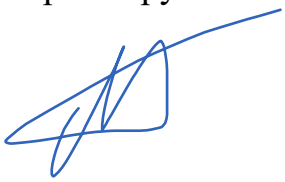
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

На правах рукописи



Успенский Михаил Борисович

Разработка и исследование методов и моделей обработки диагностической информации для обнаружения и локализации неисправностей в системах хранения данных

Специальность 05.13.01 – «Системный анализ, управление и обработка информации»

ДИССЕРТАЦИЯ

на соискание ученой степени
кандидата технических наук

Научный руководитель:
кандидат технических наук, доцент
Ицыксон Владимир Михайлович

Санкт-Петербург – 2020

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы исследования. Проблема своевременного обнаружения неисправностей в системах хранения данных (СХД) имеет в настоящее время большое значение в связи с резким ростом объема хранимой различными способами информации. По данным совместного доклада экспертов компаний IDC и Seagate, к 2025 году общий объем хранимых данных может превысить 160 зеттабайт [1]. При этом наблюдается устойчивая тенденция к увеличению доли данных, размещенных централизованно, в корпоративных и коммерческих центрах хранения и обработки данных (ЦХОД) с применением СХД различного уровня.

СХД активно используются в банковской сфере и телекоммуникационной сферах, органах государственной власти, в предприятиях оборонно-промышленного и топливно-энергетического комплексов, в сфере образования и науки для хранения персональных данных, финансовых и нормативных документов, проектной документации, обучающих материалов и т.д.

Для повышения сохранности данных и обеспечения их постоянной доступности разработчики СХД применяют все более сложные аппаратные и программные решения, направленные на создание схем избыточности и кэширования на разных уровнях топологии, поэтому современные СХД являются комплексными аппаратно-программными системами, включающими в себя множество взаимосвязанных элементов. Сложность диагностики таких систем постоянно возрастает, так как, по мере увеличения объема хранимых данных, растет сложность применяемых решений. Как следствие, растет важность своевременного и точного обнаружения неисправностей не только в носителях информации (жестких дисках, твердотельных дисках), но и в элементах, не предназначенных непосредственно для хранения данных (контроллерах хранения, элементах сетевой инфраструктуры), а также неисправностей, возникающих в результате межэлементного взаимодействия.

Для обнаружения таких неисправностей необходима разработка новых методов и моделей, обеспечивающих комплексный подход к диагностике на основе анализа расширенного множества параметров СХД. В частности, настоящая работа посвящена решению задач контроля технического состояния, а также поиска мест и определения причин возникающих неисправностей.

Степень разработанности темы исследования.

К СХД применимы актуальные подходы к решению задач диагностирования вычислительных систем, основанные на применении моделей и средств обработки данных мониторинга. Диагностическая модель при этом определяется в соответствии с ГОСТ 20911-89 как формализованное описание объекта, необходимое для решения задач диагностирования.

Известны современные работы: в области диагностики вычислительных систем с использованием моделей С. Чена [2], Л. Кейроша [3]; в области диагностики на основании анализа данных исследования Дж. Дина [4], Г. Ванга [5], Д. Дасгупты [6], Д. Ли [7], Ф. Гонсалеса [8], К. Эйрас-Франко [9]. Диагностика с использованием комбинированного подхода в различных технических объектах рассматривается в работах А. Слимани [10], Дж. Луо [11], Д. Юнга [12]. Диагностическая модель определяется в соответствии с ГОСТ 20911-89 [13] как формализованное описание объекта, необходимое для решения задач диагностирования.

Методы диагностики появления неисправностей носителей информации рассмотрены в работах, выполненных в лабораториях компаний "Майкрософт", "Гугл", "Фейсбук" и др., а также независимыми исследователями: И. Наряяном [14], Ф. Махдисолтани [15] и др.

Одним из наиболее перспективных направлений исследований является диагностика вычислительных систем с использованием журналов программного обеспечения (ПО). Различные аспекты такой диагностики рассмотрены в работах Р. Ваараанди [16], Ф. Киянга [17], Ж. Мина [18], С. Бертеро [19], С. Мессуди [20] и др.

Актуальные технические решения в области диагностики СХД отражены в пакетах программных продуктов компаний "Ай-Би-Эм", "Фуджитсу", "Эйч-Пи", "Делл", "Заббикс" и др.

Анализ состояния предметной области показал необходимость дальнейшего исследования подходов, позволяющих расширить число классов обнаруживаемых неисправностей за счёт возможности обработки разнородной диагностической информации и формализации экспертного знания о функционировании СХД, в том числе путем совместного применения диагностических моделей и методов, основанных на обработке данных мониторинга.

Целью диссертационной работы является расширение множества классов обнаруживаемых неисправностей в системах хранения данных, создание новых методов обработки и анализа диагностической информации и создание инструментальных средств для автоматизации процесса диагностирования.

Научная задача диссертационной работы заключается в разработке моделей, методов и алгоритмов, расширяющих множество возможных классов обнаруживаемых неисправностей в системах хранения данных.

Для достижения поставленных целей в работе решаются следующие задачи:

1) Анализ систем класса СХД как объекта диагностики, определение требований к методам обнаружения неисправностей в СХД и реализующему их ПО. Анализ существующих научных работ и прикладных инструментов в области диагностики вычислительных систем, выполнение сравнительного анализа их характеристик.

2) Разработка метода построения диагностических моделей для систем класса СХД, определяющих отношения между параметрами и возможными состояниями системы и ее элементов, где отношения могут быть заданы и как детерминированные связи между диагностическими сущностями, и как функции машинного обучения.

3) Разработка методов и программных средств преобразования диагностической модели к упрощенному графовому виду для её применения в составе диагностического ПО.

4) Разработка методов и инструментальных средств обнаружения неисправностей в СХД, основанных на анализе текстовой информации, получаемой в процессе мониторинга СХД методами классификации текста с использованием машинного обучения.

5) Экспериментальная проверка разработанных моделей, методов и средств на целевой платформе СХД.

Объектом исследования являются СХД и отношения между диагностическими параметрами, состоянием отдельных элементов СХД и СХД в целом.

Предметом исследования являются модели и методы обнаружения неисправностей в СХД.

Научная новизна положений, выносимых на защиту, положений, выносимых на защиту, заключается в следующем:

- В работе предложен и применен метод построения диагностических моделей систем хранения данных, основанный на использовании онтологической модели и методов обработки и анализа экспертной информации, отличающийся от существующих возможностью задания связей между объектами онтологии путем использования алгоритмов машинного обучения.

- В работе предложен и применен подход к обнаружению неисправностей в системах хранения данных, отличающийся от существующих использованием алгоритма классификации частично структурированных текстовых данных мониторинга программного обеспечения систем хранения данных, основанного на применении методов машинного обучения.

- В работе предложен алгоритм анализа, трансформации и обработки текстовой информации, получаемой в процессе мониторинга программного обеспечения систем хранения данных, отличающийся от известных тем, что позволяет обнаруживать неисправности на основании классификации частично структурированных текстов без детального анализа структуры, формата и порядка поступления сообщений мониторинга.

- В работе предлагается метод обнаружения неисправностей в системах хранения данных, отличающийся от существующих совместным применением онтологической модели и алгоритмов машинного обучения для обработки текстовой информации, получаемой в процессе мониторинга систем хранения данных.

Теоретическая и практическая значимость исследования.

Предложенный в настоящей работе подход развивает научные основы построения диагностических моделей, предназначенных для автоматического и автоматизированного обнаружения неисправностей в СХД, путем применения онтологической модели для описания отношений между состоянием объекта, его элементами и диагностическими параметрами с расширением аппарата методов онтологического моделирования, за счёт добавления возможности описания отношения между понятиями, экземплярами и атрибутами при помощи внешних процедур, реализующих применение алгоритмов машинного обучения.

Практическая значимость исследования заключается в разработке и практической реализации в рамках диагностического ПО метода обнаружения неисправностей в СХД, использование которого в процессе функционирования СХД позволит повысить надежность хранения данных и обеспечить бесперебойный доступ к ним. Полученное решение может применяться для диагностирования широкого спектра конфигураций СХД путем изменения набора элементов онтологической модели. При этом, если конфигурации СХД отличаются только применением различных количественных характеристик схем избыточности, то какие-либо действия по адаптации онтологической модели не требуются, а полученное решение сохраняет работоспособность в условиях масштабируемости СХД.

Методология и методы диссертационного исследования базируются на междисциплинарном подходе с применением методов диагностики на основании диагностических моделей и методов диагностики, основанных на анализе закономерностей в данных мониторинга, в том числе методами теории графов,

теории распознавания образов, онтологического моделирования и семантических сетей, методами обработки естественного языка с использованием средств машинного обучения.

Положения, выносимые на защиту:

1) Метод построения диагностических моделей СХД, отличающихся конфигурацией аппаратных средств, составом программных средств и параметрами использованных схем избыточности, позволяющий обнаруживать большее число типов неисправностей относительно существующих решений за счёт обеспечения возможности совместного использования диагностических параметров разного рода.

2) Метод и алгоритм анализа, трансформации и обработки текстовой информации, получаемой в процессе мониторинга СХД, позволяющий, в отличие от существующих решений, обнаруживать неисправности без детального анализа структуры данных мониторинга, формата и последовательности текстовых сообщений.

3) Комплексный метод обнаружения неисправностей в СХД, основанный на совместном использовании диагностической модели СХД и метода обработки текстовых данных мониторинга СХД с использованием алгоритмов машинного обучения, позволяющий масштабировать онтологическую модель СХД и обеспечивающий увеличение числа обнаруживаемых типов неисправностей относительно существующих средств.

Обоснованность и достоверность научных результатов научных результатов достигается за счёт использования апробированного математического аппарата, соответствия экспериментальных данных теоретическим предположениям, а также успешного применения разработанных методов и моделей в экспериментальном образце аппаратно-программного комплекса обнаружения неисправностей в СХД.

Реализация результатов работы. Результаты, полученные в настоящем исследовании, использованы для разработки опытного образца программно-

аппаратного комплекса предотвращения сбоев в СХД в ходе выполнения работ по разработке программно-аппаратного комплекса прогнозирования неисправностей, выполненных при финансовой поддержке Министерства образования и науки Российской Федерации в рамках Федеральной целевой программы «Исследования и разработки по приоритетным направлениям развития научно-технического комплекса России на 2014-2020 годы», соглашение о предоставлении субсидии от 03.10.2017 г. № RFMEFI581 17X 0023.

Апробация полученных результатов. Полученные в ходе диссертационного исследования результаты представлены на 7 российских и международных конференциях: Topical Problems of Architecture, Civil Engineering and Environmental Economics, Москва, Россия, 2018; XXIII Международная научно-практическая конференция «Системный анализ в проектировании и управлении» (г. Санкт-Петербург, Россия, 2018 г.); Международная конференция по мягким вычислениям и измерениям (г. Санкт-Петербург, Россия, 2018 г.); International Conference Cyber-Physical Systems and Control (г. Санкт-Петербург, Россия, 2019 г.); 17th IEEE International Symposium on Intelligent Systems and Informatics (г. Суботица, Сербия, 2019 г.); 33rd International Business Information Management Association Conference (IBIMA) (г. Мадрид, Испания, 2019 г.); 2019 International Scientific Conference on Energy, Environmental and Construction Engineering (EECE) (г. Санкт-Петербург, Россия, 2019 г.).

Публикации. Основные результаты по теме диссертационной работы опубликованы в 13 научных работах и приравненных к ним публикациях, в том числе 3 – в журналах из списка рекомендуемых ВАК и 5 – в журналах, индексируемых в базах SCOPUS и Web of Science. По результатам разработки практической части диссертационной работы зарегистрировано 9 программ для ЭВМ.

Личный вклад. Все результаты, представленные в настоящей диссертации, получены автором лично.

Структура и объем диссертационной работы. Диссертация состоит из введения, пяти глав, заключения и 3 приложений. Объем основной части диссертации составляет 149 страниц, полный объем диссертационной работы – 153 страницы, и включает в том числе 26 таблиц, 26 рисунков. Список литературы содержит 154 наименования.

Во **введении** обосновывается актуальность темы исследования, определяются цель и решаемые задачи, объекты и предмет исследования, формулируются положения, выносимые на защиту, их теоретическая и практическая значимость и научная новизна.

В **первой главе** приводится описание СХД как объекта диагностики с учётом особенностей реализации целевой платформы СХД. Рассмотрены основные элементы СХД, их взаимосвязь, причины возникновения неисправностей и особенности распространения ошибок в подсистемах СХД. Выполнен анализ существующих решений и актуальных научных работ в области обнаружения неисправностей в работе вычислительной техники, выполняется систематизация и классификация рассмотренных методов и приводятся их сравнительные характеристики.

На основании выявленного перечня подходов к обнаружению неисправностей в рассмотренных средствах, проведен анализ современных научных публикаций, направленный на поиск наиболее актуальных методов реализации данных подходов, в том числе перспективных алгоритмов обнаружения аномалий в диагностических данных, алгоритмов классификации и кластеризации.

По результатам анализа определена потребность в разработке новых инструментов обнаружения неисправностей, направленных на более широкое использование журналов программного обеспечения СХД (ПО СХД), в том числе с использованием онтологической модели СХД.

Во **второй главе** представлено описание метода построения диагностической модели, предназначенной для обнаружения неисправностей в СХД. Предлагается новый подход к разработке диагностической модели на

основании онтологии надежности СХД, приводится формализованное описание модели, определение ее элементов и отношений между ними. Предлагается добавление нового типа отношения, реализуемого на основании внешних процедур с использованием алгоритмов машинного обучения.

Для обеспечения возможности практического применения онтологической модели в составе программного обеспечения предложено ее приведение к упрощенному графовому виду, предполагающему преобразование традиционного rdf к формату rdf-nquad и описание модели в формате [узел]-<связь> [узел] [контекст]. Такое преобразование позволяет устранить громоздкие конструкции наследования, применяемые в формате построения онтологии OWL и свести их к упрощенному виду.

В **третьей главе** рассматривается алгоритм по динамическому обнаружению неисправностей в СХД, предназначенный для реализации онтологической условной связи, рассматривающий в качестве входных параметров блоки сообщений из журналов СПО СХД.

В **четвертой главе** описывается онтологическая диагностическая модель, архитектура программного обеспечения для обнаружения неисправностей в СХД и практические результаты применения предложенных методов обнаружения неисправностей.

В **пятой главе** приводятся практические результаты применения предложенных методов обнаружения неисправностей, делаются выводы об эффективности предложенных решений.

