МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____ 2021 г.

Визуализация результатов анализа MPEG потока при помощи кроссплатформенного фреймворка Qt

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.п.н., доцент каф. ЭВМ
_____Ю.Г. Плаксина
«___»_____2021 г.

Автор работы,
студент группы КЭ-405
_____Г.Д. Курочкин
«___»_____2021 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____2021 г.

Челябинск-2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____2021 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-405
Курочкину Глебу Денисовичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы:** «Визуализация результатов анализа MPEG потока при помощи кроссплатформенного фреймворка Qt» утверждена приказом по университету от 24 апреля 2021 г. №714-13/12

2. **Срок сдачи студентом законченной работы:** 24 мая 2021 г.

3. **Исходные данные к работе:**
   - About MPEG. – https://mpeg.chiariglione.org/about
   - Qt Documentation. – https://doc.qt.io/qt-5.15/
   - Саммерфилд, М. Qt. Профессиональное программирование. Разработка кроссплатформенных приложений на C++, 2018. - 560 с.

4. **Перечень подлежащих разработке вопросов:**
   – рассмотрение существующих научных программ для анализа MPEG;

- анализ современных сред разработки для просмотра результатов анализа MPEG потока;
- разработка собственного программного обеспечения (ПО) для визуализации результатов анализа MPEG потока;
- оценка работоспособности разработанного программного обеспечения в различных режимах и внешних условиях.

5. **Дата выдачи задания:** 1 декабря 2020 г.

Руководитель работы _____/*Ю.Г. Плаксина*/

Студент _____/*Г.Д. Курочкин* /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	01.03.2021	
Разработка модели, проектирование	01.04.2021	
Реализация системы	01.05.2021	
Тестирование, отладка, эксперименты	15.05.2021	
Компоновка текста работы и сдача на нормоконтроль	24.05.2021	
Подготовка презентации и доклада	27.05.2021	

Руководитель работы _____/*Ю.Г. Плаксина*/

Студент _____/*Г.Д. Курочкин*/

Аннотация

Г. Д. Курочкин. Визуализация результатов анализа MPEG потока при помощи кроссплатформенного фреймворка Qt. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2021, 83 83с., 15 ил., библиогр. список – 18 наим.

В рамках выпускной квалификационной работы выполнено проектирование, и реализация программного обеспечения для визуализации результатов анализа MPEG потока при помощи кроссплатформенного фреймворка Qt. Проведено исследование рынка программ для анализа MPEG с целью выявления и анализа аналогов. Рассмотрены преимущества и недостатки выбранной среды разработки. Описаны достоинства C++ и недостатки Python, с целью выбора языка программирования. Приведено обоснование способности предлагаемой архитектуры, обеспечивать успешность решения задач определённого класса.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Цифровое телевидение тесно вошло в нашу жизнь, оно является неотъемлемой частью восприятия окружающего мира, становясь одним из главных источников получаемой информации. Технология цифрового телевидения передает видеоизображение и звук с помощью цифровых сигналов по различным каналам связи. Для эффективного их функционирования необходима помощь специалистов, главная задача которых – настройка оборудования, исправление поломок на линии сети и предотвращение помех передачи данных.

Основой цифрового телевидения является стандарт Moving Picture Experts Group (MPEG), который служит для сжатия звуковых и видео данных [1]. При настройке и поддержке работы оборудования, специалисты используют приборы для диагностики данного стандарта на наличие тех или иных ошибок. На сегодняшний день данная область является узконаправленной, об этом свидетельствует весьма ограниченный выбор данной категории приборов. На территории России имеется мультисистемный анализатор ТВ сигналов ИТ-100, предназначенный для измерения параметров сигналов цифрового и аналогового телевидения. Измеритель может использоваться при контроле и настройке кабельных коаксиальных, оптоволоконных или IP сетей приема телевидения и радиовещания, отдельных элементов построения сети и прочих радиоэлектронных устройств [5].

Одна из возможностей ИТ–100 анализировать транспортный MPEG поток по стандарту TR101290. Прибор имеет возможность сохранить результат анализа MPEG потока в внутреннюю память.

На данный момент нами не было найдено программы, позволяющей получить и отобразить результаты анализа MPEG потока с помощью анализатора

ТВ сигналов мультисистемного ИТ-100. Этот факт делает представленные нами исследования по созданию данной программы актуальными и востребованными.

Целью выпускной квалификационной работы - является: разработка компьютерного приложения, обеспечивающего визуализацию результатов анализа MPEG потока, для удобного взаимодействия с пользователем(специалистом) на рабочем месте.

Для достижения поставленной цели, необходимо решить следующие задачи:

1. Обосновать выбор среды и средств реализации программы.
2. Разработать техническое задание.
3. Портировать библиотеки mpeganalyzer.
4. Реализовать модуль импорта данных.
5. Разработать архитектуру приложения.
6. Реализовать модуль отображения результатов анализа.
7. Протестировать разработанную систему.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

На сегодняшний день наиболее распространенной системой сжатия телевизионных данных является стандарт MPEG, который можно разбить на две части. Первая из которых является компрессией аудиосигналов и видеосигналов, а вторая - транспортный формат. Для достижения поставленной нами цели, более важной частью данного стандарта MPEG, является транспортный формат, который предназначен для передачи данных от источника к приемнику и описывает возможные структуры потока для передачи, компрессированной ТВ программы [18].

Компрессированные потоки данных разбиваются на фрагменты, которые заключаются в отдельные пакеты, каждый из них имеет заголовок с информацией и основную часть с данными [9]. Результирующие потоки называются PES (Packetized Elementary Stream), которые, в свою очередь, воспроизводятся мультиплексором, как транспортный поток, содержащий аудио и видео PES потоки вместе с данными синхронизации. Транспортный поток состоит из пакетов длиной 188 байт, каждый из которых имеет заголовок, содержащий информацию о данном пакете, и полезные данные [16].

Для измерения и анализа данного потока будет использоваться прибор от компании ОАО “Планар” под названием “Анализатор ТВ сигналов мультисистемный ИТ-100”, в возможности которого входит анализ MPEG транспортного потока [5]. Блоки результатов анализа будут аналогичны блокам информации в приборе:

1. Основная информация анализа.
2. Список Packet Identifier (PID) и информация о них.
3. Счетчик ошибок и детальная информация по каждому типу проверки.
4. Список, состав сервисов, информация по Elementary Stream (ES) потокам.

5. Таблицы Program Specific Information (PSI).

6. Графики измерения джиттера и интервалы меток Program Clock Reference (PCR).

В рамках выполнения выпускной квалификационной работы будет создано десктопное приложение для отображения результатов анализа, записанных данным прибором, который реализует трехуровневый анализ на соответствия требованиям стандарта TR101290. Разрабатываемое программное обеспечение (ПО) будет визуализировать данные только с файлами, записанными прибором IT-100.

1.1. ОБЗОР АНАЛОГОВ

При проведении исследования рынка программ для анализа MPEG потока, аналогично представленному в данной выпускной квалификационной работе, выяснилось, что на данный момент имеются приложения для визуализации результатов анализа MPEG потока, одним из таких приложений является «TS Analyser» от компании PROMAX, интерфейс которой, представлен на рисунке 1.1. Программа отображает все характеристики транспортного потока – графически и в виде древовидной структуры [4]. Данное ПО предоставляется бесплатно.

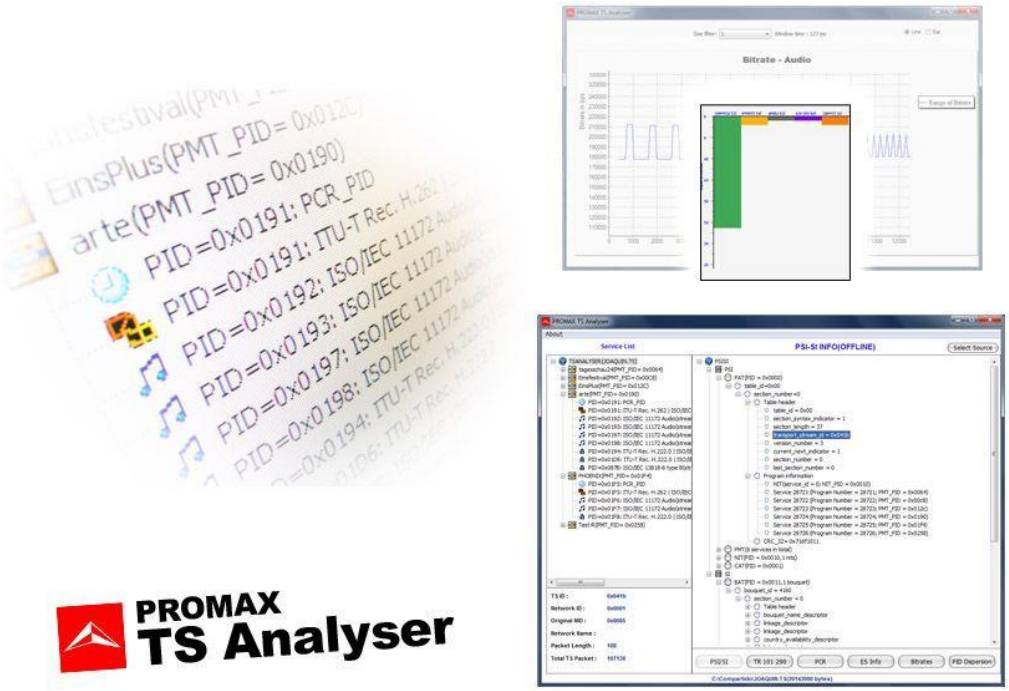

Рисунок 1.1 – Интерфейс программы «TS Analyser»

Плюсы данной программы:

- позволяет увидеть график скорости аудио и видео для всего потока или для каждого сервиса отдельно.

Минусы:

- отсутствует режим просмотра в виде списка программ потока и информации о них;
- отсутствует режим мониторинга временных характеристик PCR;
- отсутствует режим просмотра списка PID.

Данная программа работает исключительно с результатами измерения, захваченными измерительным оборудованием PROMAX для транспортного потока [4], из чего следует, что файлы, записанные с помощью других приборов, анализирующих MPEG анализ, не будут совместимы с данной программой, следовательно, отобразить результаты анализа, записанные прибором IT-100 невозможно.

Проведенный анализ показывает, что необходимо создание программы, которая выполняла бы функцию отображения результатов анализа записанных прибором IT-100.

1.2. АНАЛИЗ ОСНОВНЫХ ТЕХНОЛОГИЧЕСКИХ РЕШЕНИЙ

1.2.1 Выбор среды разработки

Для создания кроссплатформенного приложения просмотра результатов анализа MPEG потока, можно выделить две основные среды разработки, а именно Qt и Microsoft Visual Studio [15].

Qt - это объектно-ориентированная кроссплатформенная среда разработки приложений. С ее помощью можно запускать программное обеспечение, в большинстве современных операционных систем. Она включает в себя основные классы, которые могут потребоваться при разработке программ. Qt оснащен визуальной средой разработки для графического интерфейса, который позволяет создавать диалоги и формы [8].

Главные преимущества:

- возможность создания кроссплатформенного программного обеспечения;
- удобная IDE;
- достаточное количество встроенных классов для решения большинства задач;
- наличие визуальной средой разработки для графического интерфейса;
- удобная в использование документация, в которой приведено множество примеров.

Недостатки:

- потребность в большом объеме памяти, поскольку встроенные библиотеки занимают (от 15 Мб и больше);
- время запуска намного медленнее (на 50-60%) без использования Qt Quick Compiler, который поставляется только с коммерческой версией Qt [11].

Microsoft Visual Studio - это интегрированная среда разработки программного обеспечения. Данный продукт позволяет разрабатывать на разных языках программирования, как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows_Forms, а также веб-сайты и веб-приложения [2].

Достоинства:

- единые средства API для разработки программ на разных языках;
- простота стыковки разноязыковых модулей;
- многообразие готовых к использованию классов, реализующие различные алгоритмы;
- предлагаемые инструменты отладки являются наилучшим средством для отслеживания ошибок и диагностирования кода [14].

Недостатки:

- трудности при освоении функциональных возможностей программы начинающими специалистами;
- потребность в относительно большом количестве оперативной памяти (1.4 Гб и более) [13].

Qt не является языком программирования. Это фреймворк, написанный на C++. Препроцессор, MOC (Meta-Object Compiler), используется для расширения языка C ++ такими функциями, как сигналы и слоты. Перед этапом компиляции MOC анализирует исходные файлы, написанные на C ++ с расширением Qt. Таким образом, сам фреймворк и библиотеки, использующие его, могут быть

скомпилированы любым стандартным компилятором C ++, таким как MinGW и MSVC [6].

Особенности данного фреймворка:

1.IDE. Qt поставляется с собственной интегрированной средой разработки IDE, которая называется Qt Creator. Он работает на базе Linux, IOS и Windows и предлагает интеллектуальное автозавершение кода, подсветку синтаксиса, интегрированную справочную систему, интеграцию отладчика, а также интеграцию для всех основных систем контроля версий. В дополнение к Qt Creator разработчики в Windows также могут использовать надстройку Visual Studio Qt.

2.Система сборки. Совместно с Qt можно использовать любую систему сборки, но данная среда разработки предоставляет свой собственный qmake. Это кроссплатформенный интерфейс для платформенных систем сборки, таких как GNU Make , Visual Studio и Xcode .

3.Локализация. Qt обладает отличной поддержкой локализации. Инструмент Qt Linguist, lupdate, lrelease и lconvert позволяют легко переводить приложения на многие языки. Qt поддерживает большинство языков и систем письма, которые используются на сегодняшний день.

4.Виджеты. С помощью Qt GUI, виджеты могут быть написаны непосредственно на C ++, используя его модуль Widgets. Приложение также поставляется с интерактивным графическим инструментом под названием Qt Designer, который функционирует как генератор кода для графических интерфейсов на основе виджетов [7].

5.QtQuick. Другой способ написания графических интерфейсов с использованием Qt - использование модуля QtQuick. Графические интерфейсы, использующие QtQuick, написаны на QML. QML - это язык описания декларативных объектов, который интегрирует Javascript для процедурного

программирования. QtQuick предоставляет необходимые модули для разработки GUI с QML. Существует возможность писать целые приложения только в QML, но обычно в QML создаётся только GUI, а бэкэнд приложения реализован на C ++. Также Qt Creator имеет встроенный QtQuick GUI дизайнер и профилировщик.

6.GUI. Qt предоставляет модули для кроссплатформенной разработки в области сетей, баз данных, OpenGL, веб-технологий, датчиков, протоколов связи (Bluetooth, последовательных портов, NFC), обработки XML и JSON, печати, генерации PDF и многих других [6].

Среди приложений, использующих в своей работе Qt, можно выделить следующие известные кроссплатформенные программы:

1. Adobe Photoshop.
2. AMD Radeon.
3. Autodesk 3ds Max.
4. Bitcoin Core.
5. Google Планета Земля.
6. VLC медиаплеер и др.

