

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФГБОУ ВПО

«БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра «Информатика и программное обеспечение»

«УТВЕРЖДАЮ»

Зав. кафедрой «И и ПО», к.т.н., доцент

Подвесовский А.Г.

«____» _____ 2015 г.

СИСТЕМА ЭВРИСТИЧЕСКОГО АНАЛИЗА ПРИЗНАКОВ ВРЕДОНОСНОГО ПОВЕДЕНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ДИПЛОМНАЯ РАБОТА

Документы текстовые

Всего _____ листов в папке

Руководитель

к.т.н., доц. Трубаков А.О.

«____» _____ 2015 г.

Консультанты:

по экономической части

к.т.н., доц. Титарёв Д.В.

«____» _____ 2015 г.

по организационной части

к.т.н., доц. Зотина О.В.

«____» _____ 2015 г.

Нормоконтролер

к.т.н., доц. Лагереv Д.Г.

«____» _____ 2015 г.

Студент

Васин А.В.

«____» _____ 2015 г.

БРЯНСК 2015

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	5
1. АНАЛИЗ ТРЕБОВАНИЙ	6
1.1. Обзор методов защиты от вредоносного ПО	6
1.2. Обзор и сравнение аналогов	11
1.2.1. Искусственная иммунная система на основе нейросетей	12
1.2.2. Искусственная иммунная система на базе статистического метода.....	13
1.2.3. Эвристический анализ на основе n-грамм	14
1.3. Вывод.....	15
2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ.....	16
2.1. Основания для разработки	16
2.2. Назначение разработки.....	16
2.3. Общие требования	17
2.4. Функциональные требования	17
2.4.1. Требования к подсистеме обучения.....	17
2.4.2. Требования к подсистеме эвристического анализа.....	18
2.4.3. Требование к аналитическому веб-сервису	19
2.5. Требования к архитектуре и математическому аппарату	19
2.6. Требование к аппаратному и программному обеспечению	19
3. ИССЛЕДОВАНИЕ АЛГОРИТМОВ КЛАССИФИКАЦИИ	20
3.1. Описание классификаторов	20
3.1.1. Многослойная нейронная сеть	20
3.1.2. Рекуррентная нейронная сеть.....	21
3.1.3. Дерево решений ID3	22
3.2. Оценка классификаторов.....	23

3.3. Вывод.....	24
4. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА.....	25
4.1. Высокоуровневое проектирование.....	25
4.1.1. Прецеденты взаимодействия персонала с системой.....	25
4.1.2. Прецеденты взаимодействия сущностей ядра.....	25
4.1.3. Архитектура инфраструктуры.....	26
4.2. Низкоуровневое проектирование	29
4.2.1. Роли разработчиков	29
4.2.2. Структурное проектирование.....	30
4.3. Математический аппарат	35
4.3.1. Матрица накоплений	35
4.3.2. Искусственная нейронная сеть.....	36
4.4. Выбор средств разработки	37
4.5. Руководство по использованию утилит обучения.....	37
4.5.1. Утилита librarian.....	38
4.5.2. Утилита teacher.....	38
5. ЭКОНОМИЧЕСКИЙ АНАЛИЗ	40
5.1. Организационная структура проекта	40
5.2. Календарный план проекта	40
5.3. Расчёт затрат на разработку проекта	42
5.3.1. Расчёт заработной платы исполнителей работ по созданию программного продукта	43
5.3.2. Расчёт отчислений на социальные нужды	43
5.3.3. Арендные платежи за офисные помещения.....	44
5.3.4. Амортизация используемых основных средств и нематериальных активов.....	45

5.3.5. Расходы на модернизацию и приобретение основных средств	45
5.3.6. Расходы на приобретение ПО	45
5.3.7. Расходы на интернет и связь	45
5.3.8. Расходы на канцелярские товары и расходные материалы.....	46
5.3.9. Прочие расходы	46
5.3.10. Расчёт себестоимости программного продукта.....	46
6. ТЕСТИРОВАНИЕ СИСТЕМЫ.....	48
6.1. План испытаний	48
6.2. Тестирование подсистемы обучения	48
6.2.1. Тестирование утилиты librarian на небольшой выборке	48
6.2.2. Тестирование утилиты librarian на большой выборке	49
6.2.3. Тестирование утилиты teacher.....	50
7. ОРГАНИЗАЦИОННАЯ ЧАСТЬ	52
7.1. Организация рабочего места и режима труда и отдыха	53
7.2. Противопожарная безопасность	55
7.3. Электробезопасность	58
7.4. Шум и вибрация	59
7.5. Расчёт освещенности	60
ЗАКЛЮЧЕНИЕ	63
СПИСОК ЛИТЕРАТУРЫ	65
ПРИЛОЖЕНИЕ А.....	68
ПРИЛОЖЕНИЕ Б.....	69
ПРИЛОЖЕНИЕ В.....	70

ВВЕДЕНИЕ

В современном мире постоянно увеличивается сложность программного обеспечения. Вредоносное ПО тоже не стоит на месте и постоянно совершенствуется. Злоумышленники с каждым разом придумывают всё более изощрённые методы маскировки, затрудняющие обнаружение и анализ вредоносного ПО. В связи с этим, классические методы обнаружения, такие как сигнатурный анализ, становятся всё менее эффективными.

Для борьбы с современным вредоносным программным обеспечением всё чаще применяют методы проактивного анализа (методы обнаружения вредоносных программ, ещё не известных антивирусному программному обеспечению), такие как эмуляция и эвристический анализ. Брянская компания «NANO Security» использует эти технологии в своём продукте «NANO Antivirus», однако их применение вызывает ряд технических проблем.

Продукт «NANO Antivirus» уже имеет в составе своего ядра модуль эвристического анализа. В современных условиях большого разнообразия вредоносного ПО текущая версия эвристического анализатора неэффективна в использовании и очень сложна в обслуживании.

Цель данной работы — модернизация существующего эвристического анализатора.

Задачи работы:

- а) построение проекта системы эвристического анализа;
- б) разработка классификатора;
- в) экспериментальная проверка классификатора в составе ядра антивируса;
- г) анализ полученных результатов.

Объект исследования — процесс эвристического анализа исполняемых файлов.

Предмет исследования — механизм эвристического анализа файлов и автоматизация обучения эвристического анализатора.

1. АНАЛИЗ ТРЕБОВАНИЙ

В данной главе рассмотрены общие методы защиты от вредоносного ПО, произведено сравнение различных методик эвристического анализа.

1.1. Обзор методов защиты от вредоносного ПО

С развитием сети Интернет, облачных и мобильных технологий, а также развитием технологий электронной коммерции в современном мире остро стал вопрос защиты от киберугроз. Первое, что пытаются сделать киберпреступники – украсть денежные средства жертв, преимущественно с помощью вредоносных мобильных приложений. Второе – объединить зараженные устройства в одну общую сеть и использовать её для анонимной передачи данных или массовых атак [5].

Масштабы киберпреступлений хорошо видны в ежегодном отчёте **Kaspersky Security Bulletin 2013** [5]. «Лаборатория Касперского» сообщает, что их продукты отразили 5 188 740 554 атак. Из этого числа:

- а) 90,52% атак приходилось на продукт Oracle Java;
- б) 2,63% атак приходилось на компоненты Windows не являющиеся частью Microsoft Office или Internet Explorer;
- в) 2,5% атак приходилось на Android устройства;
- г) оставшаяся часть – Microsoft Office, Internet Explorer, Adobe Flash.

Брянская компания «NANO Security» [13] также занимается проблемами обеспечения кибербезопасности. Как и любой разработчик антивирусного ПО она так же сталкивается с проблемами обнаружения вредоносного ПО.

Вредоносное ПО (ВПО) – программное обеспечение, которое разрабатывается для получения несанкционированного доступа к вычислительным ресурсам ЭВМ, а также данным, которые на ней хранятся. Такие программы предназначены для нанесения ущерба владельцу информации или ЭВМ, путем копирования, искажения, удаления или подмена информации [12].

Как и любой другой вид программного обеспечения, ВПО также стремительно развивается как со стороны набора функций, позволяющие авторам вести деструктивную, шпионскую или коммерчески выгодную деятельность, так и со стороны способов маскировки вредоносного кода. В современном мире всё чаще встречаются вредоносные программы, не поддающиеся классическому сигнатурному анализу.

Сигнатура – последовательность байт, содержащая в себе вредоносный код, общий для некоторой группы файлов. В общем случае, сигнатуры формируются вирусными аналитиками, анализирующие вредоносные файлы. Однако сигнатуры могут создаваться автоматом, если процесс поиска не является сложным и является шаблонным. Сигнатурный анализ – процесс поиска известных антивирусному программному обеспечению сигнатур в исследуемом файле. Как правило, сигнатуры хранятся в антивирусных базах [18].

Основной недостаток сигнатурного анализа – чувствительность к изменениям вредоносного кода и его маскировкам. Некоторые разработчики ВПО сознательно применяют техники обфускации и полиморфизма, чтобы их ВПО оставалось не обнаруживаемым техниками сигнатурного анализа.

Полиморфизм – возможность объектов с одинаковой спецификацией иметь различную реализацию. Достигается путем замены одних инструкций на другие, носящие идентичный характер. Допустим, нужно создать код, который заполняет регистр EAX значением «0». Возможно множество реализаций [18]:

```
PUSH 0; POP EAX;
XOR EAX, EAX;
MOV EAX, 0;
AND EAX, 0;
```

Обфускация – приведение исходного текста или исполняемого кода программы к виду, сохраняющему ее функциональность, но затрудняющему анализ, понимание алгоритмов работы и модификацию при декомпиляции. Например:

```
PUSHAD; NOP; NOP; NOP; NOP; POPAD;
ADD EAX, 0xFFEEFFEE; INC EAX; OR EAX, EAX; SUB EAX, 0xFFEEFFEF;
PUSH EBX; PUSH ECX; POP ECX; POP EBX; LEA EAX, [EAX]; MOV EDX, EDX;
```

Все три последовательности инструкций не несут никакой полезной нагрузки, но вполне могут быть использованы, чтобы «замусорить» полезный код [19].

Очевидно, что сигнатурный анализ не эффективен перед техниками такого рода. Однако эмуляция и эвристический анализ могут частично решить проблему.

Эмуляция – техника контролируемого выполнения программы и отслеживания её поведения в динамике. Обычно работает совместно с эвристическим анализом.

Эвристический анализ – техника анализа кода программы в статике или в процессе эмуляции. Суть его заключается в нахождении определённых признаков. Каждый признак не является показателем вредоносности, однако по нескольким признакам можно сделать вывод о зловредности программы. Каждый признак может характеризовать как вредоносные действия, так и действия присущие безопасному ПО.

«NANO Security» использует обе техники в неклассическом анализе, однако существуют проблемы, связанные с недостатками этих техник.

Недостаток эмуляции – большие накладные расходы на анализ, следовательно, эмулируемый код работает медленней. Некоторые виды ВПО используют специальные приёмы, основанные на медленном выполнении кода. Такое ВПО определяет, что работает в эмуляторе, и просто не выполняет вредоносный код.

Самая большая техническая сложность эвристического анализа заключается в отсутствии информации о важности признака при его выявлении вирусным аналитиком. Следовательно, не известно, какое влияние должен оказывать на результат анализа этот признак.

С первым и наиболее важным недостатком эвристического анализа возможно бороться с помощью использования эвристического анализатора, способного к обучению. На данный момент «NANO Security» таковым не обладает.

Диаграмма IDEF0, иллюстрирующая модель AS-IS обслуживания существующего эвристического анализатора представлена на рис. 1.1-1.5.