1 ОБЗОР МЕТОДОВ И СРЕДСТВ ДИАГНОСТИРОВАНИЯ НЕИСПРАВНОСТЕЙ СХД

1.1 Постановка задачи обзора

В настоящей диссертационной работе рассматриваются две ключевые задачи диагностирования технических систем: обнаружение неисправностей и локализация неисправностей. В рамках данных задач выполняется определение технического состояния системы, характеристика текущего технического состояния системы и определение возможных причин возникновения неисправностей. Под неисправностью СХД при этом понимается неисправность ее отдельного элемента, отдельной подсистемы или СХД в целом.

Настоящий аналитический обзор рассматривает существующие методы обнаружения и локализации неисправностей в области вычислительной техники, реализованные в актуальных программных продуктах, которые применяются или могут применяться для решения задачи обнаружения и локализации неисправностей в СХД. Для определения современного технического уровня выполнен обзор актуальных научных публикаций, предлагающих перспективные методы и средства, которые могут быть применены для эффективной реализации данных подходов. Полученная в результате анализа оценка достоинств и недостатков этих методов и средств использована в процессе разработки оптимального подхода к обнаружению неисправностей для СХД.

1.2 СХД как объект диагностики

Целевой системой диссертационной работы является линейка (СХД) на платформе Tatlin, предоставленная для анализа в рамках исследования, выполненного при финансовой поддержке Министерства образования и науки Российской Федерации в рамках Федеральной целевой программы «Исследования и разработки по приоритетным направлениям развития научно-технического комплекса России на 2014-2020 годы» (соглашение о предоставлении субсидии от

03.10.2017 г. № RFMEFI581 17X 0023). Разработка методов и алгоритмов в рамках настоящего исследования велась таким образом, чтобы обеспечить возможность их адаптации для различных конфигураций СХД того же класса.

СХД является сложным программно-аппаратным комплексом, предназначенным для организации бесперебойного доступа и надежного хранения пользовательских данных [21]. СХД на платформе Tatlin позиционируется производителем как СХД среднего уровня (mid-range) [22], обладающая следующими особенностями реализации:

- унифицированный доступ к данным (блочный, файловый и объектный);
- программная реализация избыточности с использованием кодирования Рида-Соломона, применяемых для исправления ошибок в блоках данных[23];
- в качестве контроллеров хранения используются серверы YADRO VESNIN, использующие процессоры архитектуры OpenPOWER[24];
- одновременное использование дисков разных типов и разных интерфейсов: твердотельных (SAS, NVMe) и механических (SAS, NL-SAS, SATA);
- active-active режимы работы контроллеров.

Обобщенная структурная схема СХД такого класса представлена на рисунке 1.1:

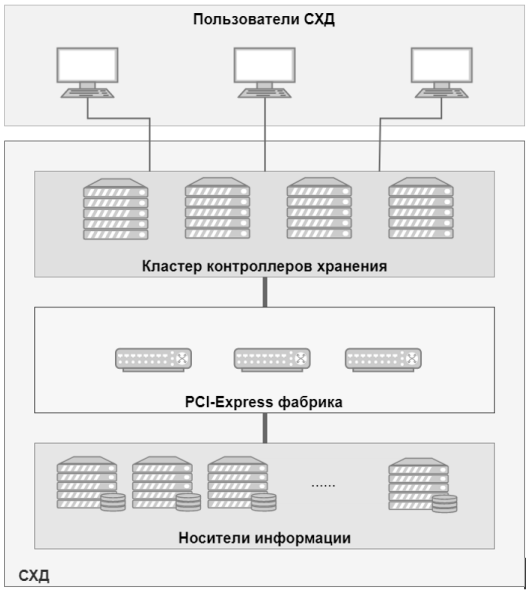


Рисунок 1.1 – Обобщенная структурная схема СХД

В данной структуре можно выделить три основных слоя:

1) Слой контроллеров хранения, содержащий кластер вычислительных узлов, предназначенный для организации и управления доступом к данным. Использование нескольких параллельно включенных контроллеров хранения позволяет обеспечить балансировку нагрузки и повышает надежность сервиса доступа к данным, так как каждый из контроллеров одновременно имеет доступ ко всем носителям информации СХД.

2) Слой фабрики включает в себя набор сетевого оборудования и предназначен для обеспечения взаимодействия контроллеров хранения и носителей информации.

3) Слой носителей информации содержит дисковые шасси, в которые вставляются носители информации.

Контроллеры хранения и **контроллер фабрики** объединены при помощи внутренней управляющей сети СХД.

Кроме того, важным компонентом СХД является его программное обеспечение (ПО), решающее задачу обеспечения хранения данных и обеспечения к ним доступа на программном уровне. Так, например, при помощи ПО решается задача построения логических сущностей, таких как пулы хранения (storage pool) и логические тома (storage volume) [25].

Пул хранения представляет собой объединение физических дисков в единую логическую неразмеченную емкость, которую далее можно разделить на логические тома. На уровне организации пула хранения применяется схема резервирования.

Логический том виден пользователю как единое дисковое пространство, при этом он объединяет несколько физических дисков и его размер не должен быть кратен размеру физического диска. Логический том защищен от потерь, возникающих из-за неисправностей физических дисков за счет резервирования, реализованного в пуле хранения.

Упрощенная (без учета затрат на резервирование) схема построения пула хранения и логических томов представлена на рисунке 1.2.

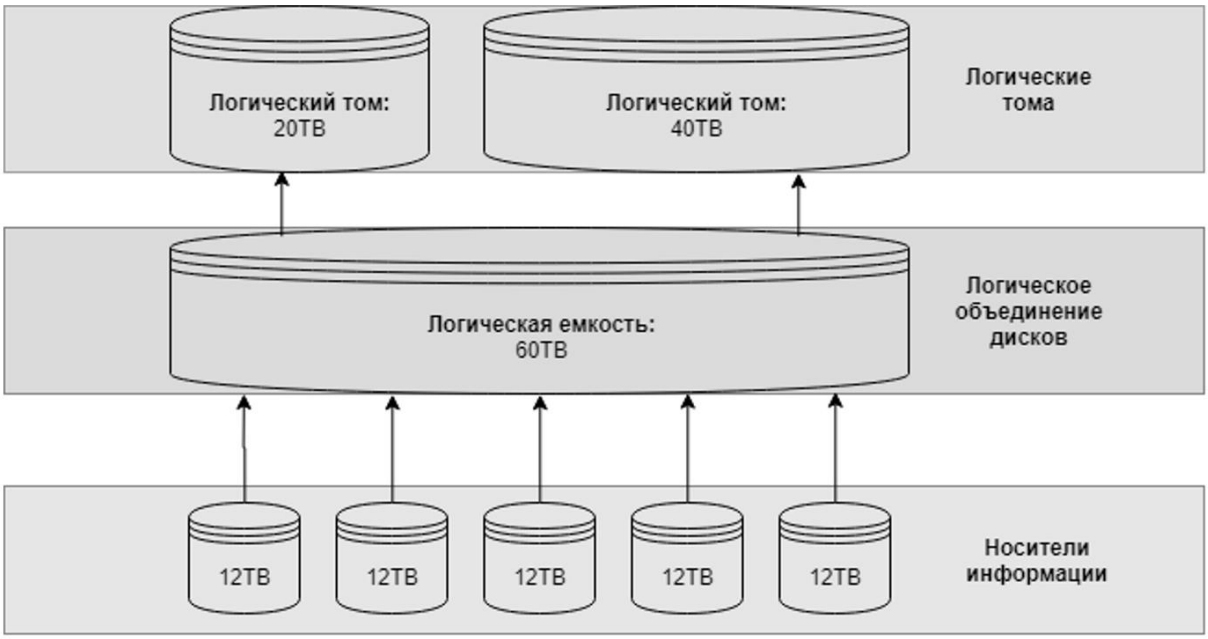


Рисунок 1.2 – Логическое объединение дисков в единую емкость

Таким образом, решение задач обнаружения и локализации неисправностей в СХД разделяется на решение задач диагностики отдельных компонентов СХД и диагностики их взаимодействия. На физическом уровне диагностируются:

- контроллеры хранения (как серверные ЭВМ, состоящие из набора аппаратных компонентов);
- сетевое оборудование;
- носители информации на физическом уровне (в т.ч. диагностика с использованием данных SMART).

На программном уровне диагностируются:

- работоспособность отдельных сервисов СПО СХД;
- работоспособность логических сущностей, предназначенных для организации хранения данных.

Важной особенностью рассматриваемой платформы является ее гибкая структура: в рамках исследования рассматривалось пять возможных конфигураций СХД (Приложение А), отличающихся друг от друга различным качественным и количественным составом компонентов. Такая особенность накладывает

определенные ограничения на разрабатываемое диагностическое решение, так как требуется обеспечить его адаптацию к изменениям состава компонентов СХД.

1.3 Технологии диагностирования СХД

Существующие научные работы в области, связанной с хранением данных, ориентированы, главным образом, на исследование различных аспектов диагностирования носителей информации, в особенности касающихся данных SMART. Можно проследить эволюцию методов, применяемых для решения данной задачи: более ранние посвящены изучению корреляции между изменением наблюдаемых параметров SMART и состоянием диска при помощи статистических методов [26] (иногда с фактическим вырождением до разработки набора правил вида «если параметр X имеет значение A, то…» [27]), тогда как более поздние рассматривают применение различных методов машинного обучения [15]. Результаты, полученные в данных научных работах, в особенности [28], [14], [29],[30] позволяют утверждать, что сформулированная таким образом задача рассмотрена исчерпывающе и не требует на текущий момент дополнительной проработки.

Однако, современная СХД, как показано в разделе 1.2, представляет собой систему, в общем случае гораздо более комплексную, чем просто набор жестких дисков, объединенных одним контейнером, причем существует тенденция к дальнейшему усложнению ее архитектуры. Это объясняется необходимостью обеспечивать с одной стороны сохранность данных, а с другой – бесперебойный доступ к ним, для чего применяются различные схемы обеспечения избыточности на уровне как отдельных компонентов, так и целых подсистем. Другой важной особенностью, влияющей на процесс обнаружения неисправностей, является необходимость обеспечивать гибкую архитектуру СХД, позволяющую за счет варьирования состава программных и аппаратных средств в каждом конкретном случае реализовывать систему, наиболее полно удовлетворяющую текущим техническим требованиям.

Все вышеперечисленное приводит к тому, что, во-первых, необходимо диагностировать не только носители информации, но и прочие активные и пассивные компоненты, в том числе программные, а во-вторых, еще и обнаруживать ошибки, возникающие в процессе межкомпонентного взаимодействия. В результате, совершенствуются и усложняются реализуемые подходы к обнаружению неисправностей, обеспечивающие различными способами решение задачи обнаружения неисправностей СХД, как сложного программно-аппаратного комплекса [31, 32].

Сформулированная таким образом задача по обнаружению неисправностей в СХД традиционно решалась в большей степени на прикладном уровне, что отражается в значительном количестве существующих на текущий момент программных и программно-аппаратных инструментов, как разработанных специально для СХД, так и применяемых для диагностики вычислительных систем в целом (например, Д. Хэйден в [33] упоминает 100 инструментов для мониторинга серверов и рабочих мест, многие из которых можно использовать для мониторинга и обнаружения неисправностей в СХД). Для оценки полноты существующих решений в настоящей работе проведен их детальный анализ, в ходе которого рассмотрены три основные группы программных средств – специализированные средства мониторинга СХД, средства мониторинга общего назначения и средства анализа данных. Кроме того, дополнительно рассмотрен набор программных пакетов, различные варианты совместного использования которых обеспечивают комплексный мониторинг и диагностику вычислительных систем, в том числе СХД. Специализированные средства диагностики, предназначенные для обнаружения неисправностей отдельных компонентов СХД, выходят за рамки настоящей работы.

По результатам анализа программных продуктов, рассмотренных в данном разделе, составлены сводные таблицы (см. таблицы №1-3), наглядно демонстрирующие их особенности. В таблицах перечислены следующие признаки, характеризующие программные продукты:

- Наименование программного комплекса;

- Открытый или закрытый исходный код программного комплекса, способ распространения (“+” – открытый/бесплатное распространение, “-“ – закрытый/продажа);

- Наличие/отсутствие встроенных средств сбора данных (“+” – есть, “-“ - нет, “-/+” – является частью программного пакета, в котором такие средства есть);

- Представленные в программном комплексе средства обнаружения неисправностей;

- Наличие в системе средств локализации неисправностей;

- Инструменты построения диаграмм и графиков для визуального контроля метрик при решении задачи обнаружения неисправностей присутствуют во всех рассмотренных программных средствах, поэтому не отражены в столбце 6 таблиц 1-3.

В процессе выполнения обзора не ставилась задача рассмотреть все существующие программные продукты (так как их число крайне велико, а набор возможностей зачастую повторяется). Вместо этого выделены наиболее часто используемые инструменты, обладающие различными характерными особенностями.

1.3.1 Специализированные средства обнаружения неисправностей в СХД

В данном разделе рассмотрены два разных типа решений: специализированные, предоставляемые крупнейшими производителями СХД и универсальные, предназначенные для мониторинга состояния отдельных моделей или целых линеек СХД, предоставляемые сторонними производителями.

К первой группе относятся инструменты, предоставляемые компаниями:

- HP (пакеты HPE InfoSight [34], HPE Active Health System (AHS) (мониторинг и сбор данных) и HPE iLO [35], реализующие функциональность по мониторингу, сбору и анализу данных, отправке уведомлений и удаленному управлению сервером),

- Dell (программа Dell CloudIQ [36] для сбора и анализа данных);

- NetApp (программа Active IQ для агрегации данных, поступающих от СХД NetApp с целью оценки состояния и построения прогнозов состояния СХД [37]);

- IBM (встроенные инструменты управляющего ПО СХД IBM Spectrum Control, специализированные сервисы Storage Insights и IBM Operations Analytics [38]);

- Hitachi (ПО Hitachi Storage Advisor [39] для сбора и анализа данных);

- Fujitsu (ПО FUJITSU Storage ETERNUS SF MA [40] для сбора и простого анализа данных).

Как правило, такие решения являются составной частью встроенного программного обеспечения СХД. Результаты проведенного анализа приведены в сводной таблице 1.

