

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего
профессионального образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра комплексной информационной безопасности
электронно-вычислительных систем (КИБЭВС)

УТВЕРЖДАЮ

заведующий каф.КИБЭВС

_____ А.А. Шелупанов

" ____ " _____ 2014г.

КОМПЬЮТЕРНАЯ ЭКСПЕРТИЗА

Отчет по групповому проектному обучению

Группа КИБЭВС-1208

Ответственный исполнитель

Студент гр. 720-1

_____ Никифоров Д. С.

" ____ " _____ 2014г.

Научный руководитель

Аспирант каф.КИБЭВС

_____ Гуляев А. И.

" ____ " _____ 2014г.

РЕФЕРАТ

Курсовая работа содержит 70 страниц, 21 рисунок, 3 таблицы, 14 источников, 1 приложение.

КОМПЬЮТЕРНАЯ ЭКСПЕРТИЗА, ФОРЕНЗИКА, ЛОГИ, QT, XML, GIT, LATEX, ЖУРНАЛЬНЫЕ ФАЙЛЫ, SKYPE, PIDGIN, WINDOWS, СЕРТИФИКАЦИЯ, РЕПОЗИТОРИЙ, XSLT, LIB, БИБЛИОТЕКИ.

Цель работы — создание программного комплекса, предназначенного для проведения компьютерной экспертизы.

Задачей, поставленной на данный семестр, стало написание программного комплекса, имеющего следующие возможности:

- 1) сбор и анализ событий системных журналов операционной системы;
- 2) сбор и анализ информации из журналов истории браузеров;
- 3) сбор и анализ истории переписки мессенджеров;
- 4) сбор и анализ событий журнальных файлов приложений;
- 5) поиск файлов по имени.

Результатами работы в данном семестре являются:

- разработка архитектуры проекта;
- использование в разработке системы контроля версий GIT;
- использование Qt — кроссплатформенной библиотеки C++;
- изучение вопроса сертификации для возможности внедрения данного комплекса в гос. структуры, занимающиеся информационной безопасностью;
- использование системы компьютерной вёрстки TeX для написания документации.

Пояснительная записка выполнена в текстовом редакторе Vim.

Список исполнителей

Моргуненко А.В. – документатор.

Никифоров Д.С. – программист, ответственный исполнитель, ответственный за написание части системы, работающей с логами системы.

Поляков И.Ю. – программист, ответственный за написание части системы, работающей с логами мессенджеров.

Пономарёв А.К. – аналитик.

Содержание

1	Введение	10
2	Назначение и область применения	11
3	Постановка задачи	11
4	Инструменты	11
4.1	Система контроля версий Git	11
4.2	Система компьютерной вёрстки T _E X	13
4.3	Qt - кроссплатформенный инструментарий разработки ПО	13
4.3.1	Автоматизация поиска журнальных файлов	16
4.3.2	Реализация сохранения результатов работы программного комплекса в XML	17
5	Технические характеристики	17
5.1	Требования к аппаратному обеспечению	17
5.2	Требования к программному обеспечению	17
5.3	Выбор единого формата выходных файлов	18
6	Разработка программного обеспечения	19
6.1	Архитектура	19
6.1.1	Основной алгоритм	19
6.1.2	Описание основных функций модуля системы	21
6.2	Сбор и анализ системных журналов Windows (III семестр)	23
6.2.1	Общие сведения о журнальных файлах	23
6.2.2	Структура журнальных файлов операционной системы Windows XP	24
6.2.3	Алгоритм работы модуля	26
6.3	Сбор и анализ системных журналов Windows (IV семестр)	28
6.3.1	Введение	28

6.3.2	Задачи на семестр	29
6.3.3	Новая архитектура	30
6.3.4	библиотеки в C++ и их использование	31
6.3.5	task = динамическая библиотека	32
6.3.6	новый механизм работы с задачами	34
6.3.7	мерджим ветки git	34
6.4	Сбор и анализ истории переписки мессенджеров (III семестр) . . .	36
6.4.1	Определение местоположения логов переписки мессенджера Skype	36
6.4.2	Определение местоположения логов переписки мессенджера Pidgin	36
6.4.3	Формат хранения переписки мессенджера Skype	38
6.4.4	Формат хранения переписки мессенджера Pidgin	38
6.4.5	Алгоритм работы модуля Skype	39
6.4.6	Алгоритм работы модуля Pidgin	41
6.4.7	Структура логов Skype, сохранённых в XML	42
6.4.8	Структура логов Pidgin, сохранённых в XML	43
6.5	Сбор и анализ истории переписки мессенджеров (IV семестр) . . .	44
6.5.1	Создание веток (branch), и распределенная работа програм- мистов.	45
6.6	Визуализация данных	46
6.7	Выбор и описание утилит для преобразования	47
6.8	XSTL	47
6.9	Результаты работы	48
6.10	Описание работы SAX и DOM модели разбора XML	48
6.11	функция чтения используемых аккаунтов и контактной книги pidgin	50
6.12	функция сбора дополнительных данных skype	53
7	Некоторые аспекты сертификации программных средств объектов информатизации по требованиям информационной безопасности	55

7.1	Схема проведения сертификации	56
7.2	Добровольная сертификация	57
7.3	Обязательная сертификация	58
7.4	Список организаций, проводящих сертификацию	59
8	Создание собственного репозитория для debian и ubuntu	64
8.0.1	Порядок конфигурирования репозитория	64
9	Цели следующего семестра	67
10	Заключение	67
	Список использованных источников	68
	Приложение А Компакт-диск	70

1 Введение

Компьютерно-техническая экспертиза – это самостоятельный род судебных экспертиз, относящийся к классу инженерно-технических экспертиз, проводимых в следующих целях: определения статуса объекта как компьютерного средства, выявление и изучение его роли в рассматриваемом деле, а так же получения доступа к информации на электронных носителях с последующим всесторонним её исследованием [1]. Компьютерная экспертиза помогает получить доказательственную информацию и установить факты, имеющие значение для уголовных, гражданских и административных дел, сопряжённых с использованием компьютерных технологий. Для проведения компьютерных экспертиз необходима высокая квалификация экспертов, так как при изучении представленных носителей информации, попытке к ним доступа и сбора информации возможно внесение в информационную среду изменений или полная утрата важных данных.

Компьютерная экспертиза, в отличие от компьютерно-технической экспертизы, затрагивает только информационную составляющую, в то время как аппаратная часть и её связь с программной средой не рассматривается.

На протяжении предыдущих семестров нами были рассмотрены такие направления компьютерной экспертизы, как исследование файловых систем, сетевых протоколов, организация работы серверных систем, механизм журналирования событий. Также нами были изучены основные задачи, которые ставятся перед сотрудниками правоохранительных органов, которые проводят компьютерную экспертизу, и набор чувствующих утилит, способных помочь эксперту в проведении компьютерной экспертизы. Было выявлено, что существует множество разрозненных программ, предназначенных для просмотра лог-файлов системы и таких приложений, как мессенджеры и браузеры, но для каждого вида лог-файлов необходимо искать отдельную программу. Так как ни одна из них не позволяет эксперту собрать воедино и просмотреть все логи системы, браузеров и мессенджеров, было решено создать для этой цели собственный автоматизированный комплекс, которому на данный момент нет аналогов.

2 Назначение и область применения

Разрабатываемый комплекс предназначен для автоматизации процесса сбора информации с исследуемого образа жёсткого диска.

3 Постановка задачи

На данный семестр были поставлены следующие задачи:

- определить средства для разработки;
- разработать архитектуру проекта;
- реализовать структуру проекта;
- определить единый формат выходных данных;
- реализовать несколько модулей.

Задачи по проектированию модулей:

- 1) сбор и анализ истории переписки пользователей системы Windows из Pidgin и Skype;
- 2) написание автоматизированных модулей для сбора истории переписки пользователей системы Windows из таких программ как Pidgin и Skype;
- 3) сбор и анализ событий журнальных файлов Windows;
- 4) написание автоматизированного модуля для сбора событий журнальных файлов Windows.

4 Инструменты

4.1 Система контроля версий Git

Для разработки программного комплекса для проведения компьютерной экспертизы решено использовать Git.

Git — распределённая система управления версиями файлов. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, как противоположность системе управления версиями Subversion (также известная как «SVN») [2].

Необходимость использования системы версий, очевидна. Так как в группе несколько программистов и тестер, мы имеем:

- возможность удаленной работы с исходными кодами;
- возможность создавать свои ветки, не мешая при этом другим разработчикам;
- доступ к последним изменениям в коде, т.к. все исходники хранятся на сервере git.keva.su ;
- исходные коды защищены, доступ к ним можно получить лишь имея RSA-ключ;
- возможность откатиться к любой стабильной стадии проекта.

Основные постулаты работы с кодом в системе Git:

- каждая задача решается в своей ветке;
- коммитим сразу, как что-то получили осмысленное;
- в master мержится не разработчиком, а вторым человеком, который производит вычитку и тестирование изменения;
- все коммиты должны быть осмысленно подписаны/прокомментированы.

Для работы над проектом нами был поднят собственный репозиторий на сервере git.keva.su. Адреса репозитория следующие:

Исходные файлы проекта:

```
git clone git@git.keva.su:gpo.git gpo.git
```

Репозиторий для тестирования проекта:

```
git clone git@git.keva.su:gpo-testdata.git gpo-testdata.git
```

4.2 Система компьютерной вёрстки T_EX

T_EX — это созданная американским математиком и программистом Дональдом Кнутом система для вёрстки текстов. Сам по себе T_EX представляет собой специализированный язык программирования. Каждая издательская система представляет собой пакет макроопределений этого языка.

L^AT_EX — это созданная Лэсли Лэмпортом издательская система на базе T_EX'a[3]. L^AT_EX позволяет пользователю сконцентрировать свои усилия на содержании и структуре текста, не заботясь о деталях его оформления.

Для подготовки отчётной и иной документации нами был выбран L^AT_EX так как совместно с системой контроля версий Git он предоставляет возможность совместного создания и редактирования документов. Огромным достоинством системы L^AT_EX то, что создаваемые с её помощью файлы обладают высокой степенью переносимости [4].

Совместно с L^AT_EX часто используется BibT_EX — программное обеспечение для создания форматированных списков библиографии. Оно входит в состав дистрибутива L^AT_EX и позволяет создавать удобную, универсальную и долговечную библиографию. BibT_EX стал одной из причин, по которой нами был выбран L^AT_EX для создания документации.

4.3 Qt - кроссплатформенный инструментарий разработки ПО

Qt - это кроссплатформенная библиотека C++ классов для создания графических пользовательских интерфейсов (GUI) от фирмы Digia. Эта библиотека полностью объектно-ориентированная, что обеспечивает легкое расширение возможностей и создание новых компонентов. Ко всему прочему, она поддерживает огромное количество платформ.

Qt позволяет запускать написанное с его помощью ПО в большинстве современных операционных систем путём простой компиляции программы для каждой ОС без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспе-

чения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Qt является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования.

Список использованных классов фреймворка QT

- `iostream`
- `QChar`
- `QCryptographicHash`
- `QDir`
- `QDirIterator`
- `QFile`
- `QFileInfo`
- `QIODevice`
- `QList`
- `QRegExp`
- `QString`
- `QTextStream`
- `QtSql/SqlDatabase`
- `QVector`
- `QXmlStreamReader`
- `QXmlStreamWriter`
- `Conversations`

Класс `QXmlStreamWriter` представляет собой XML писателя с простым потоковым.

Класс `QXmlStreamReader` представляет собой быстрый синтаксически корректный XML анализатор с простым потоковым API.

QVector представляет собой класс для создания динамических массивов.

Модуль QtSql/QtSqlDatabase помогает обеспечить однородную интеграцию БД в ваши Qt приложения.

Класс QTextStream предоставляет удобный интерфейс для чтения и записи текста.

QTextStream может взаимодействовать с QIODevice, QByteArray или QString. Используя потоковые операторы QTextStream, вы можете легко читать и записывать слова, строки и числа. При формировании текста QTextStream поддерживает параметры форматирования для заполнения и выравнивания полей и форматирования чисел. [5]

Класс QString предоставляет строку символов Unicode.

QString хранит строку 16-битных QChar, где каждому QChar соответствует один символ Unicode 4.0. (Символы Unicode со значениями кодов больше 65535 хранятся с использованием суррогатных пар, т.е. двух последовательных QChar.)

Unicode - это международный стандарт, который поддерживает большинство используемых сегодня систем письменности. Это расширение US-ASCII (ANSI X3.4-1986) и Latin-1 (ISO 8859-1), где все символы US-ASCII/Latin-1 доступны на позициях с тем же кодом.

Внутри QString использует неявное совместное использование данных (копирование-при-записи), чтобы уменьшить использование памяти и избежать ненужного копирования данных. Это также позволяет снизить накладные расходы, свойственные хранению 16-битных символов вместо 8-битных.

В дополнение к QString Qt также предоставляет класс QByteArray для хранения сырых байт и традиционных нультерминальных строк. В большинстве случаев QString - необходимый для использования класс. Он используется во всем API Qt, а поддержка Unicode гарантирует, что ваши приложения можно будет легко перевести на другой язык, если в какой-то момент вы захотите увеличить их рынок распространения. Два основных случая, когда уместно использование QByteArray: когда вам необходимо хранить сырые двоичные данные и когда критично использование памяти (например, в Qt для встраиваемых Linux-систем). [6]

Класс `QRegExp` предоставляет сопоставление с образцом при помощи регулярных выражений.

Регулярное выражение, или "regex", представляет собой образец для поиска соответствующей подстроки в тексте. Это полезно во многих ситуациях, например:

Проверка правильности – регулярное выражение может проверить, соответствует ли подстрока каким-либо критериям, например, целое ли она число или не содержит ли пробелов. Поиск – регулярное выражение предоставляет более мощные шаблоны, чем простое соответствие строки, например, соответствие одному из слов `mail`, `letter` или `correspondence`, но не словам `email`, `mailman`, `mailer`, `letterbox` и т.д. Поиск и замена – регулярное выражение может заменить все вхождения подстроки другой подстрокой, например, заменить все вхождения `&` на `&`, исключая случаи, когда за `&` уже следует `amp;`. Разделение строки – регулярное выражение может быть использовано для определения того, где строка должна быть разделена на части, например, разделяя строку по символам табуляции.

`QFileInfo` - Во время поиска возвращает полную информацию о файле.

Класс `QDir` обеспечивает доступ к структуре каталогов и их содержимого.

`QIODevice` представляет собой базовый класс всех устройств ввода/вывода в Qt.