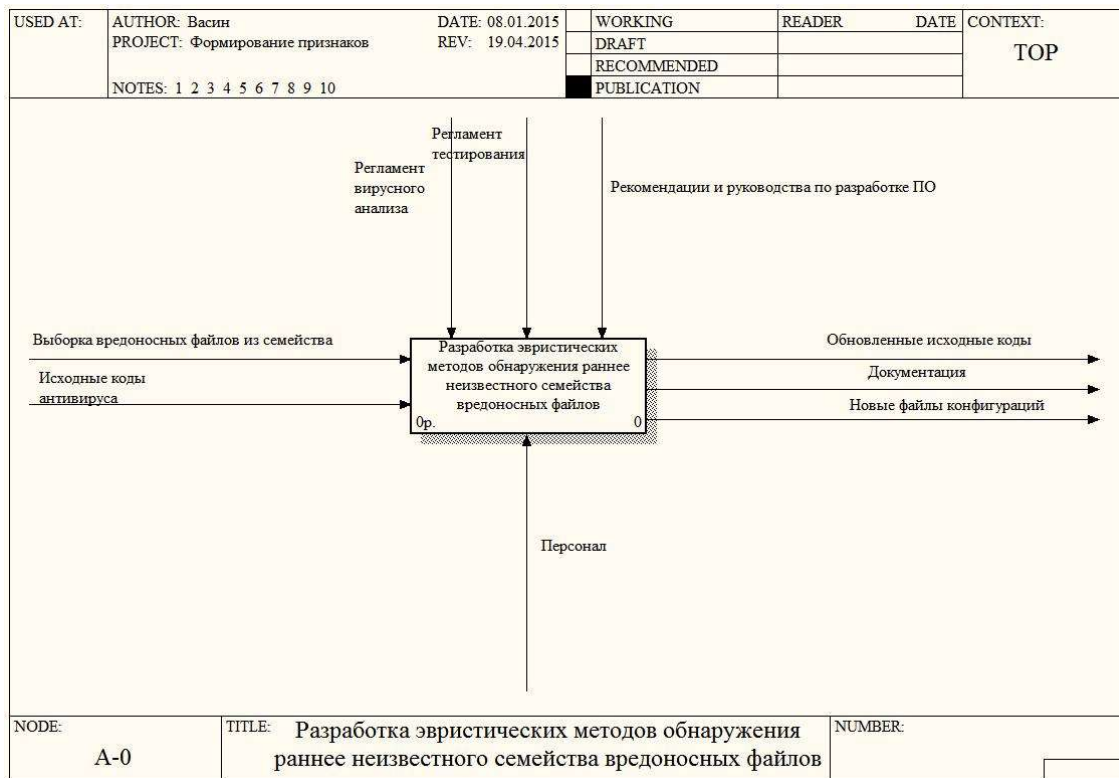


Рис. 1.1. Контекстный уровень модели

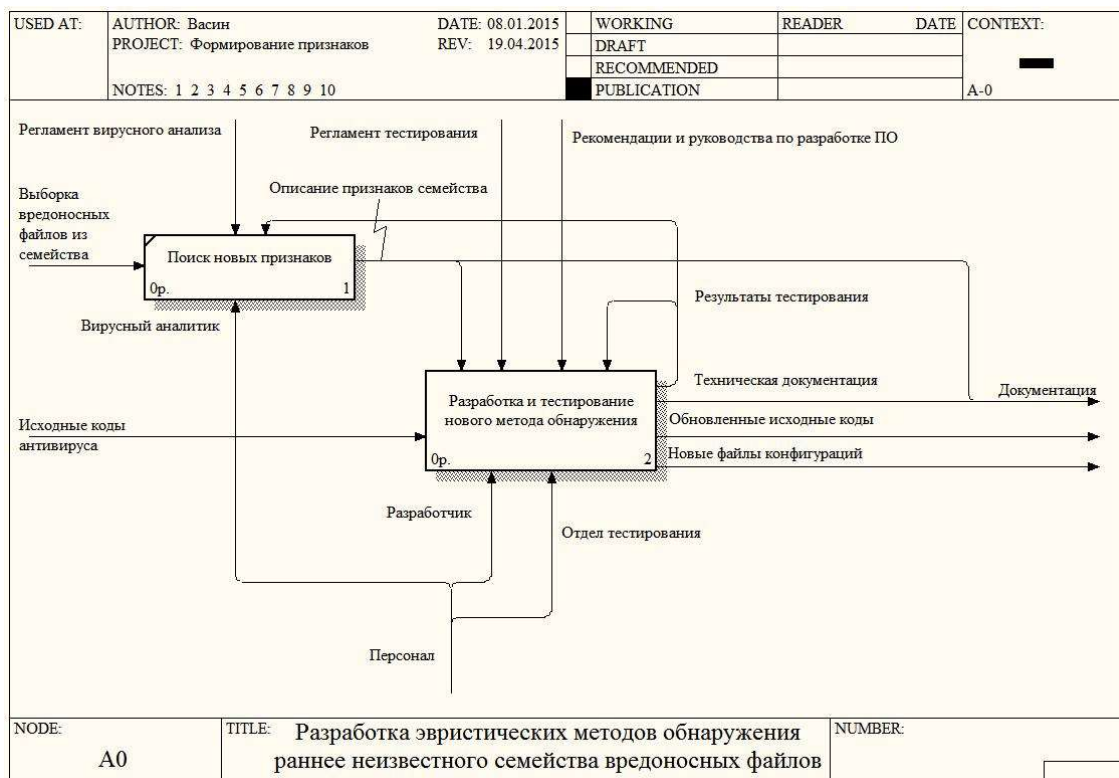


Рис. 1.2. Процесс разработки новых эвристических методов обнаружения

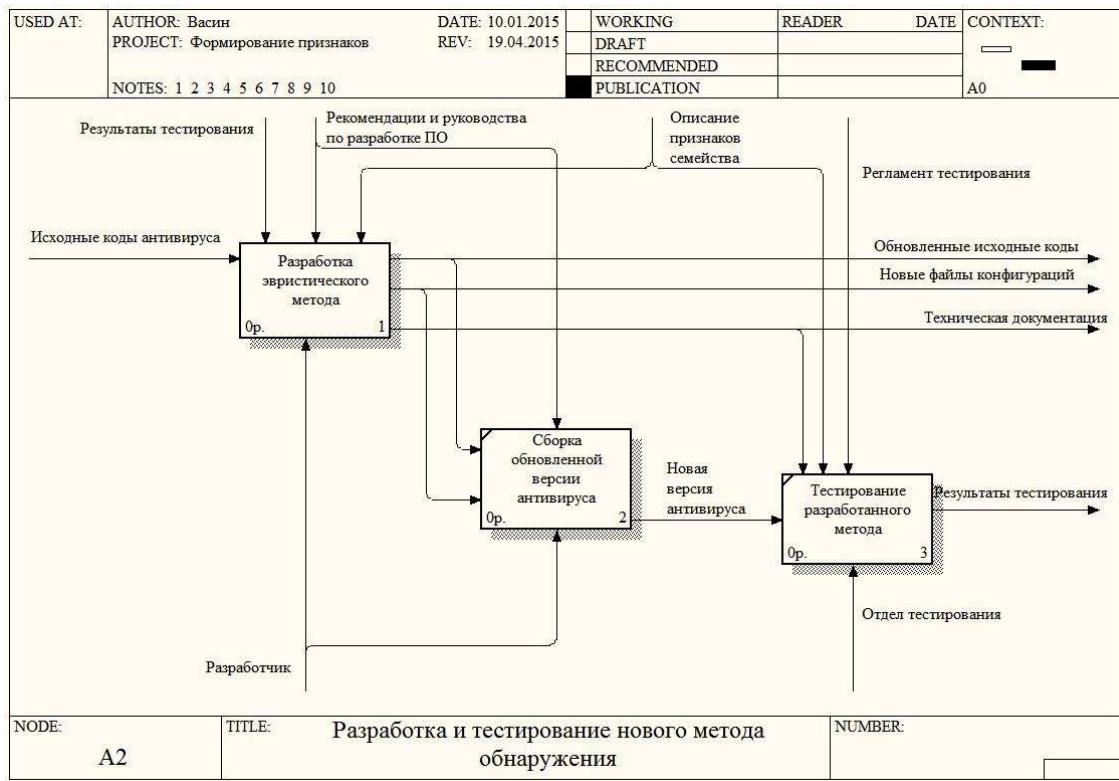


Рис. 1.3. Процесс внедрения поддержки нового эвристического метода

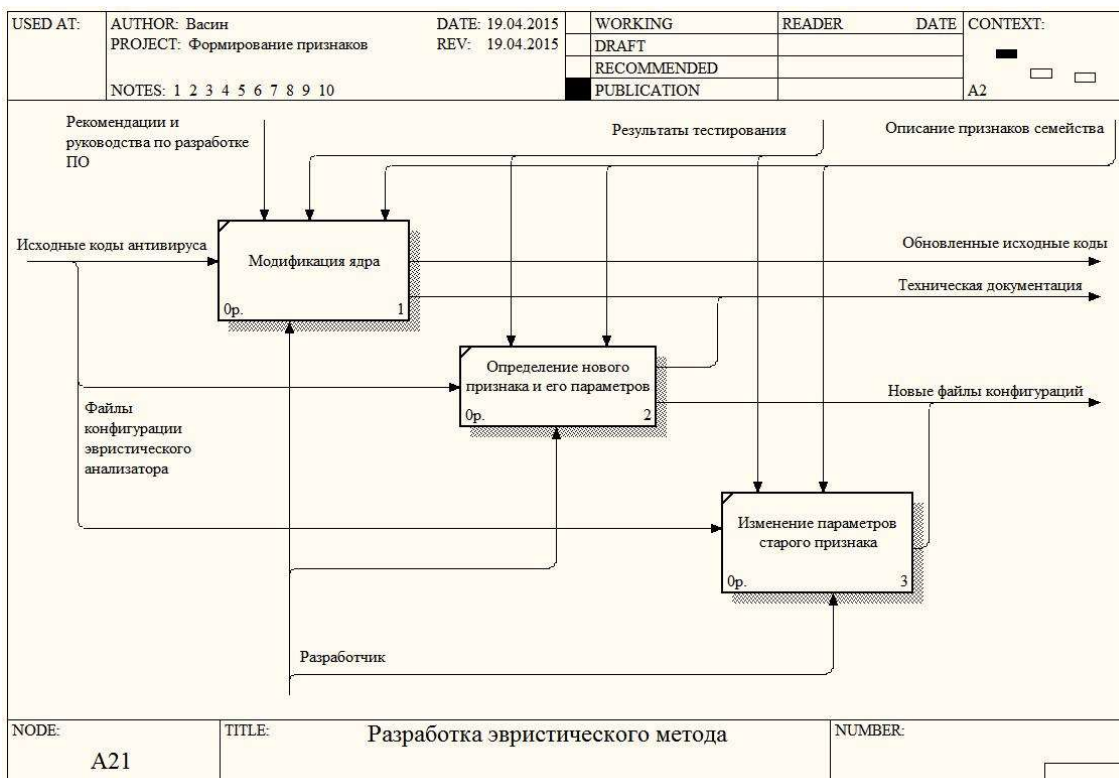


Рис. 1.4. Разработка или обслуживание одного эвристического метода

USED AT:	AUTHOR: Васин PROJECT: Формирование признаков	DATE: 09.01.2015 REV: 19.04.2015	WORKING DRAFT	READER	DATE	CONTEXT: TOP
NOTES: 1 2 3 4 5 6 7 8 9 10				RECOMMENDED		
			PUBLICATION			A-0

Разработка эвристических методов обнаружения раннее неизвестного семейства вредоносных файлов
 0р. 0

Поиск новых признаков
 0р. 1

Разработка и тестирование нового метода обнаружения
 0р. 2

- Разработка эвристического метода
- Сборка обновленной версии антивируса
- Тестирование разработанного метода

NODE: A0	TITLE: Разработка эвристических методов обнаружения раннее неизвестного семейства вредоносных файлов	NUMBER:
-------------	--	---------

Рис. 1.5. Дерево модели

Как видно из модели AS-IS, в частности на рис. 1.4, текущая реализация модуля эвристического анализа целиком конфигурируется разработчиком, а не автоматическими системами, что является существенным недостатком. Большую проблему представляют ложноположительные результаты анализа: разработчик вынужден изменять весовые коэффициенты различных признаков и зависимостей между ними так, чтобы пропали ложноположительные реакции на чистые файлы пропали, но положительные реакции на вредоносных файлах остались.

Вышеописанный недостаток создает острую необходимость в использовании другого типа эвристического анализатора – анализатора способного к обучению.

1.2. Обзор и сравнение аналогов

Вследствие технологической закрытости всех крупных коммерческих решений невозможно сравнить обучаемые эвристические анализаторы. Однако можно сравнить опубликованные методики эвристического анализа.

1.2.1. Искусственная иммунная система на основе нейросетей

Безобразов С.В. и Головки В.А. в своей статье «Алгоритмы искусственных иммунных систем и нейронных сетей для обнаружения вредоносных программ» предлагают использовать искусственную иммунную систему с набором нейросетей в качестве антител [3].

Антитело в биологии – белок, синтезированный лимфоцитами крови в ответ на вторжение чужеродных веществ или организмов (антигенов). Каждая проникающая в тело бактериальная или вирусная инфекция подстегивает производство специфических антител для борьбы с данными антигенами. После подавления инфекции антитела остаются в крови и в случае вторичного инфицирования вступают в борьбу [2].

Ключевая особенность антител заключается в том, что каждое антитело реагирует на узкий спектр антигенов. Именно по этой причине работает вакцинация: иммунитет вырабатывает антитела для борьбы с антигенами вакцины, но эти же антитела способны реагировать на антигены, для борьбы с которыми проводят вакцинацию. Это объясняется тем, что каждое антитело имеет множество молекулярных рецепторов на своей поверхности, каждый из которых реагирует на определенный пептид на поверхности антигена. Чем больше рецепторов антитела взаимодействует с пептидами антигена, тем активнее происходят химические реакции и тем выше сила связи между антителом и антигеном. Эту силу связи можно количественно измерить в виде значения аффинности – чем выше аффинность, тем прочнее связь [6].

Именно эту особенность иммунитета моделируют авторы вышеупомянутой статьи. В предложенном алгоритме предлагается создать популяцию нейросетевых детекторов (антитела), обучить их на случайно сформированных обучающих выборках, затем уничтожить нейросетевые детекторы, не способные к обучению или те, в работе которых наблюдаются дефекты (ложные срабатывания, например). Лучшие детекторы клонируются и мутируют путём дополнительного обучения на вредоносных файлах (антигенах). Авторы статьи не определяют, какие данные нужно подавать на вход нейросетевым детекторам,

а лишь указывают следующее: «На вход такого детектора в режиме функционирования подаются фрагменты проверяемого файла, которые формируются в соответствии с методом скользящего окна».

Как заявляют авторы статьи: «В отличие от известных антивирусных программ, нейросетевая искусственная иммунная система обнаруживает в среднем в 1,5 раза больше неизвестных вредоносных программ».

1.2.2. Искусственная иммунная система на базе статистического метода

Схожий подход, применяемый для построения обучаемых эвристических анализаторов, предложили в своей статье «Модель эвристического анализатора вредоносных программ на основе искусственной иммунной сети» Н.М. Кораблев и М.В. Кушнарев [7].

В данной статье авторы предлагают разбить процесс обучения на два этапа.

Первый этап заключается в выделении признаков с помощью эмуляции каждого исполняемого файла из множества вредоносных и множества чистых файлов. В результате эмуляции, для каждой программы будет получен протокол работы (последовательности вызовов API-функций). Затем специальный модуль циклично сравнивает несколько протоколов и находит в них одинаковые фрагменты (последовательности вызовов функций, общие для сравниваемых протоколов). Эти фрагменты сохраняются в специальную библиотеку. Задача библиотеки – хранить фрагменты и вычислять частоту их встречаемости. Оба множества чистых и вредоносных файлов являются подмножеством множества антигенов.

Второй этап обучения заключается в непосредственном обучении иммунной системы. Каждый протокол работы можно преобразовать в вектор, где i -й компонент этого вектора содержит либо 0, либо частоту встречаемости i -го фрагмента из библиотеки, если этот фрагмент был найден в протоколе.

После этого формируется популяция антител (изначальное состояние случайно), и вычисляются значения аффинности каждого антитела с каждым антигеном. В качестве меры аффинности авторы используют расстояние между

двумя точками в N-мерном пространстве: первая точка описывает состояние антитела, вторая точка описывает состояние антигена, полученное на основе протоколов работы.

Затем, антитела, имеющие наихудшие показатели аффинности уничтожаются (т.е. наиболее удаленные от всех антигенов), а остальные клонируются и подвергаются мутации. Также, авторы упоминают о процессе сжатия сети: в процессе обучения могут быть сформированы два антитела, которые могут находиться достаточно близко друг к другу. Одно из антител должно быть уничтожено. Процесс селекции и мутации происходит циклично, пока не будет достигнуто определенное условие, например, создано определенное количество поколений.

После обучения искусственной иммунной системы происходит определение, какие антитела реагируют на чистые файлы, а какие на вредоносные.

В процессе сканирования исполняемого файла формируется протокол работы программы, выявление в нем известных фрагментов, формирования вектора частот и подача на вход этого вектора иммунной системе. Система вычисляет аффинность антигена для каждого антитела и выносит один из следующих вердиктов: файл вредоносный, файл чистый, не определено.

1.2.3. Эвристический анализ на основе n-грамм

William Arnold и Gerald Tesauro в статье «Automatically generated Win32 Heuristic Virus Detection» предлагают следующий подход [21].

Эмулятор в процессе эмуляции вредоносных и чистых файлов выявляет некоторые признаки (множество возможных признаков может быть задан либо человеком, либо выявлено автоматически). Последовательность найденных признаков разбивается на множество непрерывных групп, каждая группа содержит в себе n признаков, т.е. формируются n -граммы (количество признаков в каждой группе одинаково и равно n). Все выявленные n -граммы сохраняются в библиотеку, которая рассчитывает частоты встречаемости каждой n -граммы. Те

n-граммы, которые имеют частоты встречаемости ниже пороговых, удаляются из библиотеки.

В результате остаются K часто встречающихся n -грамм. Для классификации файлов, авторы предлагают использовать нейросеть с K входами и сигмоидой в качестве функции активации. Если в процессе сканирования файла найдена i -я n -грамма из библиотеки, то на i -й вход нейросети подаётся 1, иначе 0. В результате нейросеть на выходе должна выдать 1, если файл вредоносный, иначе 0.

1.3. Вывод

Вследствие отсутствия открытых программных аналогов, необходимо самостоятельно разработать эвристический анализатор, используя теоретическую базу, приведенную выше.

2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Существующие в ядре антивирусного комплекса механизмы эвристического анализа устарели, так как не могут эффективно применяться в современных условиях увеличивающегося количества полиморфного и обфусцированного ВПО. Разработчикам приходится тратить большое количество времени на обслуживание системы эвристического анализа. Поэтому необходимо разработать замену, которая не будет уступать по качеству обнаружения старой системе, но при этом сократит время обслуживания.

2.1. Основания для разработки

Основанием для разработки системы эвристического анализа программного обеспечения на предмет вредоносного поведения является задание на дипломную работу, выданное Трубаковым А.О., на основании приказа по БГТУ № 418-3 от 27.05.2015 и заявки от ООО «НАНО Секьюрити» (см. приложение В).

2.2. Назначение разработки

Разрабатываемая программная система ориентирована на использование её следующим кругом лиц и программного обеспечения.

1. Подсистема эвристического анализа ориентирована на использование ядром антивирусного комплекса.
2. Подсистема обучения ориентирована на использование вирусными аналитиками и системными программистами в целях экономии рабочего времени и усилий на настройку анализаторов, а также для помощи в анализе вредоносного ПО.
3. Аналитический веб-сервис ориентирован на использование вирусными аналитиками для разработки различных экспериментальных решений и прототипов.

2.3. Общие требования

Выдвигается ключевое требование к новой системе – наличие возможности обучения.

Вся система эвристического анализа должна состоять из двух ключевых подсистем: подсистемы анализа и подсистемы обучения. Обе подсистемы должны включать в себя модуль нормализации входных данных и модуль классификации. Подсистема обучения также должна включать в себя модуль исследования зависимостей между признаками.

Система должна быть доступна для вирусных аналитиков в виде аналитического веб-сервиса.

2.4. Функциональные требования

Функционально можно выделить 3 группы: подсистема обучения, подсистема анализа и аналитическая веб-служба. Каждую функциональную группу должен реализовывать отдельный разработчик.

2.4.1. Требования к подсистеме обучения

Подсистема обучения должна включать в себя следующие утилиты: сканер антивируса, утилиту исследования зависимостей, утилиту обучения.

Сканер антивируса, необходим для формирования множества журналов признаков. *Журнал признаков* – это текстовый файл, содержащий список признаков в той последовательности, в которой они были найдены при сканировании. Для каждого сканируемого эвристическим методом объекта формируется свой журнал признаков.

Утилита исследования зависимостей нужна для выявления статистически наиболее информативных признаков и их парных комбинаций. Результатом работы утилиты должна являться *библиотека комбинаций* – текстовый файл, в котором построчно записаны пары или одиночные признаки. Наличие файла необходимо для его чтения и изменения экспертом.

Утилита обучения должна определять настройки модуля нормализации данных и настройки модуля классификации. На вход утилите подаётся информация о местонахождении журналов признаков (обучающие и тестовые) и библиотеки комбинаций. Результатом работы являются файл состояния нормализатора, файлы состояния классификатора после каждой эпохи обучения и результирующий отчёт об обучении.

2.4.2. Требования к подсистеме эвристического анализа

Подсистема должна быть реализована в виде СОМ-компонента, выполняющего следующие этапы анализа.

1. Накопление признаков для формирования журналов в процессе сканирования файла.
2. Поиск известных комбинаций, определенных в библиотеке, и формирование ненормализованного входного вектора.
3. Нормализация входного вектора.
4. Загрузка нормализованного входного вектора в классификатор и определение класса сканируемого объекта.

В соответствии с существующими механизмами, компонент должен иметь возможность загружать своё состояние из базы данных антивируса.

Компонент должен определять новый интерфейс анализа, содержащий следующие методы:

1. Метод, принимающий в качестве параметра текстовое имя признака. Задача метода – добавление заданного признака к журналу.
2. Метод, выполняющий анализ и возвращающий результат классификации в виде перечисляемого типа. Три возможных значения типа должны по смыслу отражать следующие результаты анализа:
 - а) объект безвреден;
 - б) объект вредоносен;
 - в) результат не определен.

2.4.3. Требование к аналитическому веб-сервису

Аналитический веб-сервис должен снять вычислительно дорогостоящую операцию обучения анализатора с компьютера вирусного аналитика и позволить ему работать одновременно над множеством различных вариаций анализатора. Поэтому выдвигаются следующие требования:

1. Веб-сервис должен, используя предоставленный аналитиком прототип сканера и списки хеш-сумм файлов, сформировать журналы признаков. У аналитика должна иметься возможность сохранить журналы.
2. Веб-сервис должен обучать анализатор, используя заданные аналитиком параметры.
3. Вирусный аналитик должен иметь возможность сохранить полученное состояние в виде локальной копии доступной для модификации.
4. Веб-сервис должен предоставлять программный интерфейс для использования и обучения.

2.5. Требования к архитектуре и математическому аппарату

Выдвигается важное требование к архитектуре системы – наличие гибкой возможности замены модуля анализа. Это необходимо для дальнейшего развития математического аппарата без существенных изменений в архитектуре антивирусного ядра.

2.6. Требование к аппаратному и программному обеспечению

Разработанное программное обеспечение должно работать под управлением операционной системы Windows 7 на компьютерах со следующей конфигурацией.

1. Процессор 1,2 ГГц и выше (рекомендуется 2 ГГц и выше).
2. ОЗУ 1 Гб (рекомендуется 2 Гб и выше).
3. 2 Гб свободного места на системном диске.

3. ИССЛЕДОВАНИЕ АЛГОРИТМОВ КЛАССИФИКАЦИИ

В данной главе приведены результаты исследования различных алгоритмов классификации исходя из поставленной задачи.

Цель исследования – выбор механизма классификации ПО с точки зрения вредоносного поведения.