Таблица 1 – Специализированные диагностические средства для СХД

№п.п	Наименование программного комплекса	Открытый исходный код	Встроенные средства сбора данных	Средства обнаружения неисправностей	Средства локализации неисправностей
1	2	3	4	5	6
1.	HPE InfoSight	-	-/+	Корреляционный анализ, предиктивная аналитика, алгоритмы машинного обучения для поиска решения проблем, прогнозирование потребности в свободном месте и проблем с производительностью	+
2.	EMC® Storage Monitoring and Reporting	-	+	-	-

Продолжение таблицы 1

1	2	3	4	5	6

3.	Dell CloudIQ	-	-/+	Определение аномалий с использованием машинного обучения, прогнозирование потребности в свободном месте и проблем с производительностью.	+
4.	NetApp Active IQ	-	-/+	Определение состояния здоровья компонентов СХД, диагностика производительности, конфигурации и наличии свободного места в СХД	+
5.	IBM Spectrum Control + Storage Insights + IBM Operations Analytics	-	-/+	Определение состояния здоровья компонентов СХД, определение аномалий с использованием машинного обучения, диагностика и прогнозирование производительности, конфигурации и наличии свободного места в СХД.	+
6.	Hitachi Storage Advisor	-	+	Формирование уведомлений о проблемах производительности, наличия свободного места и т.д. на основании пороговых значений	+
7	FUJITSU Storage ETERNUS SF MA	-	+	-	-

Ко второй группе относятся программные пакеты:

- AppDynamics Storage Performance Monitoring [41] (ПО, предназначенное для мониторинга и обнаружения проблем производительности в различных СХД компании NetApp);

- IntelliMagic Vision for SAN [42] (предназначено для анализа производительности фабрик и компонентов SAN с прогнозированием внештатных ситуаций);

- SevOne Storage Monitoring [43] (предназначено для отслеживания и анализа метрик производительности и формирования прогноза свободного места СХД);

- Solarwind Storage Resource Monitor [44] (система управления ресурсами СХД, включающая в себя набор средств по мониторингу здоровья);

- Centry Storage Monitoring [45] (система для мониторинга производительности и здоровья SAN СХД, предназначенная для помощи в диагностике проблем с доступностью СХД);

- eG Enterprise Storage Performance Monitoring[46] (комплексная система мониторинга производительности и здоровья СХД, включающая в себя комплекс средств агентного и безагентного мониторинга и модулей анализа собранных данных);

- VirtualWisdom[47] (платформа для управления ресурсами СХД уровня дата-центра. Наибольший интерес представляет наличие программно-аппаратных средств для сбора метрик здоровья и производительности СХД. Аппаратные агенты мониторинга (hardware probes) предназначены для мониторинга сетевой инфраструктуры путем анализа заголовков FC кадров и SCSI команд линков FibreChannel в SAN в режиме реального времени).

Результаты анализа приведены в сводной таблице 2.

Таблица 2 – Универсальные диагностические средства для СХД

№п.п	Наименование программного комплекса	Открытый исходный	Встроенные средства сбора данных	Средства обнаружения неисправностей	Средства локализации неисправностей
1	2	3	4	5	6
1.	AppDynamics Storage Performance Monitoring	-	+	-	+

Продолжение таблицы 2

1	2	3	4	5	6

2.	Centry Storage Monitoring	-	+	Отображение здоровья отдельных компонентов СХД с формированием предупреждений о внештатных ситуациях, формирование уведомлений по пороговым значениям; формирование прогноза потребности в свободном месте на дисках и проблем с производительностью	+
3.	eG Enterprise Storage Monitoring	-	+	Мониторинг конфигурации СХД, формирование уведомлений на основании пороговых значений и наличие политик формирования уведомлений, корреляционный анализ событий с учетом взаимосвязей между компонентами СХД, автоматическое ранжирование уведомлений по уровню критичности, обнаружение аномалий производительности СХД	+
4.	IntelliMagic Vision for SAN	-	+	Автоматическое выставление оценок здоровья значений метрик в соответствии с набором правил	+
5.	PRTG Network Monitor Storage Monitoring	-	+	Обнаружение сбойных дисков, формирование прогноза потребности в свободном месте на дисках и проблем с производительностью	+
6.	SevOne Storage Monitoring	-	+	Прогнозирование потребности в свободном месте на дисках и проблем с производительностью, обнаружение нормального поведения системы	+
7.	Solarwind Storage Resource Monitor	-	+	Отображение здоровья отдельных компонентов СХД с формированием предупреждений о внештатных ситуациях, формирование уведомлений по пороговым значениям	+

Продолжение таблицы 2

1	2	3	4	5	6

8.	VirtualWisdom	-	+	Корреляционный анализ событий с учетом взаимосвязей между компонентами СХД с использованием алгоритмов машинного обучения и статистического анализа, отображение здоровья отдельных компонентов СХД, обнаружение аномалий производительности СХД, формирование прогноза потребности в свободном месте на дисках и проблем с производительностью	+

Диагностические программные средства для СХД, как специализированные, так и универсальные, обязательно предоставляют решения для сбора метрик состояния СХД, обнаружения неисправностей в СХД, обнаружения причин возникновения неисправностей в СХД. При этом, поскольку такие средства, как правило, имеют закрытый исходный код и только в общих чертах раскрывают архитектурные и алгоритмические решения, позволяющие обеспечить данную функциональность, отсутствует возможность провести независимый аудит качества получаемых результатов, в том числе оценить определить необходимость и достаточность собираемых метрик и, соответственно, вероятность корректного обнаружения неисправностей и причин возникновения неисправностей. При этом программные средства, предоставляемые производителями СХД, имеют более слабое алгоритмическое обеспечение по обнаружению неисправностей, но за счет наличия у разработчиков экспертного знания о предмете анализа, обладают, как правило, более совершенными инструментами их локализации. Дополнительным ограничением как специализированных (в большей степени), так и универсальных (в меньшей степени) программных средства для диагностики СХД является условие их применимости либо на конкретных моделях СХД, либо линейках моделей СХД конкретного производителя/конкретных производителей.

1.3.2 Диагностические средства общего назначения

В данном разделе рассмотрены диагностические средства, предназначенные для мониторинга вычислительных систем, которые могут быть использованы для

мониторинга СХД, такие как **ElasticStack**[48] (включает в себя набор программных пакетов, позволяющих организовать мониторинг и диагностику СХД на базе поисковой платформы ElastisSearch), **Nagios** [49] **(о**дин из наиболее известных программных пакетов для мониторинга IT-инфраструктуры, имеет в своем составе набор инструментов для организации мониторинга здоровья и производительности СХД) и др. Результаты приведены в сводной таблице 3.

Таблица 3 – Диагностические средства общего назначения для СХД

№п.п	Наименование программного комплекса	Открытый исходный код	Встроенные средства сбора данных	Средства обнаружения неисправностей	Средства локализации неисправностей
1	2	3	4	5	6
1.	ElasticStack	+	+	Настраиваемые уведомления на основании пороговых значений, политик уведомлений и с использованием алгоритмов машинного обучения; выявление аномалий с использованием алгоритмов машинного обучения	-
2.	Nagios	+	+	Формирование уведомлений, формирование прогноза потребности в свободном месте на дисках СХД, формирование прогноза проблем производительности	-
3.	NetData[50]	+	+	Формирование уведомлений для заданных событий, пороговых значений, наличие политик формирования уведомлений, инструменты для визуального контроля возникновения аномалий	-

Продолжение таблицы 3

1	2	3	4	5	6

4.	Okmeter[51]	-	+	Обнаружение неисправностей в соответствии с набором, заранее предопределенном сценарием	+
5.	Zabbix[52]	+	+	Формирование уведомлений для заданных событий, пороговых значений, наличие политик формирования уведомлений, наличие средств выявления аномалий на основании машинного обучения;	-

Диагностические средства общего назначения чаще всего имеют открытый исходный код, более ограниченные, в общем случае, возможности по сбору метрик здоровья и производительности, которые, однако, могут расширяться при помощи плагинов (кроме того, ElasticStack предоставляет возможности извлечения метрик из журналов ПО СХД). Они имеют относительно слабо развитые средства обнаружения неисправностей в СХД и причин их возникновения (хотя ElasticStack и Zabbix содержат программные модули обнаружения аномалий, использующие алгоритмы машинного обучения). При этом разные диагностические средства общего назначения имеют специфические достоинства и недостатки архитектурных и алгоритмических решений (например, Zabbix хранит всю историю наблюдений в одной базе данных, Nagios сравнительно сложен в установке и использовании, но обладает мощной поддержкой плагинов, и т.д.). Несмотря на то, что их можно использовать практически с любой СХД, их развертывание требует больших трудовых затрат. Для эффективного применения может потребоваться доработка программного кода. Дополнительным недостатком таких систем является избыточность, обусловленная их общим назначением.

1.3.3 Средства анализа данных

Средства анализа данных общего назначения предназначены для математического анализа данных произвольного характера. Для применения таких средств для обнаружения неисправностей в работе СХД необходимо организовать сбор метрик СХД и предварительное обучение алгоритмов определения аномалий. Средства анализа данных, входящие в состав прочих программных платформ,

такие как встроенные инструменты Tick Stack и ElasticStack, описаны в предыдущем разделе и в текущем разделе не рассматриваются.

Наиболее интересными из существующих решений являются:

- Anomaly.IO [53] **(**Облачный сервис для определения аномалий. На начальном этапе выполняется обучение алгоритмов платформы для определения различных закономерностей в поступающих метриках, таких как минимумы/максимумы, тренды, корреляции и д.р. с использованием алгоритмов машинного обучения, нейронных сетей, средств теории распознавания образов, статистических моделей и собственных математических моделей. Используя выявленные закономерности Anomaly.IO в дальнейшем определяет необычное поведение поступающих метрик);

- DatastreamIO [54] **(**Перспективная платформа определения аномалий с открытым кодом. Реализует функциональность по получению метрик от внешних источников, преобразованию полученных метрик в интересующие, такие как агрегации, счетчики, сессии, группы и т.д., построению модели нормального поведения собираемых метрик при помощи машинного обучения без учителя, выявлению аномалий в полученных метриках, визуализации аномалий на диаграммах и графиках).

Преимуществом средств анализа данных является использование ими большого количества разнообразных алгоритмов выявления аномалий, в том числе вынесенных в сферу облачного вычисления, что позволяет экономить вычислительные ресурсы СХД. При этом алгоритмы выявления аномалий сами по себе не обладают знаниями о топологии и устройстве конкретных СХД, и, следовательно, в общем случае не подходят для определения типов возникающих неисправностей, путей распространения ошибки по топологии системы, конкретных компонентов в которых возникла неисправность, планирования потребности в свободном месте на диске и определения узких мест в производительности СХД. С точки зрения решения задач диагностики, это означает, что они эффективны для решения проблемы обнаружения факта

возникновения неисправности, но плохо подходят для определения ее причины. Состав исходных данных полностью определяется пользователем, поэтому полнота и охват метрик в каждом конкретном случае зависит от квалификации административного персонала. Anomaly.Io, кроме того, является облачным средством, а следовательно, также чувствительна к ограничениям на доступ к данным.

1.3.4 Модульный подход к диагностированию СХД

В настоящее время широкое распространение получила схема организации мониторинга здоровья вычислительных средств, при котором отдельные модули структуры, описанной в разделе 1.4, реализуются разными программными пакетами. В настоящем разделе приведены наиболее часто использующиеся пакеты, которые можно применять для решения такой задачи. Перечисленные решения распространяются свободно и обладают открытым исходным кодом и встроенными средствами информационного обмена с прочими сервисами. Кроме того, как правило, для них существует множество плагинов, дополняющих и модифицирующих их функциональность.

К ним относятся такие инструменты, как:

- Bosun [55] (система мониторинга и уведомлений);

- Cacti [56] (программный инструмент для сбора метрик и построения диаграмм и графиков);

- CollectD [57] (сбор статистики системы и приложений и передача ее в подключаемое хранилище);

- Graphana [58] (система построения диаграмм и графиков);

- Graphite [59] (хранение и визуализация временных рядов);

- InfluxData TICK Stack [60] (комплексное решение, при этом взаимодействующее с множеством сторонних программных пакетов. Реализует функциональность по сбору и агрегации метрик, хранению временных рядов, отображению диаграмм и графиков и отправки уведомлений, анализу данных, в

том числе с использованием средства машинного обучения (теории распознавания образов) и статистического анализа для отправки уведомлений и выполнения действий по поступившим уведомлениям, таких как динамическая балансировка нагрузки);

- StatsD [61] (агрегация метрик);

- Prometheus [62] (мониторинг, хранение данных, отправка уведомлений, построение диаграмм и графиков);

- Sensu [63] (конвейер событий мониторинга).

Вышеперечисленные программные пакеты используются как по отдельности, так и в различных сочетаниях. Так Мишель Тэн в [64] описывает опыт совместного использования StatsD для сбора данных, Sensu для управления сбором метрик и формированием уведомлений, InfluxDb для хранения временных рядов метрик и Graphana для построения диаграмм и графиков изменения метрик. Дж. Эллингвуд в [65] описывает организацию мониторинга статистики хостинга при помощи StatsD (отправка метрик), CollectD (сбор метрик) и Graphite (хранение метрик).

1.3.5 Классификация программных средств диагностирования СХД

Для систематизации полученной информации, в ходе данного анализа выявлен набор признаков, позволяющих построить их классификацию. К ним относятся: область применения СХД, способ получения диагностических данных, тип источника диагностических данных, применение агентов, способ размещения диагностического ПО.

По области применения программные средства подразделяются на:

- Диагностические программные средства для отдельных компонентов СХД.

Сюда относятся, например, средства мониторинга состояния сетевой инфраструктуры, жестких дисков и т.д. Как правило, при решении задачи обнаружения неисправностей в СХД корпоративного уровня используются в качестве источников данных для более комплексных средств мониторинга.

- Диагностические программные средства для СХД в целом.

Специализированные программные средства, такие как IBM Spectrum Control, FUJITSU Storage ETERNUS SF MA и т.д. разрабатываются производителем для поставляемых СХД, например, в составе программных средств управления ресурсами конкретных типов СХД. Универсальные программные средства, предлагаемые независимыми разработчиками, такие как Solarwind Storage Resource Monitor, SevOne Storage Monitoring, применяются соответственно для отдельных моделей или линеек моделей СХД различных производителей;

- Диагностические средства общего назначения.

Диагностические средства общего назначения (Zabbix, ElasticStack и др.), как правило, предназначены для мониторинга и диагностики ЭВМ в целом, но используются также в задачах мониторинга и обнаружения неисправностей в работе СХД.

- Средства анализа данных (Anomaly.io и др.).

Представляют собой программную платформу для выявления аномалий в данных, предоставляемых пользователем. В том числе на вход могут подаваться и данные мониторинга СХД в явном виде или предварительно обработанные. Задача установки соответствия между классами выявляемых аномалий и состояниями СХД ложится на оператора СХД.

