future_use	8	bslbf
for (i=0;i<N;i++){		
service_id	16	uimsbf
reserved_future_use	6	bslbf
EIT_schedule_flag	1	bslbf
EIT_present_following_flag	1	bslbf
running_status	3	uimsbf
free_CA_mode	1	bslbf
descriptors_loop_length	12	uimsbf
for (j=0;j<N;j++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

Table 12 – SDT definition

transport_stream_id: This is a 16-bit field which serves as a label for identification of the TS, about which the SDT informs, from any other multiplex within the delivery system.

original_network_id: This 16-bit field gives the label identifying the network_id of the originating delivery system.

service_id: This is a 16-bit field which serves as a label to identify this service from any other service within the TS. The service_id is the same as the program_number in the corresponding program_map_section.

EIT_schedule_flag: This is a 1-bit field which when set to "1" indicates that EIT schedule information for the service is present in the current TS. If the flag is set to 0 then the EIT schedule information for the service should not be present in the TS.

EIT_present_following_flag: This is a 1-bit field which when set to "1" indicates that EIT_present_following information for the service is present in the current TS. If the flag is set to 0 then the EIT present/following information for the service should not be present in the TS.

running_status: This is a 3-bit field indicating the status of the service as defined in the table below.

Value	Meaning
0	undefined
1	not running
2	starts in a few seconds (e.g. for video recording)
3	pausing
4	running
5	service off-air
6 to 7	reserved for future use

Table 13 – Running status assignments

For an NVOD reference service the value of the running_status shall be set to "0".

free_CA_mode: This 1-bit field, when set to "0" indicates that all the component streams of the service are not scrambled. When set to "1" it indicates that access to one or more streams may be controlled by a CA system.

descriptors_loop_length: This 12-bit field gives the total length in bytes of the following descriptors.

Event Information Table

Four classifications of EIT have been identified, distinguishable by the use of different table_ids:

1) actual TS, present/following event information = table_id = "0x4E";
2) other TS, present/following event information = table_id = "0x4F";
3) actual TS, event schedule information = table_id = "0x50" to "0x5F";
4) other TS, event schedule information = table_id = "0x60" to "0x6F".

Syntax	Number of bits	Identifier
event_information_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
service_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
transport_stream_id	16	uimsbf
original_network_id	16	uimsbf
segment_last_section_number	8	uimsbf
last_table_id	8	uimsbf
for(i=0;i<N;i++){		
event_id	16	uimsbf
start_time	40	bslbf
duration	24	uimsbf
running_status	3	uimsbf
free_CA_mode	1	bslbf
descriptors_loop_length	12	uimsbf
for(i=0;i<N;i++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

Table 14 – EIT definition

transport_stream_id: This is a 16-bit field which serves as a label for identification of the TS, about which the EIT informs, from any other multiplex within the delivery system.

original_network_id: This 16-bit field gives the label identifying the network_id of the originating delivery system.

segment_last_section_number: This 8-bit field specifies the number of the last section of this segment of the sub_table. For sub_tables which are not segmented, this field shall be set to the same value as the last_section_number field.

last_table_id: This 8-bit field identifies the last table_id used.

event_id: This 16-bit field contains the identification number of the described event (uniquely allocated within a service definition).

start_time: This 40-bit field contains the start time of the event in Universal Time, Co-ordinated (UTC) and Modified Julian Date (MJD). This field is coded as 16 bits giving the 16 LSBs of MJD followed by 24 bits coded as 6 digits in 4-bit Binary Coded Decimal (BCD). If the start time is undefined (e.g. for an event in a NVOD reference service) all bits of the field are set to "1".

EXAMPLE: 93/10/13 12:45:00 is coded as "0xC079124500".

duration: A 24-bit field containing the duration of the event in hours, minutes, seconds. format: 6 digits, 4-bit BCD = 24 bit.

EXAMPLE: 01:45:30 is coded as "0x014530".

running_status: This is a 3-bit field indicating the status of the event as defined in Table 13. For an NVOD reference event the value of the running_status shall be set to "0".

free_CA_mode: This 1-bit field, when set to "0" indicates that all the component streams of the event are not scrambled. When set to "1" it indicates that access to one or more streams is controlled by a CA system.

descriptors_loop_length: This 12-bit field gives the total length in bytes of the following descriptors.

Time and Date Table

Syntax	Number of bits	Identifier
time_date_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
UTC_time	40	bslbf
}		

Table 15 – TDT definition

UTC_time: This 40-bit field contains the current time and date in UTC and MJD. This field is coded as 16 bits giving the 16 LSBs of MJD followed by 24 bits coded as 6 digits in 4-bit BCD.

EXAMPLE: 93/10/13 12:45:00 is coded as "0xC079124500".

Time Offset Table

Syntax	Number of bits	Identifier
time_offset_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
UTC_time	40	bslbf
reserved	4	bslbf
descriptors_loop_length	12	uimsbf
for(i=0;i<N;i++){		
descriptor()		
}		
CRC_32	32	rpchof
}		

Table 16 – TOT definition

Descriptors definition

Video stream descriptor

The video stream descriptor provides basic information which identifies the coding parameters of a video elementary stream as described in ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 11172-2.

Syntax	Number of bits	Identifier
video_stream_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
multiple_frame_rate_flag	1	bslbf
frame_rate_code	4	uimsbf
MPEG_1_only_flag	1	bslbf
constrained_parameter_flag	1	bslbf
still_picture_flag	1	bslbf
if (MPEG_1_only_flag == 1){		
profile_and_level_indication	8	uimsbf
chroma_format	2	uimsbf
frame_rate_extension_flag	1	bslbf
reserved	5	bslbf
}		
}		

Table 17 – Video stream descriptor

multiple_frame_rate_flag: This is a 1 bit field which when set to '1' indicates that multiple frame rates may be present in the video stream. When set to a value of '0' only a single frame rate is present.

frame_rate_code: This is a 4 bit field as defined in ITU-T Rec. H.262 | ISO/IEC 13818-2, except that when the multiple_frame_rate_flag is set to a value of '1' the indication of a particular frame rate also permits certain other frame rates to be present in the video stream, as specified below:

coded as	also includes
23,976	
24,0	23,976
25,0	
29,97	23,976
30,0	23,976 24,0 29,97
50,0	25,0
59,94	23,976 29,97
60,0	23,976 24,0 29,97 30,0 59,94

Table 18 – Frame rate code

MPEG_1_only_flag: This is a 1 bit field which when set to '1' indicates that the video stream contains only ISO/IEC 11172-2 data. If set to '0' the video stream may contain both ISO/IEC 13818-2 video data and constrained parameter ISO/IEC 11172-2 video data.

constrained_parameter_flag: This is a 1 bit field which when set to '1' indicates that the video stream shall not contain unconstrained ISO/IEC 11172-2 video data. If this field is set to '0' the video stream mat contain both constrained parameters and unconstrained ISO/IEC 11172-2 video streams. If the MPEG_1_only_flag is set to '0', the constrained_parameter_flag shall be set to '1'.

still_picture_flag: This is a 1 bit field, which when set to '1' indicates that the video stream contains only still pictures. If the bit is set to '0' then the video stream may contain either moving or still picture data.

profile_and_level_indication: This is an 8 bit field which is set to the same value as the profile_and_level_indication fields in the video stream.

chroma_format: This is a 2 bit field which is set to the same value as the chroma_format fields in the ITU-T Rec. H.262 | ISO/IEC 13818-2 video stream.

frame_rate_extension_flag: This is a 1 bit flag which when set to '1' indicates that either or both of the frame_rate_extension_n and frame_rate_extension_d fields in the ITU-T Rec. H.262 | ISO/IEC 13818-2 video stream are non-zero.

Audio stream descriptor

The audio stream descriptor provides basic information which identifies the coding version of an audio elementary stream as described in ISO/IEC 13818-3 or ISO/IEC 11172-3.

Syntax	Number of bits	Identifier
audio_stream_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
free_format_flag	1	bslbf
ID	1	bslbf
layer	2	bslbf
variable_rate_audio_indicator	1	bslbf
reserved	3	bslbf
}		

Table 19 – Audio stream descriptor

free_format_flag: This is a 1 bit field which when set to '1' indicates that the audio stream data has the bitrate_index set to '0000'. If set to '0' then the bitrate_index is not '0000'.

ID: This is a 1 bit field which is set to the same value as the ID fields in the audio stream.

layer: This is a 2 bit field which is set to the same value as the layer fields in the audio stream.

variable_rate_audio_indicator: This is a 1 bit flag, which when set to '1' indicates that the associated audio stream may contain audio frames in which the bit rate changes.

In all cases the audio stream is intended to be presented without any decoding discontinuity

ISO 639 Language descriptor

The language descriptor is used to specify the language of the associated program element

Syntax	Number of bits	Identifier
ISO_639_language_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for (i=0;i<N;i++){		
ISO_639_language_code	24	bslbf
audio_type	8	bslbf
}		
}		

Table 20 – ISO 639 language descriptor

ISO_639_language_code: Identifies the language or languages used by the associated program element. The ISO_639_language_code contains a 3 character code as specified by ISO 639 part 2. Each character is coded into 8 bits according to ISO 8859-1 and inserted in order into the 24 bit field. In the case of multilingual audio streams the sequence of ISO_639_language_code fields shall reflect the content of the audio stream.

audio_type: The audio_type is an 8 bit field which specifies the type of stream defined by the following table.

value	description
0x00	Undefined
0x01	Clean effects
0x02	Hearing impaired
0x03	Visual impaired commentary
0x04 – 0x0F	reserved

Table 21 – Audio type values

Network Name descriptor

The network name descriptor provides the network name in text form.

Syntax	Number of bits	Identifier
network_name_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for (i=0;i<N;i++){		
char	8	uimsbf
}		
}		

Table 22 – Network name descriptor

char: This is an 8-bit field. A string of char fields specify the name of the delivery system about which the NIT informs.

Service descriptor

The service descriptor provides the names of the service provider and the service in text form together with the service_type.

Syntax	Number of bits	Identifier
service_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
service_type	8	uimsbf
service_provider_name_length	8	uimsbf
for (i=0;i<N;I++){		
char	8	uimsbf
}		
service_name_length	8	uimsbf
for (i=0;i<N;I++){		
Char	8	uimsbf
}		
}		

Table 23 – Service descriptor

service_type: This is an 8-bit field specifying the service type. The assignment of service_type value for a service shall be coded according to following table.

service_type	Description
0x00	reserved for future use
0x01	digital television service (see note 1)
0x02	digital radio sound service (see note 2)
0x03	Teletext service
0x04	NVOD reference service (see note 1)
0x05	NVOD time-shifted service (see note 1)
0x06	mosaic service
0x07	FM radio service
0x08	DVB SRM service [49]
0x09	reserved for future use
0x0A	advanced codec digital radio sound service
0x0B	advanced codec mosaic service
0x0C	data broadcast service
0x0D	reserved for Common Interface Usage (EN 50221)
0x0E	RCS Map (see EN 301 790)
0x0F	RCS FLS (see EN 301 790)
0x10	DVB MHP service
0x11	MPEG-2 HD digital television service
0x12 to 0x15	reserved for future use
0x16	advanced codec SD digital television service
0x17	advanced codec SD NVOD time-shifted service
0x18	advanced codec SD NVOD reference service
0x19	advanced codec HD digital television service
0x1A	advanced codec HD NVOD time-shifted service
0x1B	advanced codec HD NVOD reference service
0x1C to 0x7F	reserved for future use
0x80 to 0xFE	user defined
0xFF	reserved for future use
NOTE 1: MPEG-2 SD material should use this type. NOTE 2: MPEG-1 Layer 2 audio material should use this type.	