Класс `QCryptographicHash` предоставляет способ генерации криптографических хэшей. `QCryptographicHash` могут быть использованы для генерации криптографических хэшей двоичных или текстовых данных. В настоящее время MD4, MD5, и SHA-1 поддерживаются.[\[6\]](#)

`QChar` обеспечивает поддержку 16-битных символов Unicode.

4.3.1 Автоматизация поиска журнальных файлов

Для сканирования образа на наличие интересующих лог файлов использовался класс `QDirIterator`. После вызова происходит поочередный обход по каждому файлу в директории и поддиректории. Проверка полученного полного пути

к файлу осуществляется регулярным выражением, если условие выполняется, происходит добавление в список обрабатываемых файлов.

4.3.2 Реализация сохранения результатов работы программного комплекса в XML

Сохранение полученных данных происходит в ранее выбранный формат XML(Extensible Markup Language). Для этого используется класс QXmlStreamReader и QXmlStreamWriter. Класс QXmlStreamWriter представляет XML писателя с простым потоковым API.

QXmlStreamWriter работает в связке с QXmlStreamReader для записи XML. Как и связанный класс, он работает с QIODevice, определённым с помощью setDevice().

Сохранение данных реализованно в классе WriteMessage. В методе WriteMessages, структура которого представлена на UML диаграмме в разделе Архитектура.

5 Технические характеристики

5.1 Требования к аппаратному обеспечению

Минимальные системные требования:

- процессор 1ГГц Pentium 4;
- оперативная память 512 Мб;
- место на жёстком диске – 9 Гб.

5.2 Требования к программному обеспечению

Для корректной работы разрабатываемого программного комплекса на компьютере должна быть установлена операционная система Debian Squeeze или выше, данная система должна иметь набор библиотек QT.

5.3 Выбор единого формата выходных файлов

Для вывода результата был выбран формат XML-документов, так как с данным форматом легко работать при помощи программ, а результат работы данного комплекса в дальнейшем планируется обрабатывать при помощи программ.

XML - eXtensible Markup Language или расширяемый язык разметки. Язык XML представляет собой простой и гибкий текстовый формат, подходящий в качестве основы для создания новых языков разметки, которые могут использоваться в публикации документов и обмене данными [7]. Задумка языка в том, что он позволяет дополнять данные метаданными, которые разделяют документ на объекты с атрибутами. Это позволяет упростить программную обработку документов, так как структурирует информацию.

Простейший XML-документ может выглядеть так:

```
<?xml version="1.0"?>
<list_of_items>
<item id="1"><first/>Первый</item>
<item id="2">Второй <subsub_item>подпункт 1</subsub_item></item>
<item id="3">Третий</item>
<item id="4"><last/>Последний</item>
</list_of_items>
```

Первая строка - это объявление начала XML-документа, дальше идут элементы документа `<list_of_items>` - тег описывающий начало элемента `list_of_items`, `</list_of_items>` - тег конца элемента. Между этими тегами заключается описание элемента, которое может содержать текстовую информацию или другие элементы (как в нашем примере). Внутри тега начала элемента так же могут указывать атрибуты элемента, как например атрибут `id` элемента `item`, атрибуту должно быть присвоено определенное значение.

6 Разработка программного обеспечения

6.1 Архитектура

6.1.1 Основной алгоритм

В ходе разработки был применен видоизменённый шаблон проектирования Factory method.

Данный шаблон относится к классу порождающих шаблонов. Шаблоны данного класса - это шаблоны проектирования, которые абстрагируют процесс инстанцирования (создания экземпляра класса). Они позволяют сделать систему независимой от способа создания, композиции и представления объектов. Шаблон, порождающий классы, использует наследование, чтобы изменять инстанцируемый класс, а шаблон, порождающий объекты, делегирует инстанцирование другому объекту. Пример организации проекта при использовании шаблона проектирования Factory method представлен на рисунке 6.1.

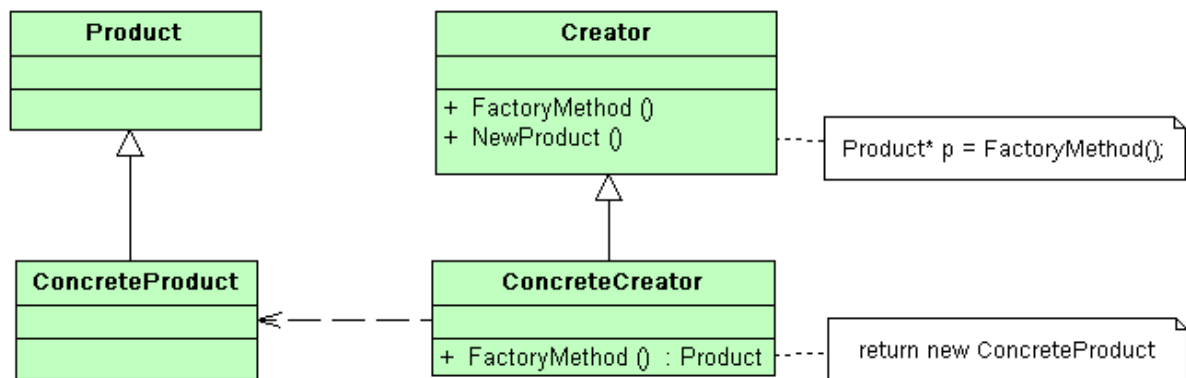


Рисунок 6.1 – Пример организации проекта при использовании шаблона проектирования Factory method

Использование данного шаблона позволило нам разбить наш проект на независимые модули, что весьма упростило задачу разработки, так как написание алгоритма для конкретного таска не влияло на остальную часть проекта. При разработке был реализован базовый класс для работы с образом диска. Данный класс предназначался для формирования списка настроек, определения операци-

онной системы на смонтированном образе и инстанционировании и накапливание всех необходимых классов-задач в очереди задач. После чего каждый task из очереди отправлялся на выполнение. Блоксхема работы алгоритма представлена на рисунке 6.2.

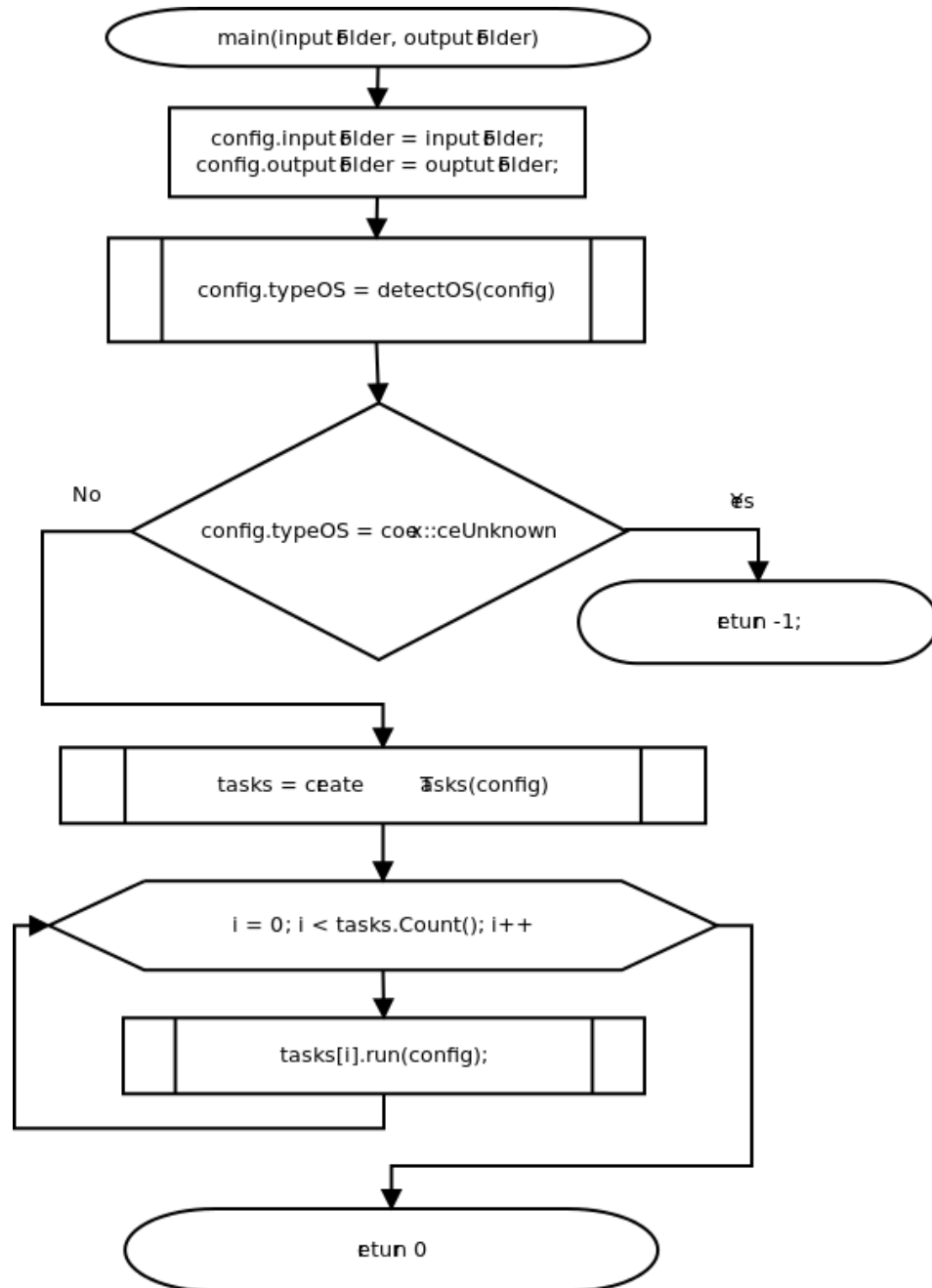


Рисунок 6.2 – Алгоритм работы с образом диска

Каждый класс-задача порождался путем наследования от базового абстрактного класса который имеет 8 методов и 3 атрибута:

- 1) QString manual() - возвращает справку о входных параметрах данного taskа;

- 2) `void setOption(QStringList list)` - установка флагов для поданных на вход параметров;
- 3) `QString command()` - возвращает команду для инициализации таска вручную;
- 4) `bool supportOS(const coex::typeOS &os)` - возвращает флаг, указывающий на возможность использования данного таска для конкретной операционной системы;
- 5) `QString name()` - возвращает имя данного таска;
- 6) `QString description()` - возвращает краткое описание таска;
- 7) `bool test()` - предназначена для теста на доступность таска;
- 8) `bool execute(const coex::config &config)` - запуск таска на выполнение;
- 9) `QString m_strName` - хранит имя таска;
- 10) `QString m_strDescription` - хранит описание таска;
- 11) `bool m_bDebug` - флаг для параметра `-debug`;

На данный момент в проекте используется восемь классов. UML-диаграмма классов представлена на рисунке 6.3.

Классы `taskSearchSyslogsWin`, `taskSearchPidginWin` и `taskSearchSkypeWin` - наследники от класса `task` являются тасками. Класс `winEventLog` и `_EVENTLOGRECORD` предназначены для конвертации журнальных файлов операционной системы Windows XP, а класс `writerMessages` для преобразования истории переписки.

6.1.2 Описание основных функций модуля системы

Любой модуль системы является классом-наследником от некоторого абстрактного класса используемого как основу для всех модулей программы (шаблон проектирования *Factory method*). Модуль содержит в себе 8 методов и 3 атрибута:

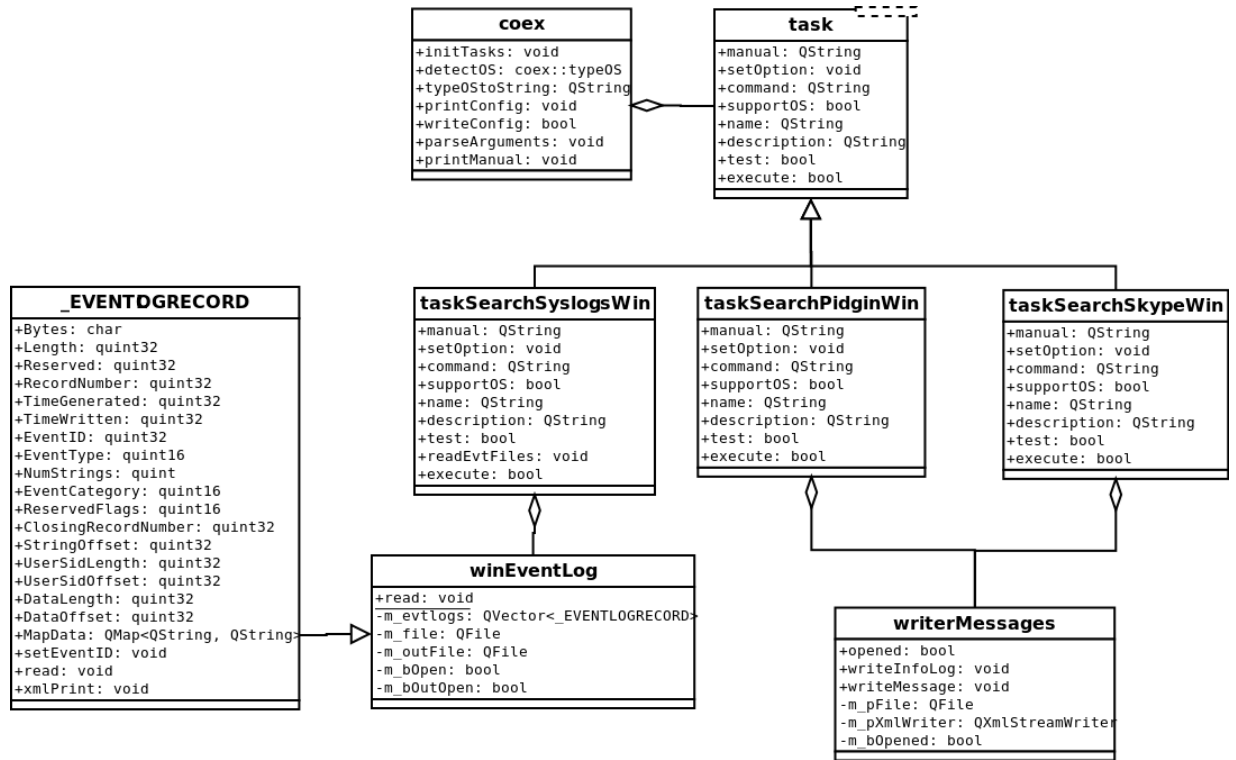


Рисунок 6.3 – UML-диаграмма классов

QString manual() - возвращает справку о входных параметрах данного таска
 void setOption(QStringList list) - установка флагов для поданных на вход параметров

QString command() - возвращает команду для инициализации таска вручную

bool supportOS(const coex::typeOS &os) - возвращает флаг указывающий на возможность использования данного таска для конкретной операционной системы

QString name() - возвращает имя данного таска

QString description() - возвращает краткое описание таска

bool test() - предназначена для проверки работоспособности таска

bool execute(const coex::config &config) - запуск таска на выполнение

QString m_strName - хранит имя таска

QString m_strDescription - хранит описание таска

bool m_bDebug - флаг для параметра -debug

6.2 Сбор и анализ системных журналов Windows (III семестр)

Анализ журналов операционных систем может помочь при решении многих задач. Примером таких задач может быть попытка восстановления системы после поломок, поиск причин неполадок системы, просмотр журналов с целью выявления активности приложений в определенные периоды и т.д. и т.п. Также, анализ журнальных файлов является неотъемлемой частью компьютерно-технических экспертиз.