Также стоит отметить, что в ходе программирования в двух данных IDE, Qt намного удобнее и функциональнее, чем Visual Studio. Таким образом, Qt – это самый оптимальный вариант для реализация данного проекта, поскольку он содержит множеством удобных в использовании библиотек, а также обладает визуальной средой разработки для графического интерфейса. Возможность запуска приложения на нескольких платформах с единой базой кода – одна из главных причин выбора Qt [10].

1.1.2 Выбор языка программирования

Qt поддерживает разные языки программирования, но изначально он создавался под C++, поскольку это эффективный и универсальный язык программирования. Декларативный язык пользовательского интерфейса QML, позволяет легко и быстро создавать пользовательские интерфейсы, также в Qt популярен язык программирования Python [12].

C++ и Python являются независимыми от платформы языками программирования, однако поддержка и развертывание платформы значительно различаются. Почти все платформы поддерживают C++ в качестве языка программирования, особенно если работают с пользовательским интерфейсом на основе Qt. Напротив, Python не поддерживается во многих встраиваемых системах.

Также важно упомянуть, что Qt / C ++ предоставляет согласованные инструменты для всех платформ. Это включает в себя инструмент развертывания, удаленную отладку и среду разработки Qt Creator, которая, на мой взгляд, является отличным выбором для разработчиков Qt / C ++.

Достоинства C++ и недостатки Python:

1. C ++ использует статическую типизацию, в то время как, в Python типы динамически назначаются и изменяются во время выполнения, что может привести к различным ошибкам во время выполнения, а также при неправильной проверке типов ввода.

2. В C ++ многие ошибки могут быть обнаружены при компиляции приложения. Подобные инструменты статического анализа доступны для Python, но многие действия могут быть оценены только во время выполнения компиляции.

3. Производительность. Практически во всех тестах программы на Python работают значительно медленнее, чем приложения на C ++.

4. Qt менее используется для Python, поэтому ошибки и проблемы возникают позже или никогда не обнаруживаются разработчиками.

5. Для C ++ в Qt имеется отличная IDE Qt Creator.

6. Поскольку Qt написан на C ++, примеры документации реализованы на этом же языке [10].

Основным преимуществом Python по сравнению с C ++ является скорость разработки проектов, так как Python:

1. Это язык сценариев, не нужно перекомпилировать код.
2. Имеет более простой синтаксис.
3. Использует слабую типизацию, т.е. исчезает необходимость определять типы данных [3].

Другое существенное различие между C++ и Python заключается в том, как реализовано управление памятью. В C++ необходимо освобождать память, в отличие от Python, в котором есть сборщик мусора. В C++ легко создавать приложения с утечками памяти, если не соблюдать осторожность в освобождении памяти. По сравнению с C++ (даже управлением памятью в Qt) Python намного проще в использовании, но при этом обладает гибкостью [10].

Рассмотрев два высокоуровневых языка программирования, мы можем сделать вывод, что оба варианта подходят для выполнения нашей задачи анализатора MPEG потока. Но проведённый нами анализ, позволяет отметить, что Qt написан на C++, как и все примеры, представленные в документации, что делает этот вариант наиболее подходящим для начинающего разработчика. Также большим преимуществом является то, что многие ошибки могут быть обнаружены при компиляции приложения, что немаловажно при разработке приложения[6].

1.3. ВЫВОД

Проанализировав среды разработки, можно сделать вывод о том, что Qt это оптимальный вариант для реализации нашего проекта.

Основание для выбора этого фреймворка:

1. Кроссплатформенность.
2. Набор встроенных классов.
3. Удобная IDE для создания графического интерфейса.
4. Низкий порог вхождения, подходящий для начинающего программиста.

Язык программирования C++, подходит практически для любых целей, от низкоуровневых утилит до сложных программных систем. Преимуществом данного языка является поддержка различных стилей и технологий программирования, ООП, обобщенное программирование, метапрограммирование и т.д.

Важным преимуществом Qt является кроссплатформенность: языковой стандарт предъявляет минимальные требования к компьютерам для запуска скомпилированных программ. Компиляторы доступны для большого количества платформ, на C ++ они разрабатывают программы для самых разных платформ и систем. Еще одним преимуществом языка C ++ является расширяемость. Есть много библиотек, которые расширяют функционал языка.

Язык C++ предназначен для максимального контроля над всеми аспектами структуры и порядка выполнения программы. Ни одна из языковых функций, приводящая к дополнительным накладным расходам, не является обязательной для использования - при необходимости язык обеспечивает максимальную эффективность программы.

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

2.1. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

Одним из самых важных этапов при постановке задачи, является определение функциональных требований, которые нужно реализовать для достижения цели, а также программных решений, которые стоит учесть при проектировании для оптимизации затрат времени на разработку.

Система должна отображать список файлов с результатами анализа и сами показатели для выбранного файла в правой части интерфейса приложения. Должна быть осуществлена поддержка двух языков, а именно: русский, английский.

Блоки результатов анализа должны быть аналогичны блокам информации в приборе IT-100. Требуется отображение следующих модулей:

1. Основная информация анализа.
2. Список PID и информация о них.
3. Счетчик ошибок и детальная информация по каждому типу проверки.
4. Список, состав сервисов, информация по ES потокам.
5. Таблицы PSI/SI.
6. Графики измерения джиттера и интервалы меток PCR.

Функциональные требования к подсистемам.

В подсистему портирования библиотеки mpeganalyzer необходимо добавить реализацию порта для Qt.

В подсистеме импорта данных из файла должны быть выделены полезные данные, такие как информация о приборе, дата и время анализа, параметры и

настройки анализатора, результаты измерений джиттера, интервалы меток PCR, полный список и формат данных необходимо изучить по исходным файлам проекта IT-100.

В подсистеме отображения результатов каждый модуль с результатами измерения должен отображаться аналогично прибору.

Временные характеристики: пользователь должен видеть, что программа выполняется, а не зависла, время подгрузок для всех модулей отображения не должно превышать 5 сек.

Объем данных: размер файла зависит от времени измерения анализа на приборе и не превышает 90 Мб.

Входные/выходные данные:

Входные данные – имя файла с результатами анализа.

Выходные данные – визуализация результатов анализа потока

2.2. НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

Программа должна работать на ОС Microsoft Windows версии не ниже Vista x32/x64.

В структуре системы должны иметься следующие подсистемы:

1. Система портирования библиотеки mpeganalyzer.

2.Система импорта данных из файла с результатами анализа транспортного потока.

3.Система отображения результатов анализа.

3. ПРОЕКТИРОВАНИЕ

3.1. АРХИТЕКТУРА ПРЕДЛАГАЕМОГО РЕШЕНИЯ

Проектирование программы включает несколько этапов:

1. Портирование библиотеки mpeganalyzer.
2. Реализация модуль импорта данных.
3. Реализация модели данных.
4. Реализация модуля отображения результатов анализа.

Графическое отображение архитектуры программного продукта отражено на рисунке 3.1

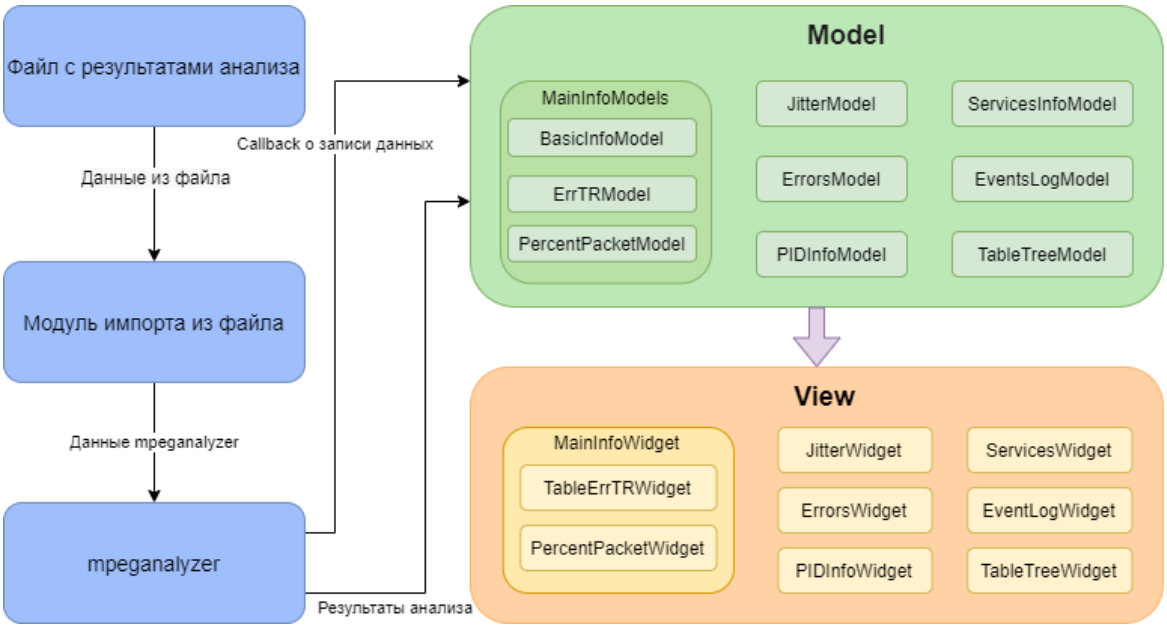

Рисунок 3.1 – Архитектура приложения

3.3.1. Этап портирования библиотеки mpeganalyzer

Для портирования библиотеки mpeganalyzer, задача которой состоит в анализе различного рода потоков, необходимо написать порт для фреймворка Qt.

Идея портирования заключается в написании реализации платформо-зависимых функций, используемых данной библиотекой.
Как правило платформо-зависимые функции состоят из следующих групп:

1. Функции работы с памятью: выделение, расширение, освобождение.

2. Функции работы с примитивами синхронизации: семафоры, мьютексы, очереди и т.п..

3. Функции работы с потоками: создание, управление, завершение.

4. Специфические функции, реализованные в API операционной системы: получение меток времени, вспомогательные функции.

Начать работу следует с изучения типовых функций каждой группы, их видов, назначения, функций Qt соответствующих группе.

Например для первой группы: функция void* malloc(size_t size) выделяет из динамической памяти size байт и возвращает указатель на выделенный блок. В Qt нет специфической реализации для выделения динамической памяти, поэтому портирование функции будет заключаться в вызове стандартной для языка C функции malloc() и free().

При портировании стоит придерживаться синтаксису языка библиотеки C, память в котором выделяется/освобождается функциями malloc/free, поэтому при написании порта используют их, а не new/delete из C++.

Другой пример, когда нужно использовать framework Qt - это работа с потоками. В библиотеке написанной на C99 не стандартизована многопоточность, поэтому мы можем использовать либо API ОС (операционной системы), либо инструменты предоставляемые кроссплатформенным фреймворком Qt, что является более оптимальным вариантом. В данном фреймворке есть класс для работы с потоками QThread, именно его мы и будем использовать для портирования группы функций.

Для реализации импорта данных нами будет использоваться ранее портированная библиотека mpeganalazer, в интерфейс которой входят функции для работы с файловым потоком.

3.3.2. Этап реализации модуля импорта данных

С помощью прибора IT-100 создается файл с записанными результатами анализа, заголовок которого соответствует 41 байту. Данный заголовок включает в себя следующие данные: версию, имя страницы, настройку анализа. Далее идут данные MPEG анализа и jitterModel, размер которых зависит от записанных результатов. Наша задача написать модуль импорта данных из файла зная вышеописанную структуру файла.

3.3.3. Этап реализации создания модели данных

Фреймворк Qt содержит набор классов, которые используют архитектуру Model View Controller (MVC) для управления отношениями между данными и способом их представления пользователю. Разделение функций, введенное этой архитектурой, дает разработчикам большую гибкость для настройки представления элементов. Это позволяет использовать широкий спектр источников данных с существующими представлениями элементов.

Данный этап подразумевает создания классов для работы с моделями данных. В фреймворке Qt все модели элементов основаны на классе QAbstractItemModel, который определяет интерфейсы для работы с представлениями и делегатами. В нашем проекте будет реализованы модели для каждого блока информации.

Для модулей «Основная информация анализа», «Список PID и информация о них», «Счетчик ошибок и детальная информация по каждому типу проверки», «Список, состав сервисов, информация по ES потокам» будут реализованы подклассы модели QAbstractTableModel. Данный момент обусловлен тем, что в

данных модулях вся информация будет отображаться в табличном представление.

Для модуля «Таблицы PSI/SI», и реализации фильтра событий модуля «Журнал событий», будут реализованы подклассы модели QAbstractItemModel, так как в обозначенных модулях вся информация будет отображаться в виде дерева.

3.3.4. Этап реализации модуля отображения результатов анализа.

В Qt предусмотрены реализации для различных видов представлений, которые основаны на базовом абстрактном классе QAbstractItemView. Данный модуль позволяет отображать данные в удобном для пользователя интерфейсе.

Подклассы представления QTableView будут реализованы для модулей «Основная информация анализа», «Список PID и информация о них», «Счетчик ошибок и детальная информация по каждому типу проверки», «Список, состав сервисов, информация по ES потокам», так как вся представленная в них информация будет отображаться в табличном представление.

Для модуля «Таблицы PSI/SI», и реализации фильтра событий модуля «Журнал событий», в которых вся информация будет отображаться в виде дерева, будут реализованы подклассы представления QTreeView.

Также стоит учесть тот факт, что модуль отображения результатов будет отвечать за режим изменения языка и стиля. Qt обеспечивает поддержку для перевода приложений на тот или иной язык, с помощью инструмента для добавления переводов Qt Linguist. Для этого создаются XML файлы, содержащие исходные фразы и их переводы с помощью данного приложения.

Таблицы стилей Qt поддерживают различные свойства, псевдосостояния и субэлементы управления, которые позволяют настраивать внешний вид. Это является одним из ключевых моментов в процессе проектирования, так как

внешний вид приложения оказывает влияние на его выбор пользователем и оказывает положительное мнение от использования.

3.2. ОПИСАНИЕ ДАННЫХ

Входными данными для визуализации результатов анализа является файл с записанной информацией, созданный анализатором ТВ сигналов ИТ-100. Структуру данного файла можно разделить на следующие модули:

1. Заголовок.
2. Данные анализатора транспортного потока.
3. Данные модели jitter.

Разберем структуру каждого модуля поподробнее.

Таблица 3.2.1 – Структура заголовка

Название	Тип данных	Описание
Version	uint_32	Версия
Name	char[33]	Имя страницы
Profile	uint_32	Настройка анализа

В итоге весь заголовок, содержащийся в таблице 3.2.1, будет занимать первоначальные 41 байт, а значит полезные данные будут располагаться, начиная с 42 байта.

Так как структура данных анализатора транспортного потока включает в себя множество полей, выделим только основные элементы структуры, которые мы будем использовать и представим их в таблице 3.2.2.

Таблица 3.2.2 – Основные данные анализатора транспортного потока

Название	Структура	Описание
tr101290Errors	X_TR101290_Errors	Счётчик обнаруженный ошибок ТС
logOfErrors	X_ERROR_LOG	Журнал зарегистрированных ошибок

mainInfo	X_MAIN_INFO	Таблица основной информации
servicesInfo	X_SERVICE_TABLE	Таблица сервисов

В таблице 3.2.3 представлено описание данных модели jitter, в структуру которой входят множество полей, поэтому выделим лишь некоторые из них.