Table 24 – Service type coding

service_provider_name_length: This 8-bit field specifies the number of bytes that follow the service_provider_name_length field for describing characters of the name of the service provider.

char: This is an 8-bit field. A string of char fields specify the name of the service provider or service.

service_name_length: This 8-bit field specifies the number of bytes that follow the service_name_length field for describing characters of the name of the service.

Service List descriptor

The service list descriptor provides a means of listing the services by service_id and service type.

Syntax	Number of bits	Identifier
service_list_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for (i=0;i<N;I++){		
service_id	16	uimsbf
service_type	8	uimsbf
}		
}		

Table 25 – Service list descriptor

service_id: This is a 16-bit field which uniquely identifies a service within TS. The service_id is the same as the program_number in the corresponding program_map_section, except that in the case of service_type = 0x04, 0x18 or 0x1B (NVOD reference services) the service_id does not have a corresponding program_number.

service_type: This is an 8-bit field specifying the type of the service. The assignment of service_type value shall be coded according to Table 24.

Satellite delivery system descriptor

This descriptor shall be used when a satellite delivery system is used to broadcast the recipient TS.

Syntax	Number of bits	Identifier
satellite_delivery_system_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
Frequency	32	bslbf
orbital_position	16	bslbf
west_east_flag	1	bslbf
Polarization	2	bslbf
If (modulation_system == "1") {		
roll off	2	bslbf
} else {		
"00"	2	bslbf
}		
modulation_system	1	bslbf
modulation_type	2	bslbf
symbol_rate	28	bslbf
FEC_inner	4	bslbf
}		

Table 26 – Satellite delivery system descriptor

frequency: The frequency is a 32-bit field giving the 4-bit BCD values specifying 8 characters of the frequency value. For the satellite_delivery_system_descriptor the frequency is coded in GHz, where the decimal point occurs after the third character (e.g. 011.75725 GHz).

orbital_position: The orbital_position is a 16-bit field giving the 4-bit BCD values specifying 4 characters of the orbital position in degrees where the decimal point occurs after the third character (e.g. 019.2°).

west_east_flag: The west_east_flag is a 1-bit field indicating if the satellite position is in the western or eastern part of the orbit. A value "0" indicates the western position and a value "1" indicates the eastern position.

polarization: The polarization is a 2-bit field specifying the polarization of the transmitted signal. The first bit defines whether the polarization is linear or circular.

Polarization	Description
00	linear – horizontal
01	linear – vertical
10	Circular – left
11	Circular – right

Table 27 – Polarization values

roll_off: This 2 bit field specifies the roll-off factor used in DVB-S2.

roll-off	Description
00	α = 0,35
01	α = 0,25
10	α = 0,20
11	reserved

Table 28 – Roll-off values

modulation_system: This is a 1-bit field. It specifies the broadcast scheme used on a satellite delivery system according to the following table.

modulation system	Description
0	DVB-S
1	DVB-S2

Table 29 – Modulation system for satellite

modulation_type: This is a 2-bit field. It specifies the modulation scheme used on a satellite delivery system according to following table.

modulation type	Description
00	Auto
01	QPSK
10	8PSK
11	16-QAM (n/a for DVB-S2)

Table 30 – Modulation type for satellite

symbol_rate: The symbol_rate is a 28-bit field giving the 4-bit BCD values specifying 7 characters of the symbol_rate in Msymbol/s where the decimal point occurs after the third character (e.g. 027,4500).

FEC_inner: The FEC_inner is a 4-bit field specifying the inner FEC scheme used according to the following table.

FEC_inner bit 3210	Description
0000	not defined
0001	1/2 conv. code rate
0010	2/3 conv. code rate
0011	3/4 conv. code rate
0100	5/6 conv. code rate
0101	7/8 conv. code rate
0110	8/9 conv. code rate
0111	3/5 conv. code rate
1000	4/5 conv. code rate
1001	9/10 conv. code rate
1010 to 1110	reserved for future use
1111	no conv. Coding
NOTE: Not all convolutional code rates apply for all modulation schemes.	

Table 31 – Inner FEC scheme

Cable delivery system descriptor

This descriptor shall be used when a cable delivery system is used to broadcast the recipient TS.

Syntax	Number of bits	Identifier
cable_delivery_system_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
Frequency	32	bslbf
reserved_future_use	12	bslbf
FEC_outer	4	bslbf
Modulation	8	bslbf
symbol_rate	28	bslbf
FEC_inner	4	bslbf
}		

Table 32 – Cable delivery system descriptor

frequency: The frequency is a 32-bit field giving the 4-bit BCD values specifying 8 characters of the frequency value. For the cable_delivery_system_descriptor, the frequency is coded in MHz, where the decimal occurs after the fourth character (e.g. 0312.0000 MHz).

FEC_outer: The FEC_outer is a 4-bit field specifying the outer Forward Error Correction (FEC) scheme used according to the following table.

FEC_outer bit 3210	Description
0000	not defined
0001	no outer FEC coding
0010	RS(204/188)
0011 to 1111	reserved for future use

Table 33 – Outer FEC scheme

modulation: It specifies the modulation scheme used on a cable delivery system according to following table.

Modulation (hex)	Description
0x00	not defined
0x01	16-QAM
0x02	32-QAM
0x03	64-QAM
0x04	128-QAM
0x05	256-QAM
0x06 to 0xFF	reserved for future use

Table 34 – Modulation scheme for cable

symbol_rate: The symbol_rate is a 28-bit field giving the 4-bit BCD values specifying 7 characters of the symbol_rate in Msymbol/s where the decimal point occurs after the third character (e.g. 027.4500).

FEC_inner: The FEC_inner is a 4-bit field specifying the inner FEC scheme used according to following table.

FEC_inner bit 3210	Description
0000	not defined
0001	1/2 conv. code rate
0010	2/3 conv. code rate
0011	3/4 conv. code rate
0100	5/6 conv. code rate
0101	7/8 conv. code rate
0110	8/9 conv. code rate
0111	3/5 conv. code rate
1000	4/5 conv. code rate
1001	9/10 conv. code rate
1010 to 1110	reserved for future use
1111	no conv. Coding
NOTE: Not all convolutional code rates apply for all modulation schemes.	

Table 35 – Inner FEC scheme

Terrestrial delivery system descriptor

This descriptor shall be used when a terrestrial delivery system is used to broadcast the recipient TS.

Syntax	Number of bits	Identifier
terrestrial_delivery_system_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
centre_frequency	32	uimsbf
Bandwidth	3	bslbf
Priority	1	bslbf
Time_Slicing_indicator	1	bslbf
MPE-FEC_indicator	1	bslbf
reserved_future_use	2	bslbf
Constellation	2	bslbf
hierarchy_information	3	bslbf
code_rate-HP_stream	3	bslbf
code_rate-LP_stream	3	bslbf
guard_interval	2	bslbf
transmission_mode	2	bslbf
other_frequency_flag	1	bslbf
reserved_future_use	32	bslbf
}		

Table 36 – Terrestrial delivery system descriptor

centre_frequency: The centre_frequency is a 32-bit field giving the centre frequency value in multiples of 10 Hz. The coding range is from minimum 10 Hz (0x00000001) up to a maximum of 42 949 672 950 Hz (0xFFFFFFFF).

bandwidth: This is a 3-bit field specifying the bandwidth in use.

Bandwidth	Bandwidth value
000	8 MHz
001	7 MHz
010	6 MHz
011	5 MHz
100 to 111	Reserved for future use

Table 37 – Signalling format for the bandwidth

priority: This 1-bit flag indicates the stream's hierarchical priority. In case the hierachy_information field is not equal to "000", if priority is set to "1", it indicates that the associated transport stream is a HP stream, and if priority is set to "0", the associated transport stream is a LP stream. In case the hierarchy_information field has the value "000", the priority flag shall be set to "1".

priority	Description
1	HP (high priority)
0	LP (low priority)

Table 38 – Signalling format for the priority

time_slicing_indicator: This 1-bit field indicates the use of the Time Slicing on the associated transport stream. If the Time_Slicing_indicator is set ("1"), Time Slicing is not used. If the Time_Slicing_indicator is cleared ("0"), at least one elementary stream uses Time Slicing.

MPE-FEC_indicator: This 1-bit field indicates the use of the MPE-FEC on the associated transport stream. If the MPE-FEC_indicator is set ("1"), MPE-FEC is not used. If the MPE-FEC_indicator is cleared ("0"), at least one elementary stream uses MPE-FEC.

constellation: This is a 2-bit field. It specifies the constellation pattern used on a terrestrial delivery system according to following table.

Constellation	Constellation characteristics
00	QPSK
01	16-QAM
10	64-QAM
11	reserved for future use

Table 39 – Signalling format for the possible constellation patterns

hierarchy_information: The hierarchy_information specifies whether the transmission is hierarchical and, if so, what the α value is. Also, the use of in-depth interleaver is indicated. When the transmission_mode indicates the use of 8k mode, only the native interleaver shall be signalled.

Hierarchy_information	α value
000	non-hierarchical, native interleaver
001	α = 1, native interleaver
010	α = 2, native interleaver
011	α = 4, native interleaver
100	non-hierarchical, in-depth interleaver
101	α = 1, in-depth interleaver
110	α = 2, in-depth interleaver
111	α = 4, in-depth interleaver

Table 40 – Signalling format for the α values and the used interleaver

code_rate: The code_rate is a 3-bit field specifying the inner FEC scheme used according to following table. Non-hierarchical channel coding and modulation requires signalling of one code rate. In this case, 3 bits specifying code_rate according to table 47 are followed by another 3 bits of value "000". Two different code rates may be applied to two different levels of modulation with the aim of achieving hierarchy. Transmission then starts with the code rate for the HP level of the modulation and ends with the one for the LP level.

code_rate	Description
000	½
001	2/3
010	¾
011	5/6
100	7/8
101 to 111	reserved for future use

Table 41 – Signalling format for each of the code rates

guard_interval: The guard_interval is a 2-bit field specifying the guard interval according to following table.

guard_interval	Guard interval values
00	1/32
01	1/16
10	1/8
11	¼

Table 42 – Signalling format for each of the guard interval values

transmission_mode: This 2-bit field indicates the number of carriers in an OFDM frame.

transmission_mode	Description
00	2k mode
01	8k mode
10	4k mode
11	reserved for future use

Table 43 – Signalling format for transmission mode

other_frequency_flag: This 1-bit flag indicates whether other frequencies are in use. The value "0" indicates that no other frequency is in use, "1" indicates that one or more other frequencies are in use.

Linkage descriptor

Linkage descriptor identifies a service that can be presented if the consumer request for additional information related to a specific entity described in SI system. For example a linkage descriptor located within the NIT shall point to a service providing additional information on the network, a linkage descriptor in the BAT shall provide a link to a service informing about the bouquet, etc

A service replacement service can also be identified using the linkage_descriptor. This replacement service may be selected automatically by the decoder when the running status of the current service is set to "not_running".