Задачи исследования:

- а) формирование списка исследуемых классификаторов;
- б) сравнение классификаторов и выбор наилучшего.

3.1. Описание классификаторов

На данном этапе развития проекта целесообразно оперировать простыми классификаторами. В качестве альтернатив рассматривались следующие алгоритмы:

- а) многослойная нейронная сеть;
- б) рекуррентная нейронная сеть;
- в) дерево решений ID3.

3.1.1. Многослойная нейронная сеть

Нейронные сети состоят из узлов, или элементов, соединенных направленными связями. Связь от элемента j к элементу i служит для распространения активации a_j от j к i . Кроме того, каждая связь имеет назначенный ей числовой вес $W_{j,i}$, который определяет силу и знак связи. Каждый элемент i прежде всего вычисляет взвешенную сумму своих входных данных. Затем он применяет к этой сумме функцию активации g , чтобы определить, какими должны быть выходные данные.

Преимущество введения скрытых слоев состоит в том, что они приводят к расширению пространства гипотез, которые могут быть представлены сетью. Каждый скрытый элемент можно рассматривать как персептрон, который представляет мягкую пороговую функцию в пространстве входов. В таком случае

любой выходной элемент должен рассматриваться как средство создания линейной комбинации нескольких таких функций с мягким порогом [16].

Пример нейросети см. на рис. 3.1.

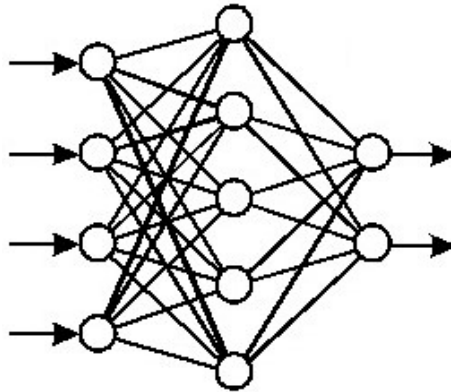


Рис. 3.1. Многослойная нейронная сеть с одним скрытым слоем

3.1.2. Рекуррентная нейронная сеть

Рекуррентная сеть подает свои выходные данные обратно на свои собственные входы. Это означает, что уровни активации сети образуют динамическую систему, которая может достигать устойчивого состояния, или переходить в колебательный режим, или даже проявлять хаотичное поведение. Более того, отклик сети на конкретные входные данные зависит от ее начального состояния, которое, в свою очередь, может зависеть от предыдущих входных данных. Поэтому рекуррентные сети (в отличие от сетей с прямым распространением) могут моделировать кратковременную память [16]. Пример рекуррентной нейронной сети см. на рис. 3.2.

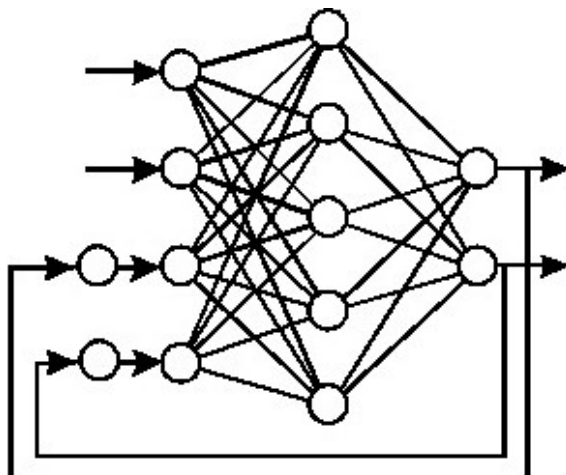


Рис. 3.2. Пример рекуррентной нейросети

3.1.3. Дерево решений ID3

Алгоритм построения деревьев ID3 – один из наиболее ранних алгоритмов обучения деревьев решений, использующих рекурсивное разбиение подмножеств в узлах дерева по одному из выбранных атрибутов. Начинает работу с корня дерева, в котором содержатся все примеры обучающего множества. Для разделения в нем выбирается один из атрибутов, и для каждого принимаемого им значения строится ветвь, и создается дочерний узел, в который распределяются все содержащие его записи.

Процедура повторяется рекурсивно до тех пор, пока в узлах не останутся только примеры одного класса, после чего они будут объявлены листьями и ветвление прекратится. Наиболее проблемным этапом здесь является выбор атрибута, по которому будет производиться разбиение. Классический алгоритм ID3 использует для этого критерий увеличения информации или уменьшения энтропии.

Пример дерева решений ID3, с помощью которого можно классифицировать риск выдачи кредита заёмщику, см. на рис. 3.3.

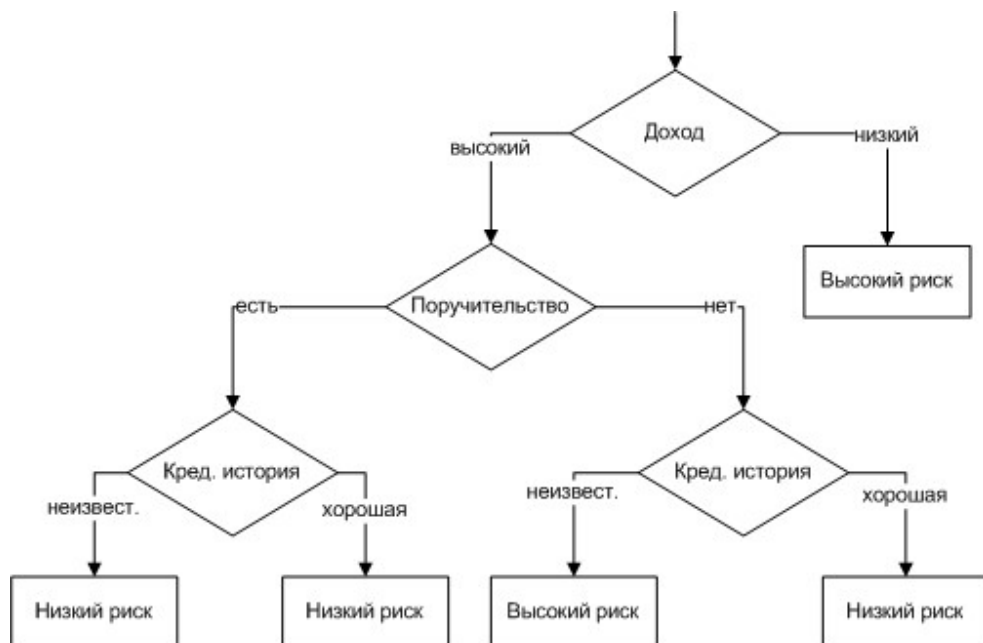


Рис. 3.3. Пример дерева решений ID3

Практическое применение классической реализации ID3 сталкивается с рядом проблем, характерных для моделей, основанных на обучении вообще и деревьев решений в частности. Основными из них являются переобучение

и наличие пропусков в данных. Поэтому алгоритм был усовершенствован, в результате чего появилась его новая версия C4.5. Она позволяет работать с пропущенными значениями признаков, а также имеет меньшую склонность к переобучению [1].

3.2. Оценка классификаторов

Результат оценки каждого классификатора приведен в табл. 3.1.

Таблица 3.1

Оценка различных классификаторов по критериям

Классификатор	Критерии		
	Определенность структуры	Чувствительность к шумам при обучении	Интерпретируемость состояния
Многослойная нейронная сеть	Определена, но может уточняться экспериментально	Низкая	Нет
Рекуррентная нейронная сеть	Определена, но может уточняться экспериментально	Низкая	Нет
Дерево ID3	Является результатом обучения	Высокая	Да

Исходя из вышеописанных оценок, очень сложно определить с классификатором, так как альтернативы несравнимы по Парето.

Рекуррентная нейросеть хороша тем, что по своему дизайну способна принимать последовательности признаков и выявлять характерные зависимости. Остальным же классификаторам приходится дополнительно подавать признаки о наличии той или иной зависимости. Недостаток такого классификатора в том, что исследователь не сможет определить, какие характерные зависимости персептрон выявил для чистого или вредоносного класса. К тому же не известно, сможет ли такая нейросеть работать с большими выборками, в которых высоко разнообразие семейств ВПО.

Дерево ID3 в процессе обучения плохо справляется с шумами и аномалиями. Из-за этого оно начинает сильно ветвиться и разрастаться в размерах, просто запоминая аномальные примеры.

Многослойная нейросеть также не лишена недостатков. Этот классификатор, так же как и рекуррентная нейросеть, может плохо обучиться на выборке, в которой содержится большое количество семейств ВПО. Фактически такому персептрону придётся сначала определить принадлежность объекта к семейству, а потом уже независимо от семейства классифицировать объект как вредоносный.

3.3. Вывод

Принято решение использовать в данной работе многослойную нейронную сеть, потому что по предварительным оценкам она не уступает другим классификаторам.

В дальнейших этапах развития проекта планируется экспериментально оценить качество остальных классификаторов, чтобы составить более точную оценку и обоснованно развивать подходящий математический аппарат.

4. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА

В данной главе в разделе высокоуровневого проектирования представлены модели ТО-ВЕ. В разделе низкоуровневого проектирования определено детальное представление подсистем, а также выделены роли исполнителей, которые будут реализовывать эти системы.

4.1. Высокоуровневое проектирование

С помощью диаграмм прецедентов и моделей потоков данных определим основные алгоритмы работы системы. С помощью диаграммы размещения определим, где и какие части системы должны функционировать.

4.1.1. Прецеденты взаимодействия персонала с системой

На диаграмме прецедентов представлены будущие сценарии работы персонала с системой эвристического анализа (см. рис. 4.1).

Вирусный аналитик, исследуя ранее не изученное семейство ВПО, выявляет характерные для этого семейства признаки и документирует их. Разработчик дорабатывает соответствующий сканер ядра таким образом, чтобы он начал выявлять новые признаки и передавать их эвристическому анализатору. Далее разработчик полностью переучивает анализатор.

В случае если анализатор выносит ложноположительные или ложноотрицательные вердикты, контроль над дообучением должен взять на себя разработчик.

4.1.2. Прецеденты взаимодействия сущностей ядра

Рассмотрим сценарии работы внутренних сущностей ядра (см. рис. 4.2).

Задача менеджера сканирования – управлять процессом сканирования. Он управляет очередью объектов сканирования, определяет форматы потоков данных (с помощью детектора формата), создаёт соответствующие сканеры и формирует отчёт пользователю.

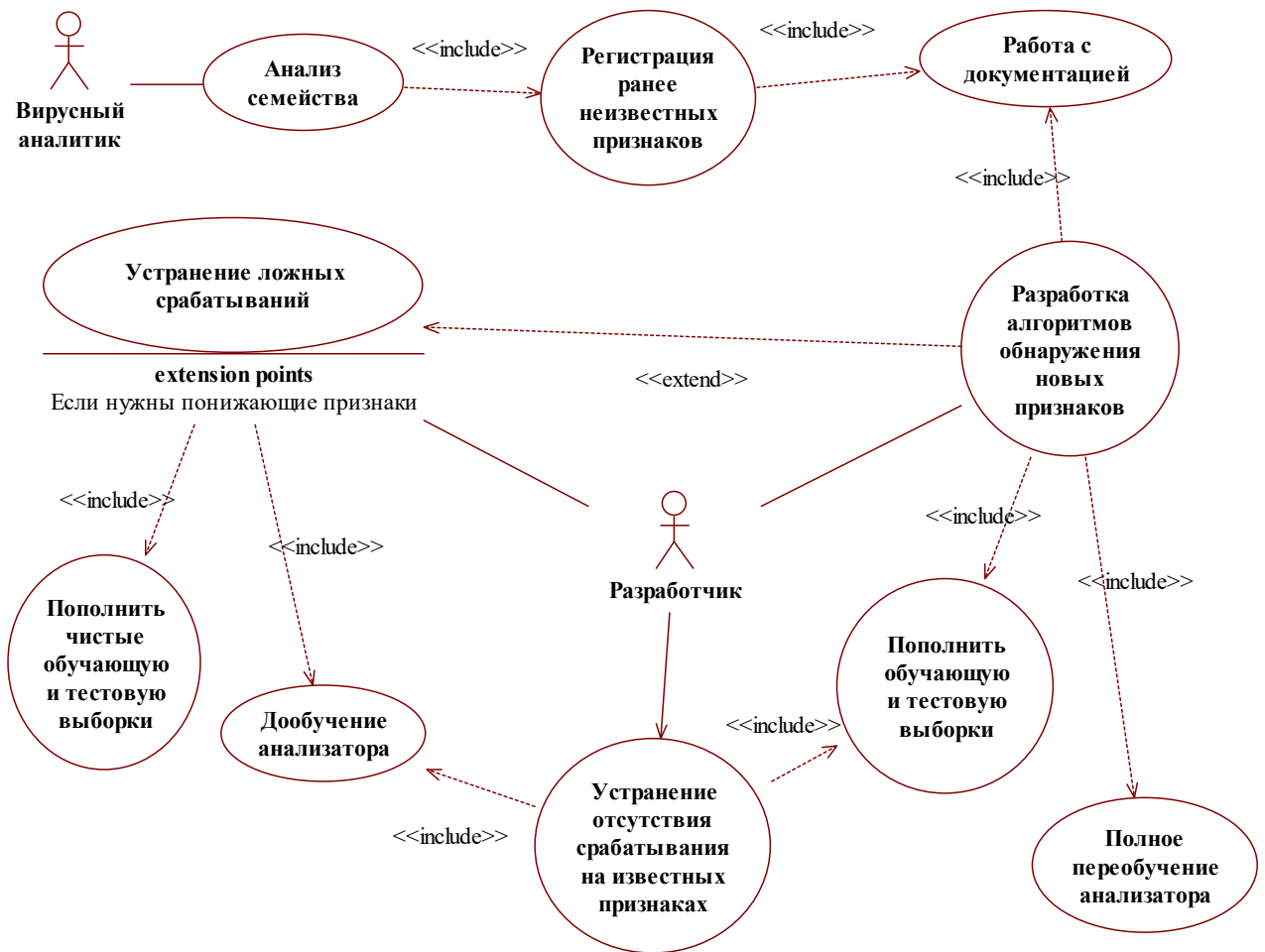


Рис. 4.1. Диаграмма прецедентов, моделирующая взаимодействие персонала с системой.

4.1.3. Архитектура инфраструктуры

Рассмотрим общее представление системы с точки зрения взаимодействия вычислительных узлов (см. рис. 4.3). Серым прямоугольником выделены узлы, представляющие интерес для данной работы.

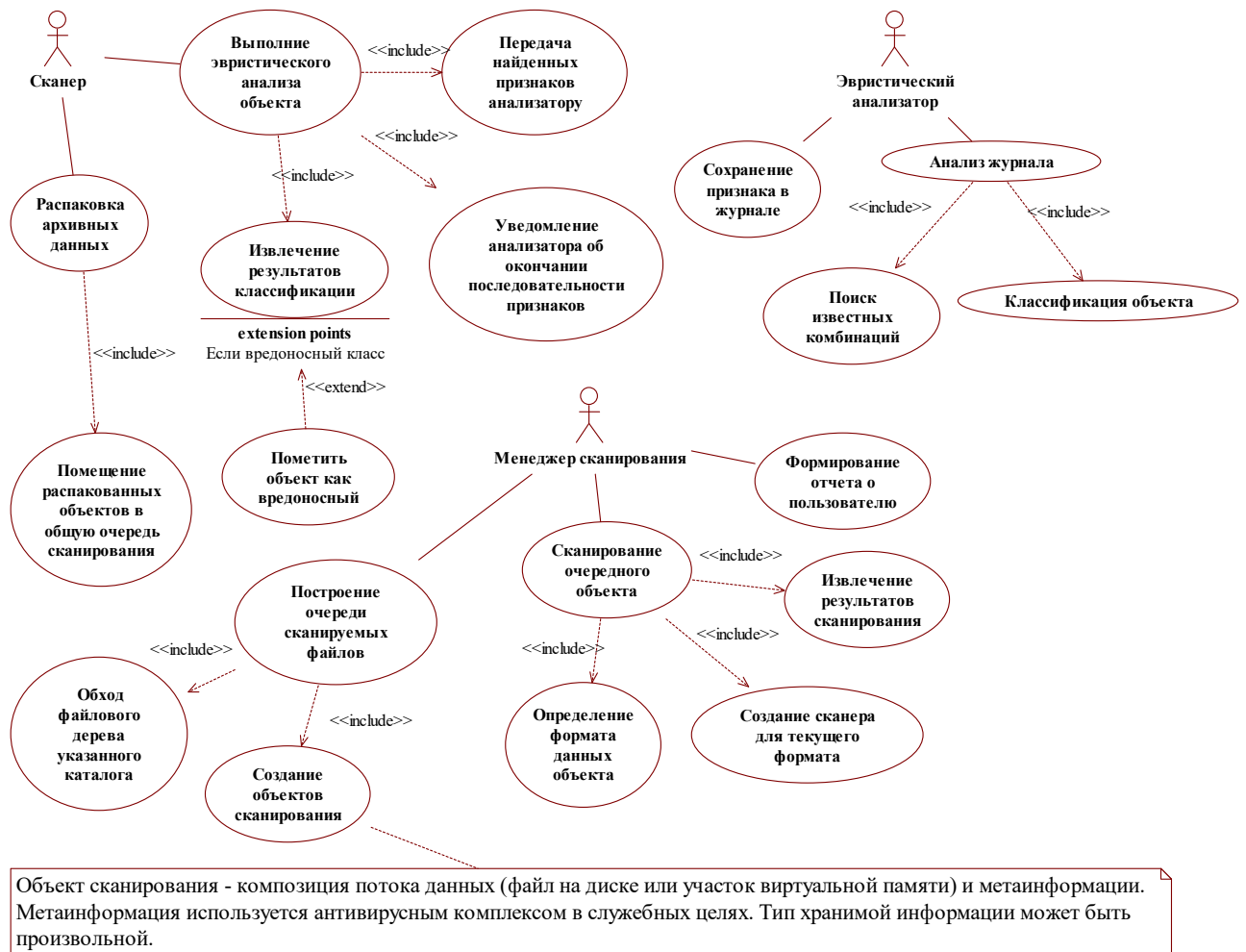


Рис. 4.2. Диаграмма прецедентов, моделирующая отношения объектов ядра

Хранилище сигнатур и контрольных сумм – сервер, куда в автоматическом режиме из разных источников попадают контрольные суммы файлов и сигнатур различных участков вредоносных файлов. Эти данные участвуют в простых системах анализа программного обеспечения.