При исследовании компьютеров задача анализа журнальных файлов ставится на этапе «выявление и изучение его роли в рассматриваемом деле», так как именно в журнальных файлах операционной системы хранится информация о действиях производимых на данном компьютере.

Для проведения компьютерных экспертиз существует множество специализированных программных средств, как платных так и свободно распространяемых. Примером такого средства является «OSForensics». Данный инструмент позволяет проводить множество различных исследований таких как просмотр содержимого оперативной памяти, поиск подозрительных файлов на жестком диске, составления списка программ установленных на исследуемом компьютере. Но бесплатная версия не предоставляет никаких средств по работе с журналами операционной системы. Данный факт подтолкнул к созданию собственного программного средства для проведения компьютерных экспертиз.

В данной работе описывается работа по созданию модуля для данного программного средства, позволяющего в автоматическом режиме находить, читать и конвертировать журнальные файлы операционной системы Windows XP в XML-документы, понятные для человека и легко обрабатываемые при помощи электронно-вычислительных средств.

6.2.1 Общие сведения о журнальных файлах

В операционной системе Windows XP по умолчанию есть четыре журнала:

- 1) Журнал приложений

- 2) Журнал безопасности
- 3) Журнал установки
- 4) Журнал системы

Каждый из этих журналов содержит определенный тип информации. Журналы приложений содержат информацию о запуске и остановке процессов приложений, изменении статуса каждого приложения, а также предупреждения и ошибки связанные с приложениями.

Журнал безопасности содержит информацию о входах в систему, и события связанные с безопасностью системы, например превышение количества попыток неправильно введенных подряд паролей.

Журнал установки содержит информацию об установке и обновлении компонентов системы.

Журнал системы хранит информацию о системных событиях. Например изменение схемы энергопотребления или различного рода предупреждения и ошибок.

Каждый журнал хранится в соответствующем файле с расширением .evt. Эти файлы хранятся в папке %windows%/system32/config. Эти файлы имеют специальную структуру.

6.2.2 Структура журнальных файлов операционной системы Windows XP

Все журнальные файлы операционной системы Windows XP имеют единую структуру. Файл представляет собой последовательность записей бинарных данных. Каждая запись – это структура имеющая семнадцать полей [8].

Первые 4 байта содержат длину события в байтах, после длинны идет 4 байтный системный код сообщения. Затем 4 байтовый номер записи, после него дата создания записи, время создания, идентификатор события, тип события и так далее. Ниже приведен список полей записи, предназначенной для считывания одного события из журнального файла.

– uint32 Length;

- uint32 Reserved;
- uint32 RecordNumber;
- uint32 TimeGenerated;
- uint32 TimeWritten;
- uint32 EventID;
- uint16 EventType;
- uint16 NumStrings;
- uint16 EventCategory;
- uint16 ReservedFlags;
- uint32 ClosingRecordNumber;
- uint32 StringOffset;
- uint32 UserSidLength;
- uint32 UserSidOffset;
- uint32 DataLength;
- uint32 DataOffset;
- byte[] Data.

Из сторонних источников [8] стало известно о пяти типах событий (поле EventType) (значение - тип события):

- 1) 0x0001 - Error event;
- 2) 0x0010 – Failure Audit event;
- 3) 0x0008 - Success Audit event;
- 4) 0x0004 - Information event;
- 5) 0x0002 - Warning event.

У поля EventID удалось определить четыре значения:

- 1) 0x00 – Success;
- 2) 0x01 – Informational;
- 3) 0x02 – Warning;
- 4) 0x03 – Error.

Среди множества полей записи события были выделены поля содержащие информацию о типе события, времени возникновения события и создания записи, пользователя от имени которого была сделана запись, а также поле Data - поле с бинарными данными, в которых записана подробная информация о событии.

6.2.3 Алгоритм работы модуля

Модуль выполняет две задачи: поиск журнальных файлов на образе диска и конвертацию каждого файла в XML-документ. Первая задача выполняется при помощи библиотек QDir и QDirIterator из Qt Framework.

QDir — библиотека позволяющая работать с конкретной директорией. Создав объект данного типа с указанием директории мы получим доступ к этой директории в программе и сможем работать в ней (просматривать содержимое; удалять, создавать или копировать файлы; создавать поддиректории). Данный объект так же поддерживать разные наборы фильтров выходных данных которые могут отсеивать ненужную информацию.

QDirIterator — библиотека, предназначенная для работы с файловой системой начиная с определенной директории как точки входа. Создав объект данного типа с указанием директории мы получим все пути которые существуют в файловой системе и начинаются с указанной директории. Данный объект поддерживает фильтрацию которая помогает выделять только необходимую информацию, исключая то, что нас не интересует, например можно вывести список только тех файлов, которые находятся в данной директории или поддиректориях, или исключить вывод символьных ссылок. Объекты данного типа используются для поиска файлов или папок на образе исследуемого диска.

Так же данные библиотеки позволяют создавать объект QFile, который позволяет работать с файлом, путь к которому передается как параметр при создании, данный объект позволяет получить базовую информацию о файле, такую как относительный или абсолютный путь до этого файла, размер файла, тип файла или его имя. Так же позволяет перемещать или копировать данный файл. Поиск работает по алгоритму представленному на рисунке 6.4.

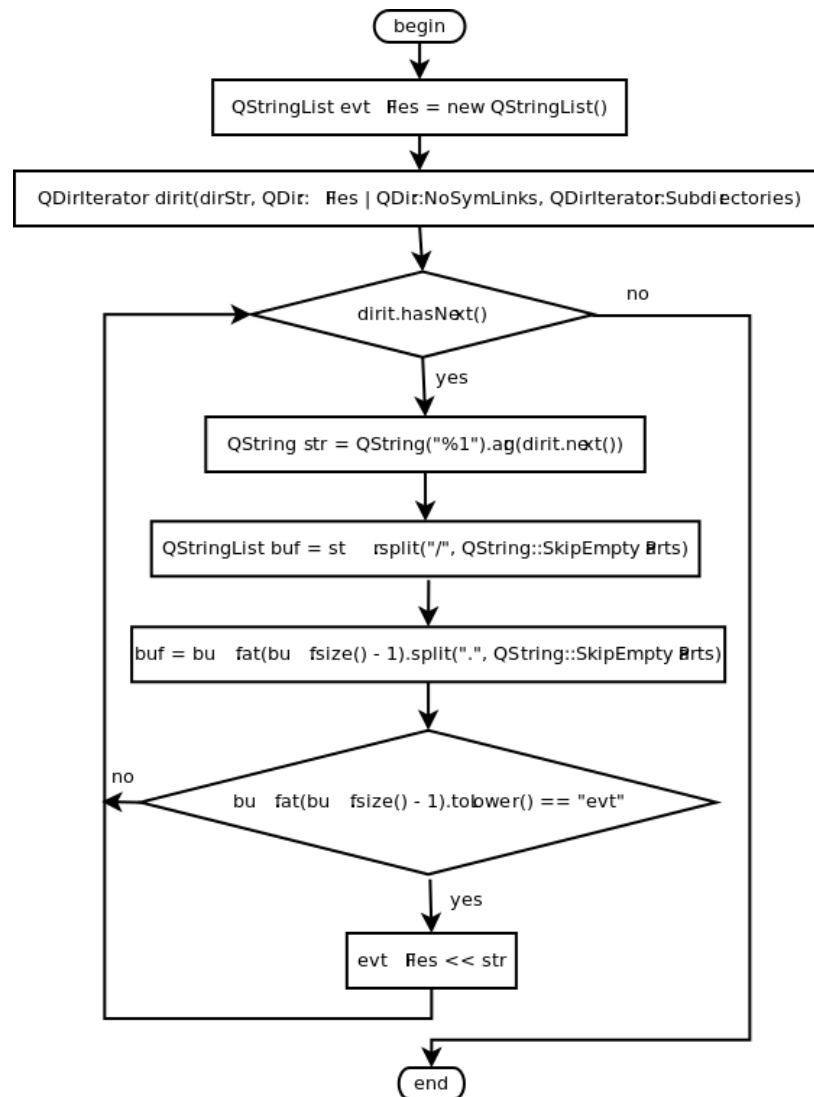


Рисунок 6.4 – Алгоритм поиска файлов с расширением *.evt

На выходе данного алгоритма мы получаем объект QStringList который содержит пути до всех найденных .evt файлов. Каждый экземпляр коллекции строк данного объекта передается в конструктор объекта winEventLog, который и конвертирует указанный файл в XML-документ. Алгоритм работы конвертора представлен на рисунке 6.5.

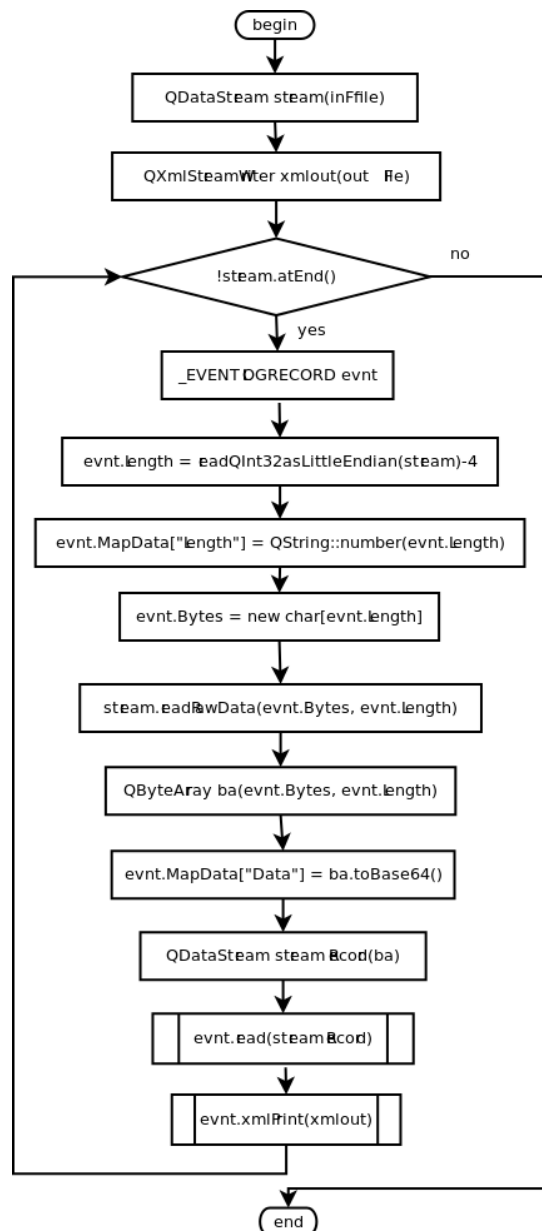


Рисунок 6.5 – Алгоритм конвертирования файлов *.evt в формат XML

Пример результата работы модуля в виде файла в формате XML представлен на рисунке 6.6

6.3 Сбор и анализ системных журналов Windows (IV семестр)

6.3.1 Введение

Продолжая разработку данного программного продукта стало ясно что, несмотря на выбранный шаблон проектирования, добавление нового функционала осложняется тем, что при добавлении новой функции приходится пересобирать весь проект. Это неправильный подход, так как не позволяет обновлять

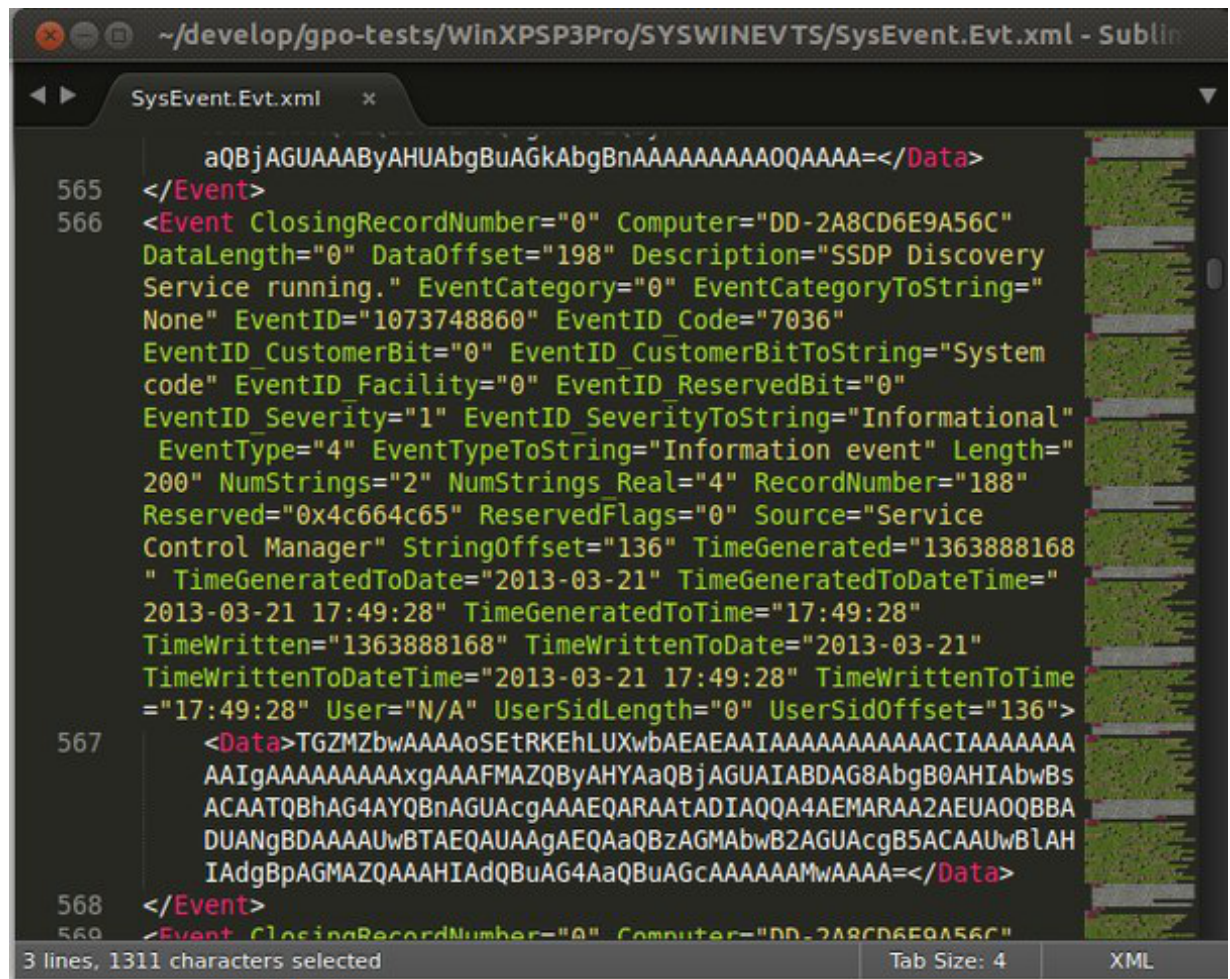


Рисунок 6.6 – Пример лога системы в формате XML

приложение. Для исправления этой ситуации необходимо доработать архитектуру приложения таким образом, что бы добавление новых функций не приводило к пересборке всего проекта. Это так же позволит проще обновлять проект в дальнейшем.