По способу реализации сбора диагностических данных:

- Без средств сбора данных.

Такие системы как Anomaly.io, получают данные от набора плагинов для различных источников информации;

- С программным сбором данных.

Такие системы как Zabbix, Cacti и др. обращаются исключительно к источникам информации, предоставляемым программным обеспечением СХД: данные о производительности СХД, наличии свободного места в СХД, состоянию процессов на контроллерах хранения (например, предоставляемые утилитами

iostat, pidstat), информационным ресурсам устройств (SMART данные носителей информации, температура) и т.д;

- С программно-аппаратным сбором данных.

В состав мониторингового комплекса VirtualWisdom могут входить аппаратные щупы (hardware probe), предназначенные для подключения в различных точках сетевой инфраструктуры СХД, передающие данные в единый центр обработки информации.

Системы с использованием или без использования агентов:

- С использованием агентов (agent-based monitoring).

На вычислительные узлы СХД устанавливаются специальные программные агенты мониторинга, собирающие какие-либо метрики;

- Без использования агентов (agentless monitoring, безагентный мониторинг).

В этом случае специализированные программные сборщики данных на объектах мониторинга не используются, а опрашиваются внутренние источники данных по специализированным протоколам, например SNMP[66], SMI-S[67] и др., а также по SSH[68], WMI[69], Telnet[70] и др.

По характеру анализируемых данных:

- Анализ журналов ПО СХД (ElasticStack).

Такое ПО как ElasticStack извлекает метрики для оценки состояния СХД из записей в журналах, отбираемых по определенным шаблонам;

- анализ метрик мониторинга (Zabbix).

Zabbix, Cacti, Nagios и др. извлекают метрики для оценки состояния СХД из любых доступных источников.

По расположению диагностического компонента системы мониторинга:

- Диагностический компонент системы мониторинга находится внутри СХД, например, на одном из контроллеров хранения;

- Диагностический компонент находится снаружи СХД в локальной сети или удаленно.

1.3.6 Методы диагностирования неисправностей СХД

На основании результатов анализа и построенной классификации, в рассмотренных программных комплексах выявлены следующие подходы к обнаружению неисправностей:

- Обнаружение неисправностей на основании заданного списка событий (например, реализовано в программном обеспечении IBM Spectrum Control) или заданных пороговых значений (метрика некоторым образом сравнивается с пороговыми значениями, в случае выхода значения метрики за пределы пороговых значений формируется уведомление администратору, см., например, FUJITSU Storage ETERNUS SF MA). Существует множество способов задания пороговых значений - в скалярном, векторном или функциональном виде;
- Выявление аномалий. В этом случае метрики анализируются на предмет наличия данных, каким-либо образом отличающихся от прочих. Решение задачи выявления аномалий может осуществляться разными способами, в простейшем случае путем отображения временных рядов получаемых метрик на графиках для визуального контроля. Более сложные решения подразумевают использование методов машинного обучения (например, с использованием методов теории распознавания образов), корреляционного анализа, статистического анализа и т.д. Выявление аномалий в том или ином виде является основным способом обнаружения неисправностей в большинстве существующих решений, включающих интеллектуальные средства анализа данных, в том числе Zabbix, Elastic, Anomaly.io, Tick Stack и др.;
- Сопоставление топологии системы с получаемыми данными мониторинга с определением перечня компонентов по пути прохождения данных и локализация неисправностей с использованием методик анализа коренных причин (root case analysis) на основании данных мониторинга и выявленных аномалий. Реализовано во многих системах с различной полнотой охвата в

автоматическом или автоматизированном режимах, например, Hitachi Storage Advisor.

Таким образом, по результатам проведенного обзора и сравнения возможностей существующих программных средств можно сделать вывод, что, хотя инструменты мониторинга позволяют выполнять сбор различного типа данных о функционировании СХД из широчайшего спектра возможных источников, возможность их совместного использования (особенно в части сочетания в одном решении учета как численных метрик, так и неструктурированных данных журналов ПО СХД) для автоматического и автоматизированного обнаружения неисправностей в рамках одного диагностического решения достаточно ограничена. Потребность в разработке решений, позволяющих использовать разнородные источники информации о состоянии СХД, обусловлена комплексной структурой современных СХД, и, соответственно, более сложными причинами возникновения неисправностей. Другой выявленной проблемой является ограниченная функциональность существующих средств в части автоматического обнаружения неисправностей – применяемые в них подходы в основном предполагают применение средств обнаружения аномалий.

Для поиска решения этих проблем выполнен анализ предметной области в двух направлениях: анализ существующих подходов к обнаружению неисправностей в технических объектах и анализ существующих подходов к обработке диагностических данных из журналов программного обеспечения.

1.4 Подходы к решению задачи обнаружения неисправностей в технических объектах

В дополнение к рассмотренным методам обнаружения неисправностей в СХД, рассмотрен ряд актуальных научных публикаций, посвященных более широкой проблеме диагностирования неисправностей в различных технических объектах, подходы из которых можно было бы применить для диагностирования

СХД. Обнаружение неисправностей в рассмотренных публикациях выполняется на базе двух основных подходов – на основании построения диагностической модели и на основании анализа данных мониторинга.

Методы обнаружения неисправностей, основанные на решении задач классификации или кластеризации, являются наиболее распространенными в современных работах, посвященных обнаружению неисправностей в области диагностики вычислительных устройств или устройств хранения данных. К ним относятся предложенный Б.Коста и др. в [71] подход к диагностике неисправностей в реальном времени, основанный на рекурсивной оценке плотности; использование адаптивной ядерной оценки плотности для определения выбросов в нелинейных системах, предлагаемое Г. Жанг и др. в [72]; метод организации анализа сетевого потока на основе кластеризированных образов Дж. Кима и др., описанный в [73]; комбинированный подход Д.Юнга и др., использующий совместно модельный подход к диагностике и классификаторы аномалий для локализации неисправностей, описанный в [12]. А. Алказем и др. в [74] рассматривают прикладной случай решения задачи обнаружения неисправностей на основании метрик производительности компонентов виртуальных машин с использованием платформы анализа больших наборов данных Hadoop MapReduce совместно с наивным классификатором Байеса в сервисе предоставления виртуальных машин и облачных хранилищ. Л. Ю и З. Лан в [75] предлагают масштабируемый непараметрический метод, предназначенный для поиска аномалий производительности в вычислительных системах большого размера, основанный на методе сравнения узлов (описывается, например, в публикации [76] З. Лан, З. Жен и Ю. Ли).

Другой большой группой являются методы, основанные на **сетях инвариантов**. З. Ванг, Ю Ли в [77] предлагают подход к определению неисправностей в программном обеспечении, использующий рассогласование динамических инвариантов. У. Чен и др. предлагают подход по определению

причинно-следственных аномалий при помощи сетей инвариантов с использованием временного и динамического анализа исчезающих корреляций [2].

В исследовании Г. Ванг, Дж. Ванг [5] описывается случай применения **метода опорных векторов** для определения аномальных виртуальных машин в облачной среде с целью предотвращения деградации производительности. С. Инь, З. Жу, С. Джинг предлагают модификацию метода опорных векторов для одного класса, направленную на решение проблемы уязвимости метода из-за выбросов в обучающей выборке [78].

Искусственные иммунные системы – это вычислительные системы, основанные на принципах работы иммунной системы. Они включают в себя в том числе негативный алгоритм отбора, иммунный сетевой алгоритм и пр. Использование иммунных систем в качестве основы для алгоритмов выявления аномалий обосновывается тем, что биологические иммунные системы так же, как и алгоритмы выявления аномалий, нацелены на выявление информации, отличной от нормальной [6], [80].

Д. Ли, С. Лю, Х. Жанг в [60] предлагают использовать для обнаружения неисправностей две разновидности алгоритма негативного отбора – FB-NSA (fixed boundary NSA) и его доработанный FFB-NSA (Fine Fixed Boundary NSA). Принципиальным отличием их от прочих работ в области алгоритмов отрицательного отбора (например, Ф. Гонсалесом, Дж. Гомесом, Д. Дасгуптой в [8] предложен детектор в виде гиперкуба, Дж. Шапиро, Г. Ламонт и Г. Петерсен в [80] предложен детектор в виде гиперэллипсоида, С. Балашандран, Д. Дасгупта, Ф. Нино и др. в [81] предложен детектор с множественной формой) заключается в том, что он направлен не на исследование формы детекторов, а на то, чтобы добиться их неизменности.

К **прочим методам** относятся предложенный З. Чен, Дж. Ванг, Дж. Жу в [82] метод обнаружения неисправностей и определения источника их возникновения на основе графической вероятностной модели; подход к диагностике жестких дисков на основании полупараметрических моделей и статистических оценок Л. Кейрош

и др., описанный в [3]; алгоритм, предложенный Дж. Дингом в [4], основанный на латентной корреляционной вероятностной модели, предназначенный для поиска аномалий в данных мониторинга состояния оборудования. С. Плаикас и И. Буталис в [83] предлагают подход к обнаружению неисправностей для одного класса, основанный на использовании генеративно-состязательных сетей [84]. Д. Мак и др. в [85] рассматривают прикладную задачу выявления ранее не выявленных аномалий в полетных данных самолетов, собранных в единой базе данных. Для решения данной задачи предлагается использовать комбинированный подход с использованием методов формализации экспертного знания и методов машинного обучения без учителя. Полученный в итоге алгоритм имеет неспециализированный характер и применим к выявлению аномалий в различных промышленных системах. Для решения проблемы большого объема анализируемых данных используется вейвлет-преобразование, позволяющее преобразовать непрерывный временной ряд в набор дискретных отсчетов, сохраняющих пространственно-временные характеристики сигнала, после чего этот набор сворачивается в матрицу сходства. К ней применяется алгоритм иерархической кластеризации, позволяющий построить набор кластеров, каждый из которых содержит схожие по параметрам сведения о полетах. Далее определяются параметры, позволяющие получить максимальное расхождение между кластерами, после чего данные анализируются экспертами и применяются для обучения алгоритмов с учителем.

С. Эйрас-Франко и др. в [9] рассматривают обнаружение аномалий в условиях смешанных численно-категориальных исходных данных большого масштаба. Для решения этой задачи предлагается комбинированная вероятностная мера, представляющая собой маргинальную плотность для непрерывных переменных и условную вероятность для категориальных переменных.

Вышеперечисленные подходы, несмотря на эффективность в решении поставленных задач, обладают рядом существенных недостатков, вытекающих в значительной степени из того, что они ориентированы либо на построение диагностической модели, либо (более распространено) на анализ данных методами

выявления аномалий. Это приводит к тому, что приоритет отдается какой-либо одной из задач диагностирования по обнаружению факта возникновения неисправностей или источника их возникновения. Перспективным представляется подход, ориентированный на решение обеих задач, то есть объединяющий применение диагностической модели и методов анализа данных.

1.5 Задача анализа журналов программного обеспечения в рамках диагностики СХД

Журналы программного обеспечения (ПО) СХД являются важнейшим источником диагностической информации, так как содержат сообщения, формируемые программным обеспечением в случае наступления каких-либо проблем в программной или аппаратной инфраструктуре СХД. Основной проблемой применения журналов в качестве источника диагностических данных является, как это ни парадоксально, чрезвычайное обилие содержащейся в них информации (проблема рассматривается, например, в публикации [86] B. Зу, посвященной обнаружению критических ситуаций при помощи анализа консольных журналов). Так, в случае целевой платформы СХД в качестве материалов для исследования получено порядка 350Гб различных журналов ПО СХД, что делает фактически невозможным их использование для решения задач диагностики без автоматической или автоматизированной обработки.

Существует три основных подхода к обнаружению неисправностей на основании журналов ПО СХД:

1. Подходы, основанные на определении структуры сообщений журналов СПО СХД.

Большинство научных исследований, посвященных автоматическому или автоматизированному использованию журналов программного обеспечения, как правило, определяют следующую схему работы: сначала выполняется предварительный анализ корпуса журналов для выявления в них шаблонов сообщений определенного вида [16,17] с предварительным решением задачи

разбора (парсинга) сообщений журналов. Затем все сообщения из массива сообщений транслируются из неструктурированного текста в цепочки последовательных событий [18]. Выделение шаблонов для построения структурированного представления текста может выполняться с применением методов теории распознавания образов [87], индивидуального определения методов извлечения признаков для каждого журнала [20, 88]. В целом, ключевым недостатком такого класса подходов можно считать высокую трудоемкость интеллектуального анализа журналов, сложность которого прямо зависит от объема анализируемых журналов.

Сюда же можно отнести наиболее простой и чаще всего применяемый в существующем программном обеспечении вариант использования журналов ПО СХД - обнаружение в журналах тех или иных событий и уведомление пользователя по факту их появления в журналах. Это требует детального анализа журналов и наличия сведений о причинах появления тех или иных сообщений (см. работу М. Синка [89], посвященную анализу журналов на основании построения набора правил).

2. Подходы, основанные на поиске аномалий в журналах ПО СХД.

Другой подход, часто применяемый в существующих программных решениях, подразумевает обнаружение в журналах аномалий без анализа структуры сообщений с уведомлением пользователя. Примерами такого подхода являются:

- работы С. Лю, З. Вей, В. Ли и др. по применению нейронных сетей для обнаружения аномалий в системных журналах [90];
- К. Яманиши в [91] по динамическому мониторингу журналов для определения сетевых проблем;
- А. Ягуба в [92] по обнаружению и прогнозированию аномалий производительности;
- работа Ф. Лопеса [93], посвященная поиску дефектов программного обеспечения.

3) Подходы, основанные на применении методов машинного обучения для анализа текста.

Обилие разных типов журналов ПО СХД с данными, поступающими из множества источников, делает задачу анализа структуры сообщений журналов весьма трудоемкой. При этом решение задачи обнаружения аномалий, как уже было показано в предыдущем разделе, не подходит для решения не только проблемы поиска причины возникновения неисправностей, но в общем случае даже типа возникшей неисправности, поэтому применяется еще один подход, подразумевающий автоматическое обнаружение типов неисправностей.

Примером использования такого подхода является работа К.Бертеро, М. Руа, К. Саваном и др. в [19], в которой рассматривается использование алгоритмов классификации для сравнения текстов журналов.