Syntax	Number of bits	Identifier
linkage_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
transport_stream_id	16	uimsbf
original_network_id	16	uimsbf
service_id	16	uimsbf
linkage_type	8	uimsbf
if (linkage_type ==0x08){		
hand-over_type	4	bslbf
reserved_future_use	3	bslbf
origin_type	1	bslbf
if (hand-over_type ==0x01		
|| hand-over_type ==0x02		
|| hand-over_type ==0x03){		
network_id	16	uimsbf
}		
if (origin_type ==0x00){		
initial_service_id	16	uimsbf
}		
}		
if (linkage_type == 0x0D){		
target_event_id	16	uimsbf
target_listed	1	bslbf
event_simulcast	1	bslbf
Reserved	6	bslbf
}		
for (i=0;i<N;i++){		
private_data_byte	8	bslbf
}		
}		

Table 44 – Linkage descriptor

transport_stream_id: This is a 16-bit field which identifies the TS containing the information service indicated.

original_network_id: This 16-bit field gives the label identifying the network_id of the originating delivery system of the information service indicated.

service_id: This is a 16-bit field which uniquely identifies an information service within TS. The service_id is the same as the program_number in the corresponding program_map_section. If the linkage_type field has the value 0x04, then the service_id field is not relevant, and shall be set to 0x0000.

linkage_type: This is an 8-bit field specifying the type of linkage e.g. to information.

Linkage_type	Description
0x00	reserved for future use
0x01	information service
0x02	EPG service
0x03	CA replacement service
0x04	TS containing complete Network/Bouquet SI
0x05	service replacement service
0x06	data broadcast service
0x07	RCS Map
0x08	mobile hand-over
0x09	System Software Update Service (TS 102 006)
0x0A	TS containing SSU BAT or NIT (TS 102 006)
0x0B	IP/MAC Notification Service (EN 301 192)
0x0C	TS containing INT BAT or NIT (EN 301 192)
0x0D	event linkage (see note)
0x0E to 0x7F	reserved for future use
0x80 to 0xFE	user defined
0xFF	reserved for future use
NOTE: A linkage_type with value 0x0D is only valid when the descriptor is carried in the EIT.	

Table 45 – Linkage type coding

hand-over_type: This is a 4-bit field specifying the type of hand-over.

Hand-over_type	Description
0x00	reserved for future use
0x01	DVB hand-over to an identical service in a neighbouring country
0x02	DVB hand-over to a local variation of the same service
0x03	DVB hand-over to an associated service
0x04 to 0x0F	reserved for future use

Table 46 – Hand-over type coding

origin_type: This is a flag specifying in which table the link is originated.

Origin_type	Description
0x00	NIT
0x01	SDT

Table 47 – Origin type coding

network_id: This is a 16-bit field which identifies the terrestrial network that supports the service indicated.

initial_service_id: This is a 16-bit field which identifies the service for which the hand-over linkage is valid.

target_event_id: This 16-bit field identifies the event_id of the event (the target event), carried on the service defined by the original_network_id, transport_stream_id and service_id, which is equivalent to the event identified by the location of this descriptor (the source event).

target_listed: This 1-bit field signals whether the service defined by the original_network_id, transport_stream_id and service_id is included in the SDT carried in that Transport Stream. When target_listed is set to 1 (one), the service shall be

included in the SDT, otherwise it may not be. In the latter case, the following conditions shall be met:

- the service_type for the service shall be 0x19 (advanced codec HD digital television service) if the events are simulcast, otherwise the service_type shall be the same as for the service where the source event is carried
- $EIT_{p/f}$ information shall be available for the service in that Transport Stream
- the service shall be running

event_simulcast: This 1-bit field shall be set to 1 (one) when the target event and the source event are being simulcast. It shall be set to 0 (zero) when the events are offset in time.

private_data_byte: This is an 8-bit field, the value of which is privately defined.

Parental rating descriptor

This descriptor gives a rating based on age and allows for extensions based on other rating criteria.

Syntax	Number of bits	Identifier
parental_rating_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for (i=0;i<N;i++){		
country_code	24	bslbf
Rating	8	uimsbf
}		
}		

Table 48 – Parental rating descriptor

country_code: This 24-bit field identifies a country using the 3-character code as specified in ISO 3166. Each character is coded into 8-bits according to ISO/IEC 8859-1 and inserted in order into the 24-bit field. In the case that the 3 characters represent a number in the range 900 to 999, then country_code specifies an ETSI defined group of countries. These allocations are found in TR 101 162.

EXAMPLE: United Kingdom has 3-character code "GBR", which is coded as: "0100 0111 0100 0010 0101 0010".

rating: This 8-bit field is coded according to following table, giving the recommended minimum age in years of the end user.

Rating	Description
0x00	undefined
0x01 to 0x0F	minimum age = rating + 3 years
0x10 to 0xFF	defined by the broadcaster

Table 49 – Parental rating

EXAMPLE: 0x04 implies that end users should be at least 7 years old.

Teletext descriptor

The Teletext descriptor shall be used in the PSI PMT to identify streams which carry EBU Teletext data. The descriptor is to be located in a program map section following the relevant ES_info_length field.

Syntax	Number of bits	Identifier
teletext_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for (i=0;i<N;i++){		
ISO_639_language_code	24	bslbf
teletext_type	5	uimsbf
teletext_magazine_number	3	uimsbf
teletext_page_number	8	uimsbf
}		
}		

Table 50 – Teletext descriptor

ISO_639_language_code: This 24-bit field contains the 3 character ISO 639-2 language code of the language of the teletext. Both ISO 639-2/B and ISO 639-2/T may be used. Each character is coded into 8 bits according to ISO/IEC 8859-1 and inserted in order into the 24-bit field.

EXAMPLE: French has 3-character code "fre", which is coded as: "0110 0110 0111 0010 0110 0101".

teletext_type: This 5-bit field indicates the type of Teletext page indicated. This shall be coded according to following table.

Teletext_type	Description
0x00	reserved for future use
0x01	initial Teletext page
0x02	Teletext subtitle page
0x03	additional information page
0x04	program schedule page
0x05	Teletext subtitle page for hearing impaired people
0x06 to 0x1F	reserved for future use

Table 51 – Teletext_type

teletext_magazine_number: This is a 3-bit field which identifies the magazine number as defined in EN 300 706.

teletext_page_number: This is an 8-bit field giving two 4-bit hex digits identifying the page number as defined in EN 300 706.

Local time offset descriptor

The local time offset descriptor may be used in the TOT to describe country specific dynamic changes of the local time offset relative to UTC.

Syntax	Number of bits	Identifier
local_time_offset_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for(i=0;i<N;i++){		
country_code	24	bslbf
country_region_id	6	bslbf
Reserved	1	bslbf
local_time_offset_polarity	1	bslbf
local_time_offset	16	bslbf
time_of_change	40	bslbf
next_time_offset	16	bslbf
}		
}		

Table 52 – Local time offset descriptor

country_region_id: This 6-bit field identifies a zone in the country which is indicated by country_code. This is set to "000000" when there are no different local time zones in the country.

Country_region_id	Description
00 0000	no time zone extension used
00 0001	time zone 1 (most easterly region)
00 0010	time zone 2
.....	....
11 1100	time zone 60
11 1101 to 11 1111	Reserved

Table 53 – Coding of country_region_id

local_time_offset_polarity: This 1-bit information indicates the polarity of the following local_time_offset and next_time_offset. If this bit is set to "0" the polarity is positive and the local time is ahead of UTC. If this bit is set to "1" the polarity is negative and the local time is behind UTC.

local_time_offset: This 16-bit field contains the offset time from UTC in the range between 0 hours and 13 hours at a time when current UTC time is early with respect to time_of_change. In conjunction with the local_time_offset_polarity, this indicates the time offset in the area which is indicated by the combination of country_code and country_region_id. These 16 bits are coded as 4 digits in 4-bit BCD in the order hour tens, hour, minute tens, and minutes.

time_of_change: This is a 40-bit field which specifies the date and time in MJD and UTC, when the time change takes place. This 40-bit field is coded as 16 bits giving the 16 LSBs of MJD followed by 24 bits coded as 6 digits in the 4-bit BCD.

next_time_offset: This 16-bit field contains the offset time from UTC in the range between 0 hours and 13 hours at a time when current UTC time is equal to or after time_of_change. In conjunction with the local_time_offset_polarity this indicates the time offset in the area which is indicated by the combination of country_code and country_region_id. These 16-bits are coded as 4-digits in 4-bit BCD in the order hour tens, hour, minute tens and minutes.

Subtitling descriptor

In the ISO/IEC 13818-1, for Program Map Table (PMT), defines the value of stream_type for any PID carrying DVB subtitle shall be "0x06" (this indicates a PES carrying private data) and this descriptor completes the information.

Syntax	Number of bits	Identifier
subtitling_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for (i= 0;i<N;I++){		
ISO_639_language_code	24	bslbf
subtitling_type	8	bslbf
composition_page_id	16	bslbf
ancillary_page_id	16	bslbf
}		
}		

Table 54 – Subtitling descriptor

subtitling_type: This 8 bit field provides information on the content of the subtitle and the intended display. The coding of this field shall use the codes defined for component_type when stream_content is 0x03.

composition_page_id: This 16-bit field identifies the composition page. DVB_subtitling_segments signalling this page_id shall be decoded if the previous data in the subtitling descriptor matches the user's selection criteria.

ancillary_page_id: This identifies the (optional) ancillary page. DVB_subtitling_segments signalling this page_id shall also be decoded if the previous data in the subtitling descriptor matches the user's selection criteria. The values in the ancillary_page_id and the composition_page_id fields shall be the same if no ancillary page is provided.

Scrambling descriptor

The scrambling descriptor indicates the selected mode of operation for the scrambling system. It is located in the program map section at the program loop level.

Syntax	Number of bits	Identifier
scrambling_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
scrambling_mode	8	uimsbf
}		

Table 55 – Scrambling descriptor

Application Signalling descriptor

This descriptor is mandatory whenever a DVB MHP data application component is defined. To avoid transition states and mismatches with the Application Information Table (AIT), it is recommended to not set the 'application_type' and 'version_no' in this descriptor, (i.e. set to 0). The stream_type in the PMT for this component/PID shall be set to 0x05 (private_sections).

This descriptor is defined in MHP standard TS 102 812.

Syntax	Number of bits	Identifier
application_signalling_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for(i=0; i<N; i++) {		
application_type	16	uimsbf
reserved_future_use	3	bslbf
AIT_version_number	5	uimsbf
}		
}		

Table 56 – Application signalling descriptor

application_type: This 16 bit field identifies the application type of an Application Information Table sub-table that is on this elementary stream.

AIT_version_number: This 5 bit field provides the "current" version number of the Application Information Table sub-table identified by the application type field.

Logical Channel Number (LCN) descriptor

This descriptor provides a default channel number label for services.

This descriptor is defined inside EACEM standard "Baseline DTT Receiver Specification".

Syntax	Number of bits	Identifier
logical_channel_descriptor {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for(i=0; i<N; i++) {		
service_id	16	uimsbf
reserved_future_use	6	bslbf
logical_channel_number	10	uimsbf
}		
}		

Table 57 – LCN descriptor

logical_channel_number: this is a 10-bit field which indicates the broadcaster preference for ordering services. For ease of use it is recommended NOT to use logical channel numbers greater than 99 where possible.

logical_channel_number	Description
0	Undefined
1 - 999	Logical channel number
1000 - 1023	Reserved for future use

Table 58 – Logical Channel Number organization

Descriptors examples

These are several examples of the information that can be found in DrTS reports about descriptors and compared with the information of TSReader.