В задачи сервера компиляции базы данных антивируса входит компоновка и упорядочивание массива сигнатур, контрольных сумм и различных настроек, взятых из репозитория, в сжатый файл базы данных. Также в задачи сервера входит формирование небольших файлов обновлений базы данных.

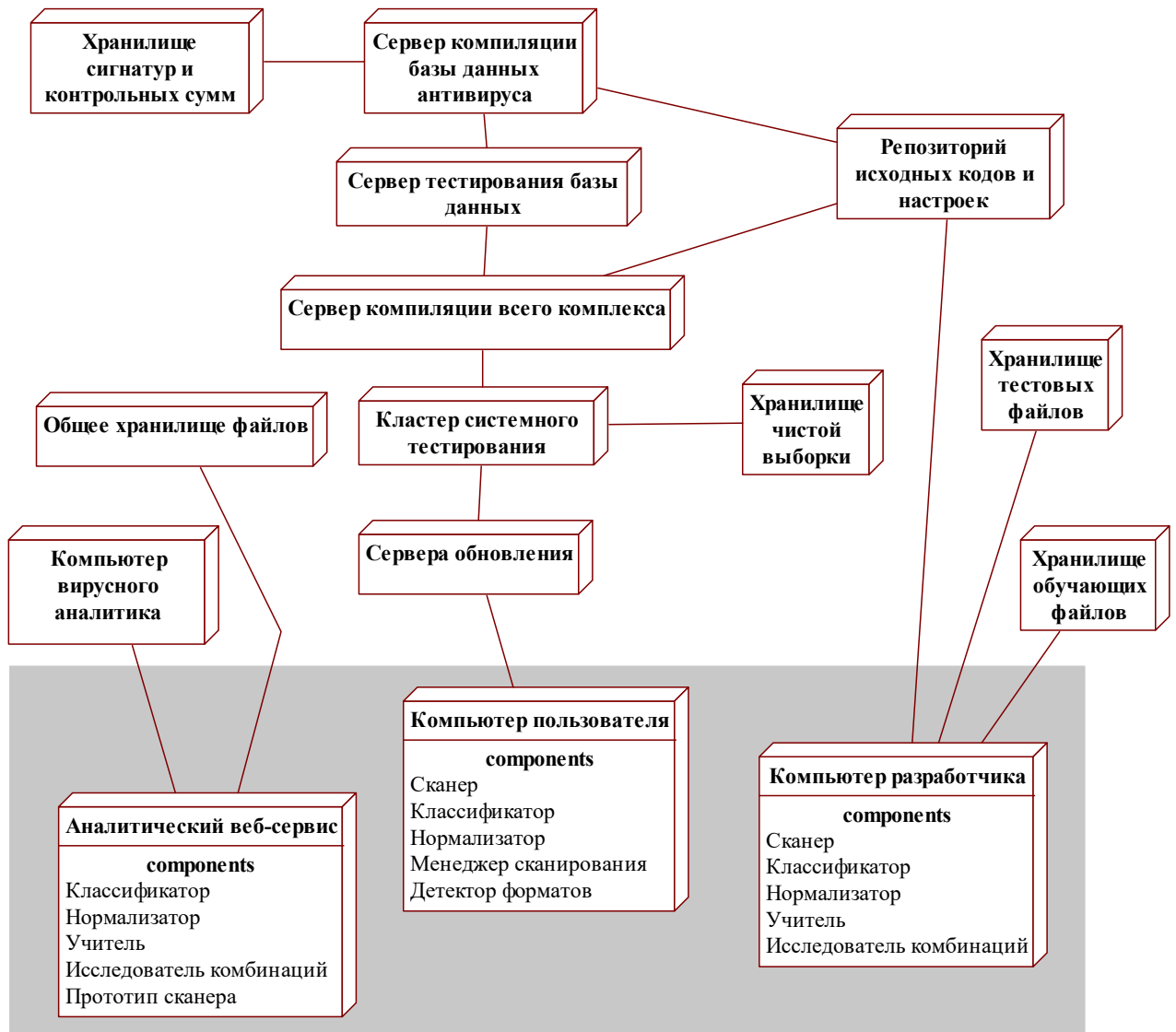


Рис. 4.3. Диаграмма развёртывания, моделирующая общий вид системы

Сервер тестирования базы данных следит за тем, чтобы в случае изменения настроек некоторых сканеров, не ухудшилось качество обнаружения небольшого множества вредоносных файлов, анализируемых этим сканером.

Сервер компиляции всего комплекса собирает ядро антивируса, драйвера, службу, графический и консольный клиенты к службе и формирует инсталлятор и деинсталлятор, а также формирует файлы обновления исполняемых файлов.

Кластер системного тестирования занимается автоматическим тестированием драйверов и службы. В этом кластере каждую ночь запускается сканирование всей коллекции чистых файлов, которое длится приблизительно 12 часов.

Сервера обновления – это удалённые машины в дата-центре с распределителем нагрузок и широкополосными соединениями к сети Интернет. На этих серверах расположены свежие версии инсталлятора, файл базы данных, файлы обновления базы данных, файлы обновления исполняемых файлов.

Чёрной рамкой выделены узлы, для которых разрабатываются модули системы эвристического анализа.

На компьютере разработчика будет проходить полное обучение и предварительное тестирование анализатора. Полное тестирование будет происходить в кластере системного тестирования.

Если выборки файлов достаточно большие, допускается, что они могут находиться на удалённых машинах.

На компьютере пользователя будет эксплуатироваться обученный анализатор в связке со сканером, обнаруживающим признаки.

Вирусные аналитики могут использовать специальную веб-службу через программный и веб-интерфейс в своих целях.

Общее хранилище файлов – сервер, позволяющий через программный интерфейс получить файл по его хеш-сумме, при условии его наличия в хранилище.

4.2. Низкоуровневое проектирование

Прежде чем приступить к детальному проектированию подсистем, необходимо определить зоны ответственности. Каждый разработчик будет реализовывать свою подсистему, поэтому для каждого из них будут определены свои проектные решения.

4.2.1. Роли разработчиков

Для реализации проекта необходимы следующие исполнители:

- а) разработчик подсистемы обучения;
- б) разработчик ядра антивируса;
- в) разработчик внутренней инфраструктуры.

Задача разработчика подсистемы обучения – реализовать математический аппарат в виде модулей исследования комбинаций, нормализации, классификации и, конечно, подсистемы обучения.

Задача разработчика ядра – интегрировать модули нормализации данных и классификации в ядро антивируса в виде СОМ-компонента, а также обеспечить доступ модулям к обученному состоянию в базе данных антивируса.

Задача разработчика внутренней инфраструктуры – интегрировать подсистемы анализа и обучения в аналитические веб-сервисы и разработать для них программный и веб-интерфейс. Эти веб-сервисы будут в дальнейшем использоваться вирусными аналитиками в экспериментальных проектах анализа.

4.2.2. Структурное проектирование

Рассмотрим проект подсистемы обучения, которую будет реализовывать соответствующий разработчик. На рис. 4.4 представлена диаграмма компонентов.

Компонент «Сканер» – уже существующий компонент, задача которого состоит в формировании журнала признаков. «Сканер» запускается при необходимости «Учителем» для формирования признаков.

Разработчик системы обучения должен разработать компоненты «Учитель», «Нормализатор» и «Классификатор» и предоставить их для интеграции разработчику ядра. Роль исследователя комбинаций выполняют класс *CBinaryMatrix*. Класс *CLibrary* отвечает за поиск известных комбинаций в журнале. Роль «Учителя» выполняет набор утилит, манипулирующий остальными сущностями. Детальное представление остальных компонентов в виде классов представлено на рис. 4.5.

Теперь рассмотрим проект подсистемы анализа, которую будет реализовывать разработчик ядра. На рис. 4.6 и рис. 4.7 приведены диаграммы, моделирующие подсистемы. На рис. 4.7 серой рамкой выделены сущности, подлежащие реализации.

Компонент «Эвристический анализатор» во время инициализации будет загружать сформированное учителем состояние из базы данных антивируса и в

процессе работы получать последовательность признаков. В конце последовательности, компонент должен отнести сканируемый объект к какому-либо классу.

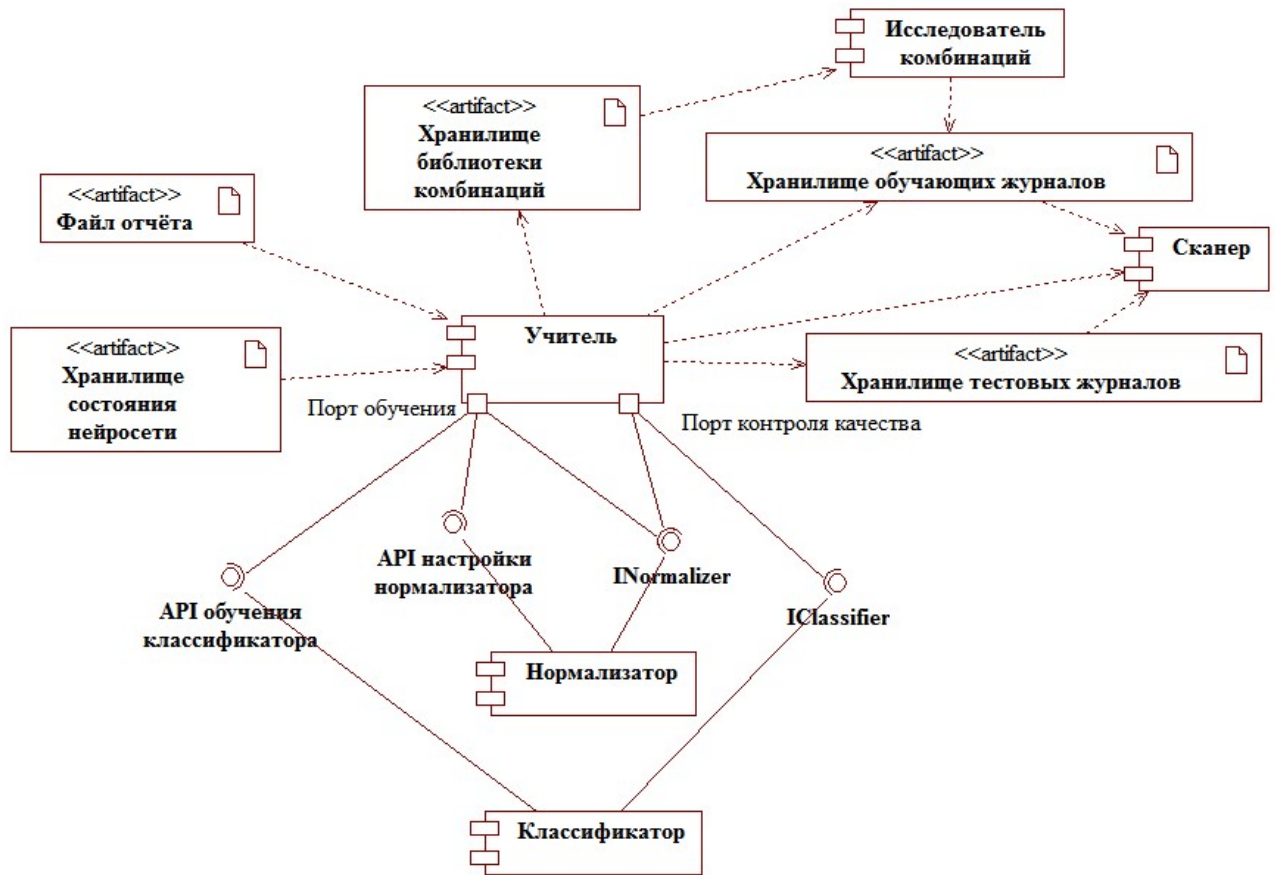


Рис. 4.4. Связи между компонентами в процессе обучения

Как отмечалось выше, вирусные аналитики могут воспользоваться системой классификации в исследовательских и экспериментальных целях. Согласно высокоуровневому дизайну (см. рис. 4.8), аналитик может либо предоставить службе уже сформированные журналы признаков и экспериментировать с конфигурацией модулей системы или предоставить прототип сканера со списком контрольных сумм файлов. Предоставление прототипа сканера поможет снять вычислительную нагрузку с машины аналитика и перенести её на сервер.

Полученное состояние может быть сохранено в файл настроек и впоследствии передано аналитической подсистеме, которая будет использовать веб-службу для своих задач.

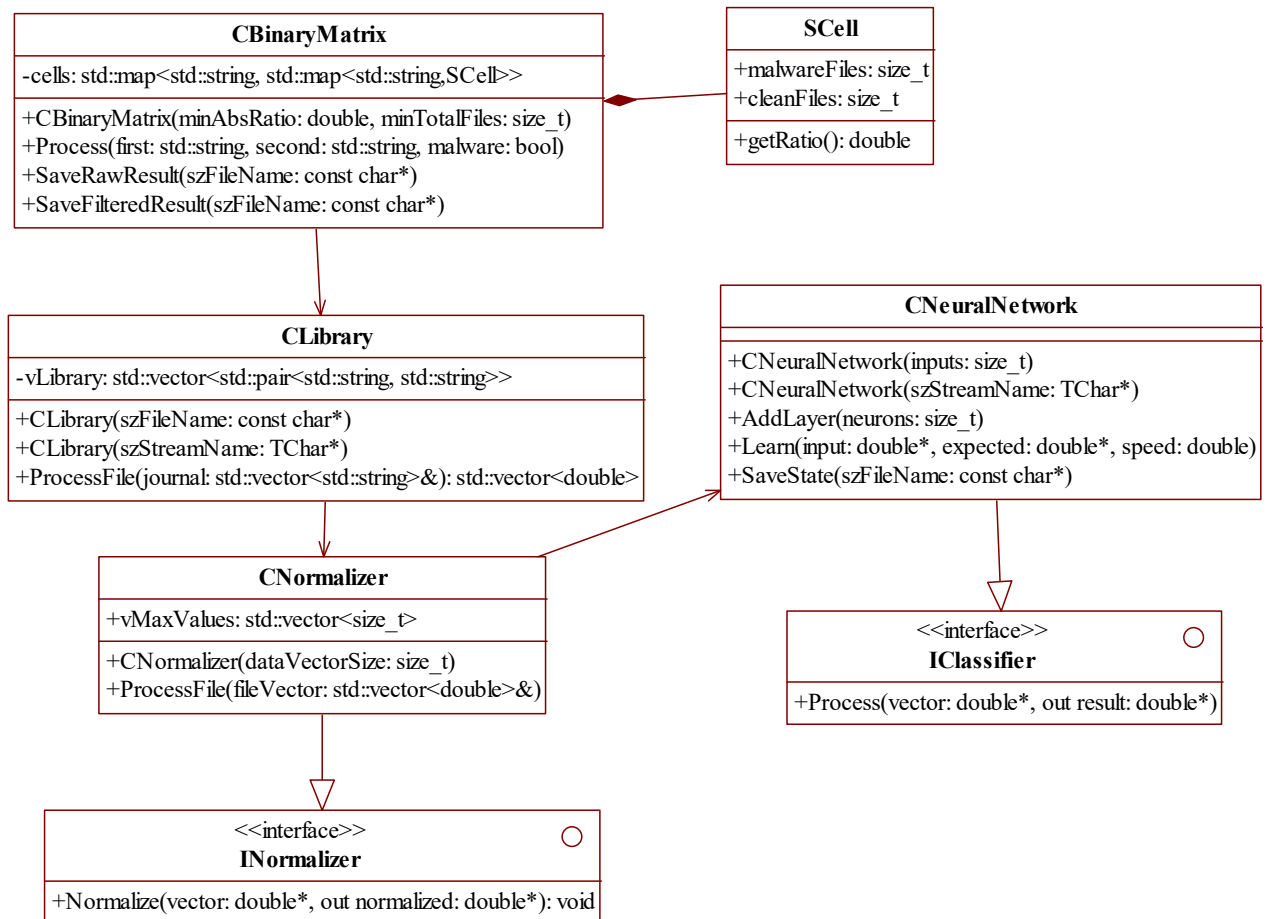


Рис. 4.5. Классы, которые необходимо реализовать разработчику подсистемы обучения