Для возможности практического применения диагностического метода, позволяющего использовать в качестве источника диагностической информации журналы ПО СХД (с учетом необходимости его применения в условиях изменения возможного состава компонентов СХД или на различных архитектурах СХД), наиболее удачным представляется третий подход, так как он исключает или сокращает потребность в анализе структуры журналов СПО СХД и при этом позволяет определить тип неисправности. Несмотря на потенциальную универсальность такого подхода, реализация, предложенная К.Бертеро, М. Руа, К. Саваном и др. работает исключительно с сообщениями одного журнала в конкретном формате (syslog [94]) и с конкретной структурой, что подразумевает весьма узкий спектр применения. Кроме того, решение предполагает сравнение журналов целиком, то есть оно не применимо для решения задачи обнаружения неисправностей в процессе работы СХД, а может использоваться только для проведения анализа по факту обнаружения неисправности.

1.6 Выводы по главе 1

В главе 1 получены следующие результаты:

1) Выполнен анализ систем класса СХД как объекта диагностики, определение требований к методам обнаружения неисправностей в СХД и реализующему их ПО.

2) Проведен анализ актуальных тенденций в области диагностики СХД, рассмотрены наиболее распространенные программные системы и определены применяющиеся в них подходы к диагностике.

3) Предложена классификация рассмотренных инструментов в соответствии с различными аспектами их устройства.

4) Рассмотрены различные научные и прикладные подходы к анализу диагностических признаков, наиболее широко применяющихся в области диагностики вычислительных систем, в том числе к анализу текстовых данных мониторинга.

5) Определена потребность в разработке инструментов, расширяющих возможности существующих решений.

Большинство из вышеперечисленных средств обнаружения неисправностей в вычислительных системах в первую очередь ориентируется на использование численных метрик, получаемых от агентов, датчиков, средств мониторинга и т.п., и в существенно меньшей степени – текстовых данных мониторинга, в том числе журналов ПО. При этом даже те средства, которые предназначены для анализа журналов, такие как logstash и elastic stack, ориентированы, в первую очередь, на применение правил и уведомлений, а также на поиск в журналах аномалий.

Повышение эффективности решения задачи обнаружения неисправностей в СХД предлагается осуществить путем разработки следующих методов:

1) Метода построения диагностических моделей СХД, позволяющих работать с разнородными диагностическими параметрами, что позволит объединить диагностику на основе численных метрик и диагностику на

основании данных, извлекаемых из журналов ПО, что обосновывается наличием в СХД обоих типов источников диагностических данных.

2) Методов и алгоритмов анализа и обработки текстовой информации мониторинга, ориентированных на извлечение диагностических признаков из журналов ПО СХД и не требующих детального анализа их структуры и содержимого. Необходимость исключить детальный анализ журналов объясняется большим объемом файлов журналов и избыточностью содержащейся в них информации.

2 МЕТОД ПОСТРОЕНИЯ ДИАГНОСТИЧЕСКИХ МОДЕЛЕЙ СХД НА БАЗЕ ОНТОЛОГИИ И УПРОЩЕННОЙ ГРАФОВОЙ МОДЕЛИ

Результаты анализа предметной области показывают, что в современных средствах диагностики вычислительных систем для обнаружения неисправностей все чаще используются алгоритмы и методы машинного обучения. Очевидным недостатком применения таких методов в задачах диагностики является то, что алгоритм машинного обучения пытается выстроить закономерности на основании данных из обучающей выборки (либо обучаясь в процессе функционирования), рассматривая входные последовательности как набор абстрактных числовых векторов, то есть отсутствует возможность каким-либо образом использовать априорные сведения о структуре и взаимодействиях компонентов системы.

Для исправления данного недостатка в настоящей работе предлагается подход, в котором приоритетным является априорное описание объекта диагностики (СХД), его структуры и состояний, а применение алгоритмов машинного обучения является одним из способов формирования связей между состоянием СХД и ее отдельных компонентов и значениями параметров, описывающих эти состояния. Разработка такого похода основывается на распространенной в настоящее время концепции совмещения алгоритмов диагностирования на базе построения диагностической модели и анализа данных о функционировании системы. Примеры такого объединения рассмотрены в работах, посвященных построению комбинированного алгоритма диагностики в разных областях техники: А. Слимани и др. в [10], Дж. Луо и др. в [11], Д. Юнгом и К. Сундстремом в [12] .

Для разработки диагностической модели были определены требования к ее реализации и структуре с учетом особенностей объекта исследования. В процессе комплексной оценки решаемых задач в рамках настоящего исследования определены следующие требования к модели СХД:

- Модель должна обеспечивать возможность оценки здоровья СХД и отдельных компонентов и модулей СХД и формирования диагностических

признаков на основании информации, получаемой из различных источников, так их как: данные мониторинга, экспертные оценки, статистические данные, исторические данные о функционировании СХД;

- Модель должна обеспечивать возможность задания связей между входными параметрами и состояниями СХД, компонентов и модулей СХД, с минимальной зависимостью от структуры и типов данных параметра;
- Модель должна позволять описывать иерархическую структуру СХД, компонентов и модулей;
- Модель должна быть достаточно гибкой и масштабируемой, чтобы обеспечивать возможность внесения изменений при изменении конфигурации или архитектуры СХД, а также повышения детализации;
- Опциональным, но крайне полезным свойством модели является возможность ее адаптации к СХД другой конфигурации с минимальным внесением изменений в использующий ее программный код.

Исходя из вышесказанного, построение модели СХД выполнялось в следующем порядке:

1. Анализ существующих подходов к построению моделей, предназначенных для формализованного представления знаний с целью определения наиболее полно соответствующего предъявляемым требованиям.
2. Подготовка к построению модели:
   2.1 Разработка описания онтологической модели, предназначенной для описания взаимосвязи между параметрами и состояниями отдельных компонентов СХД, макрокомпонентов СХД и СХД в целом.
   2.2 Анализ объекта диагностики, определение параметров, описывающих состояния компонентов СХД и СХД в целом, определение состояний СХД, определение источников получения параметров, типов данных параметров и т.д.
3. Построение модели:

3.1 Определение понятий для построения модели.

3.2 Формализация существующих экспертных знаний о способах обнаружения неисправностей СХД в терминах разработанной модели, в том числе определение объектов модели и определение связей между объектами модели.

3.3 Верификация модели.

2.1 Сравнительный анализ способов представления знаний

В настоящее время задача построения формализованного представления некоторой области знаний чаще всего решается с использованием следующих моделей:

- семантическая сеть;
- фреймовая модель;
- продукционная модель;
- логических модель;
- онтологическая модель.

Семантическая сеть [95] разрабатывалась как средство для описания закономерностей естественного языка. Она представляет собой ориентированный граф, описывающий отношение между объектами некоторой предметной области, при этом узлы соответствуют объектам, а дуги – отношениям.

Для структуризации описания семантической сети можно использовать язык описания графов rdf. Основное достоинство семантической сети – удобное для визуальной оценки отображение, недостаток – хаотичная структура, трудно поддающаяся обработке.

Фреймовая модель [96] разрабатывалась как технологическая модель памяти человек и его сознания. Единицей информации во фреймовой модели является фрейм, представляющий собой минимальное, то есть такое, которое нельзя больше сократить из-за риска потери полноты, описание некоторого объекта. Фрейм представляет собой жесткую структуру, имеющую уникальный

идентификатор и состоящую из слотов, причем допускается неограниченное количество уровней вложенности. Слоты, в свою очередь, могут быть связаны друг с другом или с другими слотами. Над слотами можно производить набор операций - запись данных, чтение данных и удаление данных.

Недостатком фреймовой модели является отсутствие механизма управления выводом.

Продукционная модель [97] фактически представляет собой систематизированный набор правил в виде перечня условий и действий по их выполнению, некоторый аналог конструкции if-else. Простая структура, модульность, хорошая читаемость делают эту модель одной из наиболее часто используемых.

Логическая модель используется для представления знаний, выраженных в виде утверждений, аксиом, которые характеризуются точностью определения смысла выражения, возможностью построения на их основе баз знаний модульного типа, компактностью записи выражений. Главная идея создания модели такого типа – рассмотреть всю информацию, которая используется для поиска оптимального плана задач, а совокупность фактов и утверждений представляется в виде формул и уравнений [98, страница 60]. Знания в этом случае представлены набором аксиом. Логическая модель имеет следующую структуру см. (1):

$$S= \langle B;F;A;R \rangle \quad (1)$$

где B – алфавит базовых понятий,

F – множество формул,

A – подмножество аксиом,

R – множество правил вывода, представляющее собой отношение между формулами.

Преимущества логической модели обусловлены тем, что она основана на хорошо проработанном и обоснованном аппарате математической логики, позволяющим достраивать некоторые знания на основании базовых аксиом и

понятий. Недостатком является невысокая наглядность и сложность описания специфических нюансов знания, что может приводить к их потере.

Онтологическая модель предназначена для формального представления некоторой области знаний с помощью концептуальной схемы, исчерпывающе описывающей данную предметную область. Онтология представляет собой описание некоторой предметной области в виде структурированного набора классов, их свойств и отношений между ними, причем позволяет использовать для описания этих отношений аппарат математической логики подобно логическим моделям.

На текущий момент именно онтологический аппарат является наиболее совершенным с точки зрения возможностей по описанию и структурированию некоторой области знаний. При помощи машинной интерпретации онтологии можно обрабатывать содержащиеся в них сведения, в том числе основные концепции, аксиомы, понятия и связи между ними.

Среди существующих онтологий есть некоторое количество относящихся к надежности технических систем. Одной из таких онтологий является ACM CS Classification [99], имеющая в своем составе некоторый набор классов, соответствующих понятиям теории надежности (такие как B_8_Performance_And_Reliability, B_8_1_Reliability_Testing_And_Fault_Tolerance, C_4_5_Reliability_Availability_And_Serviceability и т.д.). Но все существующие на текущий момент готовые онтологии не обладают необходимым набором сущностей для решения поставленной задачи.

Важным преимуществом онтологической модели является то, что она удовлетворяет требованию, предъявленному в главе 1 настоящего исследования, о необходимости обеспечения возможности работы с разнородными исходными данными, что показано, например, Д. Мерсье в [100].

2.2 Построение онтологической модели

Задача построения онтологической диагностической модели, определяемой в соответствии с ГОСТ 20911-89, фактически сводится к формализации и определению отношений между состояниями объекта, его элементами и диагностическими параметрами.

Под онтологической моделью при этом понимается база знаний на основе онтологии надежности СХД, созданной с применением традиционной методологии, интерпретируемая с использованием новых алгоритмов диагностики и анализа системных журналов для определения технического состояния системы.

Онтологическая модель должна позволять формализовывать следующие задания, необходимые для обнаружения неисправностей в вычислительных системах вообще и СХД в частности:

- Каким образом состояние подсистемы определяется состояниями входящих в нее компонентов;
- Каким образом состояние системы определяется состояниями составляющих ее подсистем;
- Как интерпретировать события системного уровня;
- Как текущая конфигурация СХД влияет на правила определения состояний подсистем и системы в целом;
- Различные способы описания состояний компонентов СХД при помощи диагностических признаков разной размерности, включающих различные типы данных.

Важным аспектом, который необходимо учесть при формировании онтологической модели, является влияние неисправностей компонентов, расположенных на нижних уровнях иерархии СХД на неисправности, возникающие на более высоких уровнях. Представленная на рисунке 2.1 схема логической подсистемы управления дисками (см. раздел 1.2) иллюстрирует взаимосвязь между параметрами и состояниями на разных уровнях иерархии компонентов. Для построения логического пула хранения физические диски

(описываются собственным набором параметров и обладают своим набором состояний) группируются с применением схемы резервирования в общую неразмеченную емкость (описывается набором собственных параметров и состояниями физических дисков, и набором собственных состояний), которая в свою очередь, разбивается на логические тома (собственные параметры, состояния общей емкости, свои состояния). При этом формируемое предельное состояние «Потеря данных» отражается на состоянии всей СХД в целом.

Рисунок 2.1 – Обнаружение неисправностей в логической подсистеме управления дисками

Подход к построению онтологической модели СХД позволяет получить модель, удовлетворяющую установленным требованиям и обладающую следующими свойствами:

- Онтологическая модель позволяет формально описывать связи между разнородными знаниями о поведении СХД и элементов СХД, получаемыми по результатам экспертных оценок, из анализа статистических данных и исторических данных о функционировании СХД, результатов моделирования и диагностических параметрах разного рода.

- Онтологическая модель позволяет формально описывать иерархическую структуру СХД, их компонентов и подсистем;

- Онтологическая модель удовлетворяет требованию масштабируемости, т.е. обеспечивает возможность адаптации для различных конфигураций и архитектур СХД, при этом масштабирование модели происходит с минимальным внесением изменений в реализующий процедуру диагностики программный код.

Онтологическая модель состоит из четырех базовых элементов: классов (classes), именованных сущностей (объектов классов, named entities), свойств объектов (object properties) и свойств данных (data properties). Кроме того, существуют также аннотационные свойства (annotation properties), но они в рамках настоящего исследования не задействованы.

Онтологическая модель СХД структурно включает в себя следующие разделы:

1. Раздел описания топологии СХД. Включает в себя классы и свойства, предназначенные для описания топологии СХД, в том числе библиотеку компонентов, макрокомпонентов, систем и подсистем СХД, как программных, так и аппаратных, а также набор свойств объектов для описания взаимосвязей между ними.
2. Раздел описания параметров СХД. Включает в себя библиотеку классов параметров, набор свойств объектов для описания отношений между параметрами и компонентами, набор свойств данных для описания связей между типами параметров и значениями параметров.
3. Раздел описания неисправностей, включающий в себя библиотеку неисправностей, набор свойств объектов для описания отношений между элементами топологии СХД, параметрами СХД и неисправностями СХД.
4. Вспомогательные разделы, например, раздел, включающий классы, свойства объектов (object property) и свойства данных (data property) онтологии для описания процесса обработки журналов ПО СХД.

Именованные сущности создаются в соответствии со структурой классов онтологии в процессе функционирования ПО диагностики для объектов, фактически существующих в системе.

В модели определен ряд понятий(классов), позволяющих формально описать структуру объекта диагностирования (СХД) и его элементов, возможные состояния объекта диагностирования и его элементов, возможные события в объекте диагностирования и его элементах, а также диагностические параметры.

Понятие **«параметр СХД»** (*P*) предназначено для описания диагностических параметров СХД. Получаемые в процессе мониторинга значения параметров СХД классифицируются в зависимости от описанных в модели возможных значений параметра *{V}* как соответствующие тому или иному событию. Параметры СХД атомарны и имеют набор абстрактных классов-родителей, позиционирующий их в структуре онтологии, но, при этом, не имеют соответствующих им именованных сущностей.