Таблица 3.2.3 – Данные модели jitter

Название	**Структура(тип данных)**	**Описание**
Ptype	E_ANJITTERIM_PTYPE	Тип отображаемого параметра
Vpoints	ARRAY_T	Набор точек для отображения в виде таблицы заданного параметра
vstarttime	int32_t	Время начала накопления UTC
vtimestep	uint32_t	Шаг точек [секунды]
vduration	uint32_t	Временной интервал за которое сделано накопление [секунды]

В качестве входных данных от пользователя возможен выбор следующих настроек для приложения: выбор языка, размер шрифта и стиль отображения.

4. РЕАЛИЗАЦИЯ

4.1. РЕАЛИЗАЦИЯ ИНТЕРФЕЙСОВ

При первом запуске приложения пользователю будет необходимо задать настройки отображения:

- язык;
- размер шрифта;
- стиль.

На рисунке 4.1 представлено окно с настройками отображения.

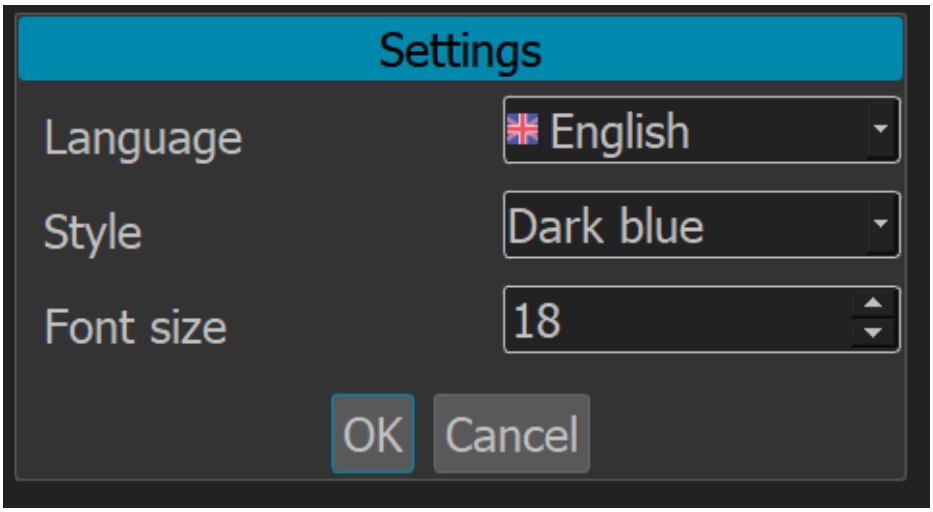

Рисунок 4.1 – Настройки отображения языка, стиля и размера шрифта

По умолчанию приложение запускается на английском языке, в стиле «Тёмно-синяя» и размером шрифта 18. Приложение поддерживает два языка: русский и английский. Есть выбор трёх стилей, а именно: «Тёмно-синяя», «Тёмно-зелёная» и «Светло-голубая».

В случае изменения настроек языка или размера шрифта, приложение будет запущено уже с новыми настройками.

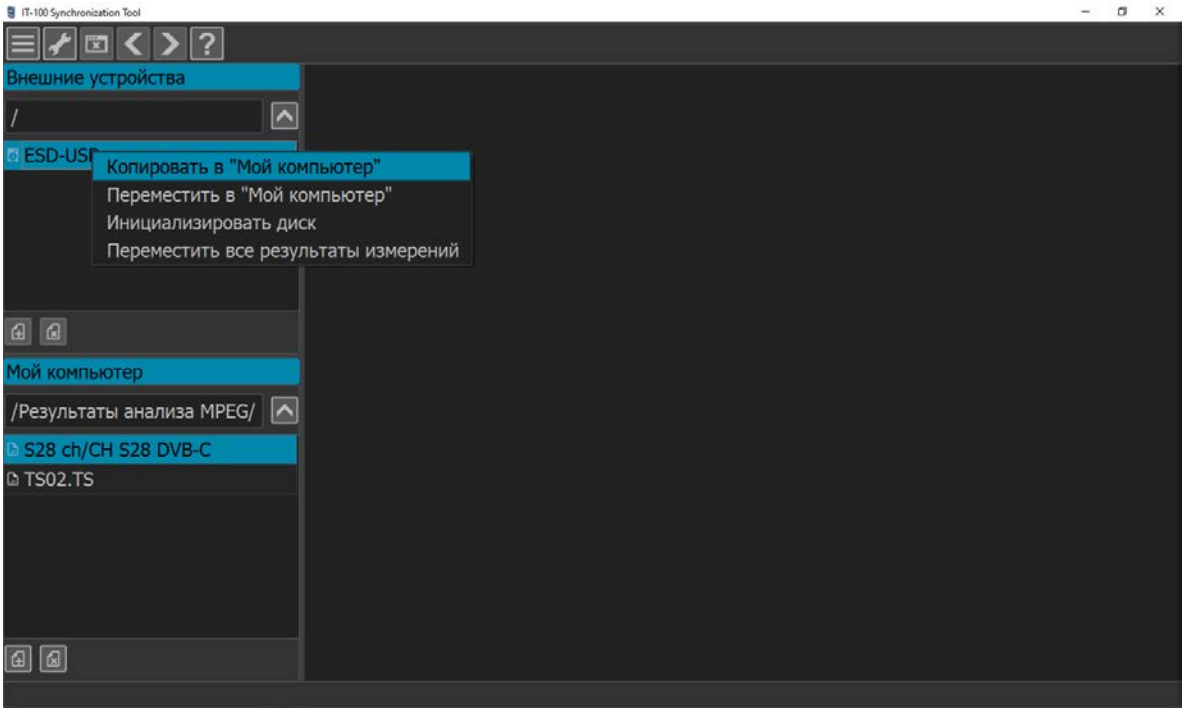

Рисунок 4.2 – Окно приложения после запуска

На рисунке 4.2 видно главное окно, в левой верхней части которого представлен список подключенных устройств по USB. Можно заметить, что в данный момент подключено одно устройство. В левом нижнем углу представлены файлы с результатами анализа скопированные на диск компьютера. Исходный код портирования и работы с файлами представлен в листинге А.1, А.2, А.3 и А.4 приложения А.

Откроем один из этих файлов щелкнув мышью два раза и увидим результаты анализа по каждому модулю в разных вкладках на рисунке 4.3. На первой вкладке «Статистика» представлена общая информация о потоке, доля пакетов и количество ошибок.

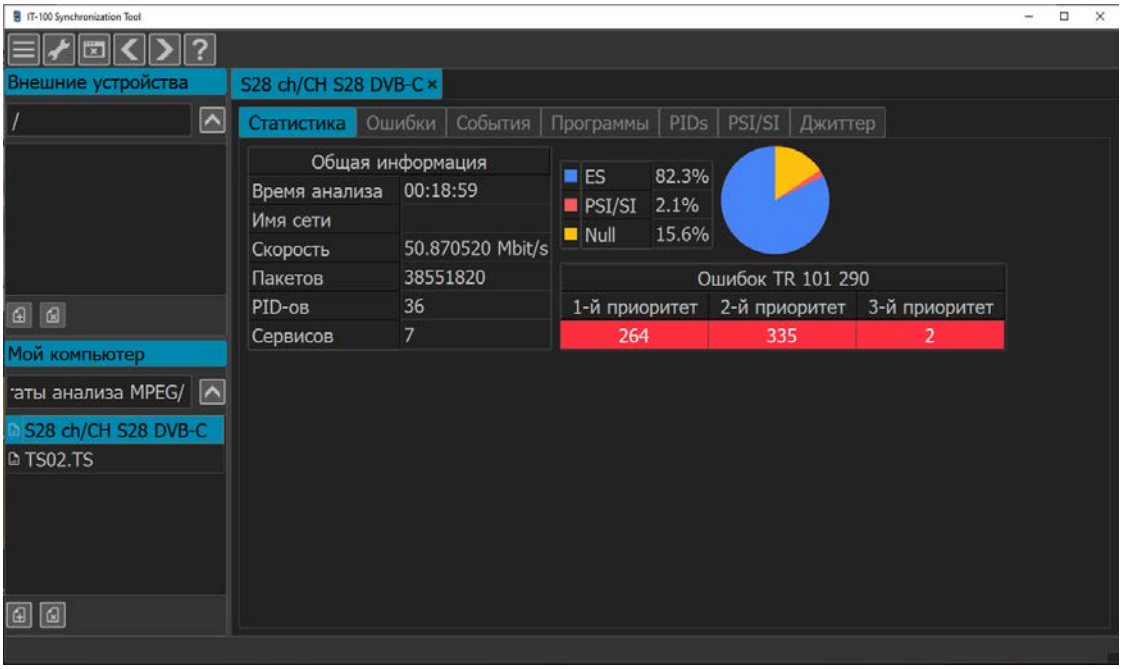

Рисунок 4.3 – Вкладка основной информации

Данное окно позволяет получить общее представление о структуре потока:

1. Время анализа.
2. Имя сети.
3. Скорость. Средняя скорость TS в Мбит/с.
4. Пакетов. Число принятых пакетов TS за время анализа.
5. PID-ов. Число PID в составе TS.
6. Сервисов. Число программ в составе TS.
7. ES. Суммарная скорость элементарных потоков (видео, аудио и данные) в процентах от общей скорости потока.
8. PSI/SI. Суммарная скорость потока таблиц сервисной информации (PAT, PMT, NIT и т.п.) в процентах от общей скорости потока.
9. Null. Скорость потока стаффинга (PID=0x1FFF) в процентах от общей скорости потока.

10. Ошибок TR 101 290. Число ошибок анализа по ETSI TR 101 290 приоритетов 1, 2 и 3, обнаруженных за все время анализа.

Исходный код модели статистики представлен в листинге Б.1 и Б.2 приложения Б.

Информации об ошибках состоит из списка проверок по ETSI TR101290 с информацией о состоянии каждой из проверок, как показано на рисунке 4.4.

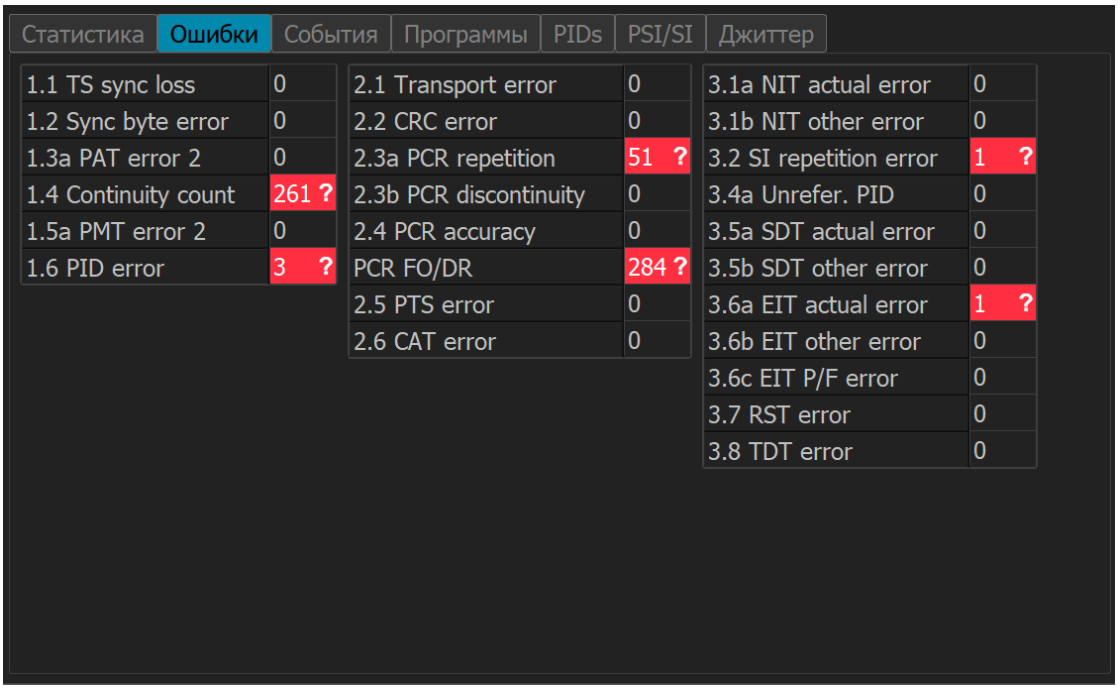

Рисунок 4.4 – Вкладка информации об ошибках ETSI TR 101 290

В представленном списке отображены проверки, аналогичные представленным в окне выбора перечня проверок ETSI TR 101 290 для анализа. Напротив каждого из пунктов проверки расположена информация о количестве зарегистрированных ошибок этого типа. В случае если зарегистрирована, по крайней мере, одна ошибка, то значение отображается на красном фоне. По нажатию на одну из зарегистрированных ошибок появляется окно с подробной информацией как показано на рисунке 4.5.

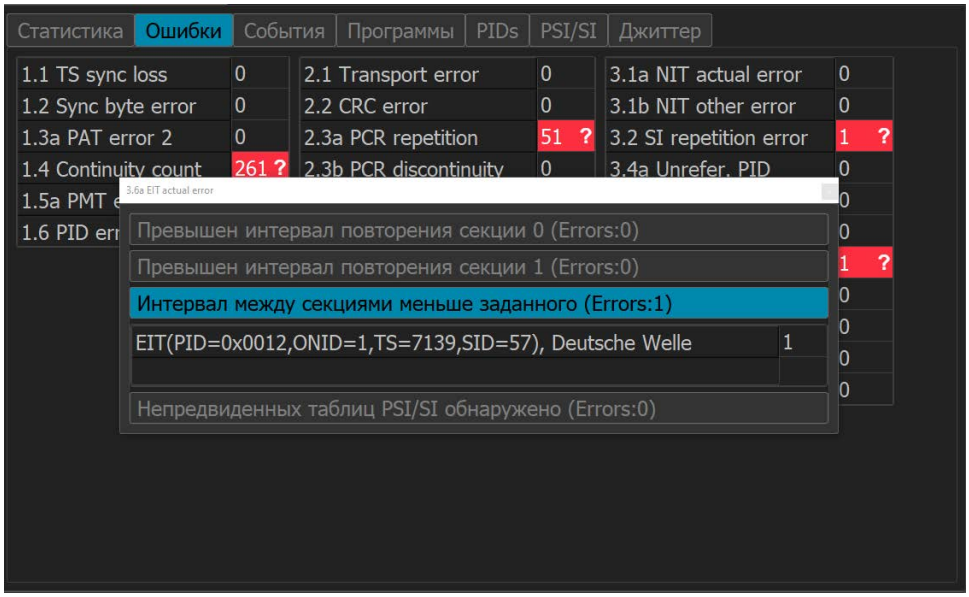

Рисунок 4.5 – Окно подробной информации об ошибках ETSI TR 101 290

Исходный код модуля ошибок представлен в листинге В.1 приложения В.