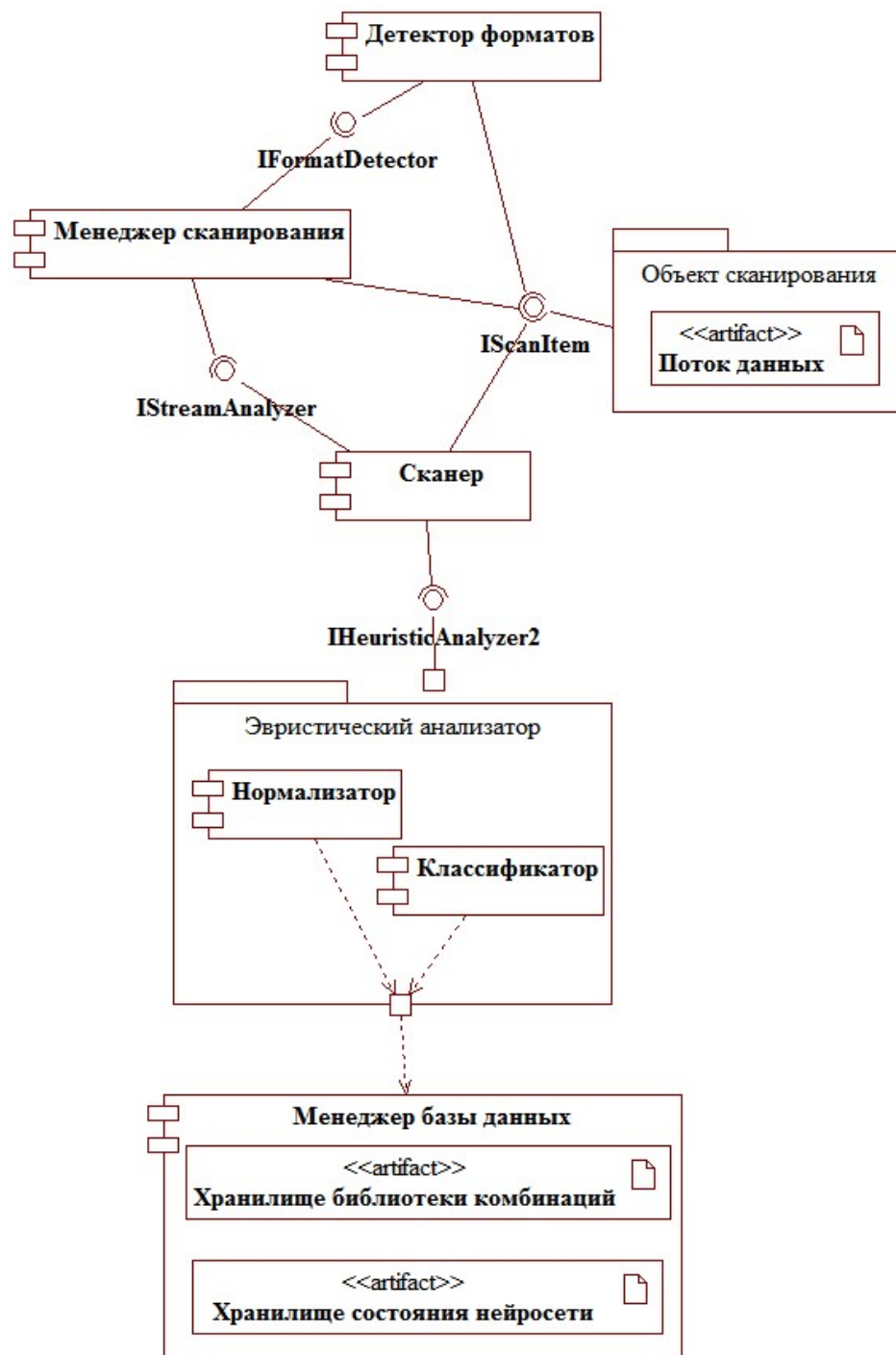


Рис. 4.6. Использование эвристического анализатора

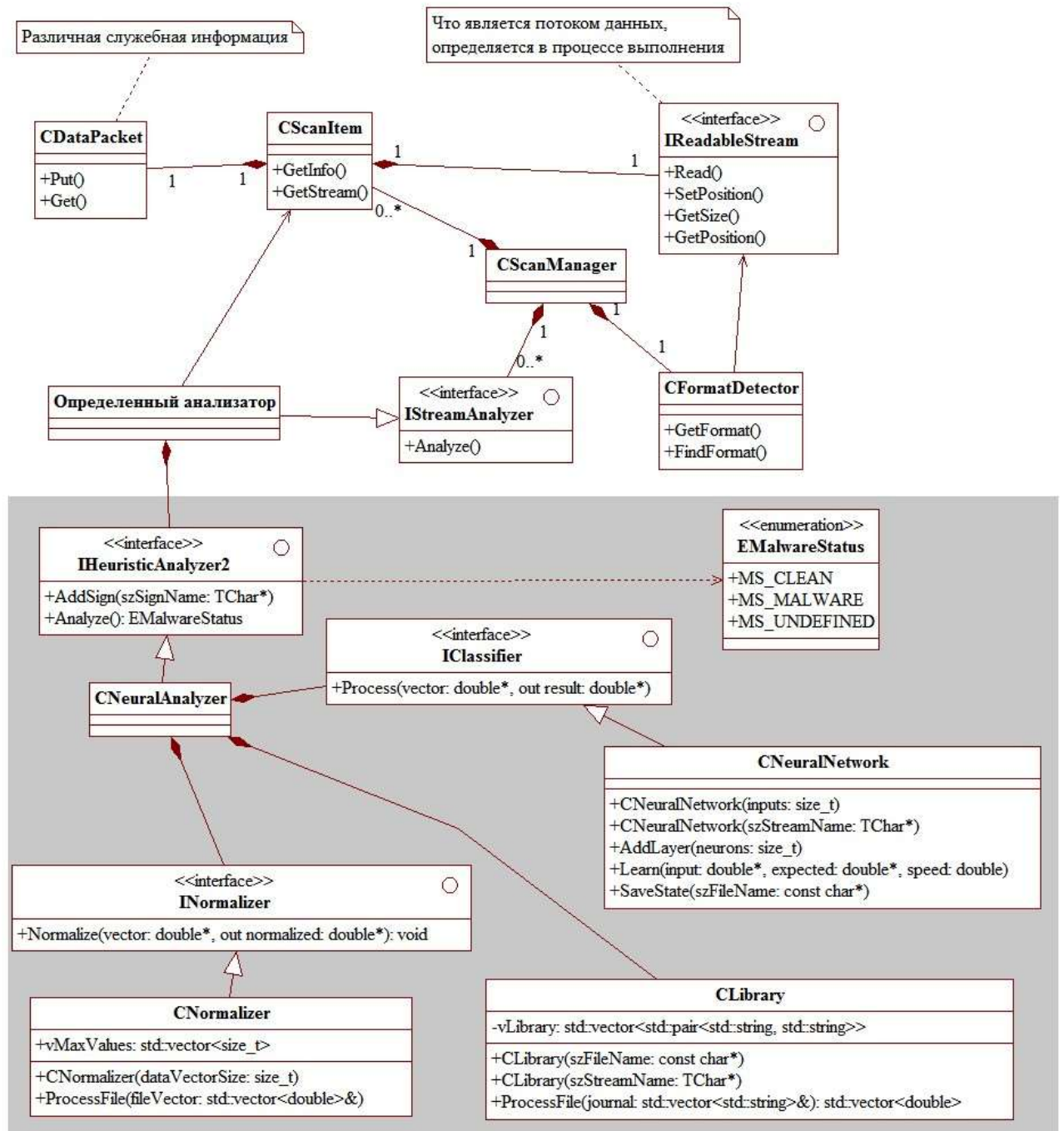


Рис. 4.7. Диаграмма классов подсистемы анализа

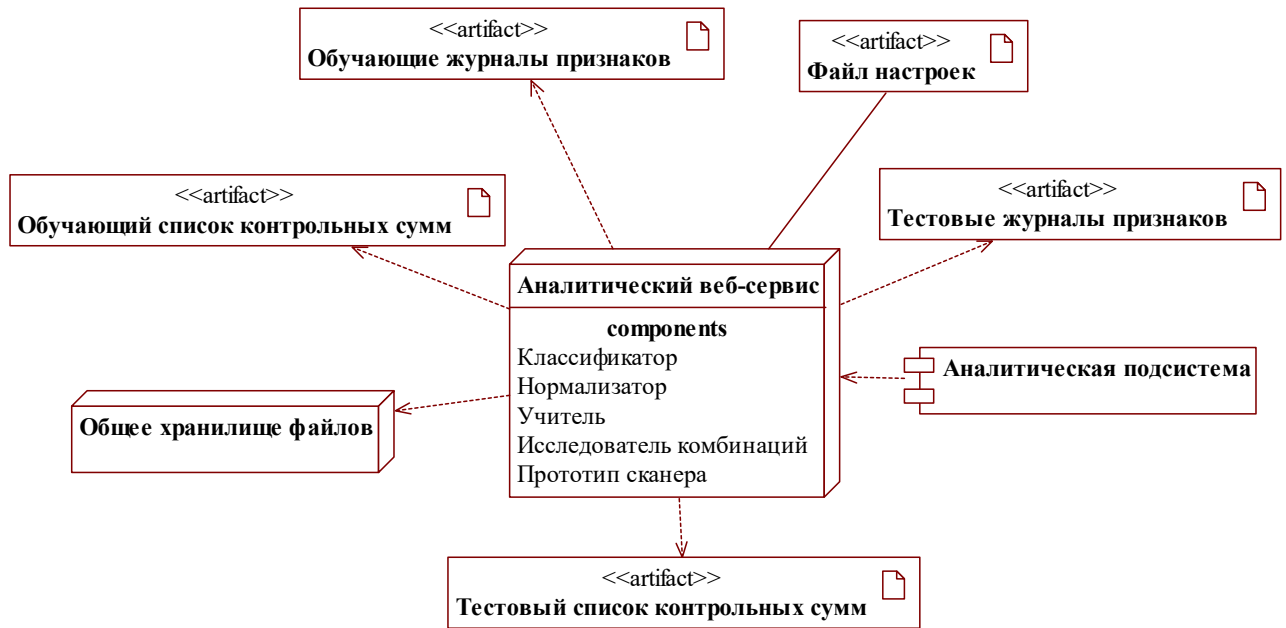


Рис. 4.8. Архитектура аналитического веб-сервиса

4.3. Математический аппарат

В данном разделе рассматриваются два математических инструмента: матрица накоплений для выявления информативных пар признаков и нейросеть как классификатор.

4.3.1. Матрица накоплений

Имея два множества журналов чистых и вредоносных файлов в обучающей выборке, можно статистически выявить наиболее информативные комбинации признаков с помощью n -мерной матрицы накоплений. Такая матрица может выявлять комбинации, содержащие от 1 до n признаков. В данной работе используется двумерная матрица.

Индексация столбцов и строк в матрице осуществляется по текстовому имени признака. Ячейка на пересечении столбцов и строк хранит два счётчика – количество чистых и вредоносных файлов, содержащие комбинацию $[i, j]$.

Комбинации выявляются по алгоритму, приведенном в листинге 1.

```

матрица.НачалоФайла (журнал.чистый_или_вредоносный) ;
для i от 0 до журнал.размер
начало
    матрица[журнал[i]][""].Задать() ;
    для j от i+1 до журнал.размер
        матрица[журнал[i]][журнал[j]].Задать() ;
конец
матрица.КонецФайла() ;

```

Листинг 1. Алгоритм поиска всех комбинаций

Журнал признаков хранит в поле *чистый_или_вредоносный* информацию о принадлежности файла к классу чистых или вредоносных файлов.

Метод *Задать()* уведомляет ячейку матрицы о факте присутствия комбинации в журнале признаков и инкрементирует соответствующий счётчик. Если комбинация встретится повторно, то метод просто проигнорирует комбинацию и счётчик для текущего файла не увеличится.

После анализа всей выборки чистых и вредоносных журналов для каждой комбинации можно вычислить отношение по формуле:

$$ratio = (malware - clean) / (malware + clean + 1),$$

где *malware* и *clean* – количество вредоносных и чистых файлов соответственно, в которых встретила эта комбинация.

Это отношение стремиться к -1, если комбинация свойственна исключительно чистым файлам, к +1, если свойственна исключительно вредоносным, и к 0, если комбинация свойственна обоим классам.

В зависимости от различных внешних условий, исследователь может задать различные параметры фильтрации для отсеивания неинформативных комбинаций. Например, исследователь, может задать следующие параметры фильтра:

$$clean + malware \geq 10000 \text{ и } abs(ratio) \geq 0.85.$$

4.3.2. Искусственная нейронная сеть

По результатам исследования принято решения в данной работе использовать классический многослойный персептрон с сигмоидной функцией активации.

Количество входов нейросети зависит от количества найденных информативных комбинаций, потому что очередной вход ассоциируется с очередной комбинацией. В процессе обучения или анализа на очередной вход нейросети подаётся нормализованное в диапазоне от 0 до 1 количество срабатываний текущей комбинации, найденной в данном журнале. Нормализация входного вектора осуществляется по алгоритму минимаксной нормализации.

Количество выходов нейросети ровно два, каждый из которых определяет принадлежность образа к текущему классу. Ожидается, что если входной вектор соответствует чистому файлу, то выходной вектор будет иметь вид (1; 0), иначе (0; 1).

Стоит отметить, что если в процессе анализа нейросеть не может достаточно точно определить класс объекта (разница значений на выходах меньше 1/3), то объект автоматически причисляется к чистому классу.

4.4. Выбор средств разработки

Единственно возможный вариант выбора языка для разработки компонентов анализа – это язык C++, потому что компоненты должны быть интегрированы в ядро антивирусного комплекса. Само ядро антивируса разработано с использованием этого языка.

Утилиты подсистемы также должны быть разработаны с использованием этого языка, так как это позволит напрямую использовать компоненты анализа.

В качестве среды разработки системного программного обеспечения используется Visual Studio 2013 Community edition, в связи с установленными требованиями компании.

4.5. Руководство по использованию утилит обучения

В данном разделе приведено руководство по использованию утилит обучения *librarian* и *teacher*.

4.5.1. Утилита *librarian*

Задача утилиты *librarian* – построение библиотеки комбинаций признаков с помощью матрицы накоплений. Утилита анализирует каждый журнал в указанных каталогах. Допустимые параметры приведены в табл. 4.1.

Таблица 4.1

Параметры командой строки утилиты *librarian*

Параметр	Значение по умолчанию	Описание
--malware	learn\malware	Задаёт директорию с вредоносными журналами
--clean	learn\clean	Задаёт директорию с чистыми журналами

Результаты общего анализа выводятся на стандартный поток вывода в виде формата csv. Каждая строка таблицы содержит следующие атрибуты: названия первого признака, название второго признака (в т.ч. пустое), количество чистых файлов с этой комбинацией, количество вредоносных файлов, общее число файлов, отношение *ratio* (см. пункт 4.3.1), абсолютное значение отношения *ratio*.

4.5.2. Утилита *teacher*

Задача утилиты *teacher* – эпохальное обучение нейросети. Допустимые параметры приведены в табл. 4.2.

Таблица 4.2

Параметры командной строки утилиты *teacher*

Параметр	Значение по умолчанию	Описание
--library	library	Задаёт имя файла библиотеки
--learn-clean	learn\clean	Задаёт директорию с чистыми обучающими журналами
--learn-malware	learn\malware	Задаёт директорию с вредоносными обучающими журналами
--test-clean	test\clean	Задаёт директорию с чистыми тестовыми журналами
--test-malware	test\malware	Задаёт директорию с вредоносными тестовыми журналами
--normalizer	normalizer.bin	Задаёт имя файла с результирующим состоянием нормализатора
--epochs	10	Количество эпох обучения
--cleans	2.0/3.0	Вероятность подачи чистого журнала в процессе обучения
--iterations	150000	Количество итераций обучения в одной эпохе
--neural-out	neural.bin	Префикс имени файла с результирующим состоянием нейросети
--neural-config	default.cfg	Имя файла с архитектурой нейросети

Результатом работы утилиты будет файл с состоянием нормализатора и множество файлов с состояниями нейросети после каждой эпохи обучения. Также утилита выведет в стандартный поток результаты тестирования после каждой эпохи обучения. В тестировании используются журналы из тестовой и обучающей выборок. В поток выведется результат классификации каждого файла, а также количества верных, ложно-положительных, ложно-отрицательных, неопределенных на чистых, неопределенно на вредоносных классификаций.

5. ЭКОНОМИЧЕСКИЙ АНАЛИЗ

В данной главе рассматриваются вопросы организации работ по созданию и внедрению программной системы, а также приводится расчёт ее себестоимости.

5.1. Организационная структура проекта

Организационная структура проекта приведена на рис. 5.1.



Рис. 5.1. Организационная структура проекта

5.2. Календарный план проекта

Продолжительность всего проекта занимает 5 недель или 25 рабочих дней. В табл. 5.1 приведен план работ проекта. Полный план проекта с развернутой диаграммой Ганта приведен в приложении А.

На рис. 5.2 приведено сокращенное представление диаграммы Ганта, на которой часть задач сгруппировано.

Исходя из длительности работ и коэффициента загрузки членов проектной команды, определим их трудозатраты при реализации проекта (см. табл. 5.2).