Понятие **«компонент СХД»** (*K*) предназначено для описания базовых структурных элементов СХД. На уровне модели компонент является атомарным и неделимым, то есть максимальная глубина поиска места отказа (неисправности) ограничивается уровнем компонентов. Для систематизации знаний в онтологической модели принята иерархическая структура понятий, формируемая в момент разработки модели, поэтому понятие «компонент» может иметь вложенные понятия. Так, класс EthernetController (предназначенный для описания аппаратного компонента «контроллер Ethernet») имеет дерево родительских классов: Component -> Network Device -> Network Interface Card -> EthernetController. При этом в процессе создания именованных сущностей создается экземпляр только класса EthernetController. Компонент СХД может быть как аппаратным, так и программным, например, одним из сервисов СПО СХД.

Понятие **«подсистема СХД»** (*Ss*) предназначено для описания группы элементов СХД, объединенных по какому-либо функциональному признаку. Подсистема СХД может использоваться для группировки понятий не только компонентов, но и подсистем, включая такие же понятия. Подсистема отличается от компонента тем, что не имеет собственных диагностических параметров для вычисления состояния.

Понятие «**система**» (*Storage*) соответствует объекту диагностирования (то есть СХД в целом) и предназначено для идентификации группы подсистем верхнего уровня.

Понятие «**состояние**» $\{D\}$ предназначено для характеристики здоровья подсистемы, компонента СХД и СХД в целом и может иметь одно из четырех значений: «работоспособное состояние», «предотказное состояние», «уязвимое состояние», «полный отказ».

Понятие «**событие в СХД**» $\{F\}$ представляет собой совокупность значений параметров СХД, состояний компонентов или подсистем СХД, характеризующих наступление какой-либо неисправности. Каждое событие при этом классифицируется как одно из состояний $\{D\}$ в зависимости от уровня его критичности. При этом работоспособное состояние компонента СХД подразумевает отсутствие диагностированных неисправностей этого компонента, подсистемы – отсутствие диагностированных неисправностей во вложенных компонентах и подсистемах.

Для определения состояния D_{Ei} в элементе Ei определяется наличие событий в дочерних элементах, начиная с элементов самого низкого уровня вложенности – параметров и далее вверх по иерархии элементов от компонентов к подсистемам.

Для определения событий S_{Ei} в элементе E_i определяются его дочерние элементы и проверяется наличие событий в них, начиная с элементов самого низкого уровня вложенности – параметров и далее вверх по иерархии элементов от компонентов к подсистемам.

Формально онтологическую модель можно описать следующим образом:

Параметр $\boldsymbol{P_i}$ СХД:

$$P_i = \{Rp_j, V_j\}_{j=1,..,N} \quad (2)$$

где N – число связей i-ого параметра СХД,

$< Rpj, Vj >$ - пара связь/значение.

Компонент $\boldsymbol{K_i}$ СХД:

$$K_i = <\{P_{ij}\}_{j=1,..,N}; \{Fe_k, De_k, Rep\}_{k=1,..,M} > \quad (3)$$

где N – число параметров,

M – число возможных событий i-ого компонента СХД,

{P_{ij}} – набор параметров компонента,

<Fe_k, De_k, Rep> - триплет событие/состояние/связь с параметрами.

Подсистема СХД ***S_i***:

$$S_i = <\{K_{ij}\}_{j=1,..,N}; \{S_k\}_{k=1,..,M}; \{Fs_t, Ds_t, [Rs_t(K_{ij}), Rs_t(S_k)]\}_{t=1,..,L} > \quad (4)$$

где N – число компонентов,

M – число вложенных подсистем,

L – число событий- i-ой подсистемы СХД,

{K_{ij}} – набор компонентов системы.

<Fs_t, Ds_t, {Rs_t, Rs_t}> - триплет <событие/состояние/связь, определяющая компонент или подсистему – источник события>,

Тогда система в целом:

$$Storage = <\{S_i\}_{j=1,..,N}; \{Fst_j, Dst_j, Rst_j\}_{j=1,..,M} > \quad (5)$$

где N – число параметров,

M – число возможных событий,

<Fst_j, Dst_j, Rst_j> - триплет событие/состояние/ связь, определяющая подсистему – источник события.

Оценка текущего состояния СХД при этом определяется как:

- набор состояний компонентов СХД *{Sc}*, подсистем СХД *{Ss}* и СХД в целом *{S}* (из списка: «работоспособное», «предотказное», «неисправность», «полный отказ»);
- набор обнаруживаемых событий в СХД *{F}*, каждое из которых соответствует одному из состояний СХД.

На рисунке 2.2 представлена визуализация фрагмента онтологии, соответствующего описанию взаимоотношения параметров, подсистем и компонентов на графе онтологии.

Элемент StoragePool может состоять из таких же элементов типа StoragePool (пул хранения), поэтому он имеет объектное свойство «strongly_depends_on» сам с

собой. В противном случае, он должен состоять из носителей информации, поэтому он связан объектным свойством «depends_with_ecc» с классом SCSIDisk (носитель информации SCSI).

У компонента есть событие SCSIDiskMediaExpectedDegradation (физическая деградация носителя диска), соответствующее его уязвимому состоянию. Так как для всего StoragePool это тоже уязвимое (warning) состояние, то для внешних связей используются объектные свойства «if_warning_manifests_in» для компонента и «if_warning_causes» для подсистемы, так как это событие влияет на состояние всей системы в целом. Событие DataVulnerability, соответствующее уязвимому состоянию подсистемы StoragePool интерпретируется как состояние Vulnerable для СХД в целом.

Уязвимое состояние SCSIDiskMediaExpectedDegradation определяется состоянием параметров (shows_in) BackgroundScanStatus и R-W-V-ErrCount [101], которые являются подклассами класса SCSILogParameter.

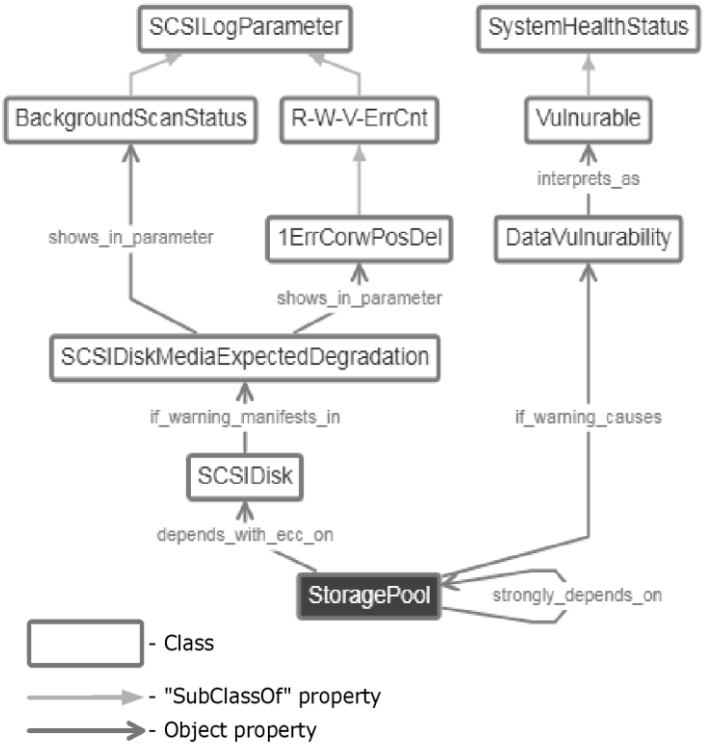


Рисунок 2.2 – Иллюстрация взаимосвязей в онтологии

Для описания отношений между объектом диагностирования, его элементами, состояниями и диагностическими параметрами применяются два типа

связей *{R}*: связи, основанные на онтологических свойствах объектов и свойствах данных, далее именуемые детерминированными связями, и связи, для описания которых требуется использование внешней процедуры, далее именуемые условными связями. В качестве внешней процедуры задается предварительно обученный алгоритм машинного обучения, принимающий на вход вектор значений диагностических параметров и классифицирующий их как одно из связанных понятий (рисунок 2.3). Процедура вызывается в процессе диагностирования при определении события, связанного с указанными диагностическими параметрами.

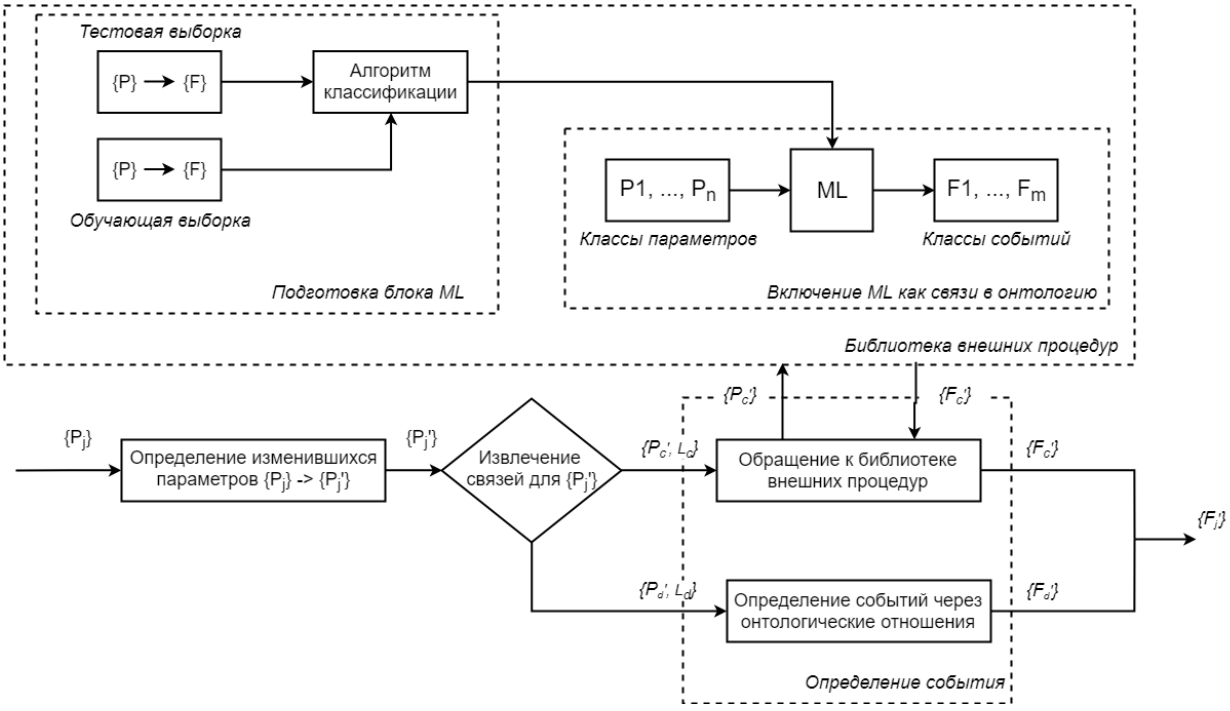


Рисунок 2.3 – Схема связи между параметрами и состояниями СХД

После извлечения из полного перечня параметров *{Pi}* тех параметров *{Pi'}*, значения которых изменились, определяются их возможные связи *{L}* с событиями в компонентах. Далее поиск произошедших событий выполняется параллельно для условных связей и соответствующих им параметров (*{Pc', Lc}*) и детерминированных связей и соответствующих параметров (*{Pd', Ld}*). При этом параметры *{Pc'}* передаются во внешние процедуры, определяемые в соответствии с *{Lc}*.

Пример такого решения на графе онтологии, см. рисунок 2.4.

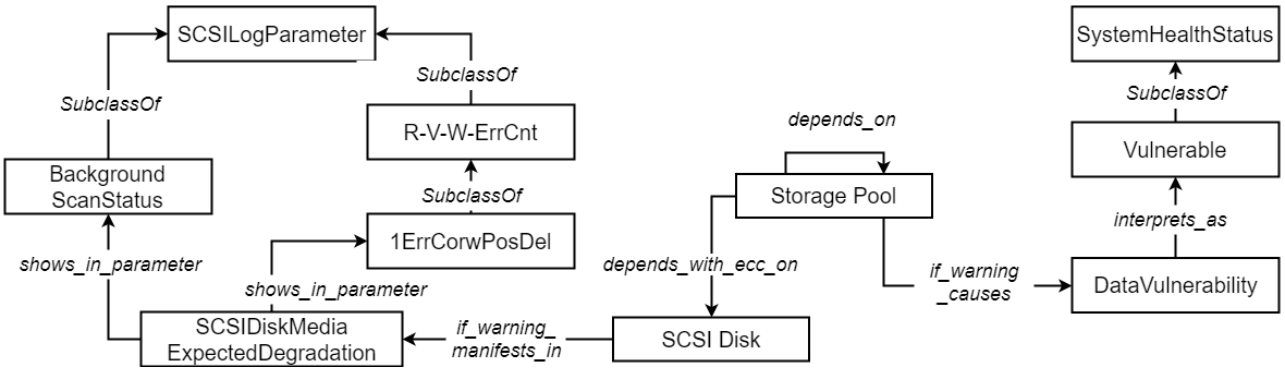


Рисунок 2.4 – Реализация сложных связей в онтологии

В приведенном примере объектное свойство «solves_with» объединяет через класс-внешнюю процедуру DiagDiskBySmart несколько уязвимых состояний (DiskDead, DiskDegradation) с набором параметров (R-W-V-ErrCnt и SolidStateMediaLog). В процессе выполнения алгоритм диагностики вызывает внешнюю процедуру, которая и выполняет процесс сопоставления значений параметров и состояний. Таким образом, с учетом того, что определение событий в подсистемах и СХД в целом выполняется аналогичным образом, функцию построения событий в СХД можно сформулировать как $S = \{La, \{Lb, \{Lc...\{Lz, P\}\}\}\}$. При этом на уровне связей определяется, какими именно состояниям соответствуют определенные события.

Разработанная онтологическая модель содержит следующий набор свойств и классов верхнего уровня, см. таблица 4:

Таблица 4 – Классы верхнего уровня СХД

Класс	**Описание**
Parameter	Параметр, информационный признак, соответствующий некоторому объекту мониторинга
Component	Компонент, некоторый объект инфраструктуры СХД
Storage subsystem	Подсистема СХД, объединяющая некоторое количество подсистем и компонентов СХД
Storage system	СХД как система, обладающая собственным набором параметров и уровнем здоровья
Component fault	Персонифицированная неисправность компонента СХД
System fault	Персонифицированная неисправность СХД в целом

Набор отношений включает в себя, см. таблицу 5.