Для полного контроля выполнения анализа используется журнал событий, представленный на рисунке 4.6, который представляет собой таблицу временных меток и описания событий, произошедших за время анализа транспортного потока.

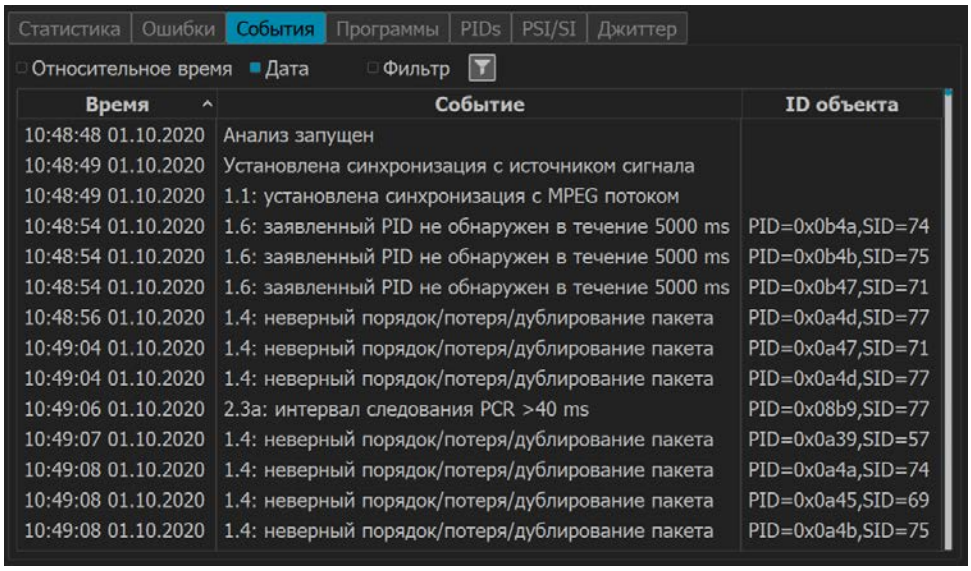

Рисунок 4.6 – Окно журнала событий

Каждое событий представлено в одной строке и содержит следующую информацию:

1. Время и дату возникновения события, или время, прошедшее с момента запуска анализа (в формате +ЧЧ:ММ:СС). Вид отображения задается на этой же вкладке с помощью флажков.
2. Идентификатор PID, с которым связано описываемое явление. Доступен для большинства событий.
3. Идентификатор программы (SID), с которой связано событие.
4. Идентификатор исходной вещательной сети (ONID), с которой связан конкретный случай.
5. Идентификатор транспортного потока (TS).
6. Номер пункта проверки согласно ETSI TR 101 290, доступный для большинства эпизодов.
7. Текстовое описание события.

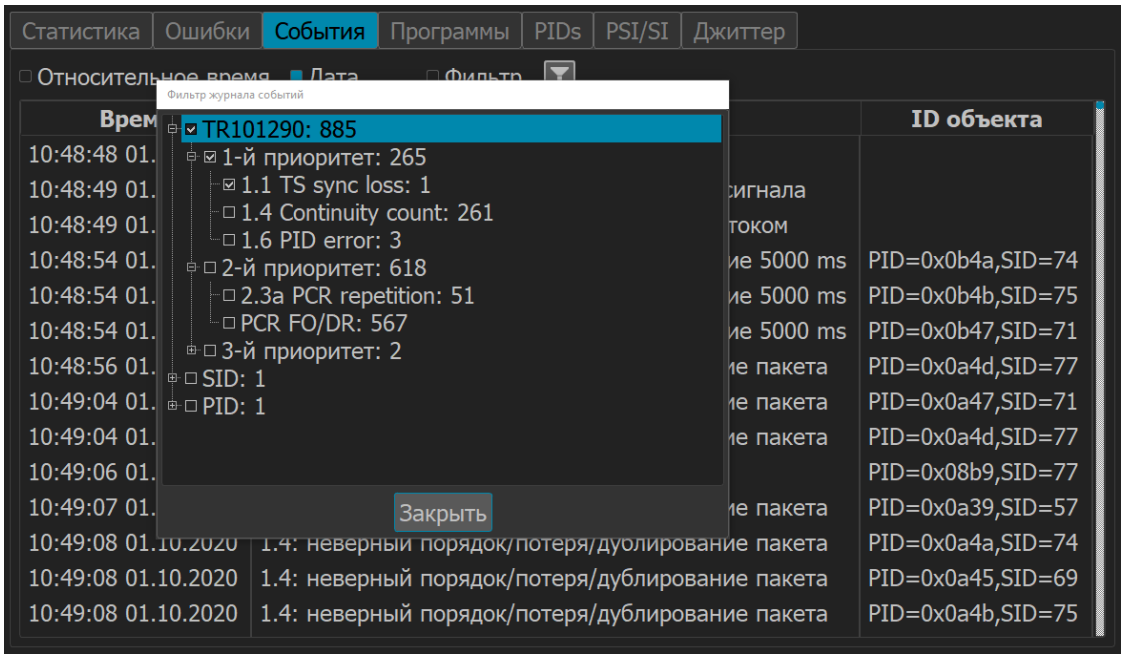

Рисунок 4.7 – Окно фильтра журнала событий

По нажатию кнопки «Фильтр» появляется окно фильтра событий (рисунок 4.7), с помощью которого можно указать только те действия, которые нужно показать в данный момент. Журнал событий в это время изменяется в реальном времени, что делает приложение максимально удобным для пользователя. В нем представлен набор типов событий, структурированный в виде дерева. События классифицированы по следующим группам:

1. TR101290. Для выбора критериев проверки ETSI TR 101 290.
2. SID для выбора программ транспортного потока.
3. PID для выбора PID транспортного потока.

Справа от каждого из узлов дерева выводится число событий, удовлетворяющих выбранному критерию, с учетом остальных параметров фильтра.

Далее идет вкладка «Программы», в которой представлен список программ и информация по каждой из них на рисунке 4.8.

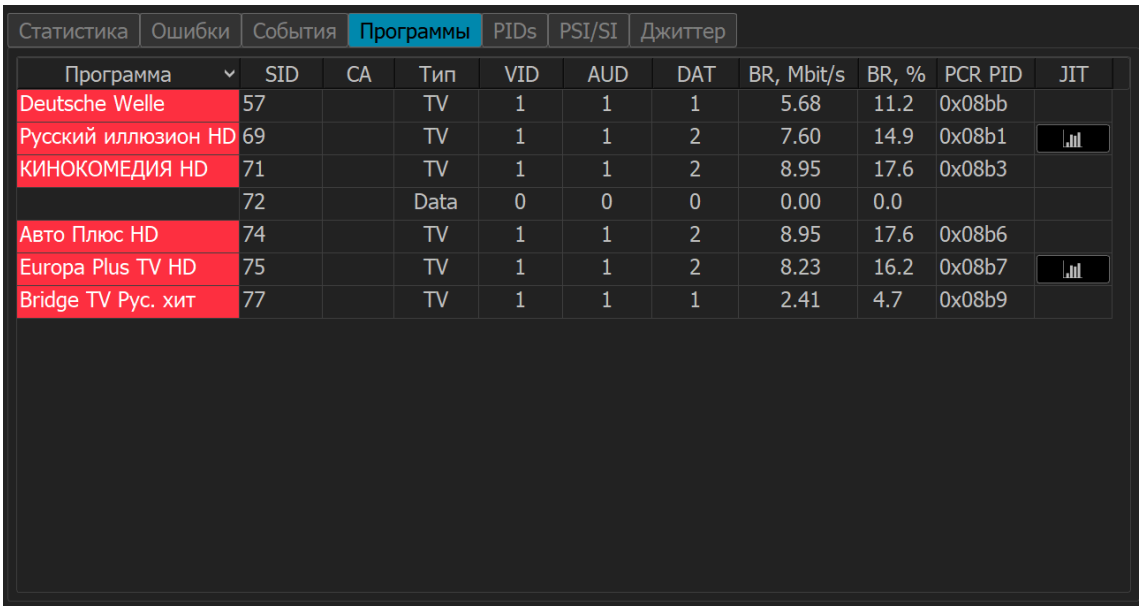

Рисунок 4.8 – Вкладка списка программ

Для каждой программы приведена следующая информация:

1. Имя программы согласно таблице SDT. В случае если один или несколько элементарных потоков программы содержат ошибки, то имя программы отображается на красном фоне.
2. SID. Идентификатор программы согласно таблицам PMT и SDT.
3. CA. Признак зашифрованной программы.
4. Тип программы: TV – телевизионная программа, Radio – радиопрограмма, Data – прочие типы программ.
5. Число элементарных потоков видео.
6. Число элементарных потоков аудио.
7. Число элементарных потоков данных.
8. Суммарная скорость всех элементарных потоков программы в мбит/с.
9. Суммарная скорость всех элементарных потоков программы в процентах относительно скорости транспортного потока.
10. PCR PID. Идентификатор PID, в котором переносятся PCR метки программы.
11. Пиктограмма состояния мониторинга временных характеристик PCR меток доступных результатов измерения.

Следом за отображением списка программ идет вкладка «PIDs», которая представляет собой таблицу идентификаторов и дополнительной информацией по каждому из них на рисунке 4.9.

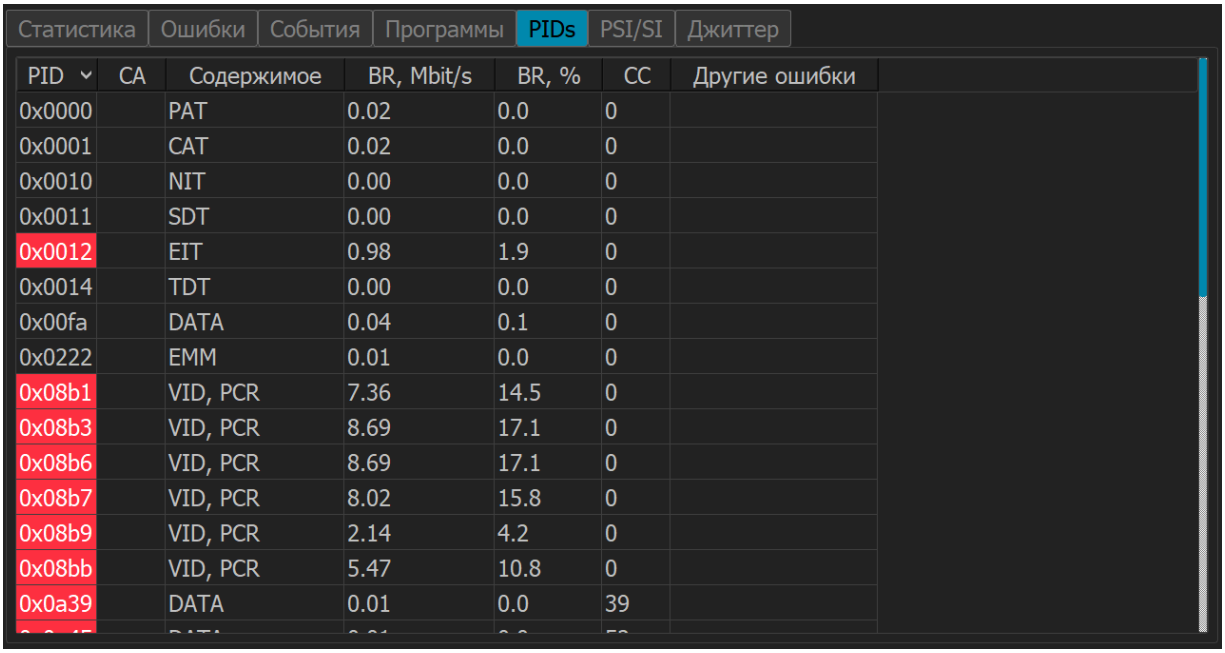

Рисунок 4.9 – Вкладка «PIDs»

Информация по каждому PID представляет собой:

1. PID. Значение идентификатора PID. В случае если PID содержит ошибки, то идентификатор отображается на красном фоне.
2. CA. Признак скремблирования информации, переносимой в этом PID.
3. Содержимое. Список объектов, переносимых в этом PID.
4. Скорость, Mbit/s.
5. Абсолютная и относительная скорость PID в процентах от скорости транспортного потока.
6. Ошибок continuity_count. Число ошибок счетчика continuity_count этого PID за все время анализа [17].
7. Другие ошибки. Признак появления ошибок.

Режим просмотра структуры сервисных таблиц (PSI/SI) представляет собой дерево узлов в формате «ключ: значение», отображающих данные таблицы на рисунке 4.10.

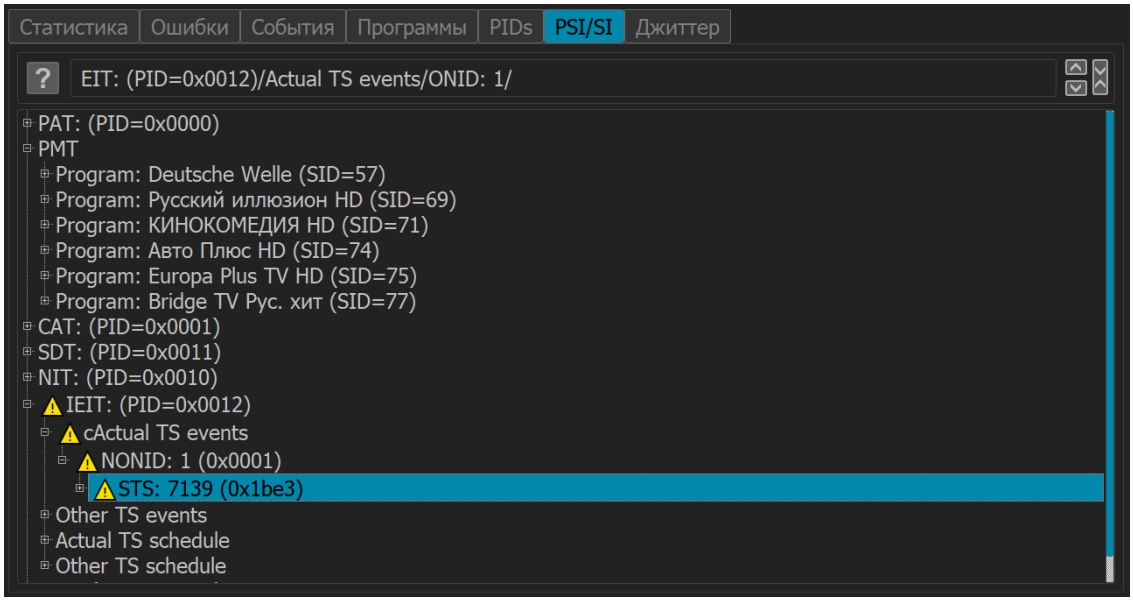

Рисунок 4.10 – Вкладка «PSI/SI»

В данной структуре отображается информация о сервисных таблицах, представленных в стандартах ISO/IEC 13818-1 и ETSI EN 300468. Корневые узлы, содержащие в себе структурированный набор данных, маркируются пиктограммой «-» (узел развернут) или «+» (узел свернут). В случае если таблица или группа таблиц сервисной информации содержит одну или несколько ошибок, то соответствующий узел маркируется пиктограммой.