Example descriptor 0x41- Service List descriptor

```
TSReader:
  <DESCRIPTOR>
  <TAG>0x41</TAG>
  <LENGTH>18</LENGTH>
  <DATA>0x10 0x43 0x01 0x10
0x83 0x01 0x10 0xff 0x01
0x11 0x3f 0x01 0x11 0x7f
0x01 0x12 0x3f 0x01</DATA>
  </DESCRIPTOR>
```

```
DrTS:
  <DESCRIPTOR>
  <DESCNAME>Service list
descriptor</DESCNAME>
  <SERVICE>0x1043 / 4163 - digital
television service</SERVICE>
  <SERVICE>0x1083 / 4227 - digital
television service</SERVICE>
  <SERVICE>0x10ff / 4351 - digital
television service</SERVICE>
  <SERVICE>0x113f / 4415 - digital
television service</SERVICE>
  <SERVICE>0x117f / 4479 - digital
television service</SERVICE>
  <SERVICE>0x123f / 4671 - digital
television service</SERVICE>
  </DESCRIPTOR>
- <DESCRIPTOR-RAW>
  <DESCNAME>Service list
descriptor</DESCNAME>
  <TAG>0x41</TAG>
  <LENGTH>18</LENGTH>
  <DATA>0x10 0x43 0x01 0x10 0x83 0x01
0x10 0xff 0x01 0x11 0x3f 0x01 0x11
0x7f 0x01 0x12 0x3f 0x01</DATA>
  </DESCRIPTOR-RAW>
```

Example descriptor 0x4a- Linkage descriptor

```
TSReader:
  <DESCRIPTOR>
  <TAG>0x4a</TAG>
  <LENGTH>12</LENGTH>
  <DATA>0x00 0x0a 0x22 0xd4
0x00 0x6e 0x09 0x04 0x08
0x00 0x46 0x00</DATA>
  </DESCRIPTOR>
```

```
DrTS:
  <DESCRIPTOR>
  <DESCNAME>Linkage
descriptor</DESCNAME>
  <TSID>10</TSID>
  <ONID>8916</ONID>
  <SERVICE>110</SERVICE>
  <LINKAGETYPE-RAW>0x9</LINKAGETYPE-
RAW>
  <LINKAGETYPE>SSU
service</LINKAGETYPE>
  <OUI>0x080046</OUI>
  </DESCRIPTOR>
- <DESCRIPTOR-RAW>
  <DESCNAME>Linkage
descriptor</DESCNAME>
  <TAG>0x4a</TAG>
  <LENGTH>12</LENGTH>
  <DATA>0x00 0x0a 0x22 0xd4 0x00 0x6e
0x09 0x04 0x08 0x00 0x46 0x00</DATA>
  </DESCRIPTOR-RAW>
```

Example descriptor 0x56- Teletext descriptor

```
TSReader:
  <DESCRIPTOR>
  <TAG>0x56</TAG>
  <LENGTH>10</LENGTH>
  <DATA>0x63 0x61 0x74 0x09
0x00 0x74 0x78 0x74 0x10
0x88</DATA>
  </DESCRIPTOR>
```

```
DrTS:
  <DESCRIPTOR>
  <DESCNAME>Teletext
descriptor</DESCNAME>
  <LANGUAGE>Catalan</LANGUAGE>
  <TXTTYPE>initial TXT page</TXTTYPE>
  <PAGENUMB>100</PAGENUMB>
  <LANGUAGE>txt</LANGUAGE>
  <TXTTYPE>TXT subt page</TXTTYPE>
  <PAGENUMB>888</PAGENUMB>
  </DESCRIPTOR>
- <DESCRIPTOR-RAW>
  <DESCNAME>Teletext
descriptor</DESCNAME>
  <TAG>0x56</TAG>
  <LENGTH>10</LENGTH>
  <DATA>0x63 0x61 0x74 0x09 0x00 0x74
0x78 0x74 0x10 0x88</DATA>
  </DESCRIPTOR-RAW>
```

Example descriptor 0x58- Local Time Offset descriptor

```
TSReader:
  <DESCRIPTOR>
  <TAG>0x58</TAG>
  <LENGTH>13</LENGTH>
  <DATA>0x47 0x42 0x52 0x02
0x01 0x00 0xd3 0x15 0x01
0x00 0x00 0x00 0x00</DATA>
  </DESCRIPTOR>
```

```
DrTS:
  <DESCRIPTOR>
  <DESCNAME>Local time offset
descriptor</DESCNAME>
  <TIME>18:46</TIME>
  <COUNTRY>GBR / United
Kingdom</COUNTRY>
  <REGION>No time zone</REGION>
  </DESCRIPTOR>
- <DESCRIPTOR-RAW>
  <DESCNAME>Local time offset
descriptor</DESCNAME>
  <TAG>0x58</TAG>
  <LENGTH>13</LENGTH>
  <DATA>0x47 0x42 0x52 0x02 0x01 0x00
0xd3 0x15 0x01 0x00 0x00 0x00
```

```
0x00</DATA>
  </DESCRIPTOR-RAW>
```

Example descriptor 0x59- Subtitling descriptor

```
TSReader:
  <DESCRIPTOR>
  <TAG>0x59</TAG>
  <LENGTH>8</LENGTH>
  <DATA>0x65 0x6e 0x67 0x10
0x00 0x01 0x00 0x01</DATA>
  </DESCRIPTOR>
```

```
DrTS:
  <DESCRIPTOR>
  <DESCNAME>Subtitling
descriptor</DESCNAME>
  <LANGUAGE>English</LANGUAGE>
  <SUBTTYPE>DVB subt (normal) with no
monitor aspect ratio
critically</SUBTTYPE>
  <COMPPAGEID>0x00 0x01</COMPPAGEID>
  <ANCILLARYPAGE>0x00
0x01</ANCILLARYPAGE>
  </DESCRIPTOR>
- <DESCRIPTOR-RAW>
  <DESCNAME>Subtitling
descriptor</DESCNAME>
  <TAG>0x59</TAG>
  <LENGTH>8</LENGTH>
  <DATA>0x65 0x6e 0x67 0x10 0x00 0x01
0x00 0x01</DATA>
  </DESCRIPTOR-RAW>
```

Example descriptor 0x65- Scrambling descriptor

```
TSReader:
  <DESCRIPTOR>
  <TAG>0x65</TAG>
  <LENGTH>1</LENGTH>
  <DATA>0x01</DATA>
  </DESCRIPTOR>
```

```
DrTS:
  <DESCRIPTOR>
  <DESCNAME>Scrambling
descriptor</DESCNAME>
 <SCRAMBLINGMODE>0x01</SCRAMBLINGMODE>
  </DESCRIPTOR>
- <DESCRIPTOR-RAW>
  <DESCNAME>Scrambling
descriptor</DESCNAME>
  <TAG>0x65</TAG>
  <LENGTH>1</LENGTH>
  <DATA>0x01</DATA>
  </DESCRIPTOR-RAW>
```

Example descriptor 0x6f- Application Signalling descriptor

```
TSReader:
  <DESCRIPTOR>
  <TAG>0x6f</TAG>
  <LENGTH>3</LENGTH>
  <DATA>0x00 0x01
0xe0</DATA>
  </DESCRIPTOR>
```

```
DrTS:
  <DESCRIPTOR>
  <DESCNAME>Application signalling
descriptor</DESCNAME>
  <APPTYPE>0x01 / 1</APPTYPE>
  <AITVERSION>0x0 / 0</AITVERSION>
  </DESCRIPTOR>
- <DESCRIPTOR-RAW>
  <DESCNAME>Application signalling
descriptor</DESCNAME>
  <TAG>0x6f</TAG>
  <LENGTH>3</LENGTH>
  <DATA>0x00 0x01 0xe0</DATA>
  </DESCRIPTOR-RAW>
```

Example descriptor 0x83- Logical Channel Number descriptor

```
TSReader:
  <DESCRIPTOR>
```

```
DrTS:
  <DESCRIPTOR>
```

```
  <TAG>0x83</TAG>
  <LENGTH>24</LENGTH>
  <DATA>0x10 0x43 0xfc 0x01
0x10 0x83 0xfc 0x02 0x10
0xff 0xfc 0x07 0x11 0x3f
0xfc 0x50 0x11 0x7f 0xfc
0x69 0x12 0x3f 0xfc
0x46</DATA>
  </DESCRIPTOR>
```

```
  <DESCNAME>Logical channel number
descriptor</DESCNAME>
  <SERVICE>0x1043 / 4163</SERVICE>
  <LCN>0x01 / 1</LCN>
  <SERVICE>0x1083 / 4227</SERVICE>
  <LCN>0x02 / 2</LCN>
  <SERVICE>0x10ff / 4351</SERVICE>
  <LCN>0x07 / 7</LCN>
  <SERVICE>0x113f / 4415</SERVICE>
  <LCN>0x050 / 80</LCN>
  <SERVICE>0x117f / 4479</SERVICE>
  <LCN>0x069 / 105</LCN>
  <SERVICE>0x123f / 4671</SERVICE>
  <LCN>0x046 / 70</LCN>
  </DESCRIPTOR>
- <DESCRIPTOR-RAW>
  <DESCNAME>Logical channel number
descriptor</DESCNAME>
  <TAG>0x83</TAG>
  <LENGTH>24</LENGTH>
  <DATA>0x10 0x43 0xfc 0x01 0x10 0x83
0xfc 0x02 0x10 0xff 0xfc 0x07 0x11
0x3f 0xfc 0x50 0x11 0x7f 0xfc 0x69
0x12 0x3f 0xfc 0x46</DATA>
  </DESCRIPTOR-RAW>
```

Example descriptor 0x55- Parental Rating descriptor

```
TSReader:
  <DESCRIPTOR>
  <TAG>0x55</TAG>
  <LENGTH>4</LENGTH>
  <DATA>0x45 0x53 0x50
0x00</DATA>
  </DESCRIPTOR>
```

```
DrTS:
  <DESCRIPTOR>
  <DESCNAME>Parental rating
descriptor</DESCNAME>
  <COUNTRY>Spain</COUNTRY>
  <RATE>0</RATE>
  <YEARS>0</YEARS>
  </DESCRIPTOR>
- <DESCRIPTOR-RAW>
  <DESCNAME>Parental rating
descriptor</DESCNAME>
  <TAG>0x55</TAG>
  <LENGTH>4</LENGTH>
  <DATA>0x45 0x53 0x50 0x00</DATA>
  </DESCRIPTOR-RAW>
```

Flow of events of use cases

Next, there are all the flows of events of use cases involved in DrTS, except for Analyse live TS that is located in the flow of events section.

Use case	Manage default settings
Actors	User
Preconditions	For executing this use case, "Configuration" tab should be selected.
Main flow	

Actor	System
1- Change settings according to correct configuration	
2- Press "Save Settings" button	3- Show WARNING message with Setting-DrTS.txt creation information
4- Accept the information of the message	5- Save settings in settings file

Subflows

A- Change Tab or close application without saving settings

Actor	System
2A- Change Tab or close application	3A- Set default settings from DrTS
From this point, this use case is finished and any other can start.	