После доработки архитектуры, необходимо будет переделать проект, переделать реализацию всего функционала, а так же свести все ветки разработки в одну, так как на данный момент не все изменения применены к проекту.

Сформулируем задачи на семестр:

6.3.2 Задачи на семестр

- 1) переработать архитектуру проекта для более легкого добавления новых функций
- 2) реализовать данную архитектуру

- 3) переделать готовые task'и для работы с текущей архитектурой
- 4) смерджить ветки разработки git

6.3.3 Новая архитектура

С увеличением количества тасков возникла проблема с обновлением кода, так как используемый подход предполагал монолитную программу и все таски хоть и были стандартизированы при помощи шаблона разработки factory, всё равно являлись частью основной программы и для добавления тасков или их обновления необходимо постоянно перекомпилировать весь проект. Данная ситуация крайне неудобна при разработке и тестировании, поэтому необходимо было разделить проект на независимые модули.

В языке программирования C++ есть возможность создания библиотек. Библиотеки - это бинарные файлы в которых хранятся коллекции классов и функций. Эти классы и функции можно использовать в любой программе в которой данная библиотека подключается. Подключать библиотеки в программу можно статически(на этапе компиляции программы) или динамически(непосредственно при выполнении программы). У нашего проекта есть "келет" таски, каждый из которых работает независимо друг от друга и обменивается данными только со скелетом.

Так как каждая библиотека является самостоятельным файлом с исполняемым кодом, то для его создания и изменения необходим отдельный проект разработки, а значит если мы захотим изменить что-либо внутри библиотеки, то практически наверняка нам не нужно будет перекомпилировать весь проект. Это значительно упрощает операцию обновления программного комплекса. Было принято решение вынести реализацию всех функций скелета в отдельную статическую библиотеку. А все таски оформить в виде динамических библиотек.

Данное решение повлекло за собой ряд проблем:

- 1) Необходимо разработать структуру динамических библиотек для тасков программного комплекса;

- 2) Необходимо изменить механизм работы с задачами;
- 3) Необходимо изменить механизм разбора и передачи параметров в основной части программного комплекса;
- 4) Разработать механизм передачи информации в механизм разбора и передачи параметров.

6.3.4 библиотеки в C++ и их использование

Библиотеки предоставляют программисту возможность использовать при разработке программы готовые фрагменты кода. В библиотеки могут быть включены подпрограммы, структуры данных, классы, макросы. В QT для создания библиотеки необходимо указать в про-файле проекта добавить строку

```
TEMPLATE = lib
```

Данная строка указывает компилятору на то, что мы компилируем библиотеку. После чего в файле исходного кода описывается всё необходимое и после компиляции мы получим файл библиотеки. Для создания динамической библиотеки необходимо добавить в про-файл ещё строку:

```
CONFIG += dll
```

А так же в заголовочном файле библиотеки поместить описание прототипов функций которые мы хотим подключать динамически в конструкцию

```
extern C
```

Это необходимо для того, что бы при сборке библиотеки функции не поменяли имена. Если не добавить данную конструкцию, то компилятор при сборке изменит имена функций и обратиться к ним динамически не получится.

Для использования разработанной статической библиотеки необходимо просто указать в файле исходного кода строку

```
include absolute/path/to/library/file"
```

Либо поместить созданную библиотеку в папку с системными библиотеками (или добавить в настройки операционной системы путь до созданной библиотеки), тогда можно подключать библиотеку так же как и стандартные библиотеки (имя

библиотеки в треугольных скобках)

Динамические библиотеки используются по-другому. Загрузка функции из динамической библиотеки происходит в три этапа:

- 1) Загрузка библиотеки;
- 2) Описание шаблона для вызова функции:
- 3) Вызов функции.

В качестве примера на рисунке 6.7 представлен пример вызова функции `foo`, которая принимает на вход один аргумент типа `int` и возвращает его удвоенное значение. функция находится в библиотеке `exampleLib`. Загрузка библиотеки производится при помощи средств, предоставляемых Qt framework:

```
#include <QLibrary>
#include <iostream>

int main()
{
    //данная строка загружает динамическую библиотеку, которая лежит
    //по указанному пути. ссылка на библиотеку хранится в переменной lib
    QLibrary lib("/home/user/mylibs/exampleLib");
    //описание переменной. на экран выведется "4"
    int a = 4;
    std::cout << a << std::endl;
    //описываем тип данных для функции, которую хотим загрузить из библиотеки
    //первым делом указываем тип возвращаемого значения, после чего имя
    //указателя на данный тип и в скобках перечисляем типы данных аргументов
    typedef int (*Fct)(int);
    //загружаем функцию с именем foo из библиотеки
    Fct fct = (Fct)(lib.resolve("foo"));
    //использование загруженной функции
    a = fct(a);
    //в консоли выведется 8
    std::cout << a << std::endl;
    return 0;
}
```

Рисунок 6.7 – Пример вызова функции `foo`

6.3.5 task = динамическая библиотека

Для того, что бы программа могла работать с каждым таском, как с динамической библиотекой, необходимо что бы все библиотеки были созданы по

одному шаблону. То есть в каждой библиотеке должен быть определенный набор функций с определенным именем и входными параметрами к которым можно обращаться при загрузке библиотеки.

В каждой библиотеке необходимо определить две функции:

- 1) функция, возвращающая информацию о таске (имя таска и возможный набор параметров, предназначенных для данного таска)
- 2) функция выполнения данного таска, которая возвращает логическое значение по окончании работы, на вход она принимает переменную со всеми настройками, введенными пользователем

На данный момент механизм разбора флагов не был реализован. Структура библиотеки представлена на рисунке 6.8.

```

1  #include <coex>
2  #include <QString>
3
4
5  //описание прототипов функций, которые будут вызывать при загрузке библиотеки
6  extern "C"
7  {
8      //получение информации о библиотеке
9      QString getLibInfo();
10     //выполнение задачи, решаемой данной библиотекой
11     bool execute(const coex::config &);
12 }
13
14 //блок со всеми ресурсами, необходимыми библиотеке для решения задачи
15 //все что описано в этом блоке доступно только внутри библиотеки
16 void supportFun1();
17 int supportFun1(int, int);
18
19 class supportClass
20 {
21 public:
22     supportClass();
23     ~supportClass();|
24
25     //class description
26 }

```

Рисунок 6.8 – Структура библиотеки

Все реализованные таски были оформлены в виде динамических библиотек. Алгоритмы обработки данных изменению не подвергались.

6.3.6 новый механизм работы с задачами

Для того, что бы было возможно добавлять в наш комплекс новые задачи без перекомпиляции скелета комплекса мы оформили их в виде динамических библиотек, каждая из которых реализована по шаблону, рассмотренному выше. Все библиотеки, которые мы хотим загружать при работе программы мы помещаем в специальную директорию `libs` рядом с исполняемым файлом. При запуске программа просматривает данную директорию и поочередно загружает каждую библиотеку и запускает её на выполнение (если это возможно).

Алгоритм работы данного механизма представлен на рисунке 6.9

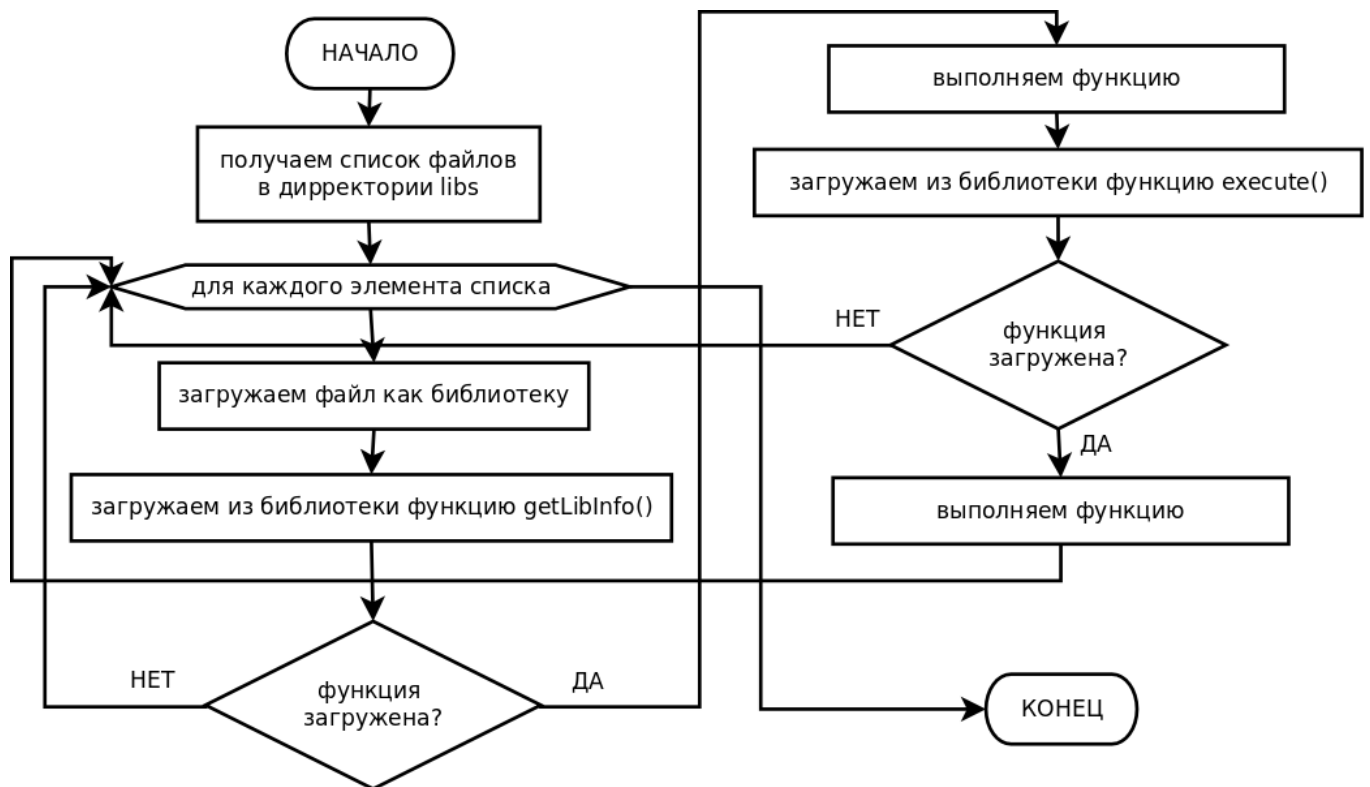


Рисунок 6.9 – Пример вызова функции foo

6.3.7 мерджим ветки git

Разработка комплекса велась двумя разработчиками, каждый из которых работал в своей ветке разработки системы контроля версий git. Необходимо было собрать все изменения проекта воедино. Обычно для этого используются инструменты, предоставляемые самим gitом, но так как в проекте была изменена архи-

тектура (по сути был создан новый проект, который делали их частей старого), то было принято решение просто получить все изменения с каждой ветки для каждого проекта и оформить последние версии тасков в виде динамических библиотек. Данный подход позволил избежать процедуры разрешения конфликтов. На данный момент новая версия проекта находится в ветке master и содержит все последние изменения предыдущего проекта. Разработка первой версии комплекса остановлена и, после того как весь код старого проекта будет использован, данная версия комплекса будет удалена.

6.4 Сбор и анализ истории переписки мессенджеров (III семестр)

Опишем задачу более подробно, разделив её на подзадачи:

- 1) определить места хранения файлов с личной перепиской пользователя;
- 2) определить формат файлов переписки;
- 3) разбор найденных файлов;
- 4) автоматизировать процесс поиска файлов;
- 5) производить сохранение полученных информации формат XML.

6.4.1 Определение местоположения логов переписки мессенджера Skype

По умолчанию файлы располагаются в директориях, для разных операционных систем возможны незначительные изменения (таблица 6.1).

Таблица 6.1 – Директории хранения логов Skype

Windows Vista/7/8	C:\Users\username\ AppData\Roaming\Skype
Windows XP	C:\Documents and Settings\username\ Application Data\Skype
Linux(Ubuntu)	/home/username/.Skype

Основная интересующая нас информация находится в main.db [9]. Алгоритм поиска логов представлен на рисунке 6.10.

6.4.2 Определение местоположения логов переписки мессенджера Pidgin

Определение месторасположения файлов переписки происходит следующим образом: для примонтированного образа запускается модуль, который сужает область поиска, сканируя только нужные директории образа.

По умолчанию файлы располагаются в директориях, для разных операционных систем возможны незначительные изменения (таблица 6.2).