Таблица 5 – Свойства верхнего уровня в онтологической модели СХД

Имя отношения	**Тип отношения**	**Пояснение**
manifests_in (if_fatal_manifests_in, if_warning_manifests_in)	объектное свойство	Определяет, какому уровню работоспособности компонента соответствует неисправность.
shows_in_parameter	объектное свойство	Определяет, какие параметры могут быть использованы для идентификации неисправности.
is_normal (is_normal_when, is_normal_below, is_normal_above)	свойство данных	Определяет диапазон нормальных значений диагностического параметра в простом событии.
solves_with	объектное свойство	Для задания ссылки на внешнее правило для события.
described_by	объектное свойство	
depends_on (strongly_depends_on, majorly_depends_on и depends_with_ecc_on)	объектное свойство	Определяет, каким образом состояние подсисте.мы СХД зависит от состояний элементов этой подсистемы.
consists_of	объектное свойство	Позволяет описать иерархию компонентов и подсистем конкретной конфигурации СХД в виде дерева.
causes (if_failed_causes, if_warning_causes)	объектное свойство	Указывает на названия системных явлений, соответствующих состояниям подсистем.
interprets_as	объектное свойство	Позволяет делать выводы о здоровье СХД как системы в целом.

Описанные взаимосвязи между элементами онтологии приведены на рисунке 2.5.

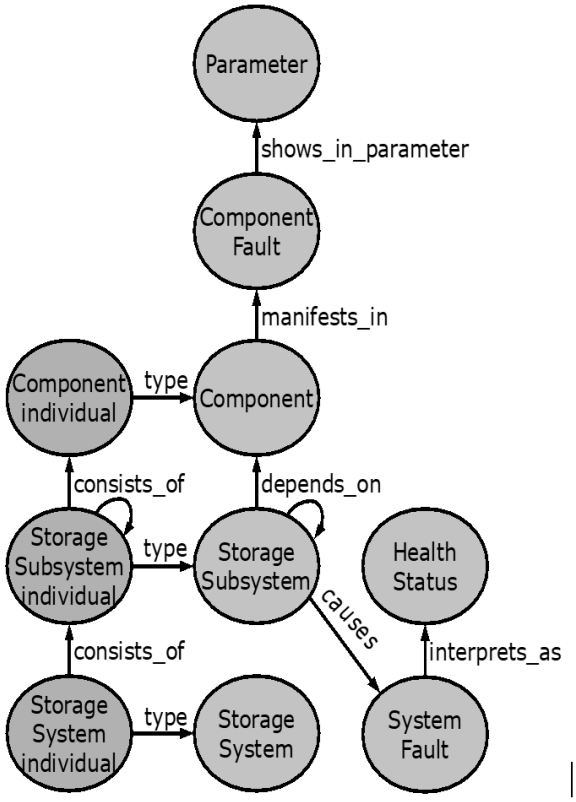


Рисунок 2.5 - Структура взаимосвязей в онтологии

В настоящей публикации рассматривается случай, когда в качестве параметров СХД применяются журналы программного обеспечения СХД. В таком случае в качестве внешнего объекта M_0 для определения связи ***{P}/{S}*** используется заранее обученный на массиве исторических данных классификатор, основанный на использовании алгоритмов машинного обучения.

Заданная таким образом модель здоровья:

- Содержит описание возможных неисправностей в СХД;
- Содержит описание соотношений между признаками и состояниями в СХД;
- Позволяет локализовать неисправность относительно элемента СХД и всех элементов более низкого уровня вложенности.

2.3 Определение источников диагностической информации

Для построения моделей используются различные подходы к формированию массива информации о параметрах и состояниях СХД, условно подразделяемые в рамках настоящей работы на:

- «пассивный» сбор данных, т.е. проводящийся без какого-либо вмешательства в работу СХД;
- построение связей «параметры-состояние» на основании технической документации и экспертных оценок;
- «активный» сбор данных, подразумевающий проведение экспериментов по имитации внештатных состояний СХД;
- сбор данных на основе моделирования.

Пассивный сбор данных предполагает фиксацию факта наступления неисправности в СХД с извлечением и хранением данных мониторинга системы. Несмотря на то, что такой способ позволяет получить наиболее достоверные результаты, его существенным недостатком является относительная редкость возникновения неисправностей в современных СХД. Примером применения такого способа является обработка таких событий, как возникновение неисправностей в физических носителях информации, так как это событие в СХД происходит относительно часто и может быть точно идентифицировано при анализе данных системы мониторинга.

Анализ технической документации позволяет учесть в модели допуски по значениям параметров, задаваемых производителями оборудования. Так, для носителей информации производителем указываются допустимые климатические параметры окружающей среды, например, для диска HGST Ultrastar 7K6000 [102] производителем указаны следующие параметры, см. таблица 6.

Таблица 6 – Климатические параметры носителя информации

Режим	**Показатель**	**Значение**
Рабочий	Температурный режим	5-60 град.С
	Относительная влажность	8-90%
	Максимальная температура влажного термометра	29 град.С
	Максимальное изменение температуры	20 град.С /час
	Высота над уровнем моря	-300-+3,048м

Продолжение таблицы 6

Режим	**Показатель**	**Значение**
Нерабочий	Температурный режим	-40-70 град. С
	Относительная влажность	5-95%
	Максимальная температура влажного термометра	35 град.С
	Максимальное изменение температуры	30 град.С /час
	Высота над уровнем моря	-300-+12000м

Указанные таким образом допуски описываются онтологическими свойствами объектов и свойствами данных, что позволяет учесть их влияние не только на конкретный диск, но и на всю систему в целом.

Задание правил на основании экспертных оценок позволяет учесть знания о СХД, получаемые путем опроса экспертов о принципах работы и взаимодействия отдельных компонентов СХД и СХД в целом. Сюда относятся, например, данные о функционировании в режиме нарушения схемы избыточности при разных значениях коэффициентов схемы избыточности, сведения о характере процессов, происходящих в СХД при нарушении кворума в кластере, оценка допустимых порогов производительности и т.д.

Активный сбор данных предусматривает проведение экспериментов по доведению СХД до предельных состояний путем применения воздействий различного типа и фиксацию значений параметров, соответствующих данным предельным состояниям. Таким образом, например, были зафиксированы параметры производительности СХД, соответствующие отключению отдельных компонентов СХД, на программном и аппаратном уровне, таких как контроллеры хранения, различной сетевой аппаратуры, дисков, входящих в пулы хранения и т.д. Сюда же относятся режимы аппаратного и программного тестирования, позволяющие зафиксировать момент наступления неисправности. Так для каждого отказа, наступившего в рамках функционального тестирования, фиксируется пакет журналов ПО с отметкой о времени прохождения сбойных тестов. Для аппаратного тестирования также может использоваться подход с использованием

многокомпонентного сигнатурного анализа, описанный в статье [103] на примере анализа цифровой схемы. Данный подход основывается на получении дерева сигнатур на основании обратной трассировки для поиска причин возникновения неисправного состояния по схеме платы в процессе аппаратного тестирования. Алгоритм проведения тестирования указан на рисунке 2.6.

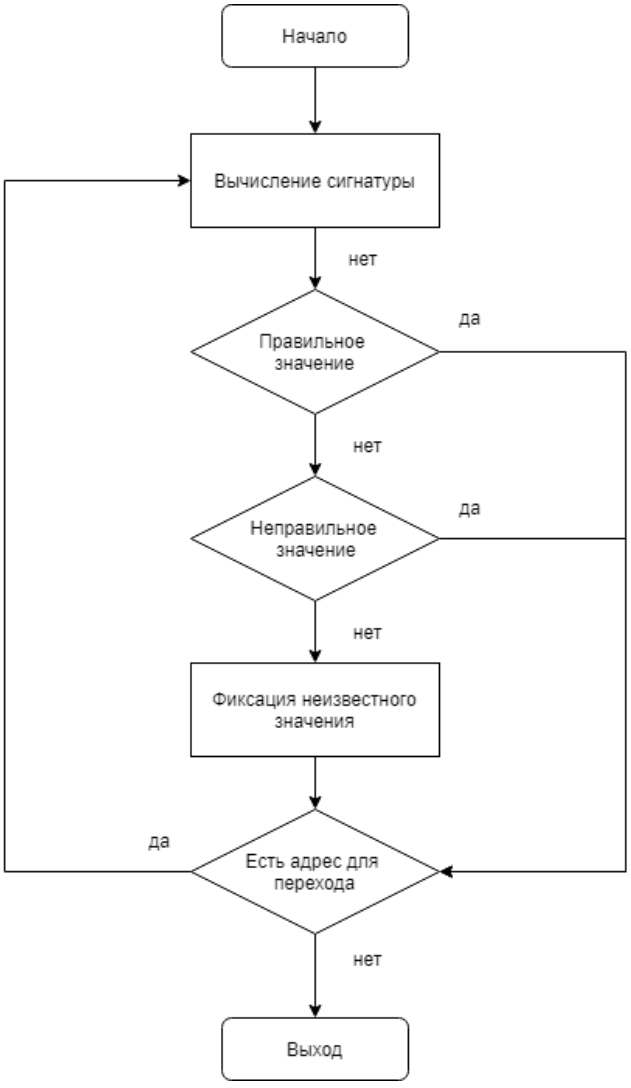


Рисунок 2.6 – Алгоритм сигнатурного анализа

Предполагается, что для каждой сигнатуры есть набор значений: правильное значение сигнатуры, соответствующее штатной работе компонента, одно из набора заданных неправильных значений, соответствующее известным неисправностям компонента, либо любое другое, подразумевающее возникновение неизвестного значения. Под сигнатурой понимается некоторое измеряемое уникальное значение, соответствующее выходному состоянию анализируемого блока или компонента

СХД. Для анализа более высокоуровневых взаимоотношений используется концепция тестового проекта, предложенная в [104], объединяющая набор тестов в ориентированный граф, по которому проходит оператор тестирования в автоматизированном режиме. При этом узлам графа соответствуют отдельные тесты, а дугам – переходы от одного теста к другому. Проход по графу осуществляется по алгоритму, аналогичному представленному на рисунке 2.6, но вместо расчета сигнатур применяются тесты. Достоинством данных методов является возможность последовательно отслеживать распространение неисправностей в СХД, недостатком – высокая сложность разработки тестов и проведения тестирования. Данные методы можно использовать не только на аппаратных объектах, но также и на их моделях.

В рамках работ по проекту в рамках субсидии от 03.10.2017 г. № RFMEFI581 17X 0023 был создан набор имитационных дискретно-событийных моделей, позволяющий построить зависимости между параметрами и состояниями отдельных типов компонентов СХД. Применение данных моделей для задач построения онтологической модели осуществляется как с использованием методов сигнатурного анализа и тестовых проектов, так и путем тренировки алгоритмов машинного обучения.

2.4 Обнаружение неисправностей в СХД с использованием данных анализа журналов ПО

2.4.1 Характеристика журналов ПО как источника данных для обнаружения неисправностей в СХД

Любое системное и многое прикладное ПО имеет один или несколько текстовых журналов, в которые добавляет записи, соответствующие определенным событиям своего жизненного цикла в некотором заданном для данного ПО формате. Подход, подразумевающий использование системных журналов в качестве источника данных для обнаружения неисправностей в СХД основывается на предположении, что можно установить однозначное соответствие между

состоянием здоровья СХД и содержимым журналов, ведущихся различными сервисами ПО СХД.

Журналы ПО в СХД формируются и размещаются на контроллерах хранения, причем на каждом контроллере размещается собственный комплект журналов. Журналы в операционных системах на базе linux, как правило, располагаются в системной папке /var/log/. Все журналы из этой папки обычно относят к одной из четырех основных категорий [105], [106]:

- журналы событий;
- журналы приложений;
- журналы сервисов;
- системные журналы.

Классификация, более подходящая для решаемой в настоящей работе задаче, предлагается в [107]. Данная классификация объединяет системные журналы с журналами событий, таким образом оставляя три основных группы журналов – журналы уровня операционной системы, уровня сервисов и уровня приложений.

Журналы уровня операционной системы в основном предназначены для фиксации активности на уровне операционной системы, такие как авторизация, управление системными демонами, выдача системных сообщений и т.д. Для ОС Ubuntu, например, это authorization log, daemon log, debug log и др., для ОС Open Suse – warn, messages, lastlog и т.д. Некоторые типы системных журналом представлены в таблице 7:

Таблица 7 – Основные системные журналы

Тип журнала, названия	**Назначение**
Журналы сообщений, поступающих в процессе работы системы	Журналы, в которые на протяжении работы системы с момента загрузки пишутся сообщения из различных источников: ядра, приложений (в том числе сведения о неисправностях, сообщения о состоянии оборудования и т.д.) (syslog, messages etc)
Журналы сообщений в момент загрузки и инициализации системы	Журналы, содержащие информацию, поступающую от системы или отдельных сервисов до завершения их запуска и инициализации (localmessages, boot.msg, boot.log)

Продолжение таблицы 7

Сообщения от устройств	Журналы содержат сведения, генерируемые драйверами устройств (в том числе о каких-либо неисправностях) (dmesg)
Сообщения о пользователях	Журналы содержат сведения о входе/выходе пользователей, последних зашедших пользователях и т.д. (utmp, wtmp, lastlog)
Сведения о неисправностях	Журнал содержит сведения о различных предупреждениях и ошибках программного обеспечения, возникших в системе. (warn, faillog)

Отдельные приложения, такие как веб-серверы, ПО кластеризации также создают и ведут свои журналы в общей папке. Каждое из таких приложений может иметь как соответствующий ему один журнал, так и несколько, для чего создается соответствующая папка в /var/log/.

В руководстве по управлению журналами операционной системы Ubuntu [108] выделяется также отдельная группа журналов – журналы, не читаемые человеком (Non-Human-Readable Logs). Они предназначены для анализа при помощи различных системных утилит. К этой группе относятся такие журналы, как faillog и др.

Реализация предлагаемого подхода к обнаружению неисправностей позволяет отказаться от явного анализа назначения и характера содержащихся в журналах сведений, поэтому детальный анализ типов шаблонов сообщений и обоснование их соответствия возникающим в СХД неисправностями не выполняется. Для формирования онтологической модели используется весь набор типов журналов, содержащихся в папке «/var/log/» контроллеров хранения, удовлетворяющий следующим требованиям:

1. Наличие в сообщениях временной метки – данное требование обосновывается необходимостью учитывать порядок слов при построении векторного представления текста, работой с временным окном

определенного размера, а не с журналом целиком и используемым методом разметки обучающей выборки;

2. Журнал должен быть в текстовом формате – требуется для унификации обработчиков сообщений;

3. Журнал должен быть включен в каждый пакет из имеющихся выборок исходных данных (см. описание исходных данных для анализа) – для построения векторного представления текста;

4. Средняя длина журнала по всем пакетам из имеющейся выборки не должна быть менее некоторого порогового значения (в рамках настоящей работы взято данное значение экспериментально установлено в 0,1% от средней длины журнала в пакете).