B- Change settings file or delete it outside DrTS control

Actor	System
5B- If settings file is changed or deleted outside DrTS control with incorrect information	6B- Show ERROR message, asking to save another time Setting-DrTS.txt from DrTS GUI
7B- Accept the information of the message	
If User doesn't change Setting-DrTS.txt with correct information, 6B will be executed every User action, launch DrTS, change Tab... until correct information.	

Postconditions	Application is ready for executing any other use case.
Extension points	1- Application/use case can be closed/cancelled/stopped at any point by pressing [x] close button of application. 2- At any point, user can change tab and start other user case.
Scenarios	User is in front his/her computer, executing DrTS.

Figure 24 – Flow of events for Manage default settings use case

Use case	Record & save TS
Actors	User
Preconditions	This use case is performed when "Manage default settings" use case has been done at least once. For executing this use case, "Recording" tab should be selected. Data in configuration file shall be correct (Settings-DrTS.txt), if not ERROR message appears. "Analysis" option in this tab shall be disabled.
Main flow	

Actor	System
1- Input desired parameters to tune or leave default parameters	2- Save parameters to be used in the execution
3- Press "Record files" button	4- Show INFO message asking to wait the results
5- Accept the information of the message	6- Start "Record TS" use case
	7- Show INFO message reporting the end of process
8- Accept the information of the message	

Subflows

A- Change default settings (except "analysis" option) for the actual execution

Actor	System
1A / 3A- Change default setting (except "analysis" option) for the actual execution	2A / 4A- Apply settings to be used without saving them in configuration file
In case of calling this subflow as step 1A, continue in main flow, step 1. In case of doing this subflow as step 3A, continue with step 3.	

B- Change default "analysis" option for the actual execution

Actor	System
1B / 3B- Change default "analysis" option for the actual execution	
This use case is finalised and shall start "Analyse live TS" use case in following step: In case of calling this subflow as step 1B, continue in main flow, step 1. In case of doing this subflow as step 3B, continue with step 3.	

Postconditions	Application is ready for executing any other use case.
Extension points	1- Application/use case can be closed/cancelled/stopped at any point by pressing [x] close button of application. 2- At any point, user can change tab and start other user case.
Scenarios	User is in front his/her computer, executing DrTS.

Figure 25 – Flow of events for Record & save TS use case

Use case	Analyse recorded TS
Actors	User
Preconditions	This use case is performed when "Manage default settings" use case has been done at least once.

	For executing this use case, "Analysis" tab should be selected. Data in configuration file shall be correct (Settings-DrTS.txt), if not ERROR message appears.

Main flow

Actor	System
1- Select and add all desired TS files	2- Save list of TS files
3- Press "Run analysis" button	4- Show INFO message asking to wait for results
5- Accept the information of the message	6- Start "Analyse TS & Create XML" use case
	7- Start "Create HTML" use case
	8- Show INFO message reporting the end of process and the results of analysis (number of TS analysed)
9- Accept the information of the message	

Subflows

A- Change default setting for the actual execution

Actor	System
1A / 3A- Change default setting for the actual execution	2A / 4A- Apply setting to be used without saving them in configuration file
In case of calling this subflow as step 1A, continue in main flow, step 1. In case of doing this subflow as step 3A, continue with step 3.	

B- Press "Run analysis" without selecting TS files

Actor	System
1B- Press "Run analysis" without selecting TS files	2B- Show WARNING message to select TS files
3B- Accept the information of the message	
After this subflow, application starts another time (step 1).	

C - Press "Reset list" button

Actor	System
3C- Press "Reset list" button	4C- TS files list is reset
This subflow restarts the use case (step 1).	

Postconditions	Application is ready for executing any other use case.
Extension points	1- Application/use case can be closed/cancelled/stopped at any point by pressing [x] close button of

	application. 2- At any point, user can change tab and start other user case.
Scenarios	User is in front his/her computer, executing DrTS.

Figure 26 – Flow of events for Analyse recorded TS use case

Use case	Analyse recorded TS (Command Line domain)
Actors	DB admin
Preconditions	This use case is performed when "Manage default settings" use case has been done at least once. The application is executed by command line, GUI is not started.

Main flow

Actor	System
1- Launch DrTS by command line	2- Check command syntax
	3- Output information about starting analysis
	4- Start "Analyse TS & Create XML" use case
	5- Delete TSReader XML

Subflows

A- Command not correct

Actor	System
	3A- Warn that command is unknown
	4A- Cancel execution

B- Parameters are not correct

Actor	System
	3B- Warn that parameters are wrong
	4B- Cancel execution

C- No correct TSReader settings

Actor	System
	4C- Warn settings are not correct
	5C- Cancel execution

D- Report files already existing

Actor	System
	4D- Warn that files are already analysed
	5D- End execution

E- No destination path is indicated

Actor	System
	3E- Warn that destination path is not indicated and analysis will be saved in

	TS location
Follow main flow in step 3.	
Postconditions	Process is closed, ready to start another execution.
Scenarios	DB admin is launching an automatic script that parses TS files in a DB to execute DrTS command for each of them from his computer to be executed in it or in a remote computer. Scenario can be simplified by DB admin executing DrTS command for each TS in his computer.

Figure 27 – Flow of events for Analyse recorded TS (Command Line domain) use case

Use case	Record TS
Actors	System (from other use cases)
Preconditions	This use case is called from other use cases. Preconditions are inherited from use case initiator.
Main flow	

Actor	**System**
1- Start Record TS	2- Convert all data to specified recorded TS name format
	3- Record files with TSReader

Subflows

A- TSReader doesn't tune a specified frequency

Actor	**System**
	3A- Avoid TSReader recording for this frequency
Continue in step 3 until finish all frequencies.	

Postconditions	Application continues the processes.
Scenarios	Some use case has been started and needs "Record TS" execution.

Figure 28 – Flow of events for Record TS use case

Use case	Analyse TS and Create XML
Actors	System (from other use cases)
Preconditions	This use case is called from other use cases. Preconditions are inherited from use case initiator.
Main flow	

Actor	**System**
1- Start Analysis TS and Create XML	2- Convert all TS files names to XML names
	3- Analyse all files with TSReader
	4- Convert all XML files names to DrTS XML names

	5- Parse and copy defined SI information for each file
	6- Inform analysis results

Subflows

A- TSReader analysis file is already created

Actor	System
	3A- Avoid TSReader analysis
Continue in step 4.	

B- DrTS XML file is already analysed

Actor	System
	5B- Avoid Parsing and DrTS XML creation
	6B- Increment files not analysed counter to inform in results
Continue in step 6.	

Postconditions	Application continues the processes.
Scenarios	Some use case has been started and needs "Analysis TS and Create XML" execution.

Figure 29 – Flow of events for Analyse TS and Create XML use case

Use case	Create HTML
Actors	System (from other use cases)
Preconditions	This use case is called from other use cases. Preconditions are inherited from use case initiator.

Main flow

Actor	System
1- Start Create HTML	2- Convert all files names to HTML names
	3- Parse and copy defined SI information for each file in HTML format

Subflows

A- HTML file is already created

Actor	System
	4A- Avoid HTML creation
Continue in step 4 until finish with all files.	

Postconditions	Application continues the processes.
Scenarios	Some use case has been started and needs "Create HTML" execution.

Figure 30 – Flow of events for Create HTML use case

Sequence diagrams

In this annex, sequence diagrams are shown for all use cases except for Analyse live TS, it is shown in Figure 14.