Перемещение по ошибкам осуществляется с помощью кнопок «вверх/вниз» расположенный на панели кнопок. На ней также расположена и маршрутная карта, на которой показаны названия корневых узлов, разделенных символом «/», которые являются родительскими для узла позиции курсора.

По нажатию на кнопку «Детальная информация», появляется окно для детального просмотра информации (рисунок 4.11).

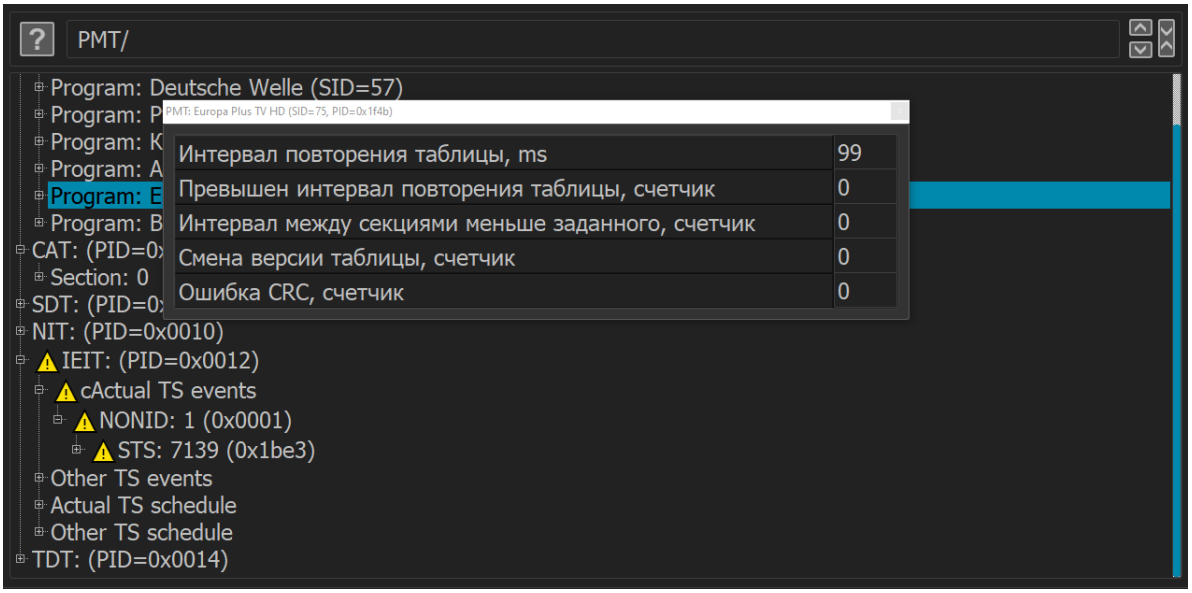

Рисунок 4.11 – Детальная информация

Детальная информация представлена в виде таблицы, в которой имеются следующие поля:

1. Интервал повторения таблицы, ms. Средний период повторения всех секций таблицы в мс.
2. Превышен интервал повторения таблицы, счетчик. Число превышений периода повторения всех секций таблицы за все время анализа.
3. Интервал между секциями меньше заданного, счетчик. Число регистраций следования любых двух секций таблицы с интервалом меньше установленного лимита за все время анализа.
4. Смена версии таблицы, счетчик. Число изменений версии таблицы за все время анализа.
5. Ошибка CRC, счетчик. Число появлений секций таблицы с ошибкой CRC за все время анализа.

Исходный код дерева таблиц представлен в листинге Г.1 приложения Г.

Далее идет вкладка под названием «Jitter» в которой представлен мониторинг временных характеристик PCR меток программы, который позволяет производить измерение интервала следования PCR меток, как представлено на рисунке 4.12.

Каждый из графиков состоит из трех трасс. В зависимости от измеряемого параметра, назначение трасс представлено в таблице 4.1.

Таблица 4.1 – таблица «Назначение трассы»

Параметр	**Назначение трассы**		
	Трасса 1	Трасса 2	Трасса 3
PCR accuracy (джиттер источника PCR)	Макс.	Ср.кв.	Мин
Интервал следования PCR	Макс	Среднее	Мин
Скорость изменения частоты PCR	Макс	Ср.кв.	Мин
Смещение частоты PCR	Макс	Среднее	Мин

Рисунок 4.12 – Временные характеристики PCR меток

В верхней части экрана можно выбрать программу, результаты измерения временных характеристик которой, хочет увидеть пользователь. Ниже располагается выбор параметра, отображаемого на графике:

1. PCR accuracy. Джиттер источника PCR меток.
2. Интервал PCR. Интервал следования PCR меток.
3. PCR drift rate. Скорость изменения частоты PCR меток.
4. PCR freq. offset. Смещение частоты PCR меток.

Справа от графика располагается дата и время начала временного интервала, число секунд временного интервала, а также результаты измерения в позиции курсора. Цвет, которым отображаются результаты измерения, соответствуют цвету трасс на графике.

5. ТЕСТИРОВАНИЕ

5.1. МЕТОДОЛОГИЯ ТЕСТИРОВАНИЯ

Для тестирования приложения был использован метод альфа-тестирования, который предполагает тестирование на поздней стадии разработки продукта и включает в себя имитацию реального пользования сотрудником или тестировщиком.

Главной целью данного тестирования является получение подробной информации о программном обеспечение, а также нахождения разного рода ошибок.

5.2. ПРОВЕДЕНИЕ ПРОЦЕДУРЫ ТЕСТИРОВАНИЯ

При тестировании принималась отладочная информация, выводимая в консоль, а также выходные данные сверялись с результатами измерения выводимыми на приборе ИТ-100, как продемонстрировано на рисунке 5.1.

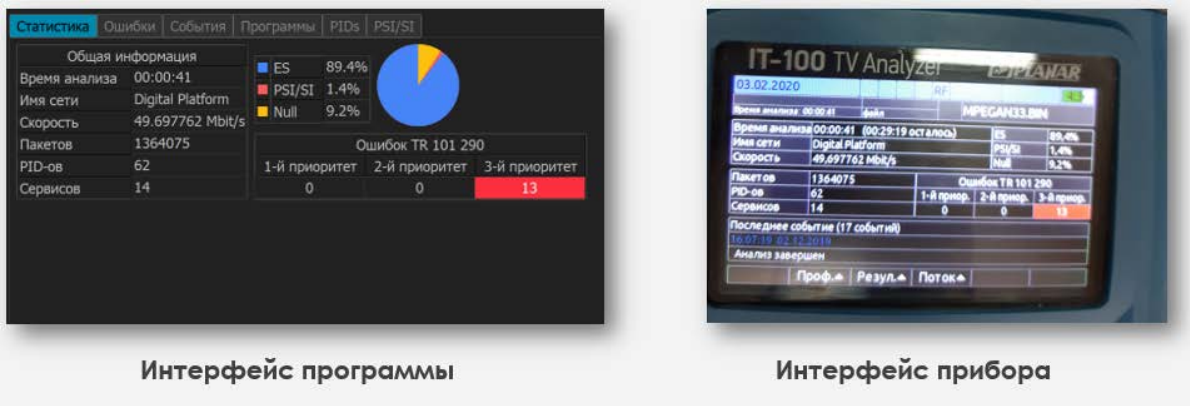

Рисунок 5.1 – Сравнение полученных данных с результатами на приборе

Для этого были записаны и проанализированы несколько транспортных потоков с помощью анализатора телевизионных сигналов ИТ-100. Далее файлы, с записанными результатами, были скопированы на компьютер и открыты в

реализованной программе. Было произведено сравнение полученных результатов с имеющимися показателями, в ходе которого не были обнаружены различия.

Таким образом, тестирование приложения для визуализации результатов анализа MPEG потока прошло успешно.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы нами было разработано приложение, обеспечивающее визуализацию результатов анализа MPEG потока, для удобного взаимодействия с пользователем(специалистом) на рабочем месте.

В ходе проектирования были решены следующие задачи:

1. Выбрана среда и средства реализации программы.
2. Разработано техническое задание.
3. Портирована библиотеки mpeganalyzer.
4. Реализован модуль импорта данных.
5. Разработана архитектура приложения.
6. Реализован модуль отображения результатов анализа.
7. Протестирована разработанная система.

Перспективы развития системы:

- написание нового модуля отображения результатов анализа по подробной информации о составе программы;
- дальнейшее усовершенствование функционала для удобного взаимодействия пользователя с программой, а именно добавление возможности перемещения в другие режимы просмотра результатов измерения, по каждому из имеющихся модулей отображения.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. About MPEG. – https://mpeg.chiariglione.org/about. Дата обращения: 20.05.21
2. Microsoft Visual Studio. – https://dic.academic.ru/dic.nsf/ruwiki/56677. Дата обращения: 20.05.21
3. Python — краткий обзор языка и его назначения. – https://techrocks.ru/2019/01/21/about-python-briefly/. Дата обращения: 20.05.21
4. TSAnalyser: Бесплатное программное обеспечение для анализа. – https://www.promax.es/ru. Дата обращения: 20.05.21
5. Анализатор ТВ сигналов мультисистемный ИТ-100. – http://www.planarchel.ru/Products/Measurement%20instrument/izmeritelnye-pribory-2/it-100/. Дата обращения: 20.05.21
6. Введение в Qt Creator. – http://doc.crossplatform.ru/qtcreator/2.0.1/creator-overview.html. Дата обращения: 20.05.21
7. Шилдт, Г. C++ для начинающих. Шаг за шагом, 2013. - 640с.
8. Краткий обзор кроссплатформенного фреймворка Qt. - https://nicknixer.ru/programmirovanie/kratkij-obzor-krossplatformennogo-frejmvorka-qt/. Дата обращения: 20.05.21
9. Локшин, Б. А. Цифровое вещание. От студии к телезрителю / Б. А. Локшин. — Компания Сайрус системс, 2001. — 446 с.
10. Саммерфилд, М. Qt. Профессиональное программирование. Разработка кроссплатформенных приложений на C++, 2018. - 560 с.
11. Шлее, М. Qt 5.10. Профессиональное программирование на C++, 2018. – 1072 с.
12. Написание приложений, основанных на Qt, на языке Python. - http//www.philosophy.ru/edu/ref/enc/k.htm1. Дата обращения: 20.05.21

13. Недостатки использования сети разработки приложений Qt. - https://www.upwork.com/hiring/for-clients/qt-cross-platform-app-development/. Дата обращения: 20.05.21

14. Описание среды разработки Microsoft Visual Studio. - https://studbooks.net/2258619/informatika/opisanie_sredy_razrabotki_studio. Дата обращения: 20.05.21

15. Популярные среды разработки и их недостатки. - https://geekbrains.ru/posts/ide_negative. Дата обращения: 20.05.21

16. Структура транспортного потока MPEG. - http://www.konturm.ru/tech.php?id=mpeg. Дата обращения: 20.05.21

17. Транспортный поток MPEG TS: основные понятия. - https://radioprog.ru/post/124. Дата обращения: 20.05.21

18. Цифровое кабельное ТВ. - https://old.telesputnik.ru/archive/144/article/112.html . Дата обращения: 20.05.21

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ИМПОРТА ДАННЫХ

Листинг А.1 – Портирование для mpeglib

```
#include <QTextCodec>
#include <QDate>
#include <QString>
#include <QTime>
#include <QLocale>

extern "C"
{
#include <stdlib.h>
#include <stdint.h>
#include <stddef.h>
#include <string.h>
#include "../TS_port.h"


static uint32_t TS_count_malloc;
static uint32_t TS_count_calloc;
static uint32_t TS_count_realloc;
static uint32_t TS_count_free;

static const char* code_page[ ] =
{
    /*TxtIso6937 */   "ISO 8859-15",//Latin-0
    /*TxtIso8859_1 */ "ISO 8859-1",
    /*TxtIso8859_2 */ "ISO 8859-2",
    /*TxtIso8859_3 */ "ISO 8859-3",
    /*TxtIso8859_4 */ "ISO 8859-4",
    /*TxtIso8859_5 */ "ISO 8859-5",
    /*TxtIso8859_6 */ "ISO 8859-6",
    /*TxtIso8859_7 */ "ISO 8859-7",
    /*TxtIso8859_8 */ "ISO 8859-8",
    /*TxtIso8859_9 */ "ISO 8859-9",
    /*TxtIso8859_10 */ "ISO 8859-10",
    /*TxtIso8859_11 */ "ISO 8859-11",
    /*TxtIso8859_13 */ "ISO 8859-13",
    /*TxtIso8859_14 */ "ISO 8859-14",
    /*TxtIso8859_15 */ "ISO 8859-15",
    /*TxtIso10646   */ "ISO 10646",
    /*TxtKsx1001_2004  */ "KSX 1001-2004",
    /*TxtGb2312_1980   */ "GB 2312-1980",
    /*TxtIso10646_Big5 */ "Big-5",
    /*TxtIso10646_Utf8 */ "UTF-8",
};

// Декодирование строки MPEG
// in_str - входная строка
// std - кодировка
// in_len - длина строки
//
// return: указатель на декодированную строку
```

```
TS_String TS_DecodeMpegStr( const uint8_t* in_str, TextStd std, uint16_t in_len )
{
    TS_String out_str = nullptr;
    int len;
    const char* codePage = code_page[std];

    QByteArray in_byte_array( reinterpret_cast<const char*>( in_str ), in_len );
    QTextCodec *codec = QTextCodec::codecForName( codePage );
    QString unicode_string = codec->toUnicode( in_byte_array );


#if ( MPEG_USE_WCHAR == 1 )

    len = unicode_string.size();
    size_t size = sizeof( TS_String ) * static_cast<unsigned int>( len + 1 );
    out_str = reinterpret_cast<TS_String>( malloc( size ) );

    if( !out_str )
    {
        return nullptr;
    }
    unicode_string.toWCharArray( out_str );

#else

    QByteArray out_byte_array = codec->fromUnicode( unicode_string );
    len = out_byte_array.size();
    size_t size = static_cast<unsigned int>( len + 1 );
    out_str = reinterpret_cast<TS_String>( malloc( size ) );
    strncpy( out_str, out_byte_array.data(), static_cast<unsigned int>( len ) );

#endif

    out_str[len] = '\0';
    return out_str;
}
//Получение времени
bool TS_RtcGetTimeStr( const TS_Time* time, TS_Char* str, size_t strSize )
{
    QTime in_time( time->uHour, time->uMin, time->uSec );
    QString time_to_string;
    time_to_string = in_time.toString( Qt :: DefaultLocaleLongDate );
    size_t len = static_cast<unsigned int>( time_to_string.size() );

    if ( ( len + 1 ) > strSize )
        return false;

    else
    {
    #if ( MPEG_USE_WCHAR == 1 )

        time_to_string.toWCharArray( str );

    #else

        QByteArray byte_array = time_to_string.toLocal8Bit();
        strncpy( str, byte_array.data(), len );
```

```
    #endif

        str[len] = 0;
        return true;
    }
}
//получение даты
bool TS_RtcGetDataStr( const TS_Date* date, TS_Char* str, size_t strSize )
{
    QDate in_date( date->wYear, date->uMonth, date->uDay );
    QString date_to_string;
    date_to_string = in_date.toString( Qt :: DefaultLocaleShortDate );
    size_t len = static_cast<unsigned int>( date_to_string.size() );

    if ( ( len + 1 ) > strSize )
        return false;

    else
    {
    #if ( MPEG_USE_WCHAR == 1 )

        date_to_string.toWCharArray( str );
    #else

        QByteArray byte_array = date_to_string.toLocal8Bit();
        strncpy( str, byte_array.data(), len );

    #endif

        str[len] = 0;
        return true;
    }
}
}
```