План работ проекта

№	Название задачи	Длительность	Предшественник	Исполнитель
1	Разработка технического задания	0,5 дней		Начальник отдела
2	Проектирование системы	3 дней		Разработчик подсистемы обучения
3	Проектирование процессов	3 дней		
4	Исследование классификаторов	3 дней		
5	Построение проекта системы	3 дней		
6	Подготовка выборок	2,5 дней		Разработчик ядра
7	Формирование выборок вредоносного и чистого ПО	1 день		
8	Формирование обучающих и тестовых журналов на основе уже существующих признаков	1,5 дней	6	
9	Реализация математического аппарата	6 дней	5;1	Разработчик подсистемы обучения
10	Реализация утилит и модулей подсистемы обучения	5 дней		
11	Тестирование системы обучения и исправление ошибок	1 день	9	
12	Интеграция подсистем в ядро	2 дней	8	Разработчик ядра
13	Тестирование ядра	2 дней	11	Разработчик ядра[50%]; Специалист отдела тестирования[50%]
14	Интеграционное тестирование	4 дней	12	
15	Контроль автоматических тестов	4 дней		Специалист отдела тестирования[25%]
16	Исправление ошибок классификации	4 дней		Разработчик подсистемы обучения[25%]
17	Исправление ошибок ядра	4 дней		Разработчик ядра[25%]
18	Оформление документации	0,5 дней	13	Разработчик ядра
19	Интеграция подсистем в аналитическую веб-службу	5 дней	17	
20	Разработка веб-интерфейса	1 день		Разработчик инфраструктуры
21	Интеграция подсистем в веб-службу	2 дней	19	Разработчик инфраструктуры
22	Тестирование и исправление ошибок	2 дней	20	Разработчик инфраструктуры[25%]; Специалист отдела тестирования[25%]
23	Оформление документации	1 день	18	Разработчик инфраструктуры

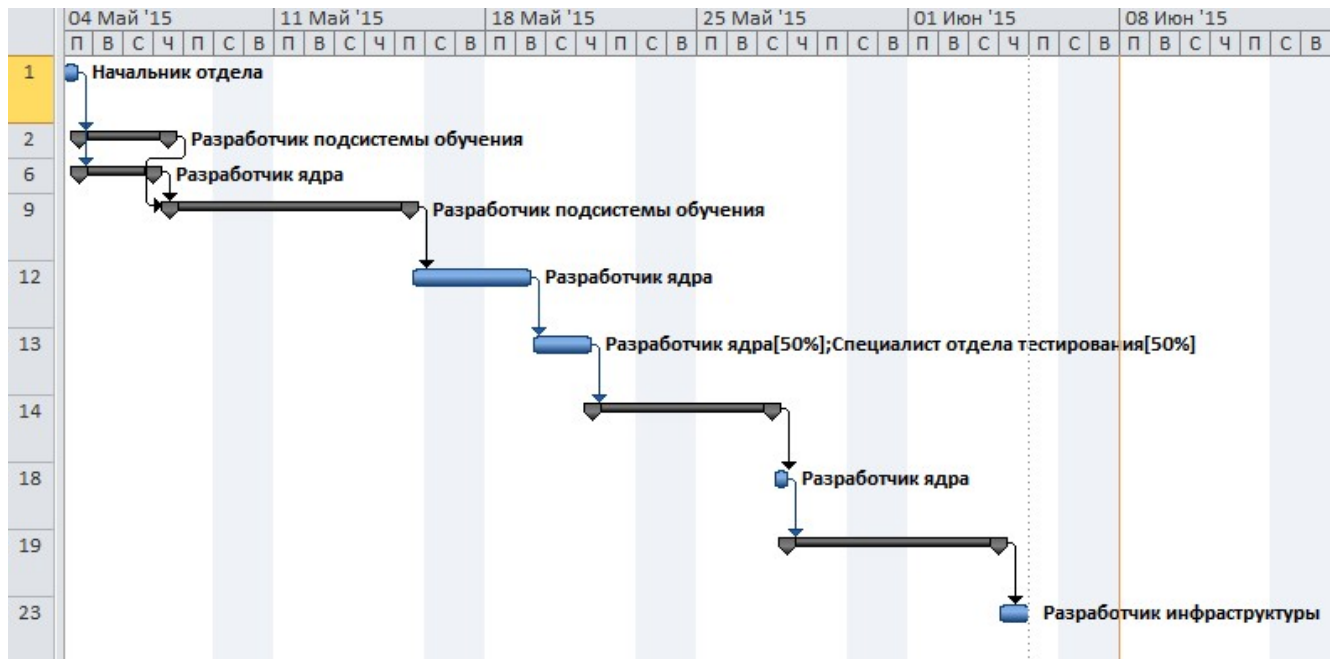


Рис. 5.2. Сокращенное представление диаграммы Ганта

Таблица 5.2

Трудозатраты исполнителей

№	Исполнитель	Трудозатраты, человеко-часов
1	Начальник отдела	4
2	Разработчик подсистемы обучения	80
3	Разработчик ядра	56
4	Разработчик инфраструктуры	36
5	Специалист отдела тестирования	20

5.3. Расчёт затрат на разработку проекта

Расчет затрат на создание и внедрение ПО включает следующие составляющие с последующим их графическим представлением в виде круговой диаграммы:

- заработная плата исполнителей работ по проекту – $ЗП_{осн}$;
- дополнительная заработная плата $ЗП_{доп}$;
- отчисления на социальные нужды (страховые взносы) – $H_{зн}$;
- арендные платежи за производственные (офисные) помещения – $A_{пм}$;
- амортизация используемых основных средств и нематериальных активов – A ;
- расходы на модернизацию и приобретение основных средств – $P_{мод}$;
- расходы на приобретение необходимого ПО – $P_{по}$;
- расходы на интернет, связь – $P_{тел}$;

- и) расходы на канцелярские товары и расходные материалы – $P_{р.м.}$;
- к) прочие расходы – $П_{р.р.}$.

5.3.1. Расчёт заработной платы исполнителей работ по созданию программного продукта

Основная ЗП определяется по формуле:

$$ЗП_{осн} = \frac{М \cdot Т}{Ч_p \cdot t_{р.д.}} \left(1 + \frac{П}{100} \right), \text{руб.},$$

где $М$ – месячная зарплата (руб.), $Т$ – общие трудозатраты (чел.-ч), $Ч_p$ – число рабочих дней в месяц, $t_{р.д.}$ – продолжительность рабочего дня в часах, $П$ – процент премии. В данной работе $Ч_p = 21 \text{ день}$, $t_{р.д.} = 8 \text{ ч}$, $П = 0$.

Значение месячной заработной платы ($М$), суммарные трудозатраты членов, а также рассчитанная по формуле основная заработная плата проектной команды приведены в табл. 5.3.

Таблица 5.3

Заработная плата проектной команды

№	Исполнитель	Месячная заработная плата (М), руб.	Трудозатраты, человеко-часов	ЗП _{осн} , руб
1	Начальник отдела	85 000	4	2023
2	Разработчик подсистемы обучения	50 000	80	23 809
3	Разработчик ядра	50 000	56	16 666
4	Разработчик инфраструктуры	45 000	36	9 643
5	Специалист отдела тестирования	40 000	20	4 761

Суммарное значение основной заработной платы проектной команды на период реализации проекта составит 56 902 (руб.).

Дополнительная заработная плата берется в размере 15% от основной.

$$ЗП_{доп} = 8 535 \text{ (руб.)}.$$

Общая заработная плата составит 65 437 (руб.).

5.3.2. Расчёт отчислений на социальные нужды

Теперь можно рассчитать величину отчислений на социальные нужды (страховые взносы), которые начисляются на заработную плату и в 2015 г. для организаций, осуществляющих деятельность в области информационных

технологий, составляют 14% по выплатам в пределах 568 тыс. руб. Структура отчислений на социальные нужды (страховые взносы) приведена в табл. 5.4.

Таблица 5.4

Структура отчислений на социальные нужды

Пенсионный фонд Российской Федерации	8,0%
для лиц 1966 года рождения и старше	
страховые взносы на страховую часть трудовой пенсии	8,0%
для лиц 1967 года рождения и моложе	
страховые взносы на страховую часть трудовой пенсии	2,0%
страховые взносы на накопительную часть трудовой пенсии	6,0%
Фонд социального страхования Российской Федерации	2,0%
Федеральный фонд обязательного медицинского страхования	4,0%

Таким образом, $H_{zn} = 9\,161$ (руб.).

5.3.3. Арендные платежи за офисные помещения

Компания, реализующая проект по разработке и внедрению ПО для автоматизации внутри складской логистики, арендует офисные помещения в г. Брянск.

Стоимость аренды составляет 300 руб/м² в месяц.

Арендная плата включает в себя оплату как площади занимаемых компанией помещений, так и электроэнергии, отопления, водоснабжения, кондиционирования и уборки помещений, вывоза и утилизации технико-бытовых отходов, парковочных мест на автостоянке.

На каждого члена проектной команды приходится 4 м² арендуемого офисного помещения. На период данного проекта члены проектной команды в других проектах не задействованы.

Исходя из изложенного выше, затраты на аренду помещений, отнесенные на проект составят $A_{nm} = 6\,000$ (руб.).

5.3.4. Амортизация используемых основных средств и нематериальных активов

При реализации проекта по разработке и внедрению ПО задействованы 5 персональных компьютеров в сборе первоначальной стоимостью 20 000 (руб.) каждый.

Срок полезного использования для задействованных в проекте основных средств определен в 3 года. Метод начисления амортизации – линейный.

Амортизационные отчисления для персонального компьютера на 1 месяц составят

$$20\,000 / 36 = 555,56 \text{ (руб.)}.$$

Таким образом, амортизационные отчисления по основным средствам составят:

$$A_{oc} = 5 * 1 * 555,56 = 2\,777,8.$$

ПО используемое при реализации проекта было полностью списано до его начала, поэтому амортизационные отчисления этого типа равны нулю.

Суммарные амортизационные отчисления составят: $A=2\,779$ (руб.).

5.3.5. Расходы на модернизацию и приобретение основных средств

При реализации проекта не планируется приобретение новых и модернизация существующих основных средств

5.3.6. Расходы на приобретение ПО

При реализации проекта не планируется приобретение ПО.

5.3.7. Расходы на интернет и связь

Так как в компании, реализующей проект не производится биллинг и тарификация телекоммуникационных услуг в разрезе сотрудников, затраты на интернет и связь войдут в прочие затраты, рассчитываемые как процент от прямых затрат.

5.3.8. Расходы на канцелярские товары и расходные материалы

Затраты на расходные материалы берутся по факту и составляют $P_{p.m.} = 700$ (руб.). К данным затратам относятся затраты на канцтовары, тонер и бумагу для принтера и т.д.

5.3.9. Прочие расходы

Прочие расходы составляют 30% от суммы следующих элементов структуры затрат: $ЗП_{осн}$, $ЗП_{доп}$, $H_{зп}$, $A_{пм}$, A , $P_{мод}$, $P_{ПО}$, $P_{тел}$ и $P_{p.m.}$.

$$P_{p.p.} = 0.3(ЗП_{осн} + ЗП_{доп} + H_{зп} + A_{пм} + A + P_{мод} + P_{ПО} + P_{тел} + P_{p.m.}).$$

Таким образом, $P_{p.p.} = 25\,223$ (руб.).

5.3.10. Расчёт себестоимости программного продукта

В себестоимость программного продукта входят следующие элементы: $ЗП_{осн}$, $ЗП_{доп}$, $H_{зп}$, $A_{пм}$, A , $P_{мод}$, $P_{ПО}$, $P_{тел}$, $P_{p.m.}$ и $P_{p.p.}$.

Сложив все элементы, можно определить себестоимость программного продукта и услуг по его внедрению: $C_{п.п.} = 109\,300$ (руб.).

Структура себестоимости программного продукта отражена в табл. 5.5 и представлена на рис. 5.3.

Таблица 5.5

Структура себестоимости программного продукта

№	Элементы себестоимости	Сумма (руб.)	% в общ. сумме себестоимости
1	Основная заработная плата исполнителя	56 902	52,06
2	Дополнительная заработная плата исполнителя	8 535	7,81
3	Отчисления на социальные нужды (страховые взносы)	9 161	8,38
4	Арендные платежи за производственные (офисные) помещения	6 000	5,49
5	Амортизация используемых основных средств и нематериальных активов	2 779	2,54
6	Расходы на модернизацию и приобретение основных средств	0	0,00
7	Расходы на приобретение ПО	0	0,00
8	Расходы на интернет, связь	0	0,00
9	Расходы на канцелярские товары и расходные материалы	700	0,64
10	Прочие расходы	25 223	23,08
Итого		109 303	100



Рис. 5.3. Структура себестоимости программного продукта

6. ТЕСТИРОВАНИЕ СИСТЕМЫ

Тестирование – необходимый процесс при создании программного продукта. Цель тестирования – обнаружить ситуацию, когда результаты работы программы не соответствуют входным данным и/или требованиям, перечисленным в техническом задании.

При тестировании программного продукта за основу был взят метод тестирования черным ящиком. Данный метод использует стратегию тестирования функционального поведения объекта с точки зрения окружающего мира. Предполагается, что специалист по тестированию ПО не знает внутреннюю структуру программного продукта.

6.1. План испытаний

Запланированы следующие этапы процесса тестирования.

1. Тестирование подсистемы обучения.
 - 1.1. Оценка качества выявления информативных пар признаков.
 - 1.1.1. На маленьких выборках.
 - 1.1.2. На больших выборках.
 - 1.2. Оценка качества работы учителя.
2. Интеграционное тестирование подсистемы обучения на больших обучающих выборках.
3. Тестирование антивируса на стандартной коллекции файлов.
4. Тестирование стабильности системы анализа в течение двух итераций цикла разработки (1 итерация = 1 неделя).

6.2. Тестирование подсистемы обучения

6.2.1. Тестирование утилиты *librarian* на небольшой выборке

В качестве входных данных при тестировании использовалось 8 журналов (4 чистых и 4 вредоносных). Фрагмент результирующего отчёта приведен в табл. 6.1.

Фрагмент выходных данных утилиты *librarian* на небольшой выборке

Первый признак	Второй признак	Чис- тых	Вредо- носных	Все- го	Отноше- ние
EEV_CALLCHARFROMCODE		0	1	1	0.500
EEV_CALLCHARFROMCODE	EEV_CALLCHARFROMCODE	0	1	1	0.500
EEV_CALLCHARFROMCODE	EEV_EXPLOIT	0	1	1	0.500
EEV_CALLCHARFROMCODE	EEV_HIDDENEVAL	0	1	1	0.500
EEV_CALLCHARFROMCODE	EEV_HIDDENUNESCAPE	0	1	1	0.500
EEV_CALLCHARFROMCODE	EEV_READVERSION	0	1	1	0.500
EEV_EXPLOIT		0	1	1	0.500
EEV_EXPLOIT	EEV_EXPLOIT	0	1	1	0.500
EEV_EXPLOIT	EEV_HIDDENUNESCAPE	0	1	1	0.500
EEV_HASUNESCAPE		0	4	4	0.800
EEV_HASUNESCAPE	EEV_HASUNESCAPE	0	4	4	0.800
EEV_HASUNESCAPE	EEV_LONGSTRING	0	4	4	0.800
EEV_HIDDENEVAL		0	1	1	0.500
EEV_HIDDENEVAL	EEV_EXPLOIT	0	1	1	0.500
EEV_PUREJS		5	4	9	-0.100

Утилита корректно подсчитала количество файлов и отношения для каждой комбинации.

6.2.2. Тестирование утилиты *librarian* на большой выборке

В ходе тестирования использовалось 32 599 журналов (20555 чистых и 12044 вредоносных). Фрагмент отчёта о работе утилиты приведен в табл. 6.2.

Текстовый поиск по файлам подтвердил корректный подсчёт файлов в случае с одиночными признаками. Также, по признаку *EEV_EXPLOIT* и комбинациям, в которых он участвует, можно косвенно судить о корректности работы утилиты. Данный признак устанавливается в том случае, если найдена эксплуатация уязвимости.

Фрагмент выходных данных утилиты *librarian* на большой выборке

Первый признак	Второй признак	Чистых	Вредо- носных	Всего	Отношение
EEV_DISPLAYNONE		132	8	140	-0.879
EEV_DISPLAYNONE	EEV_DISPLAYNONE	1	0	1	-0.500
EEV_DISPLAYNONE	EEV_HIDDENUNESCAPE	0	1	1	0.500
EEV_DISPLAYNONE	EEV_JQUERYCALL	1	0	1	-0.500
EEV_DISPLAYNONE	EEV_SETTIMEOUT	1	5	6	0.571
EEV_DISPLAYNONE	EEV_WINDOWONLOAD	1	0	1	-0.500
EEV_DISPLAYNONE	EEV_WRITEIFFRAME	0	1	1	0.500
EEV_DYNCODE		33	7	40	-0.634
EEV_DYNCODE	EEV_ADVERTISING	3	0	3	-0.750
EEV_DYNCODE	EEV_DYNCODE	9	4	13	-0.357
EEV_DYNCODE	EEV_EXPLOIT	0	1	1	0.500
EEV_DYNCODE	EEV_HIDDENEVAL	3	2	5	-0.167
EEV_DYNCODE	EEV_NATIVETOSTRING	2	0	2	-0.667
EEV_DYNCODE	EEV_READVERSION	0	1	1	0.500
EEV_DYNCODE	EEV_WINDOWONLOAD	24	0	24	-0.960
EEV_EXPLOIT		0	5300	5300	1.000
EEV_EXPLOIT	EEV_CALLBYTETOCHAR	0	4	4	0.800
EEV_EXPLOIT	EEV_BODYAPPENDCHILD	0	33	33	0.971
EEV_EXPLOIT	EEV_CALLBYTETOCHAR	0	4	4	0.800

6.2.3. Тестирование утилиты *teacher*

Помимо обучающей выборки утилита *teacher* дополнительно использовала 32599 журналов для тестирования качества обучения. Параметры обучения заданы по умолчанию. На рис. 6.1 представлен отчёт о результатах первой эпохи обучения где используются:

- а) шестнадцатеричные числа – значения хешей *SHA1* исполняемых файлов;
- б) *Exp* – ожидаемый выходной вектор из нейросети;
- в) *Res* – фактический вектор;
- г) *Correct* – количество верно классифицированных журналов;
- д) *False positive* – количество ложно-положительных результатов;
- е) *False negative* – количество ложно-отрицательных результатов;
- ж) *Undefined clean* – количество неопределенных результатов на чистых журналах;

3) *Undefined malware* – количество неопределенных результатов на вредоносных журналах.