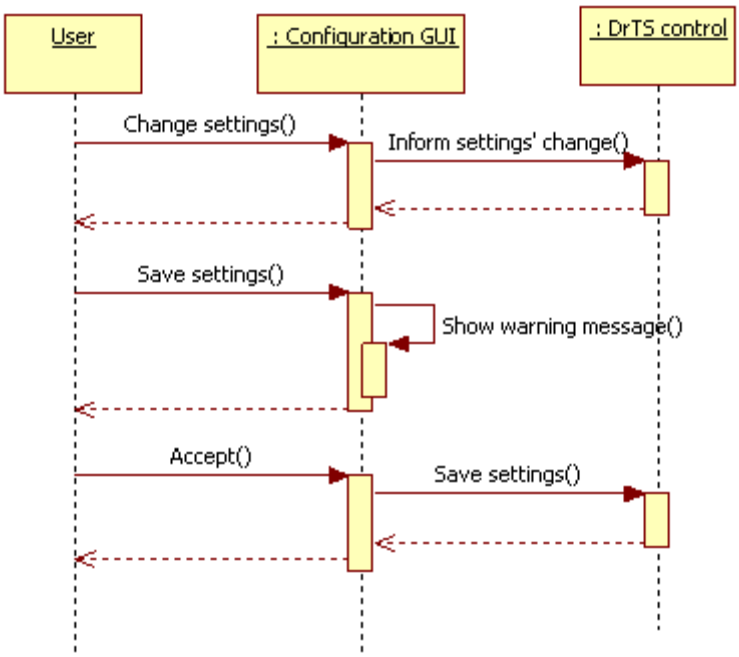


Figure 31 – Sequence diagram for Manage default settings use case

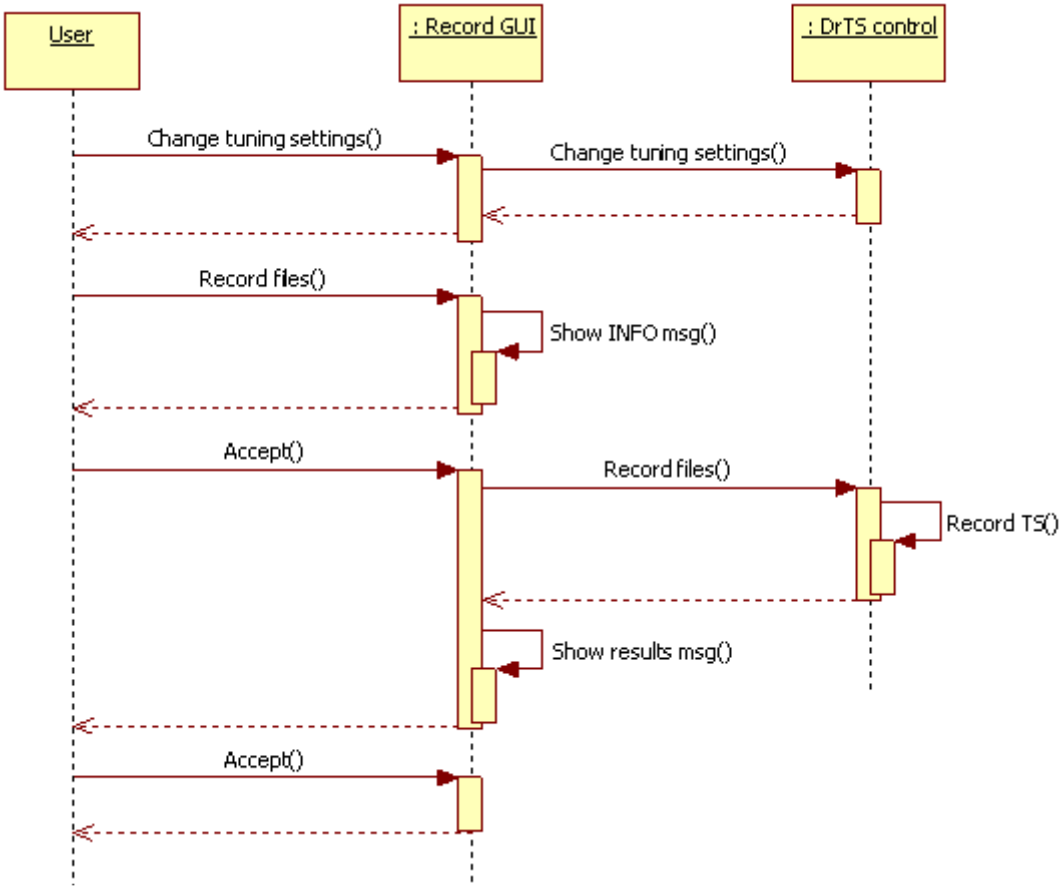


Figure 32 – Sequence diagram for Record and save TS use case

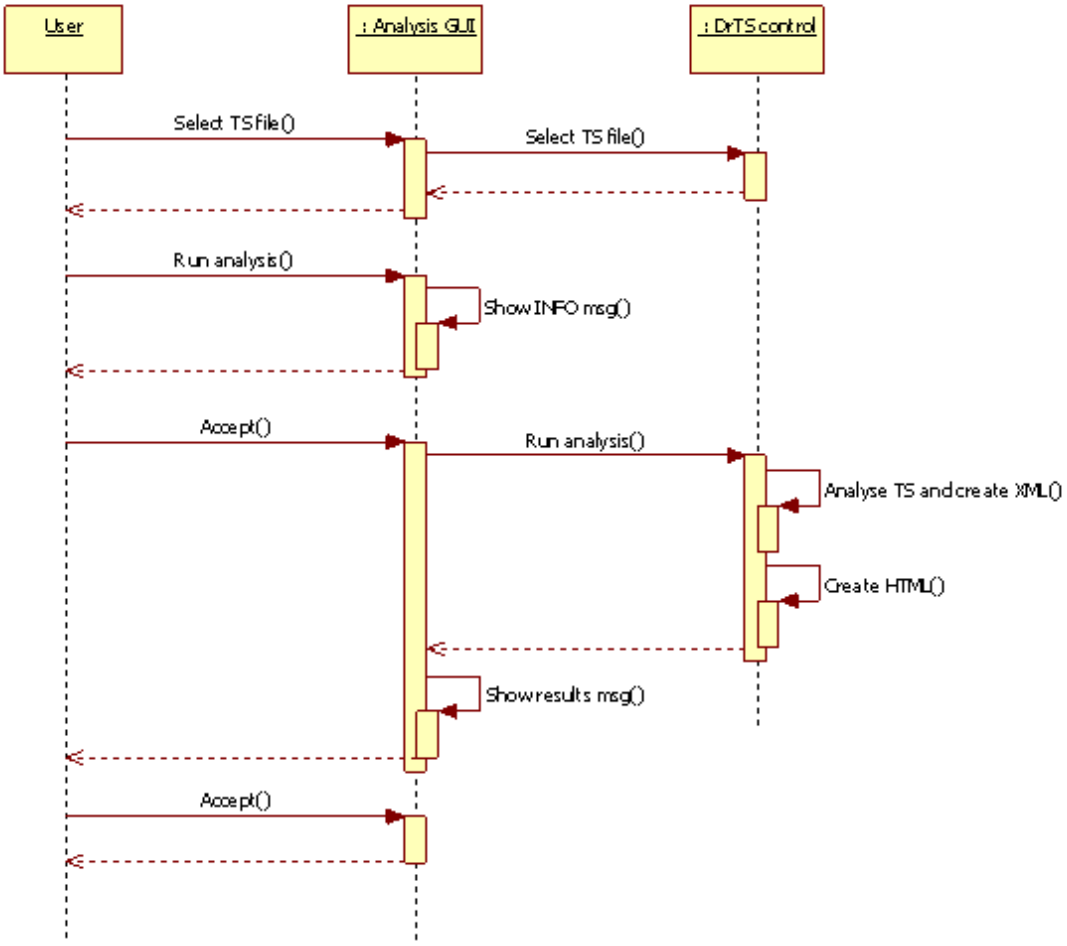


Figure 33 – Sequence diagram for Analyse recorded TS use case

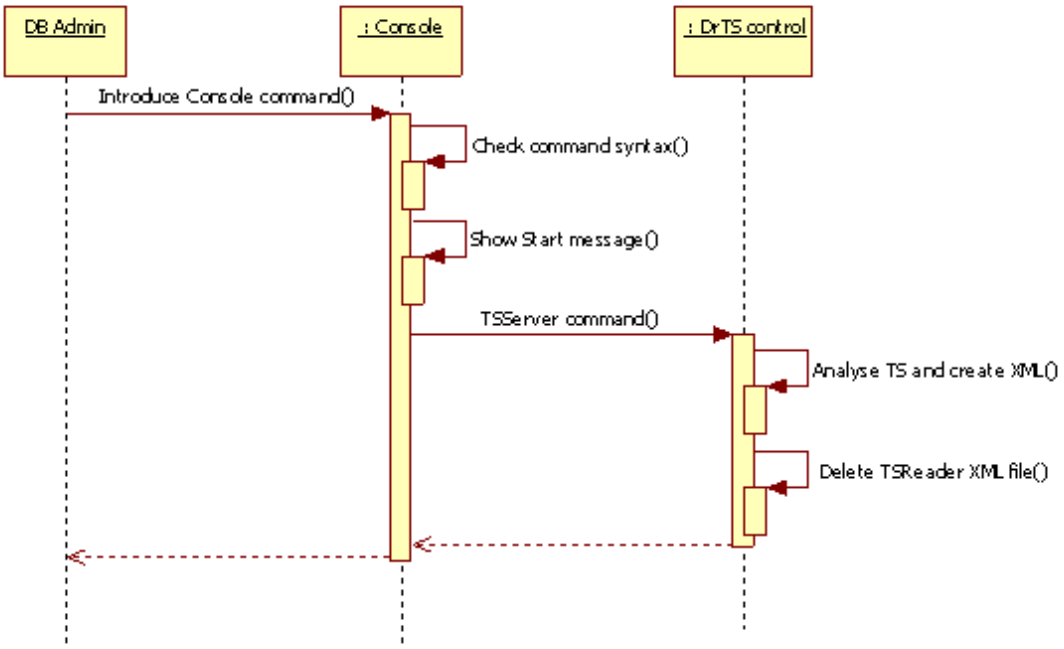


Figure 34 – Sequence diagram for Analyse recorded TS (Command Line domain) use case

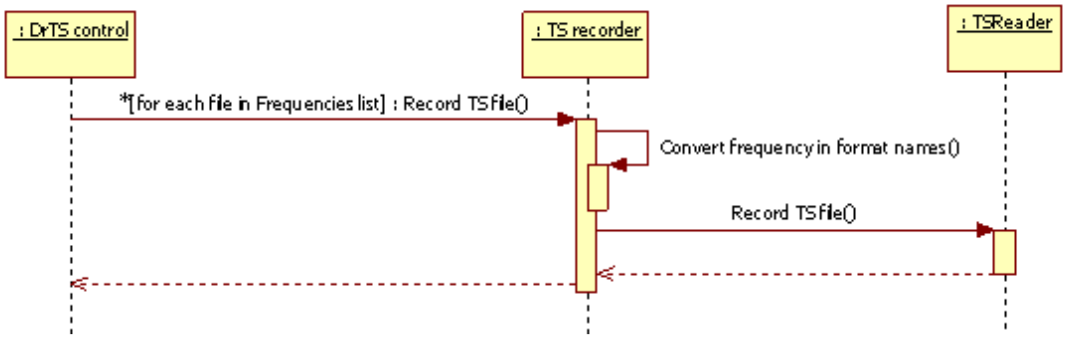


Figure 35 – Sequence diagram for Record TS use case

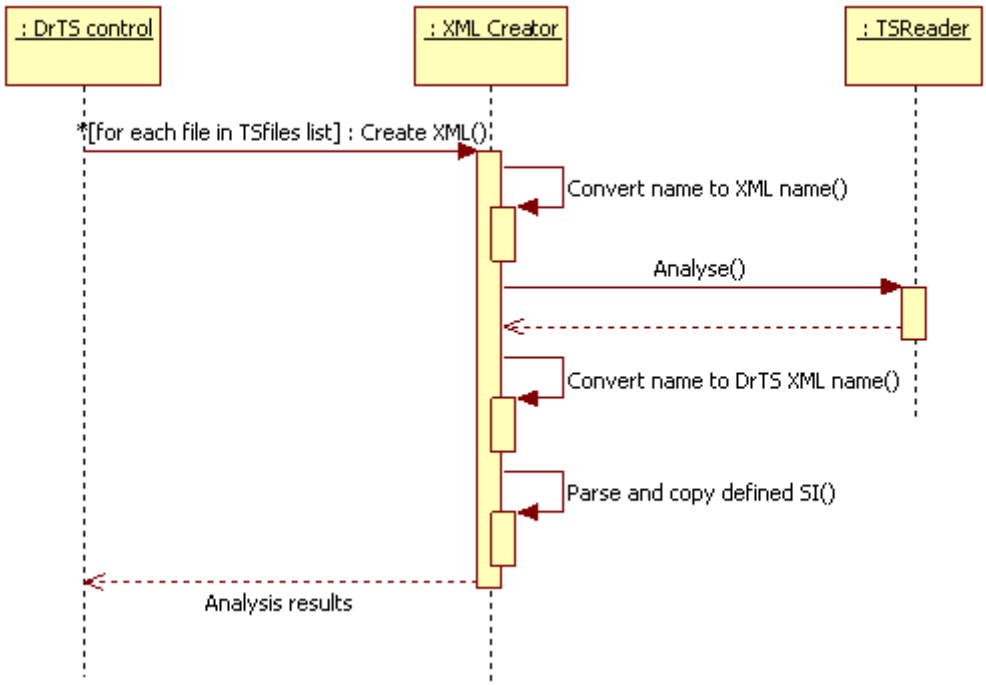


Figure 36 – Sequence diagram for Analyse TS & Create XML use case

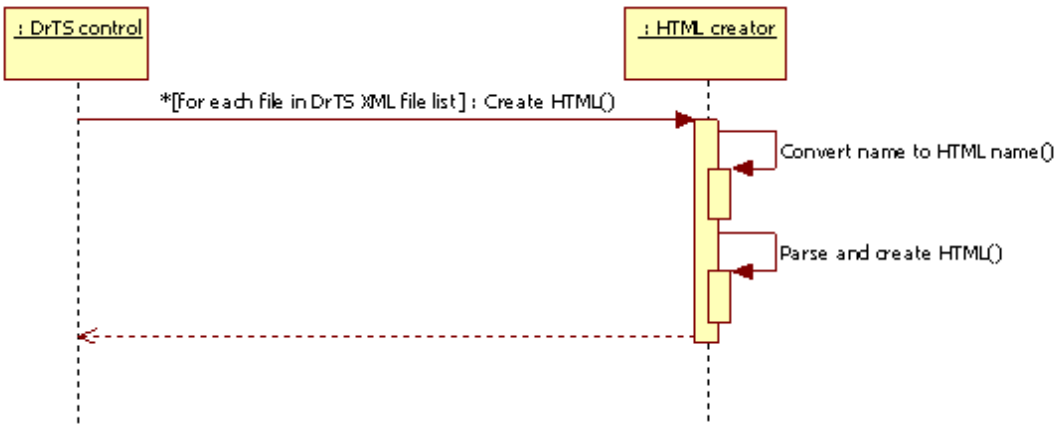


Figure 37 – Sequence diagram for Create HTML use case

GUI: additional messages

This annex shows the messages that can be found while DrTS execution, all of them can be found in flows of events description. A little description for identification is in each figure title.