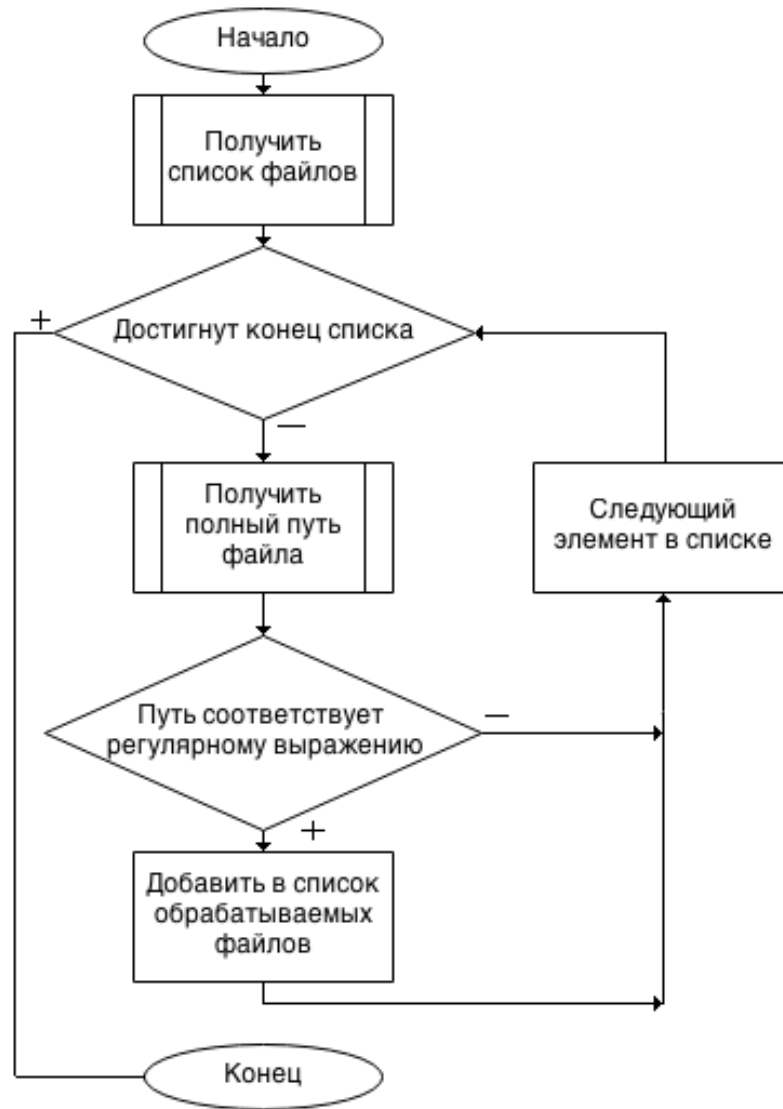


Рисунок 6.10 – Блок-схема алгоритма поиска логов

Таблица 6.2 – Директории хранения логов pidgin

Windows Vista/7/8	C:\Users\username\AppData\Roaming\.purple\logs
Windows XP	C:\Documents and Settings\username\Application Data\.purple
Windows 98/ME	C:\Windows\Profiles\username\.purple
Linux(Ubuntu)	/home/username/.purple/logs

Основная интересующая нас информация хранится в файлах с маской имени YEAR-MONTH-DATE.TIME.html(например 2013-03-02.004915+0700NOVT.htm).

6.4.3 Формат хранения переписки мессенджера Skype

Приложение «skype» хранит переписку на локальных машинах пользователей или же возможна синхронизация с машин других пользователей [9]. Формат хранения: реляционная база данных основная на СУБД SQLite. Но база данных «main.db» не является единственным местом хранения информации, Skype сохраняет сведения о работе программы во временных файлах (chatsync). Эти файлы имеют расширение «.dat» и цифробуквенные имена (например «0172b0a519e2c584»)[10].

6.4.4 Формат хранения переписки мессенджера Pidgin

Приложение «pidgin» поддерживает перечисленные ниже протоколы: [11]

- "AIM"
- "Bonjour"
- "Gadu-Gadu"
- "Google Talk"
- "Groupwise"
- "ICQ"
- "IRC"
- "MSN"
- "Mxit"
- "MySpaceIM"
- "SILC"
- "SIMPLE"
- "Sametime"
- "XMPP"

- "Yahoo!"
- "Zephyr"

Соответственно, все подключенные аккаунты всех вышеперечисленных протоколов хранятся как лог файлы на локальной машине пользователя в формате HTML и TXT. По умолчанию лог файлы хранятся в .HTML файле. Настройки программы, пользователя и подключенных аккаунтов хранятся в XML.

Примечание: Кроме account.xml - он хранит нешифрованные пароли для всех подключенных чатов [11].

6.4.5 Алгоритм работы модуля Skype

Структура рассматриваемого файла.

main.db содержит 18 таблиц. и ещё немного статистики.

- "DbMeta"
- "Contacts"
- "LegacyMessages"
- "Calls"
- "Accounts"
- "Transfers"
- "Voicemails"
- "Chats"
- "Messages"
- "ContactGroups"
- "Videos"
- "SMSes"
- "CallMembers"
- "ChatMembers"

- "Alerts"
- "Conversations"
- "Participants"
- "VideoMessages"

Таблицы которые были рассмотрены, на данный момент:

- 1) Contacts
- 2) Messages
- 3) Chats
- 4) Calls
- 5) CallMembers
- 6) Conversations

В таблице Contacts находятся все контакты, причем даже те, что были удалены, и уже не показываются в клиенте.

```
select skypeusername, fullname, languages, country, city from contacts
```

В таблицах Calls и CallMembers содержатся, соответственно, история звонков и их участников.

```
select calls.id as "ID разговора", coalesce(contacts.displayname,
accounts.fullname) as "Инициатор", strftime('%d.%m.%Y
%H:%M:%S',calls.begin_timestamp, 'unixepoch', 'localtime') as
"Дата начала", time(calls.duration, 'unixepoch') as "Длительность",
callmembers.dispname as "Подключенный участник", strftime('%d.%m.%Y
%H:%M:%S',callmembers.start_timestamp, 'unixepoch', 'localtime') as
"Дата подключения", time(callmembers.call_duration, 'unixepoch')
as "Длительность подключения" from calls inner join callmembers
on calls.id = callmembers.call_db_id left join contacts on
calls.host_identity = contacts.skypeusername left join accounts on
calls.host_identity = accounts.skypeusername
```

И, наконец, в таблицах Conversations и Messages содержатся данные переписки и сами сообщения.

```
select conversations.id as "ID переписки", conversations.displayname
as "Участники переписки", messages.from_dispname as "Автор
сообщения", strftime('%d.%m.%Y %H:%M:%S',messages.timestamp,
'unixepoch', 'localtime') as "Время сообщения", messages.body_xml
as "Текст сообщения" from conversations inner join messages on
conversations.id = messages.convo_id order by messages.timestamp
```

Для доступа ко всему содержимому базы достаточно иметь доступ к самому файлу — содержимое базы никак не шифруется и не защищается, так что любой человек, который сможет получить доступ к вашему профилю Windows, сможет найти список контактов, просмотреть историю звонков и прочитать всю переписку.

6.4.6 Алгоритм работы модуля Pidgin

Структура рассматриваемого файла.

У каждого лог файла есть заголовок находящийся между тегов title. В котором записан ID_Chat, дата начала переписки, логин пользователя и используемый протокол.

Затем идет "тело" в котором описывается обмен сообщениями в формате, время, автор сообщения и сообщение.

Пример "Заголовка" `<head><meta http-equiv = "content-type" content= "text/html; charset=UTF-8"> <title> Conversation with 0dpkhcz6clufs2kozj82uqif30@public.talk.google.com at Чт. 24 окт. 2013 23:40:21 on user.fox@gmail.com/ (jabber)</title></head>`

Пример "Тела" ` (23:44:52) user.fox@gmail.com/95C9F047: Hello)
`

Точно так же из полученного списка найденные файлы поочередно открываются для чтения. Разбор открытого файла решено осуществлять при помощи регулярных выражений, описанных в класс QRegExp.

Парсинг логов pidgin представлен на рисунке 6.11.

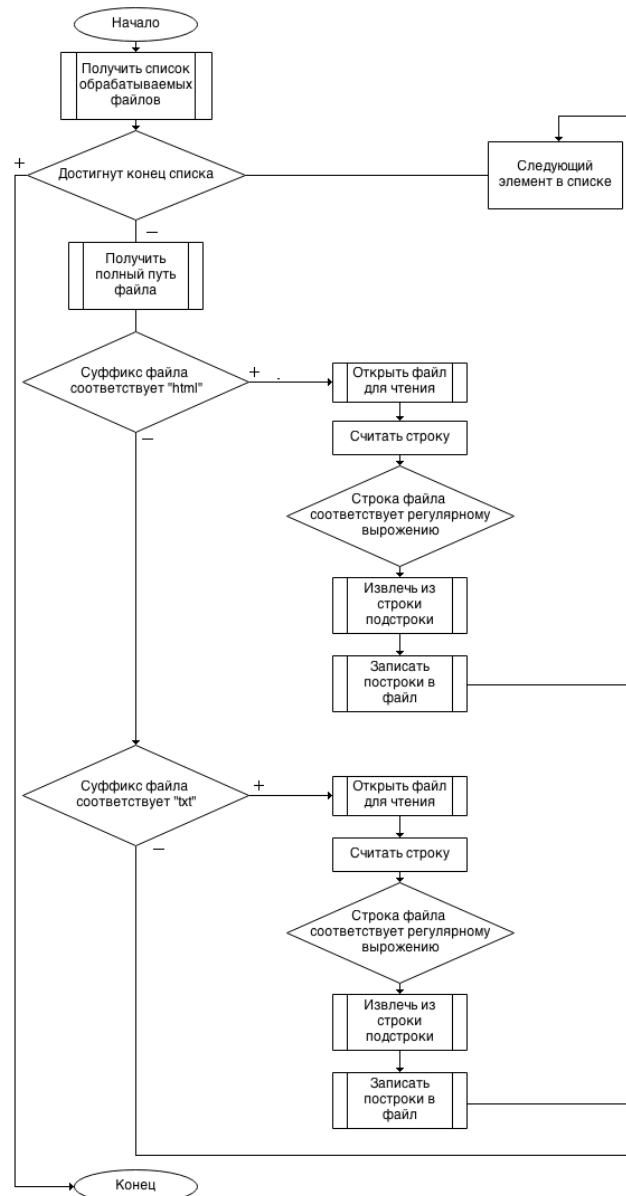


Рисунок 6.11 – Блок-схема алгоритма парсинга логов

6.4.7 Структура логов Skype, сохранённых в XML

Все документы XML начинаются с пролога (prolog). Пролог сообщает, что документ написан на XML, а также указывает, какая версия XML при этом использовалась.

Следующий элемент Messages с атрибутом Messenger, ему присваивается имя рассматриваемого приложения.

Для каждого файла БД, находящейся в обрабатываемом списке, создается элемент info account содержащий атрибуты skypeName, fullName, emails, ipcountry

которым присваивается, соответственно: логин пользователя Skype, его полное имя, email (не шифрованный), геолокация ip-адреса.

Далее следует элемент `contact` содержащий атрибуты `skypeName`, `fullName`, `languages`, `country`, `city`.

Пример выходного файла в формате XML представлен на рисунке 6.12.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <Messages Messenger="pidgin">
3   <Info chathID="" account="" data="" protocol=""/>
4   <message author="dog.3" dateTime="15:26:07" message="Сдаем чертежи на проверку. Принимает злобный
5     чел. Доходит очередь Дениса. Подходит к проверяющему, начинает открывать тубус с унылым лицом."/>
6   <message author="dog.3" dateTime="15:28:59" message="Проверяющий - неправильный чертеж! Деня в
7     шоке, только тубус открыл, даже достать не успел."/>
8   <message author="dog.3" dateTime="15:47:36" message="Деня стоял стоял, смотрит ластик на столе лежит
9     . Берет его, бросает в тубус, закрывает и трясет со всех сил. Открывает, достает чертёк и говорит &
10    quot;ИСПРАВИЛ!&quot;"/>
11   <message author="dog.3" dateTime="15:47:36" message="Единственный ушел с пятеркой..."/>
12   <Info chathID="" account="" data="" protocol=""/>
13   <message author="dog.user" dateTime="15:49:02" message="Устал сегодня а массаж никто не делает."/>
14   <message author="dog.user" dateTime="15:49:32" message="WTF?"/>
15   <message author="cat.user" dateTime="15:50:04" message="И не надейся!! Завтра не суббота!"/>
16   <message author="dog.user" dateTime="15:50:07" message="ах так!"/>
17   <message author="cat.user" dateTime="15:50:58" message="Даже не притронусь к твоей сплнице!"/>
18   <message author="dog.user" dateTime="15:51:09" message="дэ? чойто?"/>
19   <message author="cat.user" dateTime="15:52:08" message="Мне еще ни разу не делал"/>
20   <message author="cat.user" dateTime="15:53:08" message="А у меня между прочим сидячая работа"/>
21   <message author="cat.user" dateTime="15:53:37" message="давно делал. не ври а"/>
22   <message author="cat.user" dateTime="15:54:18" message="Когда еще динозавры жили?"/>
23   <message author="dog.user" dateTime="15:55:00" message="ты забыла чтоле?"/>
24   <message author="cat.user" dateTime="15:55:58" message="Я всё помню! Это был 2011 год... следующий
25     раз будет в 2016?"/>
26   <message author="dog.user" dateTime="15:57:07" message="воу воу палехче! зачем так частить?"/>
27   <Info chathID="" account="" data="" protocol=""/>
28   <message author="" dateTime="" message=""/>
29   <message author="pinkman" dateTime="17:51:20" message="OLOLO"/>
30   <message author="user.fox" dateTime="17:51:22" message="shlom!"/>
31   <message author="pinkman" dateTime="17:51:37" message=":wait:"/>
32   <message author="user.fox" dateTime="17:51:42" message=":cake:"/>

```

Рисунок 6.12 – логи Skype в формате XML

6.4.8 Структура логов Pidgin, сохранённых в XML

Все документы XML начинаются с пролога (prolog). Пролог сообщает, что документ написан на XML, а также указывает, какая версия XML при этом использовалась.

Следующий элемент `Messages` с атрибутом `Messenger`, которому присваивается имя рассматриваемого приложения.

Далее следует элемент `INFO` содержащий атрибуты `chathID`, `account`, `data`, `protocol` которым присваивается, соответственно: идентификатор чата, полная дата в формате День.Число Месяц. Год Час:Мин:Сек, аккаунт с которого происходил обмен сообщениями и протокол используемый для передачи сообщений.

Последний элемент – MESSAGE, содержащий атрибуты author, dateTime, message.

Пример выходного файла в формате XML представлен на рисунке 6.13.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <Messages Messenger="skype">
3   <contact skypeName="echo123" fullName="Echo / Sound Test Service" languages="en" country="" city=""/>
4   <contact skypeName="xartix57" fullName="ArtemDr..." languages="ru" country="ru" city="Томск"/>
5   <contact skypeName="senya_2007" fullName="" languages="" country="" city=""/>
6   <contact skypeName="kretov1" fullName="Кретов Игорь" languages="" country="" city=""/>
7   <contact skypeName="pahendral" fullName="Павел Нестеренко" languages="ru" country="ru" city="Томск"/>
8   <contact skypeName="turb8_1992" fullName="Артур" languages="ru" country="ru" city="Томск"/>
9   <contact skypeName="willer12312" fullName="Илья Довженко" languages="ru" country="ru" city="Томск"/>
10  <contact skypeName="sveta_749" fullName="" languages="" country="" city=""/>
11  <contact skypeName="sofia_78_08" fullName="София" languages="ru" country="ru" city="Нижевартовск"/>
12  <contact skypeName="kamal3134" fullName="Камиль Сатаев" languages="ru" country="ru" city="
Нижевартовск"/>
13  <contact skypeName="vlad-venom" fullName="Владислав" languages="ru" country="ru" city="Тайра"/>
14  <contact skypeName="erkin_201" fullName="" languages="" country="" city=""/>
15  <contact skypeName="jenny_sunny3" fullName="Женечка" languages="ru" country="ru" city=""/>
16  <contact skypeName="kretovigor1" fullName="Игорь Кретов" languages="ru" country="ru" city="Томск"/>
17  <contact skypeName="denas570" fullName="Баженков Александр" languages="ru" country="ru" city="Томск"/>
18  <contact skypeName="fumafu2" fullName="" languages="" country="" city=""/>
19  <contact skypeName="james_mariarti" fullName="Дмитрий Никифоров" languages="ru" country="ru" city="
Томск"/>
20  <contact skypeName="sain_070" fullName="Sain_070" languages="ru" country="ru" city="Асино - Томск"/>
21  <contact skypeName="aleksey.gulyaev" fullName="aleksey.gulyaev" languages="" country="" city=""/>
22  <contact skypeName="reducer" fullName="Redux" languages="ru" country="ru" city="Томск"/>
23  <contact skypeName="maxsim_404" fullName="Max Antonov" languages="ru" country="ru" city=""/>
24  <contact skypeName="stas56491" fullName="Станислав" languages="ru" country="ru" city="Томск"/>
25  <contact skypeName="victor.grenke" fullName="Victor Grenke" languages="ru" country="ru" city="Томск"/>
26  <contact skypeName="polyakova.svetlana1" fullName="Светлана Полякова" languages="ru" country="ru" city="
Нижевартовск"/>
27  <contact skypeName="sherbakov.viktor" fullName="Victor" languages="ru" country="ru" city="Томск"/>
28  <contact skypeName="igor.shakirov" fullName="Igor Shakirov" languages="ru" country="ru" city="Томск"/>
29  <contact skypeName="tilzaer" fullName="Igor Potapov" languages="" country="ru" city="Томск"/>
30  <contact skypeName="vadireshakirov" fullName="Vadim Shakirov" languages="ru" country="ru" city=""/>