Листинг А.2 - Портирование для mpeganalyzer

```
#include <QSemaphore>
#include <QtConcurrent/QtConcurrent>
#include <QQueue>
#include "meta/mpegan/ThreadFunc.h"

extern "C"
{
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include "../X_port.h"

//создание семафора
X_SEMAPHORE_HANDLER X_CreateSemaphore( void )
{
        return reinterpret_cast<X_SEMAPHORE_HANDLER>( new QSemaphore( 1 ) );
}
// ----------------------------------------------------------------------------
```

```
//уничтожение семафора
void X_DestroySemaphore( X_SEMAPHORE_HANDLER handler )
{
        QSemaphore* semaphore = reinterpret_cast<QSemaphore*>( handler );
        semaphore->~QSemaphore();
}
// ----------------------------------------------------------------------------


//взять паузу в мс для семафора
uint32_t X_TakeSemaphoreTimeout( X_SEMAPHORE_HANDLER handler, uint32_t ms )
{
        QSemaphore* semaphore = reinterpret_cast<QSemaphore*>( handler );
        if( ms == X_INFINIT_TIMEOUT ) ms = INFINITE;

        return semaphore->tryAcquire( 1, static_cast<int>( ms ) );
}
// ----------------------------------------------------------------------------


//Получить указатель на семафор
void X_GiveSemaphore( X_SEMAPHORE_HANDLER handler )
{
        QSemaphore* semaphore = reinterpret_cast<QSemaphore*>( handler );
        semaphore->release();
}
// ----------------------------------------------------------------------------


// создать новый поток
// stackSize – размер стэка
// priority - приоритет
//    X_THREAD_PRIORITY_LOW
//    X_THREAD_PRIORITY_NORM
//    X_THREAD_PRIORITY_HIGH
// name – имя потока
// fnct - функтор
// opaq – значение передаваемое в поток
X_THREAD_HANDLER X_RunInThread( uint32_t stackSize, uint32_t priority,
                                char* name, F_THREAD_FUNC fnct, void* opaq )
{
        (void)name;
        (void)stackSize;
        (void)priority;

        ThreadFunc* func = new ThreadFunc( fnct, opaq );

        QThreadPool* qPool = QThreadPool::globalInstance();
                qPool->start( func );

        return static_cast<X_THREAD_HANDLER>( qPool );
}
// ----------------------------------------------------------------------------


// Уничтажение потока
int32_t X_TerminateThread( X_THREAD_HANDLER handler )
{
```

```
        (void)handler;
        /*! реализация под Qt не нужна */
        return 0;
}
// ----------------------------------------------------------------------------

// Получить текущий поток
//
X_THREAD_HANDLER X_GetCurThreadHandler( void )
{
        return QThread::currentThread();
}
// ----------------------------------------------------------------------------


// создание очереди
// max_count - максимальное кол-во элементов
// item_size - размер элемента
X_QHDL X_QueueCreate( uint32_t max_count, uint32_t item_size )
{
        X_QUEUE* queue =reinterpret_cast<X_QUEUE*>( X_malloc( sizeof(*queue) ) );

        QSemaphore *spaceSem = new QSemaphore( static_cast<int>( max_count ) );
        QSemaphore *dataSem  = new QSemaphore( 0 );

        queue->queueBuf = X_calloc( max_count, item_size );
        queue->items_max = max_count;
        queue->item_size = item_size;
        queue->in_pos = 0;
        queue->out_pos = 0;
        queue->space_avail = reinterpret_cast<X_SEMAPHORE_HANDLER>( spaceSem );
        queue->data_avail  = reinterpret_cast<X_SEMAPHORE_HANDLER>( dataSem );
        queue->CS = new QMutex();

        return reinterpret_cast<X_QHDL>(queue);
}
// ----------------------------------------------------------------------------


// удаление очереди
int32_t X_QueueDestroy( X_QHDL handler )
{
        X_QUEUE* queue = reinterpret_cast<X_QUEUE*>( handler );

        if( queue == nullptr )
        {
                return -1;
        }

        QMutex*  mutex = reinterpret_cast<QMutex*>( queue->CS );


        X_free( queue->queueBuf );
        mutex->~QMutex();
        X_DestroySemaphore( queue->data_avail );
        X_DestroySemaphore( queue->space_avail );
        X_free( queue );
```

```
        return 0;
}
// ---------------------------------------------------------------------------


// добавить элемент в очередь ----------------------------------------------
// handler – укзатель на очередь
// pItem – элемент добавления
// timeout – время ожидания
int32_t X_QueuePush( X_QHDL handler, const void* pItem, uint32_t timeout )
{
        X_QUEUE* queue = reinterpret_cast<X_QUEUE*>( handler );
        if( queue == nullptr )
        {
                return -1;
        }
        QMutex*  mutex = reinterpret_cast<QMutex*>( queue->CS );


        if ( X_TakeSemaphoreTimeout( queue->space_avail, timeout ) )
        {
                mutex->lock();
                ////////////////////////////////////////////////

                uint8_t *buf = reinterpret_cast<uint8_t*>( queue->queueBuf );
                X_memcpy( buf + queue->in_pos * queue->item_size,
                          pItem,
                          queue->item_size );
                queue->in_pos = ( queue->in_pos + 1 ) % queue->items_max;

                ////////////////////////////////////////////////
                mutex->unlock();

                X_GiveSemaphore( queue->data_avail );

                return 0;
        }
        else
        {
                return -1;
        }
}
// ---------------------------------------------------------------------------


// извлеч элемент из очереди -----------------------------------------------
// handler – указатель на очередь
// timeout – время ожидания
//
// pItem – элемент извлечения
int32_t X_QueuePop( X_QHDL handler, void* pItem, uint32_t timeout )
{
        X_QUEUE* queue = reinterpret_cast<X_QUEUE*>( handler );
        if( queue == nullptr )
        {
                return -1;
        }
```

```
        QMutex*  mutex = reinterpret_cast<QMutex*>( queue->CS );


        if ( X_TakeSemaphoreTimeout( queue->data_avail, timeout ) )
        {
                mutex->lock();
                ////////////////////////////////////////////////

                uint8_t *buf = reinterpret_cast<uint8_t*>( queue->queueBuf );
                X_memcpy( pItem,
                          buf + queue->out_pos * queue->item_size,
                          queue->item_size );
                queue->out_pos = ( queue->out_pos + 1 ) % queue->items_max;

                ////////////////////////////////////////////////
                mutex->unlock();

                X_GiveSemaphore( queue->space_avail );

                return 0;
        }
        else
        {
                return -1;
        }
}
// ----------------------------------------------------------------------------

// получение занятости очереди
uint32_t X_QueueGetOccupation( X_QHDL handler )
{
        uint32_t items;
        X_QUEUE* queue = reinterpret_cast<X_QUEUE*>( handler );

        if( queue == nullptr )
        {
                return 0;
        }
        QMutex*  mutex = reinterpret_cast<QMutex*>( queue->CS );


        mutex->lock();
        ////////////////////////////////////////////////

        if( queue->in_pos >= queue->out_pos )
        {
                items = queue->in_pos - queue->out_pos;
        }
        else
        {
                items = queue->in_pos - queue->out_pos + queue->items_max;
        }

        ////////////////////////////////////////////////
        mutex->unlock();

        return items;
}
```

```
// ----------------------------------------------------------------------------

// получение даты и времени UTC в unix формате
//
int32_t X_getNowDateTime( void )
{
        return static_cast<int32_t>( QDateTime::currentSecsSinceEpoch() );
}

// бинарный поиск в отсортированном списке
// list - указатель на список
// list_len - кол-во элементов
// val - значение
// cmp_items - функция для сравнения
// nearest - соседний элемент
int32_t X_BinarySearch( void* list, int32_t list_len, void* val,
                        X_COMPARE_ITEMS_F cmp_items, int32_t* nearest )
{
        int32_t index_first, index_last;
        int32_t nearest_local;
        int32_t cmp_result;


        index_first = 0;
        index_last = list_len;
        while( index_first < index_last )
        {
                int32_t index_mid;


                index_mid = index_first + ( index_last - index_first ) / 2;
                cmp_result = cmp_items( list, index_mid, val );
                if( cmp_result <= 0 )
                {
                        /* целевое значение >= элементу => берем поддиапазон справа */
                        index_last = index_mid;
                }
                else
                {
                        /* целевое значение < элемента => берем поддиапазон слева */
                        index_first = index_mid + 1;
                }
        }

        if( index_last < list_len )
        {
                cmp_result = cmp_items( list, index_last, val );
                if( cmp_result < 0 )
                {
                        /* целевое значение ниже найденного элемента =>
                        выбираем ближайшим предыдущий элемент */
                        nearest_local = index_last == 0? 0 : index_last - 1;
                        index_last = -1;
                }
                else if( cmp_result > 0 )
                {
                        /* целевое значение выше найденного элемента =>
                        выбираем ближайшим найденный элемент */
```

```
                        nearest_local = index_last;
                        index_last = -1;
                }
                else
                {
                        /* искомый элемент найден */
                        nearest_local = index_last;
                }
        }
        else
        {
                /* значение выше всех элементов списка => ближайший является последним
                элементом */
                nearest_local = list_len - 1;
                index_last = -1;
        }

        if( nearest != nullptr )
        {
                *nearest = nearest_local;
        }

        return index_last;
}
}
```

Листинг А.3 - MpegAnFile

```
#include "MpegAnFile.h"
#include <QTextCodec>
#include <QString>
#include "Common/Constants.h"
#include <QCoreApplication>
#include <QApplication>
#include <QDebug>
#include "meta\mpegan\MpegAnalyzer.h"
extern "C"{
#include "fm\al_fmanager.h"
#include "meta\mpeganalyzer\X_MPEG_Analyzer.h"
#include "meta\mpegan\mpegan.h"
#include "meta\mpegan\logev.h"
#include "anmpegext.h"
#include "meta\mpegan\tsanevflt.h"
}

#include "Views\Files\MpegAnView.h"
#include "Models\MpegAn\ServicesInfoModel.h"
```

```
#include "Models\MpegAn\MainInfo\MainInfoModel.h"
#include "Models\MpegAn\PIDInfoModel.h"
#include "Models\MpegAn\EventsLogModel.h"
#include "Models\MpegAn\TableTreeModel.h"
#include "Models\MpegAn\ErrorsModel.h"
#include "Views\MpegAn\MainInfo\MainInfoWidget.h"
#include <QtConcurrent/QtConcurrent>

namespace Planar {
namespace Models {

MpegAnFile::MpegAnFile(TreeItem* parent, tU32 fileId):
        File(parent, File::MpegAnalysisType, fileId)
{
       _isSimple = false;
       _name = fileName();
       if (_name.isEmpty())
              _isCorrupted = true;
}
//Получение имени файла
QString MpegAnFile::fileName() const
{
       unsigned mpeganNameSize = nameBufferSize();
       char mpeganName[mpeganNameSize];
       _status = mpegan_read_name( drive(),
                                   _fileId,
                                   mpeganName,
                                   mpeganNameSize );

       if ( _status == MPEGAN_ERR_FAILED_TO_READ )
              return _name;
       if ( _status != MPEGAN_ERR_OK ) {
                     return QString();
       }
       QTextCodec* codec = QTextCodec::codecForName( Constants::deviceCodec );
       return codec->toUnicode( mpeganName );
}
//открытие файла с результатами
void MpegAnFile::open()
```

```
{
	if (_isOpened || _isNew)
		return;
	_isCorrupted = true;
   if( OpenFile( drive(), &_fileId, MPEGAN_FILE_TYPE, FM_OPEN_READ ) == FILE_OK )
	{
		File::open();
	 tU32 sizeHeader = sizeof(TSANAM_STORE_T);

	 if( movePointer( sizeHeader ) == FILE_OK)
		{
			createModels();
			_mpegAn = new MpegAnalyzer(this);
			_isCorrupted = false;
		}

	}
	else
		_isCorrupted = true;

}
//закрытие файла
void MpegAnFile::close()
{
	if(!_isOpened ) return ;

	if ( CloseFile( drive(), _fileId ) == FILE_OK )
	{
		QFuture<int> future = QtConcurrent::run( X_tsClose, _mpegAn->getXHandle() );
		File::close();
	}
	//delete _mpegAn;
	return;
}
//переместить указатель в файле
int MpegAnFile::movePointer( int size )
{
   return ReadFileEx( drive(), _fileId, size, 0 , nullptr );
```

```
}
//создание моделей
void MpegAnFile::createModels()
{
	_servicesInfoModel = new ServicesInfoModel(this);
	_mainInfoModel     = new MainInfoModel(this);
	_pidInfoModel      = new PIDInfoModel(this);
	_logEventModel     = new EventsLogModel(this);
	_tableTreeModel    = new TableTreeModel(this);
	for(int i =0;i< 3;i++)
		_errorsModel.push_back(new ErrorsModel(this,i));
}

} // namespace Models
} // namespace Planar
```

Листинг А.4 - MpegAnalyzer

```
#include "MpegAnalyzer.h"
#include <QtConcurrent/QtConcurrent>
#include "Models/MpegAn/EventsLogModel.h"
#include "Models/MpegAn/JitterModel.h"
#include "MpegAnEvent.h"
#include <QThread>
#include <QApplication>

namespace Planar {
namespace Models {

MpegAnalyzer::MpegAnalyzer(MpegAnFile *file):
   _file(file),
   _tsHandle(nullptr),
   _servicesTable(new X_SERVICE_TABLE{}),
   _mainInfo(new X_MAIN_INFO{}),
   _eventLog(nullptr),
   _errors(new X_TR101290_ERRORS),
   tsan_callbacks{ nullptr, x_clbfnc_cmd_complete, nullptr, nullptr }
{
   int32_t err;
       X_TS_PARAMS params {.mode = tsModeViewResult,
```

```
                        .callback = tsan_callbacks,
                        .aux = this};

   _tsHandle = X_tsOpen( &params, &err );
       if ( err != X_ERR_OK )return;

       X_tsReadAllResultsSync( _tsHandle, file, F_X_STREAM_READ );//считывание всех данных
из файла
      tsUpdate();

       setPidList();
       _eventLog = X_tsGetEventsLog( _tsHandle );//получение журнала событий
       file->getLogEventModel()->setEventLog( _eventLog );

       _htsanam = tsanam_impl_create( _tsHandle );
       ANJITTERMODEL_CFG_T cfg{ .htsanam = _htsanam,
                         .handle_event = anjitterim_handle_event};
       _jitter = anjitter_model_create( &cfg );
       _jitter->sread( _jitter, file, F_X_STREAM_READ );

       X_tsGetCopyTr101290Errors( _tsHandle, _errors );//получение ошибок

}
MpegAnalyzer::~MpegAnalyzer()
{
       tsanam_impl_destroy(_htsanam);
       delete _servicesTable;
       delete _mainInfo;
}

void MpegAnalyzer::setMainInfo( const X_MAIN_INFO* mainInfo )
{
       *_mainInfo = *mainInfo;
}

void MpegAnalyzer::setServicesTable( const X_SERVICE_TABLE* servicesInfo )
{
       *_servicesTable = *servicesInfo;
```

```
}

void MpegAnalyzer::setErrLog( const X_ERROR_LOG* errLog )
{
       _eventLog = errLog;
}

void MpegAnalyzer::setPidList()
{
       maxCountPid = anmpegext_get_pid_list( _tsHandle, 0, nullptr );
       _pidList = new ANMPEGEXT_PID_INFO_T[maxCountPid];
       _pidDetailInfo = new ANMPEGEXT_PID_DETAIL_INFO_T[maxCountPid];
       anmpegext_get_pid_list( _tsHandle, maxCountPid, _pidList );

       for ( size_t i = 0 ; i < maxCountPid ; i++ ){
              anmpegext_get_pid_info( _tsHandle, _pidList[i].oid,
                                      &_pidDetailInfo[i] );
   }
}

}
}
```

