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
«_____»_____2015г.

КОМПЬЮТЕРНАЯ ЭКСПЕРТИЗА

Отчет по групповому проектному обучению
Группа КИБЭВС-1401

Ответственный исполнитель
студент гр. 722
_____О.В. Лобанов
«_____»_____2015г.

Научный руководитель
аспирант каф. КИБЭВС
_____А.И. Гуляев
«_____»_____2015г.

2015

Список исполнителей

Лобанов О.В. – программист, ответственный исполнитель, ответственный за разработку функций сбора информации из реестра.

Шиповской В.В. – программист, ответственный за написание части системы для сбора и обработки информации из браузера Google Chrome.

Серяков А.В. – программист, ответственный за написание части системы для сбора информации из почтового клиента MS Outlook.

Боков И.М. – программист, ответственный за написание части системы для для нахождения медиа-файлов (аудио, видео, изображение) и извлечения мета-данных из них.

Кучер М.В. – программист, ответственный за написание части системы для сбора информации об установленном ПО по остаточным файлам.

Терещенко Ю.А. – программист, ответственный за написание части системы для сбора информации из почтового клиента Mozilla Thunderbird.

Мейта М.В. – документатор, ответсвенный за верстку необходимой документации в системе LaTeX.

Содержание

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 разработчиками данного проекта были рассмотрены такие направления компьютерной экспертизы, как исследование файловых систем, сетевых протоколов, организация работы серверных систем, механизм журналирования событий. Также были изучены основные задачи, которые ставятся перед сотрудниками правоохранительных органов, проводящими компьютерную экспертизу, и набор существующих 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, не имеющий на данный момент аналогов в РФ.

1 Назначение и область применения

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

2 Постановка задачи

На данный семестр были поставлены следующие задачи:

– изучение архитектуры проекта «Компьютерная экспертиза» новыми участниками проектной группы;

– изучение теоретического материала и основных инструментов разработки;

– определение индивидуальных задач для каждого участника проектной группы;

– исследование предметных областей в рамках индивидуальных задач;

– реализация новых программных модулей и доработка уже существующих;

– изучение инструментов для генерации документации к программному коду проекта.

Задачи по проектированию модулей:

1) сбор и анализ информации из реестра Windows;
2) сбор и анализ информации из браузера Google Chrome;
3) сбор и анализ информации из почтового клиента Mozilla Thunderbird;
4) сбор и анализ информации из почтового клиента MS Outlook;
5) поиск медиа-файлов (аудио, видео, изображение) и извлечение мета-данных из них;
6) сбор информации об установленном ПО по остаточным файлам.

3 Инструменты

3.1 Система контроля версий Git

Для разработки программного комплекса для проведения компьютерной экспертизы было 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].

При работе над одним проектом команде разработчикоа необходим инструмент для совместного написания, бэкапирования и тестирования программного обеспечения. Используя Git, 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;

– необходимо делать коммит как только был получен осмысленный результат;

– ветка master мержится не разработчиком, а вторым человеком, который производит вычитку и тестирование изменения;

– все коммиты должны быть осмысленно подписаны/прокомментированы.

3.2 Система компьютерной вёрстки TEX

3.3 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 огромнейшее количество платформ.

Qt позволяет запускать написанное с его помощью ПО в большинстве современных операционных систем путём простой компиляции программы для каждой ОС без изменения исход-

ного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, 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 фраемворка QT

– iostream
– QChar
– QCryptographicHash
– QDateTime
– QDir
– QDirIterator
– QFile
– QFileInfo
– QIODevice
– QList
– QRegExp
– QString
– QTextStream
– QtSql/QSqlDatabase
– QVector
– QMap
– 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/QSqlDatabase 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. [3]

Класс QString предоставляет строку символов Unicode.

Класс QMap — контейнерный класс для хранения элементов различных типов данных.

Класс QDateTime используется для работы с форматом даты, в который записывается информация о файле.

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 использующихся 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).[4]

Класс QRegExp предоставляет сопоставление с образцом при помощи регулярных выражений.

Регулярное выражение, или "regexp", 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.[4]

QChar обеспечивает поддержку 16-битных символов Unicode.

3.3.1 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 фай-

лов.

3.3.2 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.

3.4 GitLab

Для работы над проектом проектной группой был поднят собственный репозиторий на сервере git.keva.su. Адреса репозиториев следующие:

http://gitlab2.keva.su/gpo/coex

Репозиторий для тестирования проекта:

git clone git@gitlab2.keva.su:gpo/coex.git**Исходные файлы проекта можно найти здесь:**

http://gitlab2.keva.su/gpo/coex

Репозиторий для тестирования проекта:

git clone git@gitlab2.keva.su:gpo/coex.git

4 Технические характеристики

4.1 Требования к аппаратному обеспечению

Минимальные системные требования:

– **процессор 1ГГц Pentium 4;**

– **оперативная память 512 Мб;**

– **место на жёстком диске – 9 Гб.**

4.2 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.

4.3 Выбор единого формата выходных файлов

Для вывода результата был выбран формат XML-документов, так как с данным форматом лего 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 [5]. 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.

5 Разработка программного обеспечения

5.1 Архитектура

5.1.1 Основной алгоритм

В ходе разарботки был применен видоизменнённый 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 5.1.

Рисунок 5.1 – Пример организации проекта при использовании шаблона проектирования Factory method

Использование данного шаблона позволило разбить 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 клас предназначался для формирования списка настроек, определения операционной системы на смонтированном образе и инстанционировании и накапливание всех необходимых классов-тасков в очереди тасков. После чего каждый таск из очереди отправлялся на выполнение. Блоксхема работы алгоритма представлена на рисунке 5.2.

Рисунок 5.2 – Алгоритм работы с образом диска

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

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

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 5.3.

Рисунок 5.3 – UML-диаграмма классов

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

5.1.2 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:

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 такска 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 такска

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

5.2 Сбор информации из браузера Google Chrome

5.3 Плагин SearchProgram

5.4 Сбор информации из почтового клиента MS Outlook

5.5 Сбор информации из почтового клиента Mozilla Thunderbird

5.6 Поиск медиа-файлов и извлечение мета-данных

5.7 Сбор и анализ информации из реестра ОС MS Windows

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 для дальнейшего развития проекта.

Список использованных источников

1 **Федотов Николай Николаевич. Форензика - компьютерная криминалистика. Юрид. мир, 2007. 432 с.**

2 **Scott Chacon. Pro Git : professional version control. 2011. URL: http://progit.org/ebook/progit.pdf.**

3 **Qt Documentation [Электронный ресурс] // qt-project.org:[сайт]. 2013. URL: http://qt-project.org/doc.**

4 **Всё о кроссплатформенном программировании - Qt [Электронный ресурс] // doc.crossplatform.ru:[сайт]. 2013. URL: http://doc.crossplatform.ru/qt.**

5 **Справочник по XML-стандартам [Электронный ресурс] // msdn.microsoft.com:[сайт]. URL: http://msdn.microsoft.com/ru-ru/library/ms256177(v=vs.110).aspx.**

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;

– документацию к проекту в html-формате.