```

Рисунок 6.13 – логи Pidgin в формате XML

На текущий момент полностью реализован импорт контактной книги из приложения Skype, импорт звонков и переписки находится в режиме разработки.

6.5 Сбор и анализ истории переписки мессенджеров (IV семестр)

Задачи на данный семестр:

- 1) преобразование собранной информации в читаемый вид;
- 2) добавление новых функций к уже имеющимся.

Для упрощения разобьем задачу, на подзадачи:

- 1) Создание веток (branch), и распределенная работа программистов;
- 2) Визуализация данных;
- 3) Выбор и описание утилит для преобразования;

- 4) XSTL;
- 5) Показать результат;
- 6) Функция чтения контактной книги pidgin;
- 7) Функция сбора используемых учётных записей pidgin;
- 8) Описание работы SAX и DOM модели разбора XML;
- 9) Функция сбора дополнительных данных skype.

6.5.1 Создание веток (branch), и распределенная работа программистов.

Допустим мы приступаем к задаче. которая займет больше одного дня\...

Но если задача длинная, и мы будем коммитить полусделанную задачу, то это будет как минимум мешать другим разработчикам, как максимум сделает проект неработоспособным.^[2]

Однако не делать коммиты несколько дней тоже плохой вариант. Вот для этого и существуют другие ветки. Они позволяют нам вносить изменения, не мешая и этом остальным разработчикам. Или же работать над несколькими ветками одновременно.

Ветка - это снимок репозитория в сделанный в прошлом, в который не попадают коммиты из основной ветки после момента создания.

В проекте использовались две ветки:

- 1) master
- 2) skype_to_XML

В ветке master"елась работа по переносу проекта с модульной структуры на плагииную. skype_to_XML"елась работа по расширению функционала модулей клиентов - мгновенного обмена сообщениями, и VoIP программ. Велась работа по ознакомлению с языком разметки xslt, с последующей генерацией отчета в xhtml, основанный, на собранных данных, хранящихся в xml.

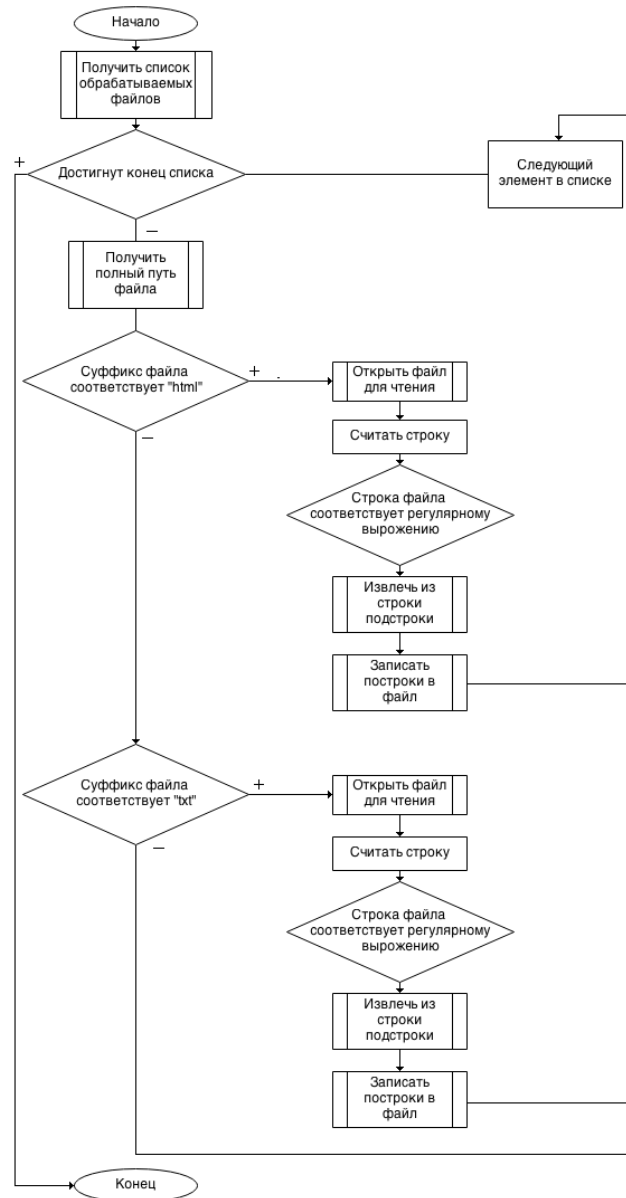


Рисунок 6.14 – Результат работы git branch

6.6 Визуализация данных

Визуализация собранной информации очень важный момент создания подобного рода комплексов, т.к. это является главной задачей нашего ПО.

Список возможных вариантов конвертации XML

- 1) CSV
- 2) HTML
- 3) TXT
- 4) SQL

5) PNG

6) И многое другое...

Из рассматриваемого списка удобным для восприятия, хранения структурированной информации и программно-независимымоказался - HTML.

6.7 Выбор и описание утилит для преобразования

XML можно хранить в виде БД, а затем выводить записи БД в виде веб-страниц, используя к примеру PHP. Возникает вопрос, зачем мы храним данные в промежуточном формате XML, если в результате данные хранятся в БД. Работа в таком формате займет большее время, и сложнее ко всему. Для прямого преобразования XML в XHTML, мы можем воспользоваться языком преобразования XSLT.

6.8 XSTL

XSLT (eXtensible Stylesheet Language Transformations) — это расширяемый язык стилей для преобразований, который использует для описания преобразований структуры документов.[\[12\]](#)

Хотя XSLT не позиционируется, как язык запросов для XML, можно смело сравнить его с языком SQL, в котором определяются запросы к реляционным базам данных.[\[12\]](#)

При применении таблицы стилей XSLT, состоящей из набора шаблонов, к XML-документу образуется конечное дерево, которое может быть сериализовано в виде XML-документа, XHTML-документа (только для XSLT 2.0), HTML-документа или простого текстового файла. Правила выбора (и, отчасти, преобразования) данных из исходного дерева пишутся на языке запросов XPath. XSLT имеет множество различных применений, в основном в области веб-программирования и генерации отчётов. Одной из задач, решаемых языком XSLT, является отделение данных от их представления, как часть общей парадигмы MVC (англ. Model-view-controller). Другой стандартной задачей является

преобразование XML-документов из одной XML-схемы в другую.[12]

6.9 Результаты работы

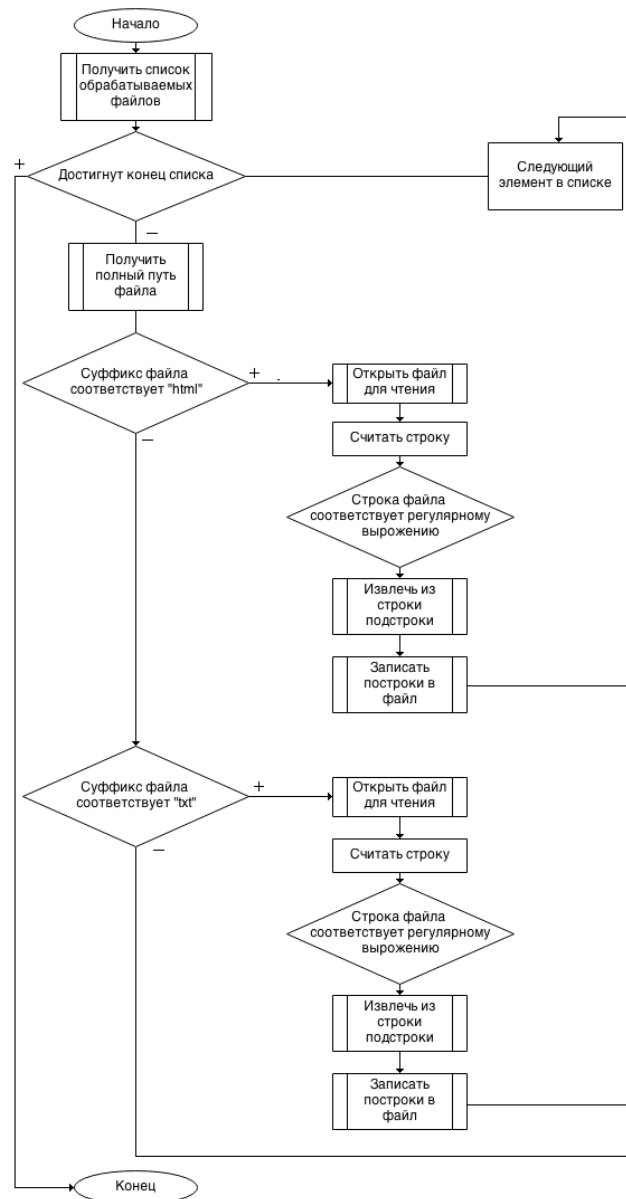


Рисунок 6.15 – Результат заглавной генерированной страницы отчета

6.10 Описание работы SAX и DOM модели разбора XML

SAX расшифровывается как Simple API for XML, что означает буквально “простой прикладной интерфейс программирования для XML

SAX (англ. «Simple API for XML») — способ последовательного чтения/записи XML-файлов.

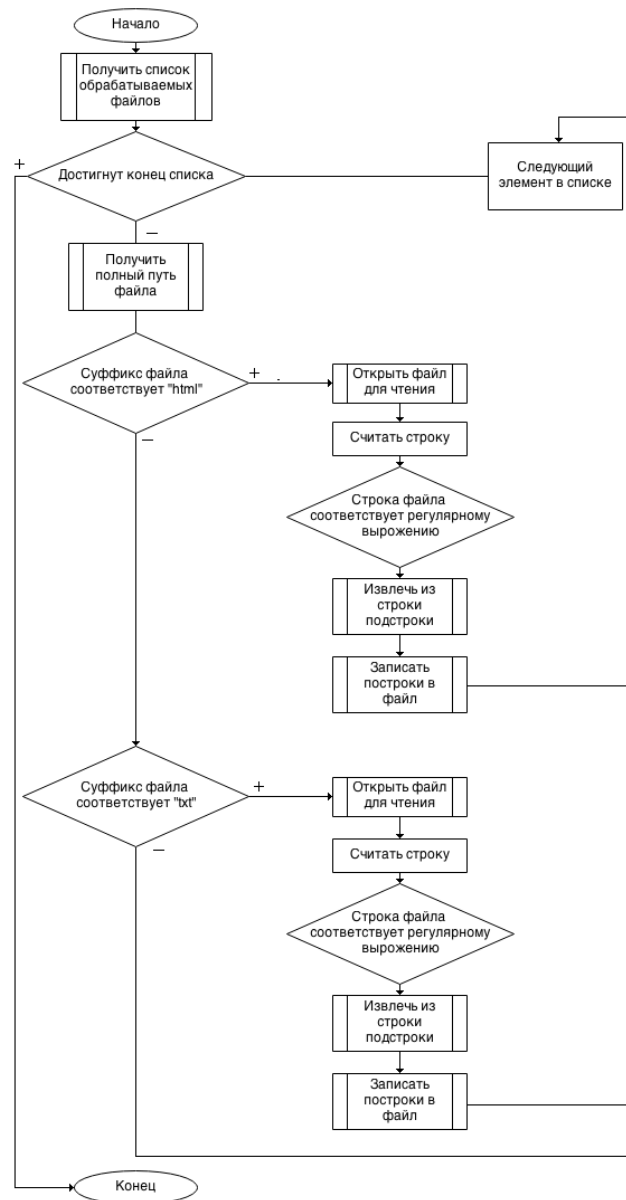


Рисунок 6.16 – Результат сбора данных контактной книги pidgin

Всё, что делает SAX-парсер, это сообщает вызвавшему приложению о встреченных распознанных элементах XML-разметки или о встреченных ошибках. Связь парсера с вызывающим приложением, как правило, осуществляется посредством функций обратного вызова.