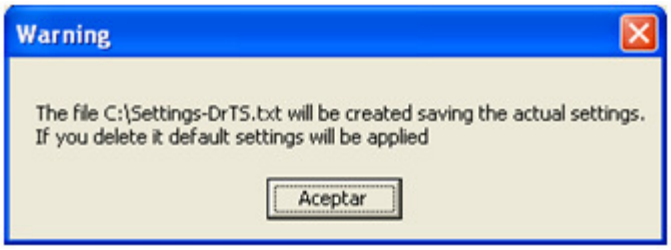


Figure 38 – Warning message when settings are going to be saved

Figure 39 – Error message at starting the application when settings file is not correct

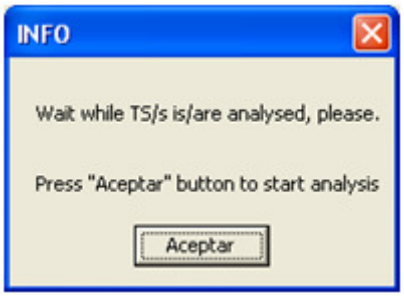


Figure 40 – Info message shown when starts a process with analysis delay

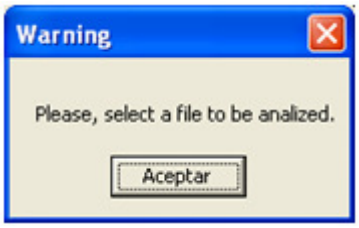


Figure 41 – Warning message when the process is going to be executed without selecting files

Figure 42 – Error message when settings file is not correct

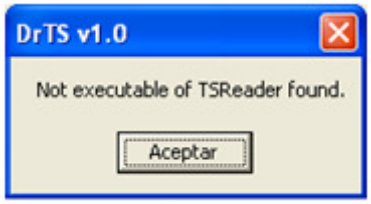


Figure 43 – Supplementary message when the instructions in error message is ignored and process is launched anyway

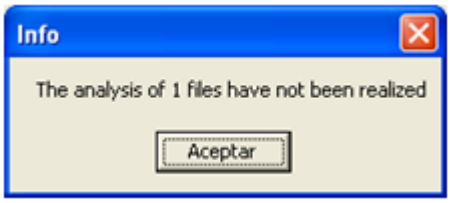


Figure 44 – Results message when application finalises

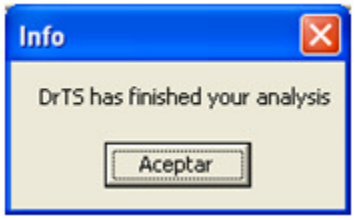


Figure 45 – Info message for the end of the process

HTML report for full profile

Here is a portion of HTML report for full profile. In this file, most of the services self information has been deleted to save space.

Sony BCNTec SW group

TS Reader - v.2.7.45h

General Information			
File:	prueba.ts		
Frequency (MHz):	650	Channel:	43
Frequency in NIT:	794	Channel:	61
Date (TDT):	2007/03/14	Hour (TDT):	13:26
Date (TOT):	2007/03/14	Hour (TOT):	14:26
Country:	ESP / Spain	Region:	No time zone
Descriptors List that are defined:			

Tag:	0x58	Local time offset descriptor
Raw data (without tag nor length):		0x45 0x53 0x50 0x02 0x01 0x00 0xd3 0xa8 0x01 0x00 0x00 0x02 0x00

Network Information		NIT version:	9
Network Name:	TELEVISIO DE CATALUNYA	Network ID:	12674
Network Type:	DVB-T	Original Net ID:	8916
Descriptors List that are defined:			
Tag:	0x40	Network name descriptor	
Raw data (without tag nor length):		0x17 0x05 0x54 0x45 0x4c 0x45 0x56 0x49 0x53 0x49 0x4f 0x20 0x44 0x45 0x20 0x43 0x41 0x54 0x41 0x4c 0x55 0x4e 0x59 0x41	
Descriptors List that are defined:			
Tag:	0x5a	Terrestrial delivery descriptor	
Raw data (without tag nor length):		0x04 0xbb 0x8c 0x40 0x1f 0x81 0x1a 0xff 0xff 0xff 0xff	

Stream Information	TS ID:	97	SDT version:	5
Service ID	**Service Name (short & long name)**		**Type**	
801	TV3 - TVC		digital television service	
802	33 - TVC		digital television service	
803	3/24 - TVC		digital television service	
804	K3/300 - TVC		digital television service	

ES information					
Service ID:	801	**Service Name (short name):**		TV3	
EIT availability:		Scheduled:	Yes	Now/Next:	Yes
PMT PID:	110		**PCR PID:**	111	
INDEX	**PID**	**Stream Type: RAW - TYPE**			
1	111	2	VIDEO		
2	112	3	AUDIO		
3	114	3	AUDIO		
4	115	6	AUDIO		
5	116	3	AUDIO		
6	113	6	TELETEXT		
7	601	12	12		
8	701	11	11		
9	704	11	11		

10	702	11	11
11	703	11	11
12	801	6	SUBTITLES
13	901	5	5

Additional ES information					
Service ID:	801		PMT PID:	110	
PID	111	**Stream Type: RAW - TYPE**		2 - VIDEO	
Horizontal-resolution:	720		Vertical-resolution:	576	
Aspect Ratio:	4:3		Frame Rate:	25	
Descriptors List that are defined:					
Tag:	0x11	STD descriptor			
Raw data (without tag nor length):	0xff				
Tag:	0x52	Stream identifier descriptor			
Raw data (without tag nor length):	0x0b				
Tag:	0x07	Target background grid descriptor			
Raw data (without tag nor length):	0x0b 0x40 0x24 0x02				
PID	112	**Stream Type: RAW - TYPE**		3 - AUDIO	
Audio Language:	Catalan				
Audio Type:	MPEG	Layer:	II	Mode:	Joint Stereo
Bit Rate:	192		Sample Rate:	48	
Descriptors List that are defined:					
Tag:	0x0a	ISO language descriptor			
Raw data (without tag nor length):	0x63 0x61 0x74 0x00				
Tag:	0x52	Stream identifier descriptor			
Raw data (without tag nor length):	0x0c				
PID	114	**Stream Type: RAW - TYPE**		3 - AUDIO	
Audio Language:	v.o				
Audio Type:	MPEG	Layer:	II	Mode:	Single Channel
Bit Rate:	96		Sample Rate:	48	
Descriptors List that are defined:					
Tag:	0x0a	ISO language descriptor			
Raw data (without tag nor length):	0x76 0x2e 0x6f 0x00				
Tag:	0x52	Stream identifier descriptor			
Raw data (without tag nor length):	0x0e				

PID	115	**Stream Type: RAW - TYPE**		6 - AUDIO	
Audio Language:		ac3			
Audio Type:	AC3				
Bit Rate:	192		Sample Rate:	48	
Descriptors List that are defined:					
Tag:	0x0a	ISO language descriptor			
Raw data (without tag nor length):		0x61 0x63 0x33 0x00			
Tag:	0x05	Registration descriptor			
Raw data (without tag nor length):		0x41 0x43 0x2d 0x33			
Tag:	0x6a	AC3 audio descriptor			
Raw data (without tag nor length):		0x00			
PID	113	**Stream Type: RAW - TYPE**		6 - TELETEXT	
Language:	esl	TXT type:	initial TXT page	Page numb:	100
Descriptors List that are defined:					
Tag:	0x56	Teletext descriptor			
Raw data (without tag nor length):		0x65 0x73 0x6c 0x09 0x00			
Tag:	0x45	VBI data descriptor			
Raw data (without tag nor length):		0x01 0x1a 0xe7 0xe8 0xe9 0xea 0xeb 0xec 0xed 0xee 0xef 0xf3 0xf4 0xf5 0xf6 0xc7 0xc8 0xc9 0xca 0xcb 0xcc 0xcd 0xce 0xcf 0xd3 0xd4 0xd5 0xd6 0x04 0x01 0xf0			
PID	702	**Stream Type: RAW - TYPE**		11 - 11	
Descriptors List that are defined:					
Tag:	0x13	Carousel Identifier descriptor			
Raw data (without tag nor length):		0x00 0x00 0x00 0x02 0x00			
Tag:	0x14	Association tag descriptor			
Raw data (without tag nor length):		0x00 0x02 0x00 0x00 0x08 0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff			
Tag:	0x52	Stream identifier descriptor			
Raw data (without tag nor length):		0x02			
Tag:	0x66	Descriptor 0x66			
Raw data (without tag nor length):		0x00 0x07			
PID	801	**Stream Type: RAW - TYPE**		6 - SUBTITLES	
Language:	Catalan	Subt type:	DVB subt (normal) with no monitor aspect ratio critically		
Composit page_id:		0x00 0x01	Ancillary page_id	0x00 0x01	
Descriptors List that are defined:					
Tag:	0x59	Subtitling descriptor			

Raw data (without tag nor length):	0x63 0x61 0x74 0x10 0x00 0x01 0x00 0x01		
PID	901	**Stream Type: RAW - TYPE**	5 - 5
Application descriptor:			
Application type:	0x01 / 1	AIT version:	0x0 / 0
Descriptors List that are defined:			
Tag:	0x6f	Application signalling descriptor	
Raw data (without tag nor length):	0x00 0x01 0xe0		

Actual event information					
Service ID:		801			
ID:	24162	**Name:**	El medi ambient - Jane Goodall dels ximpanzès als humans		
Description:	Sense Descripció.				
Date (UTC):		2007-03-14	Time (UTC):	13:23:47	
Date (local):		2007-03-14	Time (local):	14:23:47	
Duration:	00:06:18	Status:	4	CA mode:	0
Parental rating descriptor					
Country:	Spain	Rate:	0	Years:	0
Descriptors List that are defined:					
Tag:	0x54	Content descriptor			
Raw data (without tag nor length):	0x00 0x00				
Tag:	0x55	Parental rating descriptor			
Raw data (without tag nor length):	0x45 0x53 0x50 0x00				

Sony BCNTec SW group

Figure 46 – Portion of HTML report for full profile

XML report example

In this annex, example of apportion of XML report is shown.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!-- Sony BCNTec SW group -->
<MPEG-TABLES>
```