ПРИЛОЖЕНИЕ Б
ИСХОДНЫЙ КОД ОСНОВНОЙ МОДЕЛИ

Листинг Б.1 - MainInfoModel

```
#include "MainInfoModel.h"
#include "Common/Constants.h"
#include "meta\mpegan\MpegAnalyzer.h"
#include "middleware\meta\mpegan\MpegAnEvent.h"

namespace Planar {
namespace Models {

MainInfoModel::MainInfoModel(MpegAnFile* parent):
   QObject(parent),
_parent(parent)
{
_mainInfo = new X_MAIN_INFO{};

_basicInfo  = new BasicInfoModel(this);//создание основной модели данных
_errTR      = new ErrTRModel(this);// создание модели ошибок TR
_percPacket = new PercentPacketModel(this);// создание модели PP

}
MainInfoModel::~MainInfoModel()
{
delete _mainInfo;
}

void MainInfoModel::customEvent(QEvent* event)
{
if ( event->type() == (QEvent::Type)MpegAnEvent::MainInfoType )
{
       *_mainInfo = *(_parent->getMpegAn()->getMainInfo());
       emit dataChanged();
}
QObject::customEvent( event );
}

ErrTRModel* MainInfoModel::getErrTr()const
```

```
{
return _errTR;
}

X_MAIN_INFO* MainInfoModel::getMainInfo()const
{
	return  _mainInfo;
}

PercentPacketModel* MainInfoModel::getPacketModel() const
{
	return _percPacket;
}
BasicInfoModel* MainInfoModel::getBasicInfoModel()const
{
	return _basicInfo;
}

}
}
```

Листинг Б.2 - PrecentPacketModel

```
#include "PercentPacketModel.h"
#include "Common/Constants.h"
#include "MainInfoModel.h"
#include <QColor>

namespace Planar {
namespace Models {

PercentPacketModel::PercentPacketModel(MainInfoModel* parent):
   QAbstractTableModel(parent),
   _parent(parent)
{

}

QVariant PercentPacketModel::headerData(int section, Qt::Orientation orientation,
int role)const
```

```
{
unsigned idx = unsigned(section);

if ( (role == Qt::DisplayRole) && (orientation == Qt::Vertical) &&
     (section >= 0) && (idx < Constants::mainInfoPercentPacket.size()) )
       return TRCONST(Constants::mainInfoPercentPacket.at(idx));

return QVariant();
}

int PercentPacketModel::rowCount(const QModelIndex& parent)const
{
if (!parent.isValid())
       return Constants::mainInfoPercentPacket.size();

return 0;
}

int PercentPacketModel::columnCount(const QModelIndex& parent) const
{
if (!parent.isValid())
       return 1;
return 0;
}

QVariant PercentPacketModel::data(const QModelIndex& index, int role) const
{
int row = index.row();
int column = index.column();
QVariant unswer = 0;

if ( !index.isValid() ||
     !hasIndex(row, column, index.parent())||
     ( (role != Qt::DisplayRole   ) &&
       (role != Qt::BackgroundRole) ) )
return QVariant();

if (role == Qt::DisplayRole) {
       switch (row) {
```

```
                    case ES:
                           unswer                                                    =
QString("%1%").arg(static_cast<double>(*getPercentES())/10);
                           break;
                    case PSI:
                           unswer                                                    =
QString("%1%").arg(static_cast<double>(*getPercentPSI())/10);
                           break;
                    case Stuff:
                           unswer                                                    =
QString("%1%").arg(static_cast<double>(*getPercentStuff())/10);
                           break;
             }
      }
             return unswer;
      }

      Qt::ItemFlags PercentPacketModel::flags(const QModelIndex &index) const
      {
             return Qt::NoItemFlags;
      }

      uint16_t* PercentPacketModel::getPercentES()const
      {
             return &_parent->getMainInfo()->percentES;
      }

      uint16_t* PercentPacketModel::getPercentPSI()const
      {
             return &_parent->getMainInfo()->percentPSI;
      }

      uint16_t* PercentPacketModel::getPercentStuff()const
      {
             return &_parent->getMainInfo()->percentStuff;
      }

      }
      }
```

ПРИЛОЖЕНИЕ В
ИСХОДНЫЙ КОД МОДЕЛИ ОШИБОК

Листинг В.1 - ErrorsModel

```
#include "ErrorsModel.h"
#include "Common\Constants.h"
#include <QColor>
#include <QDebug>
extern "C"{
#include "Models\MpegAn\Tree.h"
}
namespace Planar {
namespace Models {

ErrorsModel::ErrorsModel(MpegAnFile* parent, unsigned index):
    QAbstractTableModel(parent),
    _parent(parent),
    _modelIndex(index)
{
}

int ErrorsModel::rowCount(const QModelIndex &parent) const
{
if (parent.isValid())
       return 0;

       return Constants::errNames.at(_modelIndex)->size();
}

int ErrorsModel::columnCount(const QModelIndex &parent) const
{
if (parent.isValid())
       return 0;

return 1;
}

QVariant ErrorsModel::headerData( int section,
```

```
                                    Qt::Orientation orientation,
                                    int role ) const
{
unsigned idx = unsigned(section);
if ( role == Qt::DisplayRole && orientation == Qt::Vertical)
{
       return Constants::errNames.at(_modelIndex)->at(idx);
}
return QVariant();
}

QVariant ErrorsModel::data(const QModelIndex &index, int role) const
{
if (!index.isValid() ||
        !hasIndex(index.row(), index.column(), index.parent()))
       return QVariant();
if( role == Qt::DisplayRole )
{
       switch (_modelIndex)
       {
       case First:
              return dataFirst(index, role);
       case Second:
              return dataSecond(index, role);
       case Third:
              return dataThird(index, role);
       }
}
if( role == Qt::BackgroundColorRole )
{
       if(index.data(Qt::DisplayRole).toBool())
              return QVariant(QColor(253,47,64));
}

if ( role == Qt::ForegroundRole )
{
       if (index.data(Qt::DisplayRole).toBool())
              return QVariant(QColor(Qt::white));
```

```
}

return QVariant();
}

Qt::ItemFlags ErrorsModel::flags(const QModelIndex &index) const
{
if( data(index,Qt::DisplayRole) != 0 )
       return Qt::ItemIsEnabled;
return Qt::NoItemFlags;
}
//Получение ошибок 1-ого приоритета
QVariant ErrorsModel::dataFirst(const QModelIndex& index, int role) const
{
int row = index.row();

if( role == Qt::DisplayRole )
{
       uint64_t errscount = 0;
       const X_TR101290_1ts* errs = &_parent->getMpegAn()->getErrors()->first;

       switch( row )
       {
           case 0:
                  errscount += errs->tsSyncLossCnt;
             break;
           case 1:
                  errscount += errs->syncByteErrorCnt;
             break;
           case 2:
                  errscount += errs->patErrors2.falseTableIdFoundCnt;
                     errscount += errs->patErrors2.maxIntervalErrorCnt;
                     errscount += errs->patErrors2.scramblingErrorCnt;
             break;
           case 3:
                  errscount += errs->continuityCounterErrorCnt;
             break;
           case 4:
                  errscount += errs->pmtErrors2.maxIntervalErrorCnt;
```

```
errscount += errs->pmtErrors2.scramblingErrorCnt;
             break;
          case 5:
                errscount += errs->pidErrorCnt;
             break;
       }

       return errscount;

}

return QVariant();
}
//Получение ошибок 2-ого приоритета
QVariant ErrorsModel::dataSecond(const QModelIndex& index, int role) const
{
int row = index.row();

if( role == Qt::DisplayRole )
{
       uint64_t errscount = 0;
       const X_TR101290_2nd* errs = &_parent->getMpegAn()->getErrors()->second;

       switch( row )
       {
           case 0:
                 errscount += errs->transportErrorCnt;
             break;
           case 1:
                 errscount += errs->crcErrors.patCnt;
                    errscount += errs->crcErrors.catCnt;
                    errscount += errs->crcErrors.pmtCnt;
                    errscount += errs->crcErrors.nitCnt;
                    errscount += errs->crcErrors.nitOthCnt;
                    errscount += errs->crcErrors.eitCnt;
                    errscount += errs->crcErrors.eitOthCnt;
                    errscount += errs->crcErrors.eitSchCnt;
                    errscount += errs->crcErrors.eitOthSchCnt;
                    errscount += errs->crcErrors.batCnt;
```

```
errscount += errs->crcErrors.sdtCnt;
                    errscount += errs->crcErrors.sdtOthCnt;
                    errscount += errs->crcErrors.totCnt;
              break;
           case 2:
                 errscount += errs->pcrRepetitionCnt;
              break;
           case 3:
                 errscount += errs->pcrDiscontinuityErrorCnt;
              break;
           case 4:
                 errscount += errs->pcrAccuracyErrorCnt;
              break;
           case 5:
                 errscount += errs->pcrOveralErrorCnt;
              break;
           case 6:
                 errscount += errs->ptsErrorCnt;
              break;
           case 7:
                 errscount += errs->catErrors.falseTableIdFoundCnt;
                    errscount += errs->catErrors.scramblingErrorCnt;
              break;
       }

       return errscount;
}

return QVariant();
}
//Получение ошибок 3-ого приоритета
QVariant ErrorsModel::dataThird(const QModelIndex& index, int role) const
{
int row = index.row();

if( role == Qt::DisplayRole )
{
       uint64_t errscount = 0;
       const X_TR101290_3rd* errs = &_parent->getMpegAn()->getErrors()->third;
```

```
switch( row )
{
    case 0:
          errscount += errs->nitActualErrors.falseTableIdFoundCnt;
             errscount += errs->nitActualErrors.maxIntervalErrorCnt;
             errscount += errs->nitActualErrors.minIntervalErrorCnt;
      break;
    case 1:
          errscount += errs->nitOtherErrors.maxIntervalErrorCnt;
      break;
    case 2:
          errscount += errs->siRepetitionCnt;
      break;
    case 3:
          errscount += errs->unreferencedPidCnt;
      break;
    case 4:
          errscount += errs->sdtActualErrors.falseTableIdFoundCnt;
             errscount += errs->sdtActualErrors.maxIntervalErrorCnt;
             errscount += errs->sdtActualErrors.minIntervalErrorCnt;
      break;
    case 5:
          errscount += errs->sdtOtherErrors.maxIntervalErrorCnt;
      break;
    case 6:
          errscount += errs->eitActualErrorCnt.falseTableIdFoundCnt;
             errscount += errs->eitActualErrorCnt.maxIntervalSec0ErrorCnt;
             errscount += errs->eitActualErrorCnt.maxIntervalSec1ErrorCnt;
             errscount += errs->eitActualErrorCnt.minIntervalErrorCnt;
      break;
    case 7:
          errscount += errs->eitOtherErrorCnt.maxIntervalSec0ErrorCnt;
             errscount += errs->eitOtherErrorCnt.maxIntervalSec1ErrorCnt;
      break;
    case 8:
          errscount += errs->eitPFErrorCnt;
      break;
    case 9:
          errscount += errs->rstErrorCnt.falseTableIdFoundCnt;
```

```
                    errscount += errs->rstErrorCnt.minIntervalErrorCnt;
                break;
            case 10:
                  errscount += errs->tdtErrorCnt.falseTableIdFoundCnt;
                    errscount += errs->tdtErrorCnt.maxIntervalErrorCnt;
                    errscount += errs->tdtErrorCnt.minIntervalErrorCnt;
                break;
        }

        return errscount;

}

return QVariant();
}
//получение конкретной ошибки любого приоритета
const X_ListOIDs* ErrorsModel::trerrstree_node_get_object( int row)
{
const X_ListOIDs* list;
switch (_modelIndex)
{
case First:
        list = prio1_sub_nodes_get_oid_list(row);
        break;
case Second:
        list = prio2_sub_nodes_get_oid_list(row);
        break;
case Third:
        list = prio3_sub_nodes_get_oid_list(row);
        break;
}

return list;
}
//дополнительная информация для ошибок 1-ого приоритета
const X_ListOIDs* ErrorsModel::prio1_sub_nodes_get_oid_list( int row )
{
const X_ListOIDs* oidlist = NULL;
const X_TR101290_1ts* errs = &_parent->getMpegAn()->getErrors()->first;
```

```
switch( row )
{
    case TR101290_CONTINUITY_COUNT:
           oidlist = &errs->contCountErrOidList;
      break;
    case TR101290_PID_ERR:
           oidlist = &errs->pidErrOidList;
      break;
}

return oidlist;
}
//получение доп.инф. для 2-ого приоритета
const X_ListOIDs* ErrorsModel::prio2_sub_nodes_get_oid_list( int row)
{
const X_ListOIDs* oidlist = NULL;
const X_TR101290_2nd* errs = &_parent->getMpegAn()->getErrors()->second;


switch( row )
{
    case TR101290_PCR_REPETITION:
           oidlist = &errs->pcrRepErrOidList;
      break;
    case TR101290_PCR_DISCONTINUITY:
           oidlist = &errs->pcrDiscontErrOidList;
      break;
    case TR101290_PCR_ACCURACY:
           oidlist = &errs->pcrAccuracyErrOidList;
      break;
    case TR101290_PCR_FO_DR:
           oidlist = &errs->pcrOveralErrOidList;
      break;
    case TR101290_PTS_ERR:
           oidlist = &errs->pcrPtsErrOidList;
      break;
}

return oidlist;
```

```
}
//получение доп.инф. для 3 приоритета
const X_ListOIDs* ErrorsModel::prio3_sub_nodes_get_oid_list( int row )
{
const X_ListOIDs* oidlist = 0;
const X_TR101290_3rd* errs = &_parent->getMpegAn()->getErrors()->third;


switch( row )
{
    case TR101290_NIT_OTHER_ERR:
           oidlist = &errs->nitOtherErrors.maxIntervalErrOidList;
      break;
    case TR101290_SI_REPETITION_ERR:
           oidlist = &errs->siRepErrOidList;
      break;
    case TR101290_UNREFERENCED_PID:
           oidlist = &errs->unrefPidErrOidList;
      break;
    case TR101290_SDT_OTHER_ERR:
           oidlist = &errs->sdtOtherErrors.maxIntervalErrOidList;
      break;
    case TR101290_EIT_P_F_ERR:
           oidlist = &errs->eitPFErrOidList;
      break;
}

return oidlist;
}
/*-------------------------------------------------------------------------*/
//получение списка ошибок
QList<const X_ListOIDs*>* ErrorsModel::getOidList(QStringList& list, int row)
{
QList<const X_ListOIDs*>*listOIDs = nullptr;
switch (_modelIndex)
{
case First:
       listOIDs = getPrio1OidList(list,row);
       break;
```

```
case Second:
       listOIDs = getPrio2OidList(list,row);
       break;
case Third:
       listOIDs = getPrio3OidList(list,row);
       break;
}

return listOIDs;
}
//получение списка для 1-ого приоритета
QList<const X_ListOIDs*>* ErrorsModel::getPrio1OidList(QStringList& list,int row)
{
TS_Char str[96];
QList<const X_ListOIDs*>*listOIDs = nullptr;
X_TR101290_ERRORS* errors = _parent->getMpegAn()->getErrors();

switch( row )
{
    case TR101290_PAT_ERR2:
    {
           for(int i = 0;i<3;i++)
              {
                     fill_pat_err2_name(errors, i, str, sizeof(str));
                     uint64_t errCount = get_pat_err2_counter(errors,i);
                     list.append( QString(QString::fromWCharArray(str))
                                  + " (Errors:"+QString::number(errCount)
                                  + ")");
              }
    }
       break;
    case TR101290_PMT_ERR2:
    {
           listOIDs = new QList<const X_ListOIDs*>;
              for(int i = 0;i<2;i++)
              {
                     fill_pmt_err2_name(errors, i, str, sizeof(str));
                     uint64_t errCount = get_pmt_err2_counter(errors,i);
                     list.append( QString(QString::fromWCharArray(str))
```

```
                                + " (Errors:"+QString::number(errCount)

                                + ")" );
                        listOIDs->append(pmt_err2_sub_nodes_get_oid_list(errors,i));
                }
        }
           break;
        default:
               return nullptr;

}
return listOIDs;

}
//получение списка для 2-ого приоритета
QList<const X_ListOIDs*>* ErrorsModel::getPrio2OidList(QStringList& list,int row)
{
TS_Char str[96];
QList<const X_ListOIDs*>*listOIDs = nullptr;
X_TR101290_ERRORS* errors = _parent->getMpegAn()->getErrors();

switch( row )
{
    case TR101290_CRC_ERR:
    {
           listOIDs = new QList<const X_ListOIDs*>;
              for(int i = 0;i<13;i++)
              {
                     fill_crc_err_name( errors, i, str, sizeof(str) );
                     uint64_t errCount = get_crc_err_counter( errors, i );
                     list.append( QString( QString::fromWCharArray( str ) )
                                + " (Errors:"
                                + QString::number( errCount )
                                + ")" );
                     listOIDs->append( crc_err_sub_nodes_get_oid_list( errors, i ));
              }
    }
       break;
```

```
    case TR101290_CAT_ERR:
    {
           listOIDs = new QList<const X_ListOIDs*>;
             for(int i = 0;i<2;i++)
             {
                    fill_cat_err_name(errors, i, str, sizeof(str));
                    uint64_t errCount = get_cat_err_counter(errors,i);
                    list.append( QString( QString::fromWCharArray(str) )
                                 + " (Errors:"
                                 + QString::number(errCount)
                                 + ")" );
                    listOIDs->append(cat_err_sub_nodes_get_oid_list( errors, i ));
             }
    }
       break;
    default:
           return nullptr;

}
return listOIDs;

}
//получение списка ошибок для 3-его приоритета
QList<const X_ListOIDs *> *ErrorsModel::getPrio3OidList(QStringList &list, int row)
{
TS_Char str[96];
QList<const X_ListOIDs*>*listOIDs = nullptr;
X_TR101290_ERRORS* errors = _parent->getMpegAn()->getErrors();

switch( row )
{
    case TR101290_NIT_ACTUAL_ERR:
    {
           for(int i = 0;i<3;i++)
             {
                    fill_nitact_err_name(errors, i, str, sizeof(str));
                    uint64_t errCount = get_nitact_err_counter(errors,i);
                    list.append( QString(QString::fromWCharArray(str))
                                 + " (Errors:"
```

```
                                + QString::number(errCount)
                                + ")");
            }
}
   break;
case TR101290_SDT_ACTUAL_ERR:
{
       for(int i = 0;i<3;i++)
          {
                 fill_sdtact_err_name(errors, i, str, sizeof(str));
                 uint64_t errCount = get_sdtact_err_counter(errors,i);
                 list.append( QString(QString::fromWCharArray(str))
                                + " (Errors:"
                                + QString::number(errCount)
                                + ")");
            }
}
   break;
case TR101290_EIT_ACTUAL_ERR:
{
       listOIDs = new QList<const X_ListOIDs*>;
          for(int i = 0;i<4;i++)
          {
                 fill_eitact_err_name(errors, i, str, sizeof(str));
                 uint64_t errCount = get_eitact_err_counter(errors,i);
                 list.append( QString(QString::fromWCharArray(str))
                                + " (Errors:"
                                + QString::number(errCount)
                                + ")");
                 listOIDs->append(eitact_err_sub_nodes_get_oid_list(errors,i));
            }
}
   break;
case TR101290_EIT_OTHER_ERR:
{
       listOIDs = new QList<const X_ListOIDs*>;
       for(int i = 0;i<2;i++)
          {
```

```
                    fill_eitoth_err_name(errors, i, str, sizeof(str));
                    uint64_t errCount = get_eitoth_err_counter(errors,i);
                    list.append( QString(QString::fromWCharArray(str))
                                + " (Errors:"
                                + QString::number(errCount)
                                + ")");
                    listOIDs->append(eitoth_err_sub_nodes_get_oid_list(errors,i));
             }
    }
       break;
    case TR101290_TDT_ERR:
    {
           for(int i = 0;i<3;i++)
             {
                    fill_tdt_err_name(errors, i, str, sizeof(str));
                    uint64_t errCount = get_tdt_err_counter(errors,i);
                    list.append( QString(QString::fromWCharArray(str))
                                + " (Errors:"
                                + QString::number(errCount)
                                + ")");
             }
    }
       break;
    default:
           return nullptr;

}
return listOIDs;

}
}
}
```