Отчёт демонстрирует факт способности нейросети осуществлять классификацию уже после первой эпохи обучения.

```

65172 FFBC01CD9788386F75F6DE7E51D791561A009157 Exp: (0.000; 1.000) res: (0.004; 0.996) correct
65173 FFC7867BDF735068AF574B3051F28EC6E429482E Exp: (0.000; 1.000) res: (0.004; 0.996) correct
65174 FFCB0A0797CA2839BB040A740FD8D9124BC60366 Exp: (0.000; 1.000) res: (0.004; 0.996) correct
65175 FFAEA681A04AA302B0AA01ADA2BA7CB0EBA47607 Exp: (0.000; 1.000) res: (0.004; 0.996) correct
65176 FFBFD71BA8905C8D88619D5B3662E56DCD285584 Exp: (0.000; 1.000) res: (0.032; 0.969) correct
65177 FFCF8077AE0F35BD082B223BA378CAB7EBF1AC1 Exp: (0.000; 1.000) res: (0.032; 0.969) correct
65178 FFDA68AFA0D28C1AA13818BB542FA23AFEF3F80D Exp: (0.000; 1.000) res: (0.032; 0.969) correct
65179 FFD9AB1891CB1D4B10C14CB62B4DDBF78091CFAD Exp: (0.000; 1.000) res: (0.004; 0.996) correct
65180 FFCDD5D7ED6ABE963E10FE745F2A7D1223BFC0BE Exp: (0.000; 1.000) res: (0.004; 0.996) correct
65181 FFDA950A69250F5E9C9CC10D7785178383C99F19 Exp: (0.000; 1.000) res: (0.032; 0.969) correct
65182 FFD49DAD7C8B1782514A62A1B0A3F0F9D1CCE451 Exp: (0.000; 1.000) res: (0.032; 0.969) correct
65183 FFE233677E8E47E79883A1821A2A2DF53D105B35 Exp: (0.000; 1.000) res: (0.032; 0.969) correct
65184 FFE02EA580F001E18878DD12449E45C65085235E Exp: (0.000; 1.000) res: (0.032; 0.969) correct
65185 FFDE06D4B80E943B41DAF485F8AA0BA20B7E65EC Exp: (0.000; 1.000) res: (0.032; 0.969) correct
65186 FFE2936BD13F612C6B4D54998478A233DBBF80D2 Exp: (0.000; 1.000) res: (0.032; 0.969) correct
65187 FFE79FE3FEBD903B3FFADC42407D02E7C2EF731A Exp: (0.000; 1.000) res: (0.004; 0.996) correct
65188 FFE59BA1B86DF6360B446541ED0E6BDA2B0D6FCB Exp: (0.000; 1.000) res: (0.032; 0.969) correct
65189 FFE29753F770A2A798BC0BBA7DD3F746B35DA8F2 Exp: (0.000; 1.000) res: (0.032; 0.969) correct
65190 FFF7FD6DBE146BC23AF31408E71ED65DCD941833 Exp: (0.000; 1.000) res: (0.004; 0.996) correct
65191 FFEBAD058AF2B517188CF598EBF595FE6177FF5 Exp: (0.000; 1.000) res: (0.004; 0.996) correct
65192 FFEDBED2AF936651CFB0403ABDCB4800027CBA40 Exp: (0.000; 1.000) res: (0.004; 0.996) correct
65193 FFF196BEC0EC9771761CF87D5C35C1E58A25FBF9 Exp: (0.000; 1.000) res: (0.004; 0.996) correct
65194 FFF5C8414CFDBC83E63E85C7B1310BC1AA1E1949 Exp: (0.000; 1.000) res: (0.032; 0.969) correct
65195 FFFE3CF80E5F2E94AE3B0E98BAB0E1AF386ADEBA Exp: (0.000; 1.000) res: (0.032; 0.969) correct
65196 FFF32FF06DA1BEB6DCCEEE58581C167651B2F3DD Exp: (0.000; 1.000) res: (0.032; 0.969) correct
65197 FFF378D71505132E44EA27426281201DBE62A8FE Exp: (0.000; 1.000) res: (0.036; 0.965) correct
65198 FFEA4CBBEDBD5262761701E3657FEB80D4F5242 Exp: (0.000; 1.000) res: (0.004; 0.996) correct
65199 Correct: 63905; False positive: 41; False negative: 1222; Undefined clean: 2; Undefined malware: 28
65200

```

Рис. 6.1. Отчёт о результатах первой эпохи обучения

7. ОРГАНИЗАЦИОННАЯ ЧАСТЬ

В современном мире персональные компьютеры широко используются в различных областях деятельности человека. Появление компьютеров в нашей жизни изменило характер труда, это потребовало комплексного решения проблем эргономики, гигиены и организации труда, регламентации режимов труда и отдыха.

При работе с компьютером человек подвергается воздействию ряда опасных и вредных производственных факторов:

- а) нарушение микроклимата помещения;
- б) повышенный уровень шума на рабочем месте;
- в) повышенный уровень статического электричества;
- г) повышенный уровень электромагнитных излучений;
- д) повышенная напряженность электрического поля;
- е) повышенная напряженность магнитного поля;
- ж) отсутствие или недостаток естественного света;
- з) недостаточная освещенность рабочей зоны;
- и) повышенная яркость света;
- к) повышенная пульсация светового потока.

Для уменьшения негативных последствий работы с ЭВМ следует выбирать рациональные режимы труда и отдыха, использовать защитные средства, осуществлять комплексные оздоровительно-профилактические мероприятия. Безопасные условия труда на ЭВМ регламентируют СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к ПЭВМ и организации работы».

Ответственность за выполнение настоящих правил возлагается на юридических лиц и индивидуальных предпринимателей, осуществляющих разработку, производство и эксплуатацию ПК, занимающихся проектированием и реконструкцией помещений, предназначенных для работы с ПК.

Последствия несоблюдения требований безопасности приводит к тому, что через период времени здоровье человека подвергается опасности. Он начинает испытывать усталость, раздражительность, головные боли, дискомфорт в области

спины и в шеи и это далеко не все последствия, которые ожидают человека, несоблюдающего требования безопасности.

7.1. Организация рабочего места и режима труда и отдыха

Требования к организации и оборудованию рабочего места сотрудника вычислительного центра (ВЦ) приведены в ГОСТ 12.2.032-78. Высота рабочей поверхности стола для пользователей должна регулироваться в пределах 680-800 мм; при отсутствии таковой возможности высота рабочей поверхности стола должна составлять 725 мм.

Модульными размерами рабочей поверхности стола для ПК, на основании которых должны рассчитываться конструктивные размеры, следует считать: ширину 800, 1200, 1400 мм, глубину 800 и 1000 мм при нерегулируемой высоте, равной 725 мм.

Рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной – не менее 500 мм, глубиной на уровне колен – не менее 450 мм и на уровне вытянутых ног – не менее 650 мм.

Рабочий стул (кресло) должен быть подъемно-поворотным и регулируемым по высоте и углам наклона сиденья и спинки, а также – расстоянию спинки до переднего края сиденья.

Рабочее место необходимо оборудовать подставкой для ног, имеющей ширину не менее 300 мм, глубину не менее 400 мм, регулировку по высоте в пределах до 150 мм и по углу наклона опорной поверхности подставки до 20 градусов.

Поверхность подставки должна быть рифленой и иметь по переднему краю бортик высотой 10 мм.

Клавиатуру следует располагать на поверхности стола на расстоянии 100-300 мм от края, обращенного к пользователю, или на специальной регулируемой по высоте рабочей поверхности, отделенной от основной столешницы.

Данные требования описаны в санитарных нормах и правилах (СанПиН) для работников вычислительных центров от 22-05-95.

Искусственное освещение в помещениях эксплуатации ПЭВМ осуществляется системой общего равномерного освещения. В производственных и административно-общественных помещениях, в случаях преимущественной работы с документами, разрешено применение системы комбинированного освещения (к общему освещению дополнительно устанавливаются светильники местного освещения, предназначенные для освещения зоны расположения документов).

Освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300-500 лк, также допускается установка светильников местного освещения для подсветки документов, но с таким условием, чтобы оно не создавало бликов на поверхности экрана и не увеличивало освещенность экрана более чем на 300 лк. В качестве источников света при искусственном освещении должны применяться преимущественно люминесцентные лампы типа ЛД. При устройстве отраженного освещения в административно-общественных помещениях допускается применение металлогалогенных ламп мощностью до 250 Вт. Допускается применение ламп накаливания в светильниках местного освещения.

Общее освещение следует выполнять в виде сплошных или прерывистых линий светильников, расположенных сбоку от рабочих мест, параллельно линии зрения пользователя при рядном расположении ПЭВМ. При периметральном расположении компьютеров линии светильников должны располагаться локализовано над рабочим столом, ближе к его переднему краю, обращенному к оператору.

Для обеспечения нормируемых значений освещенности в помещениях использования ПК следует проводить чистку стекол оконных рам и светильников не реже двух раз в год и проводить своевременную замену перегоревших ламп.

Помещения, где размещаются рабочие места с ПК, должны быть оборудованы защитным заземлением в соответствии с техническими требованиями по эксплуатации.

Согласно СанПиН 2.2.2/2.4.1340-03, экран монитора должен находиться от глаз пользователя на расстоянии 600-700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.

Для обеспечения оптимальной работоспособности и сохранения здоровья профессиональных пользователей, на протяжении рабочей смены должны устанавливаться регламентированные перерывы. Время регламентированных перерывов в течение рабочей смены следует устанавливать, в зависимости от ее продолжительности, вида и категории трудовой деятельности.

Продолжительность непрерывной работы перед монитором без регламентированного перерыва не должна превышать 1 час.

В случаях, когда характер работы требует постоянного взаимодействия с монитором ПК (набор текстов или ввод данных) с напряжением внимания и сосредоточенности, при исключении возможности периодического переключения на другие виды трудовой деятельности, не связанные с ПВМ, рекомендуется организация перерывов на 10-15 минут через каждые 45-60 минут работы.

7.2. Противопожарная безопасность

Пожарная безопасность – состояние объекта, при котором исключается возможность пожара, а в случае его возникновения предотвращается воздействие на людей опасных факторов пожара и обеспечивается защита материальных ценностей.

Пожарная безопасность обеспечивается системой предотвращения пожара и системой пожарной защиты. Во всех служебных помещениях обязательно должен быть «План эвакуации людей при пожаре», регламентирующий действия персонала случае возникновения очага возгорания и указывающий места расположения пожарной техники.

Пожары в ВЦ представляют особую опасность, так как сопряжены с большими материальными потерями. Характерная особенность ВЦ – небольшие площади помещений. Как известно пожар может возникнуть при взаимодействии горючих веществ, окисления и источников зажигания. В помещениях ВЦ

присутствуют все три основных фактора, необходимые для возникновения пожара.

Горючими компонентами на ВЦ являются: строительные материалы для акустической и эстетической отделки помещений, перегородки, двери, полы, изоляция кабелей и др.

Противопожарная защита – это комплекс организационных и технических мероприятий, направленных на обеспечение безопасности людей, на предотвращение пожара, ограничение его распространения, а также на создание условий для успешного тушения пожара.

Источниками зажигания в ВЦ могут быть электронные схемы от ЭВМ, приборы, применяемые для технического обслуживания, устройства электропитания, кондиционирования воздуха, где в результате различных нарушений образуются перегретые элементы, электрические искры и дуги, способные вызвать загорания горючих материалов.

В современных ЭВМ очень высокая плотность размещения элементов электронных схем. В непосредственной близости друг от друга располагаются соединительные провода, кабели. При протекании по ним электрического тока выделяется значительное количество теплоты. При этом возможно оплавление изоляции. Для отвода избыточной теплоты от ЭВМ служат системы вентиляции и кондиционирования воздуха. При постоянном действии эти системы представляют собой дополнительную пожарную опасность.

Для большинства помещений ВЦ установлена категория пожарной опасности В.

Одной из наиболее важных задач пожарной защиты является защита строительных помещений от разрушений и обеспечение их достаточной прочности в условиях воздействия высоких температур при пожаре. Учитывая высокую стоимость электронного оборудования ВЦ, а также категорию его пожарной опасности, здания для ВЦ и части здания другого назначения, в которых предусмотрено размещение ЭВМ должны быть 1 и 2 степени огнестойкости.

Для изготовления строительных конструкций используются, как правило, кирпич, железобетон, стекло, металл и другие негорючие материалы. Применение дерева должно быть ограничено, а в случае использования необходимо пропитывать его огнезащитными составами. В ВЦ противопожарные преграды в виде перегородок из несгораемых материалов устанавливают между машинными залами.

К средствам тушения пожара, предназначенных для локализации небольших загораний, относятся пожарные стволы, внутренние пожарные водопроводы, огнетушители, сухой песок, асбестовые одеяла и т. п.

В зданиях ВЦ пожарные краны устанавливаются в коридорах, на площадках лестничных клеток и входов. Вода используется для тушения пожаров в помещениях программистов, библиотеках, вспомогательных и служебных помещениях. Применение воды в машинных залах ЭВМ, хранилищах носителей информации, помещениях контрольно-измерительных приборов ввиду опасности повреждения или полного выхода из строя дорогостоящего оборудования возможно в исключительных случаях, когда пожар принимает угрожающе крупные размеры.

При этом количество воды должно быть минимальным, а устройства ЭВМ необходимо защитить от попадания воды, накрывая их брезентом или полотном.

Для тушения пожаров на начальных стадиях широко применяются огнетушители. По виду используемого огнетушащего вещества огнетушители подразделяются на следующие основные группы.

Пенные огнетушители, применяются для тушения горящих жидкостей, различных материалов, конструктивных элементов и оборудования, кроме электрооборудования, находящегося под напряжением.

Газовые огнетушители применяются для тушения жидких и твердых веществ, а также электроустановок, находящихся под напряжением.

В производственных помещениях ВЦ применяются главным образом углекислотные огнетушители, достоинством которых является высокая эффективность тушения пожара, сохранность электронного оборудования, диэлектрические свойства углекислого газа, что позволяет использовать эти

огнетушители даже в том случае, когда не удастся обесточить электроустановку сразу.

7.3. Электробезопасность

Электрический ток представляет собой скрытый тип опасности, т.к. его трудно определить в токо- и нетоковедущих частях оборудования, которые являются хорошими проводниками электричества. Смертельно опасным для жизни человека считают ток, величина которого превышает 0,05А, ток менее 0,05А – безопасен (до 1000 В). С целью предупреждения поражений электрическим током к работе должны допускаться только лица, хорошо изучившие основные правила по технике безопасности.

В соответствии с правилами электробезопасности в служебном помещении должен осуществляться постоянный контроль состояния электропроводки, предохранительных щитов, шнуров, с помощью которых включаются в электросеть компьютеры, осветительные приборы, другие электроприборы.

Электрические установки, к которым относится практически все оборудование ЭВМ, представляют для человека большую потенциальную опасность, так как в процессе эксплуатации или проведении профилактических работ человек может коснуться частей, находящихся под напряжением. Специфическая опасность электроустановок – токоведущие проводники, корпуса стоек ЭВМ и прочего оборудования, оказавшегося под напряжением в результате повреждения (пробоя) изоляции, не подают каких-либо сигналов, которые предупреждают человека об опасности. Реакция человека на электрический ток возникает лишь при протекании последнего через тело человека. Исключительно важное значение для предотвращения электротравматизма имеет правильная организация обслуживания действующих электроустановок ВЦ, проведения ремонтных, монтажных и профилактических работ.

В зависимости от категории помещения необходимо принять определенные меры, обеспечивающие достаточную электробезопасность при эксплуатации и ремонте электрооборудования. В ВЦ разрядные токи статического электричества

чаще всего возникают при прикосновении к любому из элементов ЭВМ. Такие разряды опасности для человека не представляют, но кроме неприятных ощущений они могут привести к выходу из строя ЭВМ. Для снижения величины возникающих зарядов статического электричества в ВЦ покрытие технологических полов следует выполнять из однослойного поливинилхлоридного антистатического линолеума.

Другим методом защиты является нейтрализация заряда статического электричества ионизированным газом. В промышленности широко применяются радиоактивные нейтрализаторы. К общим мерам защиты от статического электричества в ВЦ можно отнести общие и местное увлажнение воздуха.

7.4. Шум и вибрация

Шум ухудшает условия труда, оказывая вредное действие на организм человека. Работающие в условиях длительного шумового воздействия испытывают раздражительность, головные боли, головокружение, снижение памяти, повышенную утомляемость, понижение аппетита, боли в ушах и т. д.

Такие нарушения в работе ряда органов и систем организма человека могут вызвать негативные изменения в эмоциональном состоянии человека вплоть до стрессовых. Под воздействием шума снижается концентрация внимания, нарушаются физиологические функции, появляется усталость в связи с повышенными энергетическими затратами и нервно-психическим напряжением, ухудшается речевая коммутация. Все это снижает работоспособность человека и его производительность, качество и безопасность труда. Длительное воздействие интенсивного шума на слух человека приводит к его частичной или полной потере.

Уровень шума на рабочем месте математиков-программистов и операторов видеоматериалов не должен превышать 50дБА, а в залах обработки информации на вычислительных машинах – 65дБА.

Для снижения уровня шума стены и потолок помещений, где установлены компьютеры, могут быть облицованы звукопоглощающими материалами.

Уровень вибрации в помещениях вычислительных центров может быть снижен путем установки оборудования на специальные виброизоляторы.

7.5. Расчёт освещенности

Тип лампы: люминесцентная;

Тип светильника: ПВЛ-1;

$E = 300$ лк;

$h_p = 3$ м;

$\lambda = 1,5$;

$a = 7$ м, $b = 5$ м.

В зависимости от типа светильника выбираем коэффициент λ . Он определяет такое соотношение максимального расстояния между светильниками $L_{св(max)}$ и высотой их подвеса над рабочей поверхностью h_p , которое обеспечит равномерность освещения в помещении.

$$L_{св(max)} = \lambda \cdot h_p = 1,5 \cdot 3 = 4,5 \text{ м.}$$

Определяем расстояние $L_{1(max)}$ от стены до первого ряда светильников:

$$L_{1(max)} = (0,2 \dots 0,3) \cdot L_{св(max)} = 0,25 \cdot 4,5 = 1,125 \text{ м.}$$

Определяем общее число рядов светильников (по ширине помещения):

$$n_{ш(min)} = \frac{b - 2 \cdot L_{1(max)}}{L_{св(max)}} + 1 = \frac{5 - 2 \cdot 1,125}{4,5} + 1 = 1,61 \approx 2,$$

и число светильников в ряду (по длине помещения):

$$n = \frac{a - 2 \cdot L_{1(max)}}{L_{св(max)}} + 1 = \frac{7 - 2 \cdot 1,125}{4,5} + 1 = 2,05 \approx 2,$$

где **a** – длина, **b** – ширина помещения, для которого рассчитывается система освещения.