DOM (от англ. Document Object Model — «объектная модель документа») — это не зависящий от платформы и языка программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому HTML, XHTML и XML-документов, а также изменять содержимое, структуру и оформление таких документов.^[12]

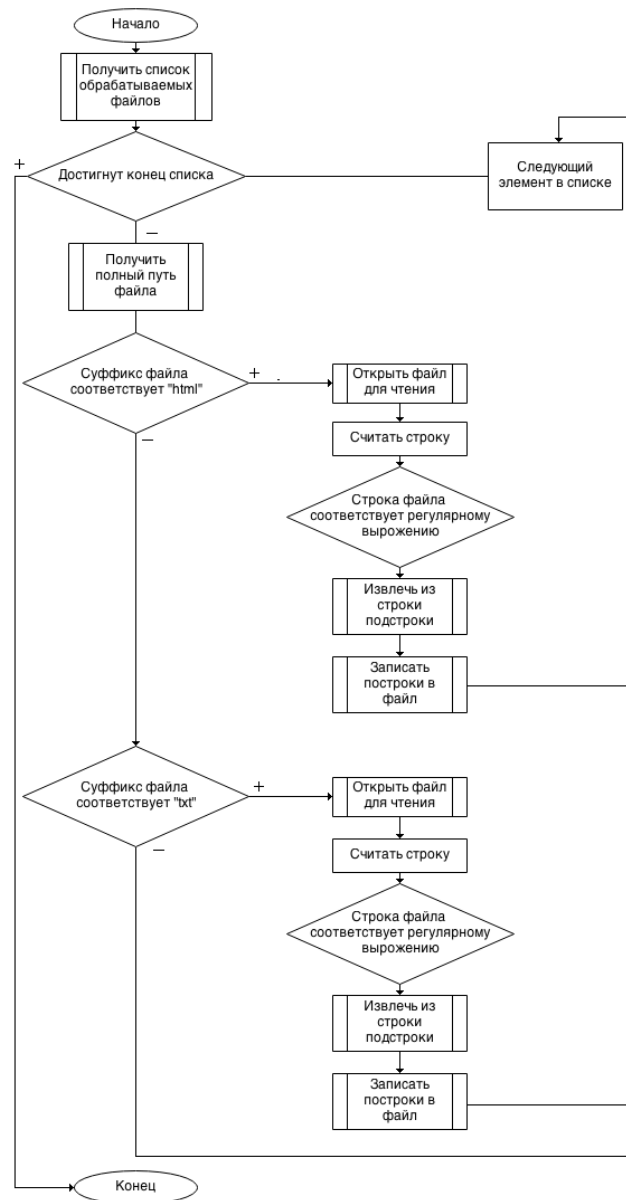


Рисунок 6.17 – Результат сбора данных аккаунта pidgin

6.11 функция чтения используемых аккаунтов и контактной книги pidgin

Интересующие нас контактная книга находятся в в файле blist.xml, используемые аккаунты хранятся в файле accounts.xml.

Оба файла имеют идентичную древовидную структуру, которая начинается в "орне" разветвляется до "истьев"; как и положено в XML документе.

Для разбора данного файла использовалась модель SAX. Реализованная в библиотеке QT, в классе QDom. При проходе по дереву XML мы реагируем на события [5]:

– Старт документа

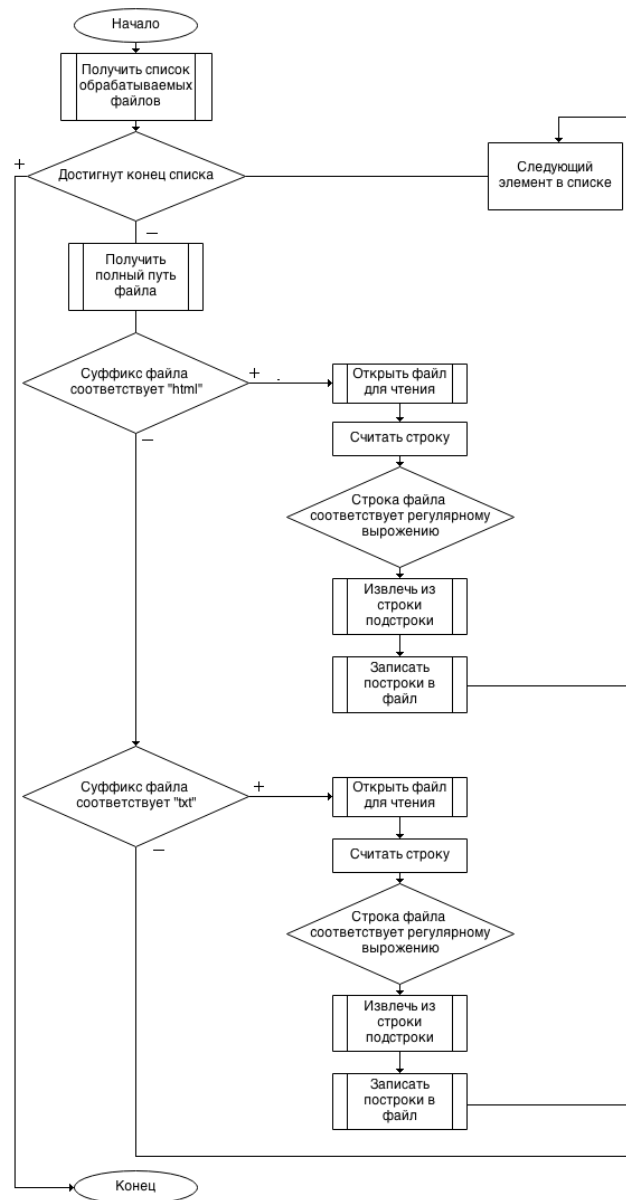


Рисунок 6.18 – Результат сбора данных контактной книги skype

- Старт элемента
- CDATA (анг. Character Data) Символьный тип данных
- Комментарии
- Конец документа
- Конец элемента
- Ошибка чтения

При проходе парсера по файлу мы реагируем на нужные нам элементы, атрибуты и прочее. Список интересующих элементов файла asoumt.xml

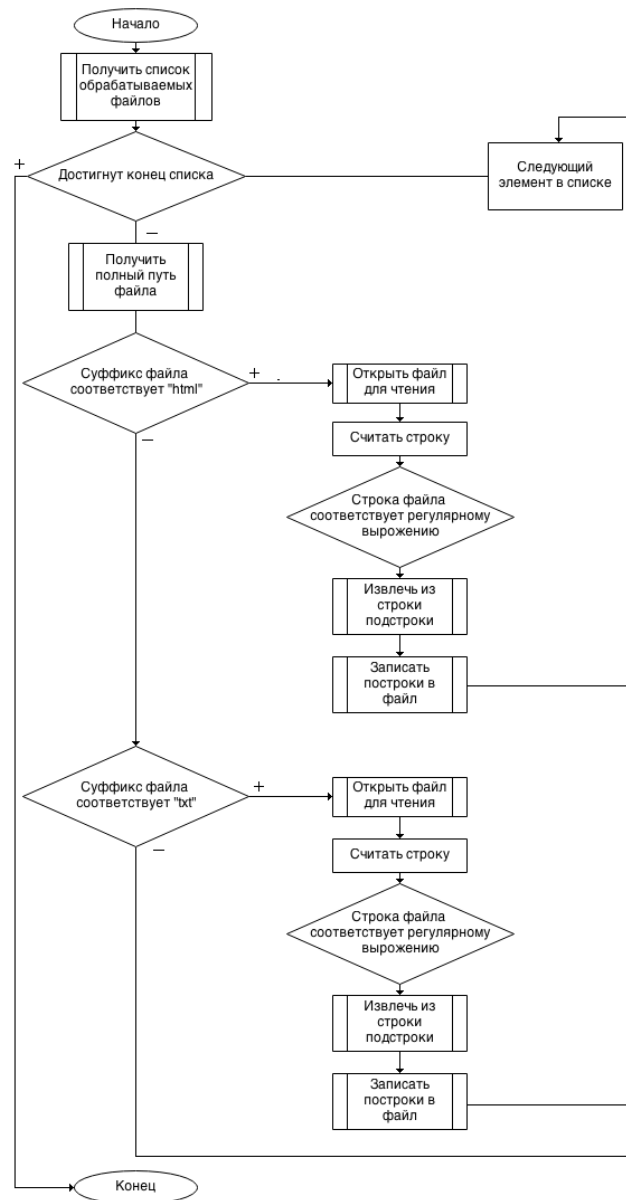


Рисунок 6.19 – Результат сбора данных аккаунта skype

- элемент `protocol` - используемый протокол для передачи сообщений.
- элемент `name` Имя или другое идентифицирующий элемент.
- элемент `password` Пароль.
- элемент `alias` Имя используемое при переписке.

Список интересующих элементов файла blist.xml

- элемент buddy: атрибут account - используемый акаунт.
- элемент buddy: атрибут proto - используемый протокол для передачи сообщений.

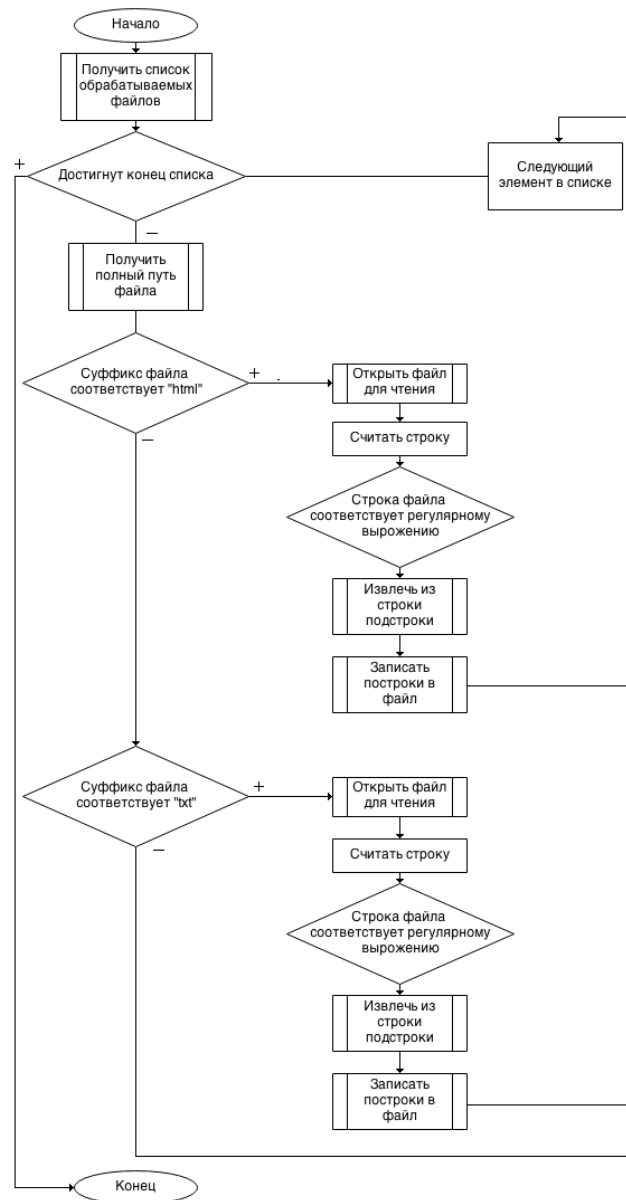


Рисунок 6.20 – Результат сбора данных звонков skype

- элемент `setting`: атрибут `last_seen` - здесь хранится время последнего появления контакта в сети интернет. Данные хранятся в `UnixTime`.
- элемент `alias` - Имя контакта используемое при переписке.

Затем данные вычитываются, и сохраняются в XML отчет.

6.12 функция сбора дополнительных данных skype

К имеющемуся списку функций работы с логами VoIP skype в этом семестре добавилось.

Функция извлечения информации о аккаунтах, когда-либо авторизовавшихся на

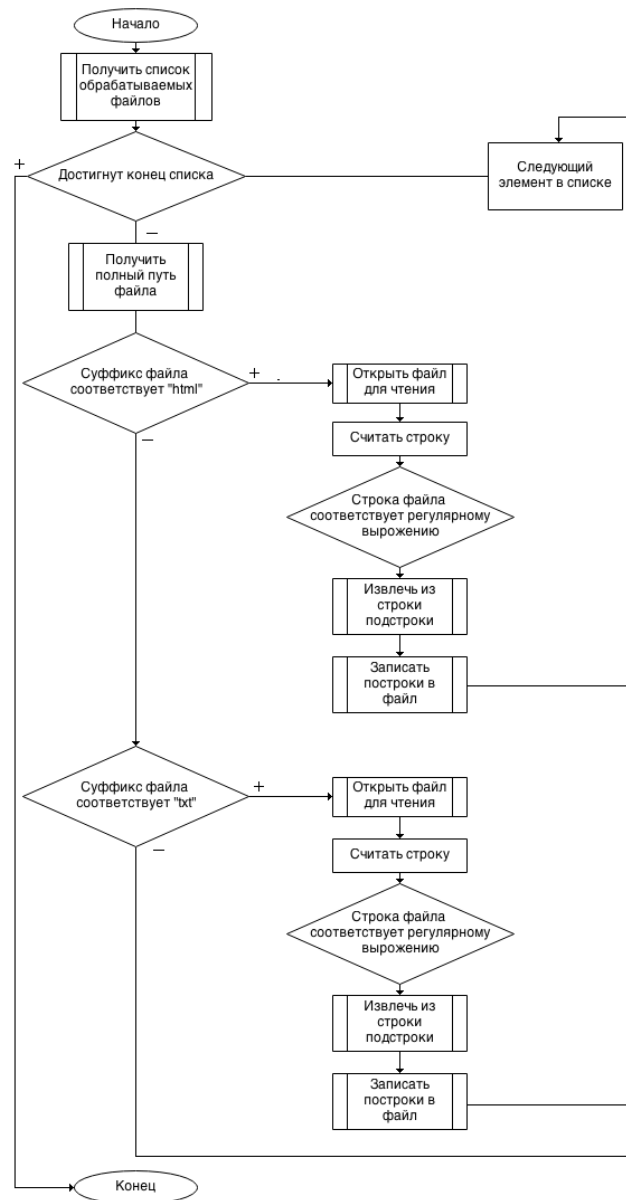


Рисунок 6.21 – Результат сбора установленного ПО

данном компьютере. Т.е при авторизации на компьютере создается база данных формата SQLite main.db. Одна из таблиц account содержит в себе информацию о используемом\.

- полное имя
- логин skype
- почта
- хеш-сумма используемого пароля
- ЯЗЫК

- данные о геолокации (Страна, город, часовой пояс)

Функция извлечения информации о звонках, входящих или исходящих когда-либо производившихся на данном компьютере. Таблица calls содержит в себе информацию данного типа.

- пользователь совершающий звонок
- пользователь принимающий звонок
- время звонка

Функция извлечения переписки между пользователями skype. Таблица chats содержит в себе информацию данного типа.

- пользователь передающий сообщение
- пользователь принимающий сообщение
- время создания сообщения

Данные вычитываются посредством SQL запросов, реализованных через библиотеку Qt, в классе QSql*, и затем так же сохраняются в XML отчет.