```
    <TSREADER>
        <VERSION>2.7.45h</VERSION>
    </TSREADER>
    <TUNED-MULTIPLEX>
        <FREQUENCY>0</FREQUENCY>
        <CHANNEL>0</CHANNEL>
    </TUNED-MULTIPLEX>
    <PAT>
        <VERSION>2</VERSION>
        <TRANSPORT-STREAM-ID>97</TRANSPORT-STREAM-ID>
    </PAT>
    <NIT>
        <VERSION>9</VERSION>
        <NIT-PID>16</NIT-PID>
        <NIT-ENTRY>
            <NETWORK-ID>12674</NETWORK-ID>
            <ORIGINAL-NETWORK-ID>8916</ORIGINAL-NETWORK-ID>
            <NETWORK-NAME>&#x17;&#x05;TELEVISIO DE CATALUNYA</NETWORK-NAME>
            <TRANSPORT-STREAM-ID>97</TRANSPORT-STREAM-ID>
            <NETWORK-TYPE>DVB-T</NETWORK-TYPE>
            <FREQUENCY>794.00</FREQUENCY>
            <CHANNEL>61</CHANNEL>
            <NETWORK-DESCRIPTORS>
                <DESCRIPTOR-RAW>
                    <DESCNAME>Network name descriptor</DESCNAME>
                    <TAG>0x40</TAG>
                    <LENGTH>24</LENGTH>
                    <DATA>0x17 0x05 0x54 0x45 0x4c 0x45 0x56 0x49 0x53 0x49
0x4f 0x20 0x44 0x45 0x20 0x43 0x41 0x54 0x41 0x4c 0x55 0x4e 0x59 0x41</DATA>
                </DESCRIPTOR-RAW>
            </NETWORK-DESCRIPTORS>
            <TS-DESCRIPTORS>
                <DESCRIPTOR-RAW>
                    <DESCNAME>Terrestrial delivery descriptor</DESCNAME>
                    <TAG>0x5a</TAG>
                    <LENGTH>11</LENGTH>
                    <DATA>0x04 0xbb 0x8c 0x40 0x1f 0x81 0x1a 0xff 0xff 0xff
0xff</DATA>
                </DESCRIPTOR-RAW>
            </TS-DESCRIPTORS>
        </NIT-ENTRY>
    </NIT>
    <PMTs>
        <CHANNEL>
            <SERVICE-NUMBER>801</SERVICE-NUMBER>
            <PMT-PID>110</PMT-PID>
            <PCR-PID>111</PCR-PID>
            <SHORT-NAME>TV3</SHORT-NAME>
            <LONG-NAME>TVC</LONG-NAME>
            <EIT-SCHEDULE>1</EIT-SCHEDULE>
            <EIT-PRESENT-FOLLOWING>1</EIT-PRESENT-FOLLOWING>
            <FREE-CA-MODE>0</FREE-CA-MODE>
            <ELEMENTARY-STREAM>
                <INDEX>1</INDEX>
                <PID>111</PID>
                <STREAM-TYPE-RAW>2</STREAM-TYPE-RAW>
                <STREAM-TYPE>VIDEO</STREAM-TYPE>
                <SCRAMBLED>0</SCRAMBLED>
                <HORIZONTAL-RESOLUTION>720</HORIZONTAL-RESOLUTION>
                <VERTICAL-RESOLUTION>576</VERTICAL-RESOLUTION>
                <ASPECT-RATIO>4:3</ASPECT-RATIO>
                <FRAME-RATE>25</FRAME-RATE>
                <DESCRIPTORS>
                    <DESCRIPTOR-RAW>
                        <DESCNAME>STD descriptor</DESCNAME>
```

```
                        <TAG>0x11</TAG>
                        <LENGTH>1</LENGTH>
                        <DATA>0xff</DATA>
                    </DESCRIPTOR-RAW>
                    <DESCRIPTOR-RAW>
                        <DESCNAME>Stream identifier descriptor</DESCNAME>
                        <TAG>0x52</TAG>
                        <LENGTH>1</LENGTH>
                        <DATA>0x0b</DATA>
                    </DESCRIPTOR-RAW>
                    <DESCRIPTOR-RAW>
                        <DESCNAME>Target background grid descriptor</DESCNAME>
                        <TAG>0x07</TAG>
                        <LENGTH>4</LENGTH>
                        <DATA>0x0b 0x40 0x24 0x02</DATA>
                    </DESCRIPTOR-RAW>
                </DESCRIPTORS>
            </ELEMENTARY-STREAM>
...
</CHANNEL>
    </PMTs>
    <SDT>
        <VERSION>5</VERSION>
        <SDT-PID>17</SDT-PID>
        <SDT-ENTRY>
            <SERVICE-ID>801</SERVICE-ID>
            <SHORT-NAME>TV3</SHORT-NAME>
            <LONG-NAME>TVC</LONG-NAME>
            <EIT-SCHEDULE>1</EIT-SCHEDULE>
            <EIT-PRESENT-FOLLOWING>1</EIT-PRESENT-FOLLOWING>
            <FREE-CA-MODE>0</FREE-CA-MODE>
            <SDT-DESCRIPTORS>
                <SERVICE>
                    <TYPE-RAW>1</TYPE-RAW>
                    <TYPE>digital television service</TYPE>
                </SERVICE>
                <DESCRIPTOR-RAW>
                    <DESCNAME>Service descriptor</DESCNAME>
                    <TAG>0x48</TAG>
                    <LENGTH>11</LENGTH>
                    <DATA>0x01 0x04 0x05 0x54 0x56 0x43 0x04 0x05 0x54 0x56
0x33</DATA>
                </DESCRIPTOR-RAW>
            </SDT-DESCRIPTORS>
        </SDT-ENTRY>
...
</SDT>
    <TDT>
        <YEAR>2007</YEAR>
        <MONTH>3</MONTH>
        <DAY>14</DAY>
        <HOUR>13</HOUR>
        <MINUTE>26</MINUTE>
        <SECOND>33</SECOND>
    </TDT>
    <TOT>
        <YEAR>2007</YEAR>
        <MONTH>3</MONTH>
        <DAY>14</DAY>
        <HOUR>13</HOUR>
        <MINUTE>26</MINUTE>
        <SECOND>32</SECOND>
        <DESCRIPTOR>
            <DESCNAME>Local time offset descriptor</DESCNAME>
            <TIME>14:26</TIME>
```

```
            <COUNTRY>ESP / Spain</COUNTRY>
            <REGION>No time zone</REGION>
        </DESCRIPTOR>
        <DESCRIPTOR-RAW>
            <DESCNAME>Local time offset descriptor</DESCNAME>
            <TAG>0x58</TAG>
            <LENGTH>13</LENGTH>
            <DATA>0x45 0x53 0x50 0x02 0x01 0x00 0xd3 0xa8 0x01 0x00 0x00 0x02
0x00</DATA>
        </DESCRIPTOR-RAW>
    </TOT>
    <EIT-SCHEDULED-ACTUAL />
</MPEG-TABLES>
```

Figure 47 – Example of a portion of XML report

Bibliography

Books

[1] HERVÉ BENOIT, "Televisión Digital, MPEG-1, MPEG-2, Sistema Europeo DVB" - Editorial Paraninfo. Madrid, 1.998

Documents and papers

[2] XAVIER FUSTAGUERAS, DTV course from "Col·legi Oficial d'Enginyers Tècnics de Telecomunicació de Catalunya (COETTC)", Juny 2005

[3] Internal course documentation of DTV from SONY

[4] UPC Software Engineering course documentation to Sony

Internet webpages

[5] TSReader home page – http://www.tsreader.com/

[6] Hauppauge home page – http://www.hauppauge.com

[7] DVB home page – http://www.dvb.org

[8] MPEG home page – http://www.mpeg.org

[9] TinyXML home page - http://www.grinninglizard.com/tinyxml/

[10] Digital TV Status home page - http://en.dtvstatus.net/

[11] Code Project home page - http://www.codeproject.com

Standards and official documentation

[12] ETSI EN 300 468 v1.10.1 – Digital Video Broadcasting (DVB): Specification for Service Information (SI) in DVB systems

[13] ISO/IEC 13818-1 – Information technology – Generic coding of moving pictures and associated audio information: Systems

[14] ETSI TR 101 211 v1.7.1 – DVB: Guidelines on implementation and usage of Service Information

[15] ETSI ETR 162 – Digital broadcasting systems for TV, sound and data services; Allocation of SI codes for DVB systems

[16] ETSI TR 101 154 – DVB: Implementation guidelines for the use of MPEG-2 (Moving Pictures Experts Group) Systems, Video and Audio in satellite, cable and terrestrial broadcasting applications

[17] ETSI ETR 211 – Guidelines on implementation and usage of Service Information (SI)

[18] ETSI TS 102 812 – Digital Video Broadcasting (DVB): Multimedia Home Platform (MHP) Specification 1.1.1

[19] ETSI TS 102 323 – DVB: Carriage and signalling of TV – Anytime information in DVB transport streams

[20] ETSI EN 301 192 – DVB: DVB specification for data broadcasting

[21] ETSI TS 102 006 – DVB: Specification for System Software Update in DVB systems

[22] ISO 639-2 – Codes for the representation of names of languages – Part 2: Alpha-3 code

[23] EACEM TR-030 – Baseline DTT Receiver Specification

[24] ITU-R 601 Recommendation

Summary of figures

Summary of tables

Resum:

El problema de controlar les emissions de televisió digital a tota Europa pel desenvolupament de receptors robustos i fiables és cada vegada més significant, per això, sorgeix la necessitat d'automatitzar el procés d'anàlisi i control d'aquests senyals.

Aquest projecte presenta el desenvolupament software *d'una aplicació que vol solucionar una part d'aquest problema. L'aplicació s'encarrega d'analitzar, gestionar i capturar senyals de televisió digital.*

*Aquest document fa una introducció a la matèria central que és la televisió digital i la informació que porten les senyals de televisió, concretament, la que es refereix a l'estàndard "*Digital Video Broadcasting*". A continuació d'aquesta part, l'escrit es concentra en l'explicació i descripció de les funcionalitats que necessita cobrir la aplicació, així com introduir i explicar cada etapa d'un procés de desenvolupament* software*.*

Finalment, es resumeixen els avantatges de la creació d'aquest programa per l'automatització de l'anàlisi de senyal digital partint d'una optimització de recursos.

Resumen:

El problema de controlar las emisiones de televisión digital de toda Europa para el desarrollo de receptores robustos y fiables es cada vez más notable, por ello, surge la necesidad de automatizar el proceso de análisis y control de estas señales.

Este proyecto presenta el desarrollo software *de una aplicación que pretende solucionar parte del problema. La aplicación se encarga de analizar, gestionar y capturar señales de televisión digital.*

*Este documento hace una introducción en la materia central que es la televisión digital y la información que transportan las señales de televisión, concretamente, la que se refiere al estándar "*Digital Video Broadcasting*". A continuación de esta parte, el escrito se centra en la explicación y descripción de las funcionalidades que necesita cubrir la aplicación, así como introducir y explicar cada etapa de un proceso de desarrollo de* software*.*

Finalmente, se resumen las ventajas de la creación de este programa para la automatización del análisis de señal digital a partir de una optimización de recursos.

Summary:

The problem of controlling all European digital television broadcastings for sturdy and reliable receivers' development is every time more remarkably, for this reason, the necessity of analysis and control process automation of these signals appears.

This project presents the software development of an application that tries to solve part of the problem. The application is in charge of analyse, manage and record digital television signals.

This essay introduces the main subject that it is digital television and the information that television signals carries, specifically, the information related to the standard "Digital Video Broadcasting". Following this section, the document focuses in the explanation and description of application scope functionalities, and also wants to introduce and explain each stage of a software development process.

Finally, the advantages of program creation for the automation of digital signal analysis from an optimization of resources are summarised.