Полученные результаты округляем до ближайшего целого числа, после чего определяем общее расчётное минимальное количество светильников, которое необходимо разместить в помещении:

$$n_{общ (min)} = n_{ш (min)} \cdot n_{д (min)} = 2 \cdot 2 = 4,$$

$$S = a \cdot b = 7 \cdot 5 = 35 \text{ м.}$$

По площади помещения S и высоте подвески светильника h_p определяем показатель помещения i :

$$i = \frac{S}{h_p * (a + b)} = \frac{35}{3 * (7 + 5)} = 0,97 \approx 1.$$

Находим значения коэффициентов отражения потолка ρ_n , стен ρ_c и полов $\rho_{пол}$ помещения, для которого рассчитывается осветительная установка:

$$\rho_n = 50\%;$$

$$\rho_c = 30\%;$$

$$\rho_{пол} = 10\%.$$

В зависимости от типа светильника и вида лампы определяем коэффициент использования светового потока η_n по показателю помещения i и коэффициентам отражения потолка ρ_n , стен ρ_c и полов $\rho_{пол}$. $\eta_n = 0,28$.

Определяем коэффициент запаса k , учитывающий снижение уровня освещённости из-за неблагоприятных условий эксплуатации осветительной установки: наличия дыма, копоти, пыли, повышенной концентрации химических веществ и т. д.; из-за старения и выхода из строя ламп $k = 1,3$.

Решаем сколько источников света x будет в светильнике $x = 4$.

Назначаем коэффициент z , характеризующий неравномерность освещённости (коэффициент отношения средней освещённости к максимальной):

$$z = 1,1 - \text{для люминесцентных ламп.}$$

Рассчитываем требуемый световой поток одной лампы:

$$\Phi_{расч} = \frac{E * S * k * Z}{\eta_n * n_{общ(min)} * x} = \frac{300 * 35 * 1,3 * 1,1}{0,28 * 4 * 4} = 3352 \text{ лм.}$$

По рассчитанному световому потоку лампы $\Phi_{расч}$ подбираем стандартную лампу со световым потоком $\Phi_{табл.}$, значение которого близко к значению $\Phi_{расч}$ (желательно в пределах $-10...+20\%$).

После выбора стандартных ламп рассчитываем число светильников, необходимых для обеспечения заданной освещённости E . Полученное число $n_{расч}$ округляют до ближайшего целого значения $n_{пр}$, при этом отклонение между

принятым количеством светильников $n_{\text{пр}}$ и расчётным $n_{\text{расч}}$ допускается в пределах от -10 до $+20\%$:

$$\begin{aligned}\Phi_{\text{табл1}} &= 3390, \\ n_{\text{расч1}} &= \frac{E * S * k * z}{\Phi_{\text{табл1}} * \eta_{\text{и}} * \chi} = \frac{300 * 35 * 1,3 * 1,1}{3390 * 0,28 * 4} = 3,955, \\ n_{\text{пр1}} &= 4, \\ \Delta_1 &= \frac{n_{\text{пр1}} - n_{\text{расч1}}}{n_{\text{расч1}}} * 100\% = 1,13\%, \\ \Phi_{\text{табл2}} &= 3380, \\ n_{\text{расч2}} &= \frac{E * S * k * z}{\Phi_{\text{табл2}} * \eta_{\text{и}} * \chi} = \frac{300 * 35 * 1,3 * 1,1}{3380 * 0,28 * 4} = 3,966, \\ n_{\text{пр2}} &= 4, \\ \Delta_2 &= \frac{n_{\text{пр2}} - n_{\text{расч2}}}{n_{\text{расч2}}} * 100\% = 1,86\%.\end{aligned}$$

Рассчитываем полную мощность проектируемой системы освещения:

$$N_1 = n_{\text{пр1}} * \chi * N_{\text{лампы1}} = 4 * 4 * 65 = 1040 \text{ Вт},$$

$$N_2 = n_{\text{пр2}} * \chi * N_{\text{лампы2}} = 4 * 4 * 80 = 1280 \text{ Вт}.$$

Результаты расчётов приведены в табл. 7.1.

Таблица 7.1

Результаты расчётов параметров осветительной установки

№	Тип лампы	Световой поток лампы Φ , лм	Количество светильников		Отклонение $n_{\text{пр}}$ от $n_{\text{расч}}$, %	Полная мощность N , Вт
			расчётное $n_{\text{расч}}$	принятое $n_{\text{пр}}$		
1	ЛД-65	3390	3,955	4	1,13	1040
2	ЛДЦ-80	3380	3,966	4	0,86	1280

Вывод: оптимальным вариантом осветительной установки является ЛД-65, так как значение её полной потребляющей мощности меньше, чем у осветительной установки ЛДЦ-80.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы был разработан опытный образец системы эвристического анализа. Данный образец включает в себя подсистему обучения и подсистему анализа. Разработчик ядра интегрировал подсистему анализа в тестовую версию антивируса, что позволило сделать выводы о качестве работы системы на практике.

В качестве предметных областей для оценки качества обучения и анализа были выбраны JAR архивы (скомпилированные программы на Java) и файлы, содержащие в себе скрипты на языке JavaScript (PDF, HTML и JS файлы). Выбор этих областей был осуществлён по причине большого количества признаков, которые способны обнаружить специальные сканеры в ядре антивируса, и достаточного крупной коллекции экземпляров, участвующих в тестировании антивирусного комплекса.

Для тестирования были взяты следующие выборки файлов:

- а) вся коллекция чистых файлов (4 491 017 экземпляров);
- б) файлы с кодом JavaScript, которые полностью обнаруживаются старым анализатором (24 088);
- в) файлы с кодом JavaScript, которые полностью обнаруживаются сигнатурами (5572);
- г) все вредоносные JAR архивы (14508);
- д) все чистые JAR архивы (7453);

Сканирование этих выборок дали следующие результаты соответственно:

- а) новый анализатор ложно среагировал на 0,0038% (174) файлов против 0,0003% (12) старого;
- б) новый анализатор обнаружил 83% (20 375) ВПО против 100% старого;
- в) новый анализатор обнаружил 20% (1 083) ВПО против 11% (612) старого;
- г) новый анализатор обнаружил 99,06% (14372) ВПО против 99,11% (14380) старого;

д) новый анализатор ложно среагировал на 0,09% (7) против 0,17% (13) старого.

Анализ ложных срабатываний в первом случае показал, что обнаруженные файлы действительно безвредны, однако они используют различные техники по сокрытию и запутыванию участков кода, на что и среагировал новый анализатор.

Анализ 17% ложноотрицательных результатов в случае с кодом на JavaScript привёл к выводу, что выборка содержит очень много семейств ВПО. Эта выборка используется при интеграционном тестировании, поэтому важно, чтобы она подтверждала способность обнаружения этих семейств антивирусом. А так как часть этой выборки участвовала в обучении, то можно утверждать, что нейросеть в единственном экземпляре не справляется с обобщением такого большого количества подклассов ВПО. Случай с JAR архивами косвенно подтверждает это утверждение – выборка вредоносных JAR файлов целиком содержит в себе несколько десятков семейств ВПО, в то время как выборка файлов с кодом на JavaScript содержит около тысячи. Одно из решений этой проблемы – обучение нескольких классификаторов, каждый из которых будет специализироваться на своей группе семейств.

Таким образом, были получены хорошие практические результаты, которые позволяют уже сейчас отделу вирусных аналитиков использовать систему в решении узкоспециализированных задач, таких как:

- а) предварительное обнаружение вредоносных файлов из проходящего потока в антивирусную лабораторию;
- б) выявление файлов с зашифрованным или запутанным кодом;
- в) оценка полноты выявленных признаков отдельного семейства ВПО.

Планы дальнейшего развития проекта:

- а) практическая оценка других алгоритмов классификации;
- б) разработка эвристического анализатора, как множество узкоспециализированных классификаторов;
- в) полноценная интеграция в ядро антивируса.

СПИСОК ЛИТЕРАТУРЫ

1. Алгоритм ID, ID3 algorithm. – 2015. – Режим доступа: <http://www.basegroup.ru/glossary/definitions/id3/>
2. Антитело. – 2015. – Режим доступа: <http://dic.academic.ru/dic.nsf/ntes/228/АНТИТЕЛО>
3. Безобразов, С.В. Алгоритмы искусственных иммунных систем и нейронных сетей для обнаружения вредоносных программ / С.В. Безобразов, В.А. Головкин. – 2015. – Режим доступа: <http://ees.kdu.edu.ua/wp-content/uploads/2013/04/86.pdf>
4. Бокс, Д. Сущность технологии СОМ / Д. Бокс. – СПб.: Питер, 2001. – 400 с.
5. Гарнаева М., Функ К. Kaspersky Security Bulletin 2013. Основная статистика за 2013 год / М. Гарнаева, К. Функ. – 2015. – Режим доступа: <http://securelist.ru/analysis/ksb/19145/kaspersky-security-bulletin-2013-osnovnaya-statistika-za-2013-god/>
6. Искусственные иммунные системы и их применение / Под ред. Д. Дасгупты. – М.: ФИЗМАТЛИТ, 2006. – 344 с.
7. Кораблев, Н.М., Модель эвристического анализатора вредоносных программ на основе искусственной иммунной сети / Н.М. Кораблев, М.В. Кушнарёв // Системи обробки інформації. – 2015. – № 8. – С. 216-222.
8. Майерс, С. Эффективное использование C++. 55 верных советов улучшить структуру и код ваших программ / С. Майерс. – М.: ДМК Пресс, 2014. – 300 с.
9. Майерс, С. Эффективное использование STL. Библиотека программиста / С. Майерс. – СПб.: Питер, 2002. – 224 с.
10. Макконнелл, С. Совершенный код. Мастер-класс / С. Макконнелл. – М.: Русская редакция, 2010. – 896 с.
11. Объектно-ориентированный анализ и проектирование с примерами приложений (UML 2). Третье издание / Буч Г., Максимчук Р., Энгл М., Янг Б., Коналлен Дж., Хьюстон К. – М.: Издательский дом «Вильямс», 2010. – 720 с.

12. Официальный сайт информационного портала SecurityLab. – 2014. – Режим доступа: <http://www.securitylab.ru/news/tags/%E2%F0%E5%E4%E5%ED%E5%F1%ED%E5+%CF%CE/>
13. Официальный сайт проекта НАНО Антивирус. – 2015. – Режим доступа: <http://www.nanoav.ru/>
14. Паттерны объектно-ориентированного программирования / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидерс. – СПб.: Питер, 2010. – 366 с.
15. Разработка классических приложений. – 2015. – Режим доступа: <https://msdn.microsoft.com/ru-ru/windows/desktop/aa904962>
16. Рассел, С. Искусственный интеллект: современный подход, 2-е изд. / С. Рассел, П. Норвиг. – М.: Издательский дом «Вильямс», 2015. – 1408 с.
17. Рихтер, Дж. Windows для профессионалов. Создание эффективных Win32-приложений с учётом специфики 64-разрядной версии Windows / Дж. Рихтер. – СПб.: Питер, 2001. – 752 с.
18. Сигнатурный метод в Kaspersky Endpoint Security 8 для Windows. – 2015. – Режим доступа: <http://support.kaspersky.ru/7421>
19. Сноу Дж.,][-препарация: вскрываем хитрый Sality.aa: Учимся распознавать полиморфизм и обфускацию кода на примере известного вируса / Дж. Сноу. – 2015. – Режим доступа: <http://hacker.ru/54790/>
20. Уильямс, Э. Параллельное программирование на C++ в действии. Практика разработки многопоточных программ / Э. Уильямс. – М.: ДМК Пресс, 2014. – 672 с.
21. Arnold W., Automatically generated Win32 heuristic virus detection // W. Arnold, G. Tesauero // Virus bulletin conference. – 2000. – С. 51-60.
22. Compiler Options. – 2015. – Режим доступа: <https://msdn.microsoft.com/en-us/library/9s7c9wdw.aspx>
23. ISO/IEC 14882:2014. Committee Draft, Standard for Programming Language C++. – 2015. – Режим доступа: <http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3690.pdf>

24. Microsoft API and reference catalog. – 2015. – Режим доступа:
<https://msdn.microsoft.com/en-us/library/>
25. Welcome to Visual Studio 2013. – 2015. – Режим доступа:
<https://msdn.microsoft.com/en-us/library/dd831853.aspx>

ПРИЛОЖЕНИЕ А

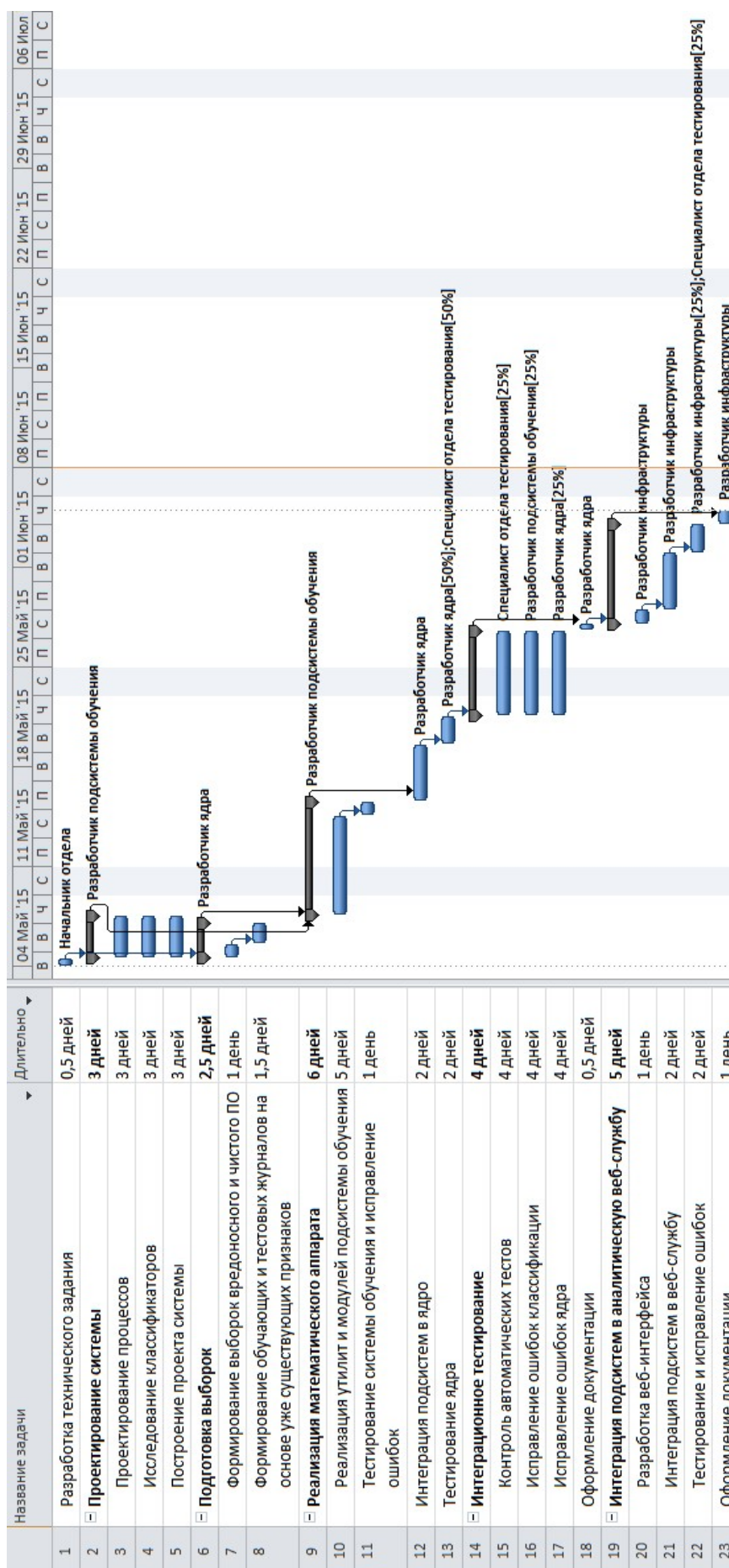


Рис. А.1. План проекта



Общество с ограниченной ответственностью

«НАНО Секьюрити»

ИНН 3255508325, КПП 325501001
241035, г. Брянск, ул. 50-й Армии, д. 6
+7 (4832) 53-41-64, +7 (905) 103-07-37
e-mail: company@nanoav.ru

Зав. кафедрой «И и ПО»
к.т.н., доц. Подвесовскому А.Г.

ЗАЯВКА

на разработку системы эвристического анализа вредоносного программного обеспечения.

Просим поручить студенту Васину А.В. разработать новую систему эвристического анализа, которая позволит:

- а) модернизировать существующий эвристический анализатор в ядре антивирусного комплекса;
- б) автоматизировать процесс настройки классификатора;
- в) реализовать веб-службу эвристического анализа для использования аналитическим отделом при разработке экспериментальных решений и прототипов;
- г) экспериментировать с различными видами классификаторов в целях дальнейшей модернизации эвристического анализатора.

Для этого необходимо реализовать подсистему обучения, которая позволит настраивать эвристический анализатор, создать подсистему анализа, встраиваемую в ядро антивируса, и предложить общий вид архитектуры веб-службы анализа. В архитектуре системы необходимо учесть возможность замены модуля классификации.

Директор
Дата



Богданов Д.Е.
2015 г.

ПРИЛОЖЕНИЕ В



Общество с ограниченной ответственностью

«НАНО Секьюрити»

ИНН 3255508325, КПП 325501001
241035, г. Брянск, ул. 50-й Армии, д. 6
+7 (4832) 53-41-64, +7 (905) 103-07-37
e-mail: company@nanoav.ru

«УТВЕРЖДАЮ»

Директор ООО «НАНО Секьюрити»

Богданов Д.Е.


« 25 » _____ 2015 г.

АКТ

Внедрения результатов дипломной работы студента Васина А.В.
«Система эвристического анализа признаков вредоносного поведения программного обеспечения»

Настоящим актом удостоверяется, что программная система, разработанная студентом Васиным А.В. в рамках выполнения дипломной работы на тему «Система эвристического анализа признаков вредоносного поведения программного обеспечения», внедрена в тестовую эксплуатацию в ООО «НАНО Секьюрити». Программная система позволит улучшить качество обнаружения вредоносного программного обеспечения и частично автоматизировать настройку эвристического анализатора.

Начальник отдела разработки системного ПО
Дата

 Подпруджников Ю.В.
« 25 » _____ 2015 г.