ПРИЛОЖЕНИЕ Г
ИСХОДНЫЙ КОД МОДЕЛИ ДЕРЕВА ТАБЛИЦ

Листинг Г.1 - TableTreeModel

```
#include "TableTreeModel.h"
#include <QHash>

namespace Planar {
namespace Models {

TableTreeModel::TableTreeModel(MpegAnFile *parent ):
   QAbstractItemModel(parent),
   _parent(parent)
{
       _hash = new QHash<tVirtualTreeNodeId, tVirtualTreeNodeId*>; //создание хэш дерева
       nodes = new QHash<tVirtualTreeNodeId, QModelIndex>; //создание хэша индексов в дереве
}

TableTreeModel::~TableTreeModel()
{
QHashIterator<tVirtualTreeNodeId, tVirtualTreeNodeId*> iterator(*_hash);
while (iterator.hasNext()) {
       iterator.next();
       if(iterator.key() == _rootId || iterator.key() == _firstChildId)
              continue;
       delete iterator.value();
}
_hash->clear();
delete _hash;
delete nodes;
mpegTblTreeDelete(_tree);
}
//создания корня дерева
void TableTreeModel::setRoot()
{
_context.htsanam = _parent->getMpegAn()->getHtsanam();
_tsanState.hcontext = static_cast<H_TSAN_CONTEXT>( &_context );
```

```
	_tree = mpegTblTreeCreate( &_tsanState );
	if (!_tree->meth->fGetNode( _tree, FIRST_NODE_ID, &_firstRootChild ))
		return;

	_firstChildId = _firstRootChild.id;
	_rootId =~_rootId;

	_hash->insert( _rootId, &_rootId );
	_hash->insert( _firstChildId, &_firstChildId );

	_countChildRoot = 1;

	tVirtualTreeNode nextNode,
				currentNode;
	currentNode = _firstRootChild;
	while( _tree->meth->fGetNextSiblingNode( _tree, &currentNode, &nextNode ) )
	{
		_countChildRoot++;
		currentNode = nextNode;
	}
	}
	//получение 1-ого дочернего элемента
	void TableTreeModel::getFirstChild( const tVirtualTreeNodeId& parentId,
							tVirtualTreeNodeId&
firstChildId )const
	{
	if ( parentId == _rootId )
	{
		firstChildId = _firstChildId;
	}
	else
	{
		tVirtualTreeNode firstChildNode,
					nodeParent;

		_tree->meth->fGetNode( _tree, parentId, &nodeParent );
		_tree->meth->fGetChildNode( _tree, &nodeParent, &firstChildNode );
		firstChildId = firstChildNode.id;
```

```
	}
	}

	QVariant TableTreeModel::data( const QModelIndex &index, int role ) const
	{
	if (!index.isValid()) {
		return QVariant();
	}

	if (role == Qt::DisplayRole)
	{
		tVirtualTreeNodeId key = nodeByIndex( index );
		tVirtualTreeNode node;

		_tree->meth->fGetNode( _tree, key, &node );
		mpegTblTreeOnDraw( &node, _tree );

		return getText(&node);
	}
	return QVariant();
	}
	// индекс элемента
	QModelIndex TableTreeModel::index( int row, int column,const QModelIndex &parent
)const
	{
	if ( !hasIndex( row, column, parent ) || column != 0 )
		return QModelIndex();

	tVirtualTreeNodeId firstChildId;
	tVirtualTreeNode firstChildNode;
	QModelIndex createdIndex;

	tVirtualTreeNodeId keyParent = nodeByIndex( parent );

	getFirstChild( keyParent, firstChildId );

	_tree->meth->fGetNode( _tree, firstChildId, &firstChildNode );
```

```
for( int i = 0; i < row ; i++ )
{
	_tree->meth->fGetNextSiblingNode( _tree, &firstChildNode, &firstChildNode );
}

if (_hash->contains(firstChildNode.id))
{
	createdIndex = createIndex( row, column, _hash->value(firstChildNode.id));
}
else
{
	tVirtualTreeNodeId* nodeID = new tVirtualTreeNodeId;
	*nodeID = firstChildNode.id;
	_hash->insert( *nodeID, nodeID );

	createdIndex = createIndex( row, column, nodeID );
	nodes->insert( *nodeID, createdIndex );
}

return createdIndex;
}
//получение родительского узла по индексу
QModelIndex TableTreeModel::parent( const QModelIndex &child )const
{
QModelIndex createdIndex;
tVirtualTreeNodeId keyChild = nodeByIndex( child );
tVirtualTreeNode childNode;
_tree->meth->fGetNode( _tree, keyChild, &childNode );

tVirtualTreeNode parentNode;

if (!_tree->meth->fGetParentNode( _tree, &childNode, &parentNode ))
{
	return QModelIndex();
}

tVirtualTreeNode prevNode;
tVirtualTreeNode currentNode = parentNode;
int row = 0;
```

```
while( _tree->meth->fGetPrevSiblingNode( _tree, &currentNode, &prevNode ) )
{
	row++;
	currentNode = prevNode;
}

if (_hash->contains(parentNode.id))
{
	createdIndex = createIndex( row, 0, _hash->value(parentNode.id));
}
else return QModelIndex();

return createdIndex;
}
//получение кол-во элементов в узле
int TableTreeModel::rowCount( const QModelIndex &parent )const
{
tVirtualTreeNodeId keyID = nodeByIndex(parent);
if (keyID == _rootId)
	return _countChildRoot;

return childCount( &keyID );
}

int TableTreeModel::columnCount( const QModelIndex &parent )const
{
Q_UNUSED(parent)
return 1;
}

tVirtualTreeNodeId TableTreeModel::nodeByIndex(const QModelIndex &index) const
{
if(!index.isValid())
	return -1;
tVirtualTreeNodeId *id = static_cast<tVirtualTreeNodeId*>(index.internalPointer());
return *id;
}
```

```
	int TableTreeModel::childCount( const tVirtualTreeNodeId* parentId ) const
	{
	int count = 0;
	tVirtualTreeNode firstChildNode,
	                    currentNode,
	                                  parentNode,
	                                    nextNode;
	if ( *parentId == _rootId)
	{
		return _countChildRoot;
	}
	else
	{
		_tree->meth->fGetNode( _tree, *parentId, &parentNode );

		if( _tree->meth->fGetChildNode( _tree, &parentNode, &firstChildNode ) )
		{
			currentNode = firstChildNode;
			count++;
			while(    _tree->meth->fGetNextSiblingNode(    _tree,    &currentNode,
&nextNode ) )
			{
				count++;
				currentNode = nextNode;
			}
		}
	}
		return count;
	}
	QString TableTreeModel::getText( const tVirtualTreeNode* node ) const
	{
	QString text;
	node->value[0] ? (text = QString("%1:%2").arg(QString::fromWCharArray(node->key),
	                                                QString::fromWCharArray(node-
>value)))
	                  : (text = QString::fromWCharArray(node->key));
		return text;
	}
```

```
//получение строки пути до данного индекса
void TableTreeModel::getNodeNameForPath( const QModelIndex &index,
                                         TS_Char* nodeName )
{
   tVirtualTreeNodeId keyID = nodeByIndex(index);
   tVirtualTreeNode node;
   _tree->meth->fGetNode( _tree, keyID, &node );
   mpegTblTreeGetNodeNameForPath( _tree, nodeName,
                                  MAX_TABLE_NAME_STR_SIZE * sizeof(TS_Char), &node );
}
//получение следующей ошибки
QModelIndex TableTreeModel::getNextError(const QModelIndex &index)
{
if(!index.isValid())
       return QModelIndex();
tVirtualTreeNodeId keyID = nodeByIndex(index);
tVirtualTreeNodeId idNextErr;
idNextErr = mpegTblTreeGetNextError( _tree, keyID );

if (idNextErr == INVALID_NODE_ID)
       return QModelIndex();
       return nodes->value(idNextErr);
}
//получение индекса пред. ошибки
QModelIndex TableTreeModel::getPrevError(const QModelIndex &index)
{
if(!index.isValid())
       return QModelIndex();
tVirtualTreeNodeId keyID = nodeByIndex(index);
tVirtualTreeNodeId idPrevErr = INVALID_NODE_ID;
idPrevErr = mpegTblTreeGetPrevError(_tree, keyID );

if (idPrevErr == INVALID_NODE_ID)
       return QModelIndex();

return nodes->value(idPrevErr);
}
}
}
```