7 Некоторые аспекты сертификации программных средств объектов информатизации по требованиям информационной безопасности

Под сертификацией принято понимать независимое подтверждение соответствия тех или иных характеристик оборудования или информационной системы некоторым требованиям. В нашем случае речь идет о программном комплексе для проведения компьютерной экспертизы, во время экспертизы программный комплекс хранит и обрабатывает информацию, которая может относиться к персональным данным, коммерческой или государственной тайне и др. а также на основании результатов работы программы специалист проводящий экспертизу пишет заключение, которое может быть использовано как доказательство в суде. Поэтому к данной системе могут быть применены требования, касающиеся безопасности информации.

Для обеспечения высокого показателя качества необходима сертификация технологии и системы обеспечения качества их проектирования, разработки и сопровождения.

Проведение сертификации системы обычно планируется для достижения одной или нескольких целей:

- определения соответствия или несоответствия элементов системы установленным требованиям;
- определения эффективности внедренной системы с точки зрения соответствия поставленным целям для обеспечения качества продукции;
- обеспечения возможности улучшить свою систему;
- определения соответствия системы регламентирующим требованиям.

Первоначальные затраты на их проведение могут нести инициаторы испытаний, заказчик, а также ее разработчики и поставщик.

7.1 Схема проведения сертификации

Особенность любых сертификационных испытаний — это независимость испытательной лаборатории сертифицирующей организации, осуществляющей независимый контроль результатов испытаний.

Схема проведения сертификации:

- 1) Заявитель (в нашем случае разработчик) подает в федеральный орган (ФСБ, ФСТЭК или Минобороны) заявку на проведение сертификационных испытаний.
- 2) Федеральный орган определяет аккредитованную испытательную лабораторию и орган по сертификации.
- 3) Испытательная лаборатория совместно с заявителем проводит сертификационные испытания. Если в процессе испытаний выявляются те или иные несоответствия заявленным требованиям, то они могут быть

устранены заявителем в рабочем порядке, либо может быть принято решение об изменении требований к продукту, например о снижении класса защищенности.

- 4) Материалы испытаний передаются в орган по сертификации, который проводит их независимую экспертизу. В экспертизе участвуют не менее двух независимых экспертов.
- 5) Федеральный орган по сертификации на основании заключения органа по сертификации оформляет сертификат соответствия. При каких-либо несоответствиях, федеральный орган может провести дополнительную экспертизу с привлечением экспертов из различных аккредитованных лабораторий и органов.

В случае возникновения инцидентов на объектах информатизации, связанных с утечкой информации, регулирующие органы могут проинспектировать лабораторию, которая проводила испытания, и приостановить лицензию и аттестаты аккредитации.

Деятельность российских систем сертификации в РФ регламентируется Федеральным законом № 184 «О техническом регулировании». Сертификация средств защиты информации может быть добровольной или обязательной — проводимой главным образом в рамках Минобороны, ФСБ и ФСТЭК. К компетенции ФСБ относятся средства защиты информации, применяемые в органах государственной власти. Система сертификации средств защиты информации Минобороны, в свою очередь, ориентирована на программные изделия, применяемые на объектах военного назначения. [13]

7.2 Добровольная сертификация

Добровольная сертификация применяется с целью повышения конкурентоспособности продукции, расширения сферы ее использования и получения дополнительных экономических преимуществ. Результаты сертификации должны оправдывать затраты на ее проведение вследствие получения пользователем

ми продукции с показателями более высокого качества. Таким сертификационным испытаниям подвергаются компоненты операционных систем и пакеты прикладных программ широкого применения. Добровольные системы сертификации средств защиты информации на сегодняшний день пока еще не получили широкого распространения. Такого рода системой является «АйТи-Сертифика», но более известным средством сертификации является ЦЕРБЕР. К сожалению, несмотря на то что в добровольных системах можно получить сертификат на соответствие любому нормативному документу по защите конфиденциальной информации, при аттестации объектов информатизации такие сертификаты ФСТЭК России не признаются.

7.3 Обязательная сертификация

Обязательная сертификация необходима для программных продуктов и их производства, выполняющих особо ответственные функции, в которых недостаточное качество, ошибки или отказы могут нанести большой ущерб или опасны для жизни и здоровья людей. Примером такой сертификации служат программные продукты в авиации, в атомной энергетике, в военных системах и т.д.

Существуют два основных подхода к сертификации (таблица 7.3):

Таблица 7.3 – Основные подходы к сертификации

Подход	Документы
Функциональное тестирование	в соответствии с положениями стандарта ГОСТ Р 15408
Структурное тестирование	по ГОСТ Р 51275-99

Функциональное тестирование средств защиты информации, позволяющее убедиться в том, что продукт действительно реализует заявленные функции. Это тестирование проводится в соответствии с определенными нормативными документами Гостехкомиссии России. Если же не существует документа, которому сертифицируемый продукт соответствовал бы в полной мере, то функциональные

требования могут быть сформулированы в явном виде – в технических условиях, или в виде задания по безопасности.

Структурное тестирование программного кода на отсутствие недекларированных возможностей. Недекларированными возможностями, например, являются программные закладки, которые при возникновении определенных условий позволяют осуществлять несанкционированные воздействия на информацию. Выявление недекларированных возможностей - проведение серии тестов исходных текстов программ.

В большинстве случаев средство защиты информации должно быть сертифицировано как в части основного функционала, так и на предмет отсутствия недекларированных возможностей. Делается исключение для систем обработки персональных данных второго и третьего класса с целью снижения затрат на защиту информации для небольших частных организаций. Если программное средство не имеет каких-либо механизмов защиты информации, оно может быть сертифицировано только на предмет отсутствия недекларированных возможностей. [14]

7.4 Список организаций, проводящих сертификацию

Так как в некотором будущем в нашей архитектуре должен появиться модуль системы защиты данных от НСД рассмотрим несколько организаций, которые могут провести сертификацию нашего программного комплекса (таблица 7.4):

Таблица 7.4 – Организации проводящие сертификацию

Организация	Описание	Сертификация продуктов
Радиофизические тестовые технологии	Данный орган по сертификации программных средств аккредитован Ростехрегулированием(Госстандартом) в системе сертификации ГОСТ Р.	<p>1 операционные системы и средства их расширения, 2 системы программирования и утилиты, 3 сетевое программное обеспечение,</p> <p>4 системы управления базами данных (СУБД),</p> <p>5 электронные таблицы,</p> <p>6 базы данных и информационно-справочные системы,</p> <p>7 электронные архивы.</p> <p>8 и другие.</p>

АНО МИЦ	<p> Центр зарегистриро- вал в Росстандарте систему доброволь- ной сертификации программного обеспе- чения и аппаратно- программных ком- плексов </p>	<p> 1 ПО, используемое для моделирования технологических процессов, матема- тического и иного моделирования; 2 ПО передачи, хранения, актуализации, за- щиты, обеспечения доступа и использо- вания измерительной, вычислительной и иной информации; 3 ПО баз данных; 4 и другие. </p>
---------	---	--

Агенство РСТ	уполномоченный агент Регионального органа по сертификации и тестированию РОСТЕСТ-МОСКВА, одного из крупнейших сертификационных и испытательных центров в Европе. Сертификаты РОСТЕСТ-МОСКВА пользуются особым доверием потребителей и контролирующих организаций.	1 электронные базы данных и информационные системы; 2 системы управления базами данных; 3 сервисные программы (утилиты); 4 ПО для организации и работы с сетью; 5 и другие.
ЗАО «НПО «Эшелон»	аккредитовано в качестве органа по сертификации ФСТЭК России и испытательных лабораторий в системах обязательной сертификации Минобороны России, ФСБ России, ФСТЭК России и добровольной сертификации «АйТи-Сертифика».	1 соответствие требованиям по защищенности от несанкционированного доступа к информации; 2 отсутствие недекларированных возможностей (в т.ч. программных закладок), 3 соответствие заданию безопасности (по требованиям ГОСТ Р ИСО/МЭК 15408);

		<p>4 соответствие реальных и декларируемых в документации функциональных возможностей (в т. числе и на соответствие требованиям Технических Условий);</p> <p>5 и другие.</p>
<p>Центр сертификации «Мостест»</p>	<p>Структурное подразделение органа по сертификации «Гортест», предоставляет полный комплекс услуг, связанных с подготовкой и оформлением различных разрешительных документов.</p>	<p>1 автоматизированных систем, в т.ч. проектирования (САПР) и управления различными технологическими процессами (АСУ ТП), систем управления отраслями и объединениями, ПО для технологической подготовки производства;</p> <p>2 систем программирования, утилит, операционных систем;</p> <p>3 сетевого ПО и приложений мультимедиа-и т.д.;</p>

		4 систем управления базами данных, баз данных, информационных и справочных систем; 5 электронных изданий, архивов, таблиц; 6 и другое.
--	--	--

В данном разделе отчета была исследована информация о сертификации программных средств и др. Также были найдены организации, которые могут произвести сертификацию нашего программного комплекса. Это необходимо для возможности использовать данный комплекс для работы с информацией различного уровня конфиденциальности. Что в свою очередь даст возможность использовать данный программный комплекс в государственных и коммерческих учреждениях для проведения компьютерной экспертизы. Также сертификация обеспечит высокий показатель качества программного комплекса для выхода на рынок.

8 Создание собственного репозитория для debian и ubuntu

С развитием разрабатываемого программного комплекса возникла необходимость организовать свой репозиторий. Свой репозиторий нужен для того, чтобы выложить программное обеспечение в открытый доступ в удобном для установки и обновления виде.

8.0.1 Порядок конфигурирования репозитория

Для начала следует выбрать директорию, в которой мы хотим выкладывать наши deb пакеты. Пусть это будет `/var/www/repositories/debian`.

```
prefix=/var/www/repositories/debian
```

```
mkdir -p $prefix/main
```

Копируем все необходимые deb пакеты в /var/www/repositories/debian/main, там можно устроить любую структуру каталогов. Сделаем подкаталоги для каждого пакета и положим в них файлы .deb, .dsc, .changes и .tar.gz. Далее переходим в корень нашего репозитория.

```
cd $prefix
```

Создаем индексные файлы. Для бинарных пакетов сделать это можно с помощью команды dpkg-scanpackages.

```
dir=$prefix/dists/unstable/main/binary-i386
```

```
mkdir -p $dir
```

```
dpkg-scanpackages -arch i386 main /dev/null > $dir/Packages
```

```
gzip -9c <$dir/Packages >$dir/Packages.gz
```

```
bzip2 -9c <$dir/Packages >$dir/Packages.bz2
```

Прделаем это для каждой архитектуры, для которой есть пакеты в main. Затем необходимо создать индексные файлы для исходных текстов

```
dir=$prefix/dists/unstable/main/source
```

```
mkdir -p $dir
```

```
dpkg-scansources main /dev/null > $dir/Sources
```

```
gzip -9c <$dir/Sources >$dir/Sources.gz
```

```
bzip2 -9c <$dir/Sources >$dir/Sources.bz2
```

И наконец, необходимо создать файл \$prefix/dists/unstable/main/Release, который имеет следующую структуру:

```
Archive: unstable Suite: unstable Component: main Origin:
название моей организации Label: Debian репозиторий моей организации
Architecture: архитектуры
```

Создадим такой файл и заполним необходимые поля, затем дадим команды

```
dir=$prefix/dists/unstable/main apt-ftparchive release $dir
»$dir/Release
```

Последняя команда вычисляет MD5Sum, SHA1 и SHA256 для файлов

Packages* и Sources*.

Теперь осталось только подписать репозиторий. Сделаем это следующей командой:

```
gpg -abs -o dir/Release.gpg dir/Release
```

Для того, чтобы aptitude, apt и др. смогли найти новый репозиторий, добавим необходимые записи в /etc/apt/sources.list.

```
deb http://example.com/repository/debian unstable main deb-src  
http://example.com/repository/debian unstable main
```

9 Цели следующего семестра

- 1) тестирование системы;
- 2) подготовка deb-пакета, лицензии;
- 3) написание публикации.

10 Заключение

В данном семестре нашей группой была выполнена часть работы по созданию автоматизированного программного комплекса для проведения компьютерной экспертизы, проанализированы дальнейшие перспективы и поставлены цели на следующий семестр.

Список использованных источников

- 1 Федотов Н. Н. Форензика - компьютерная криминалистика. — Юрид. мир, 2007. — Р. 432. — ISBN: 9785911590154.
- 2 Chacon S. Pro Git : professional version control. — 2011. — URL: <http://progit.org/ebook/progit.pdf>.
- 3 Львовский С. Набор и вёрстка в системе L^AT_EX. — МЦНМО, 2006. — Р. 448. — ISBN: 5-94057-091-7.
- 4 Котельников И. А.; Чеботаев П. З. L^AT_EX_{2 ϵ} по-русски. — Сибирский Хронограф, 2004. — Р. 489.
- 5 Qt Documentation [Электронный ресурс]. — 2013. — URL: <http://qt-project.org/doc/>.
- 6 Всё о кроссплатформенном программировании - Qt [Электронный ресурс]. — 2013. — URL: <http://doc.crossplatform.ru/qt/>.
- 7 Справочник по XML-стандартам [Электронный ресурс], Microsoft. — URL: [http://msdn.microsoft.com/ru-ru/library/ms256177\(v=vs.110\).aspx](http://msdn.microsoft.com/ru-ru/library/ms256177(v=vs.110).aspx).
- 8 Windows Event Log Format [Электронный ресурс]. — 2005. — URL: http://www.whitehats.ca/main/members/Malik/malik_eventlogs/malik_eventlogs.html.
- 9 Официальное сообщество Skype [Электронный ресурс]. — URL: <http://community.skype.com/>.
- 10 Михайлов И. Ю. Компьютерно-техническая экспертиза. — 2007 – 2013. — URL: <http://computer-forensics-lab.org/>.
- 11 Официальное сообщество Pidgin. — URL: <https://developer.pidgin.im/>.
- 12 Валиков А. Н. Технология XSLT. — БХВ-Петербург, 2002. — Р. 544. — ISBN: 5941571291.
- 13 Сертификация программного обеспечения, центр сертификации Мостест [Электронный ресурс]. — URL: <http://mostest.su/>

uslugi-po-sertifikacii-oformlenie-sertifikatov-otkaznyx-pisem-i-dr/
sertifikaciya-programmnogo-obespecheniya.

- 14 Сертификация средств защиты информации (СЗИ) по требованиям безопасности информации: сертификация и сертификационные испытания в системах сертификации ФСТЭК России, Минобороны России, ФСБ России, «АйТи-Сертифика» [Электронный ресурс]. — URL: <http://www.npo-echelon.ru/services/certification/>.

Приложение А
(Обязательное)
Компакт-диск

Компакт-диск содержит:

- электронную версию пояснительной записки в форматах *.tex и *.pdf;
- актуальную версию программного комплекса для проведения компьютерной экспертизы;
- тестовые данные для работы с программным комплексом.