Дополнительно журнал может быть исключен из рассмотрения, если:

1. Нет корреляции между сообщениями в данном журнале и какой-либо из известных неисправностей;

2. Сообщения из данного журнала полностью дублируются в каком-либо другом журнале;

3. Количество сообщений журнала, содержащих текстовые токены после обработки, не достаточно, чтобы удовлетворить требованию 4 из предыдущего списка.

2.5 Форматы сообщений журналов ПО

На текущий момент, существует несколько распространенных форматов журналов, подавляющее большинство из которых является в том или ином виде подвидом формата сообщений протоколов упоминавшегося ранее протокола Syslog. Существует несколько спецификаций, описывающих форматы сообщений Syslog – Syslog Protocol (RFC 5424), BSD syslog (RFC 3164)[109], Common Logfile Format (RFC 6872)[110]. Кроме того, RFC 5427 [111] описывает текстовые соглашения для управления Syslog такие, как важность и категорию сообщений. Некоторые приложения, генерирующие большое количество различных журналов,

например, ПО веб-сервер Nginx, имеют свои собственные спецификации форматов сообщений, но при этом допускают и экспорт в одной из стандартных форм.

В соответствии со спецификацией формата syslog сообщение состоит из трех основных областей – области заголовка, области структурированных данных и области сообщения. В соответствии с особенностями предлагаемого в настоящей работе подхода к анализу журналов, подразумевающего работу с журналом как с текстом, в дальнейшем изложении «заголовком» сообщения будет называться объединение структурированной части и заголовка, сообщение – «текстовой частью» сообщения. Заголовок сообщения при этом подлежит разбору и последующему удалению, а текстовая часть сообщения обрабатывается и анализируется при помощи классификаторов. В общем случае, даже если журнал не соответствует в явном виде спецификации формата Syslog, в его заголовке сообщения можно выделить все или некоторые из полей следующих типов:

- Временная метка (TimeStamp).

Поле содержит информацию о том, когда произошло событие, сообщение о котором помещено в журнал. Сама временная метка в общем случае может иметь достаточно произвольный формат, в том числе с указанием года в начале каждого файла журнала и месяца/числа/времени – в каждом сообщении журнала. Некоторые примеры из встреченных в журналах – «Feb 26 17:21:22», «2019-08-05T09:53:24.081021+03:00», «2019/08/05 09:32:46».;

- Идентификатор генератора сообщения (Hostname).

Поле может быть ip-адресом или строковым идентификатором;

- Идентификатор приложения/процесса (Application/PROCID).

Строковое поле, состоящее из имени приложения (строковое значение) и заключенного в квадратные скобки идентификатора процесса;

- Уровень важности сообщения (Severity Level).

Может задаваться цифровым кодом или строковым значением (например, строковому представлению уровня важности «[ERROR]» в соответствии со спецификацией формата Syslog сопоставляется целочисленный код «3»).

Кроме того, многие приложения имеют формат сообщений, в целом аналогичный syslog, но включающий специфические поля или стандартные поля, но в специфическом формате. Так, например, формат журнала error.log ПО веб-сервера Nginx по умолчанию генерирует сообщения в следующем формате:

*[yyyy/mm/dd hh:mm:ss] [[level]] [pid#tid:] [*cid] [message]*.

Здесь первые два поля соответствуют временной метке и уровню важности сообщения в обычном формате, а далее идут идентификатор процесса в формате [Идентфикатор процесса#Идентификатор сообщения:] и идентификатор соединения в формате [*Идентификатор сообщения]. А также идентификаторы уровня важности задаются константными значениями, специфичными для данного ПО.

Формат сообщений **common logfile format (clf)** чаще всего используется ПО веб- серверов и имеет следующую структуру [100]:

[remotehost] [rfc931] [authuser] [date] ["request"] [status] [bytes].

Здесь *[remotehost]* – имя или ip-адрес удаленного сетевого узла, *[rfc931]* – идентификатор пользователя в соответствии с RFC 1413 [112], *[authuser]* – идентификатор пользователя-автора запроса, *[date]* - дата (по умолчанию в формате *%d/%b/%Y:%H:%M:%S %z*), *["request"]* – текст запроса, *[status]* – HTTP Status Code ответа на запрос и последнее поле – размер в байтах отданного клиенту. Данный формат более структурирован, чем syslog, и не имеет неструктурированной текстовой части. Тем не менее, из него можно извлечь определенные сведения о неисправностях. Например, комбинация кодов 5xx или 4xx поля *[status]* с запросом к сервису системного ПО СХД может характеризовать программную (неисправность сервиса) или аппаратную (потеря узла, на котором расположен сервис) неисправность.

Наличие априорного знания о формате журналов при решении задачи предотвращения неисправностей в СХД необходимо для извлечения из сообщений журналов метаданных и диагностических признаков. Так как велика вероятность того, что метаданные и диагностические признаки для выявления неисправностей

могут быть обнаружены не только в системных журналах, но и журналах приложений, которые могут иметь свой собственный, не соответствующий какому-либо определенному стандарту формат сообщения, была предусмотрена возможность описания такого формата в рамках онтологической модели.

2.6 Приведение онтологической модели к графовому виду

Для решения задачи построения онтологической модели было использовано инструментальное средство Protégé 5.5 [113], применение которого в настоящее время фактически является стандартом для построения онтологий [114]. Важными особенностями данного пакета, обуславливающими его распространенность, являются:

- возможность автоматической верификации непротиворечивости аксиом и правил онтологии;
- удобный интерфейс редактирования;
- наличие графового визуализатора;
- наличие встроенного языка запросов SPARQL;
- наличие внутренних средств верификации непротиворечивости.

При этом, генерируемый Protégé 5.5 формат онтологии OWL [115], удобный с точки зрения построения модели и наполнения ее сторонними экспертами, достаточно громоздок и неудобен для применения в составе диагностического программного обеспечения. Получаемая структура данных, описывающая иерархическую связь между классами и объектами, содержит избыточные с точки зрения решения задачи диагностики сущности. Например, для определения какого-либо свойства в онтологии создается подкласс «[Имя-свойства]» наследуемый от корневого класса «Ограничение», причем в цепочке предков конкретного свойства может быть несколько уровней вложенности. Кроме того, язык SPARQL, используемый для построения запросов, также достаточно громоздок и неудобен.

Решение, предложенное в настоящей работе, заключается в преобразовании онтологической модели в упрощенный графовый вид и работа с ней в рамках графовой базы данных в процессе эксплуатации программного комплекса.

Для построения упрощенного графового представления модели предполагается, что классам и экземплярам объектов онтологии соответствуют узлы, а связям – дуги графа. При этом, каждой дуге может соответствовать объект, содержащий дополнительные данные, такие как информация о родительских классах связей и т.п. Полученное описание модели соответствует формату rdf-nquad [116], в виде [узел]-<связь> [узел] [контекст]. В таблице 8 приведен пример эквивалентных записей в онтологическом и графовом виде.

Таблица 8 – Преобразование онтологии в упрощенное графовое представление

Модель	**Фрагмент записи**
Онтология	<owl:DatatypeProperty rdf:about="is_normal_when"> <rdfs:subPropertyOf rdf:resource="is_normal"/> <rdfs:range> <rdfs:Datatype> <owl:unionOf rdf:parseType="Collection"> <rdf:Description rdf:about="integer"/> <rdf:Description rdf:about="string"/> </owl:unionOf> </rdfs:Datatype> </rdfs:range> <rdfs:label>is_normal_when</rdfs:label> </owl:DatatypeProperty> <owl:Class rdf:about="Health"> <rdfs:subClassOf rdf:resource="REST"/> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="is_normal_when"/> <owl:hasValue rdf:datatype="string">OK</owl:hasValue> </owl:Restriction> </rdfs:subClassOf> </owl:Class>
Упрощенный граф	Health <is_normal > “OK” [class=”when”] Health <subClassOf> REST .

Используемая в настоящей работе реализация базы данных dgraph [117] в качестве языка запросов использует GraphQL+-[118], являющийся подмножеством языка запросов GraphQL[119]. Это позволяет сделать запросы намного более

читаемыми и уменьшить их размер по сравнению с традиционными запросами на SPARQL[120], см. таблица 9.

Таблица 9 – Сравнение запросов SPARQL и GraphQL+-

SPARQL	GraphQL+-

SELECT DISTINCT ?entity ?symptom_property ?symptom ?parameter ?param_property ?value
WHERE{
?entity rdfs:label "scsi-35000cca03e938d00" .
?entity a owl:NamedIndividual .
?entity a ?subcomponent .
?subcomponent rdfs:subClassOf ?component .
	{ find_component (func: eq(name, "scsi-35000cca03e938d00")) {
	type {
	subClassOf {
	name
	}
	}
	}
	}

?parameter rdfs:subClassOf ?tmp2 .
?tmp2 owl:hasValue ?value .
?param_property rdf:type owl:DatatypeProperty .
?param_property rdfs:subPropertyOf :is_normal .
?tmp2 owl:onProperty ?param_property .
 	
}
ORDER BY ?symptom_property ?parameter ?param_property
	{ find_param_property(func: eq(name, "Health")) {
	name
	subClassOf{
	hasValue
	onProperty{
	name
	}
	}
	}
	}

SELECT ?class ?property ?component
WHERE{
:SCSIDisk rdfs:subClassOf* ?component .
?property rdfs:subPropertyOf :depends_on .
?tmp owl:onProperty ?property .
?tmp owl:someValuesFrom ?component .
?class rdfs:subClassOf ?tmp .
?class rdfs:subClassOf+ :StorageSubsystem .
}
GROUP BY ?class ?property ?component
ORDER BY ?class
	{ find_dependencies(func: eq(name, "SCSIDisk"))
	{
	~someValuesFrom{
	onProperty{ name }
	~subClassOf{ name }
	}
	}
	}

Такое решение имеет сразу несколько ключевых преимуществ:

- в отличие от онтологий, графовая модель идеально приспособлена для поиска связей и кратчайших путей между узлами, что повышает скорость работы диагностической процедуры [121];
- использование графового представления позволяет существенно упростить структуру онтологической модели путем замены онтологических свойств данных и свойств объектов на взвешенные дуги, в следствие чего уменьшается размер и сложность онтологической модели, используемой в составе СПО СХД;
- использование онтологического представления в процессе создания, редактирования и обновления позволит использовать средства проверки непротиворечивости для верификации онтологии;
- структура модели при этом удовлетворяет заданным ограничениям свойств и сохраняет иерархию классов.

2.7 Выводы по главе 2

В главе 2 были решены следующие задачи:

1. Разработан метод построения диагностических моделей на основе онтологии с возможностью преобразования в упрощенную графовую модель, позволяющих решать задачу диагностики вычислительных систем вообще и СХД в частности за счет интеграции и совместного использования формальных описаний неисправностей в СХД, полученных:

- в процессе мониторинга функционирования СХД;
- программного и аппаратного тестирования СХД и ее отдельных компонентов;
- на основании формализации экспертных оценок и технической документации;
- на основании результатов моделирования.

Такая интеграция позволяет получить максимальный охват возникающих в системе неисправностей и локализовать как источник возникновения

нештатного состояния, так и его возможное распространение по другим компонентам и подсистемам СХД путем трассировки связей онтологической модели, приведенной к упрощенному графовому виду. Приведены примеры графов отдельных разделов онтологии, сформулированы подходы к получению информации для построения моделей.

2. Предложен подход, предполагающий применение средств машинного обучения для характеристики отношений между понятиями онтологии, позволяющий оценивать неявные связи, отслеживание которых невозможно без применения такой технологии.

3. Определен подход к использованию журналов СПО СХД в качестве источников диагностической информации.

4. Сформулирована концепция упрощенной графовой модели, предоставляющая возможность применения онтологии, кроме всего прочего, для хранения правил локализации источника возникновения и распространения по топологии СХД неисправностей, выявленных при помощи алгоритма классификации СПО СХД, правил разбора заголовков и текстовой части журналов СПО СХД и настроек диагностической процедуры. Размер модели при приведении ее к графовому виду сокращается в среднем на 22%.

5. Полученный подход, в сравнении с существующими решениями по построению диагностических моделей (см. патенты [122, 123, 124]), является предпочтительным, так как обеспечивает возможность формирования иерархической структуры компонентов и подсистем СХД, использования гетерогенных данных мониторинга СХД в качестве источника диагностических данных, возможность использования модели в автоматической процедуре обнаружения неисправностей.

3 МЕТОД ОБНАРУЖЕНИЯ НЕИСПРАВНОСТЕЙ В СХД НА ОСНОВАНИИ АНАЛИЗА ТЕКСТОВОЙ ИНФОРМАЦИИ, ПОЛУЧАЕМОЙ В ПРОЦЕССЕ МОНИТОРИНГА СХД

3.1 Использование журналов ПО СХД как источника текстовой информации для извлечения диагностических параметров онтологической модели СХД

Применение журналов ПО СХД в качестве источника для извлечения диагностических параметров онтологической модели СХД определяется тем, каким образом можно связать журнал ПО СХД с возникающей в системе неисправностью. В рамках разработанной и представленной в главе 2 онтологической модели предполагается два варианта связи между параметром и событием:

- структурная связь, реализованная на базе объектных свойств онтологии;
- условная связь, реализуемая через промежуточный класс, содержащий внешнюю процедуру обработки.

Журналы ПО СХД могут содержать в себе информацию, которую можно связать с событиями в системе как первым, так и вторым способом. Так как разработанный в настоящей работе способ обнаружения неисправностей направлен на динамическое обнаружение неисправностей в СХД, то вместо работы с отдельными сообщениями или с журналами в целом он в качестве источника для извлечения параметров принимает блоки сообщений журналов ПО СХД, соответствующих временному окну постоянного размера (в простейшем случае данное окно может быть единичной длины, т.е. включать в себя только одно сообщение).

Для работы с текстами журналов ПО СХД как с параметрами, связанными с событиями через структурные связи, необходимо выполнить анализ журналов СХД в соответствии с набором детерминированных правил (на практике это сводится к

проверке наличия во временном окне сообщений определенного типа). Описание данных правил задается в рамках разрабатываемой онтологической модели.

Для работы с текстами журналов СХД как с параметрами, связанными с событиями условными связями, сам текст временного окна рассматривается как анализируемый параметр.

Один и тот же журнал ПО СХД может одновременно быть источником данных и первого, и второго типа, поэтому для практического решения задачи их извлечения предусмотрена универсальная процедура, представленная на рисунке 3.1.

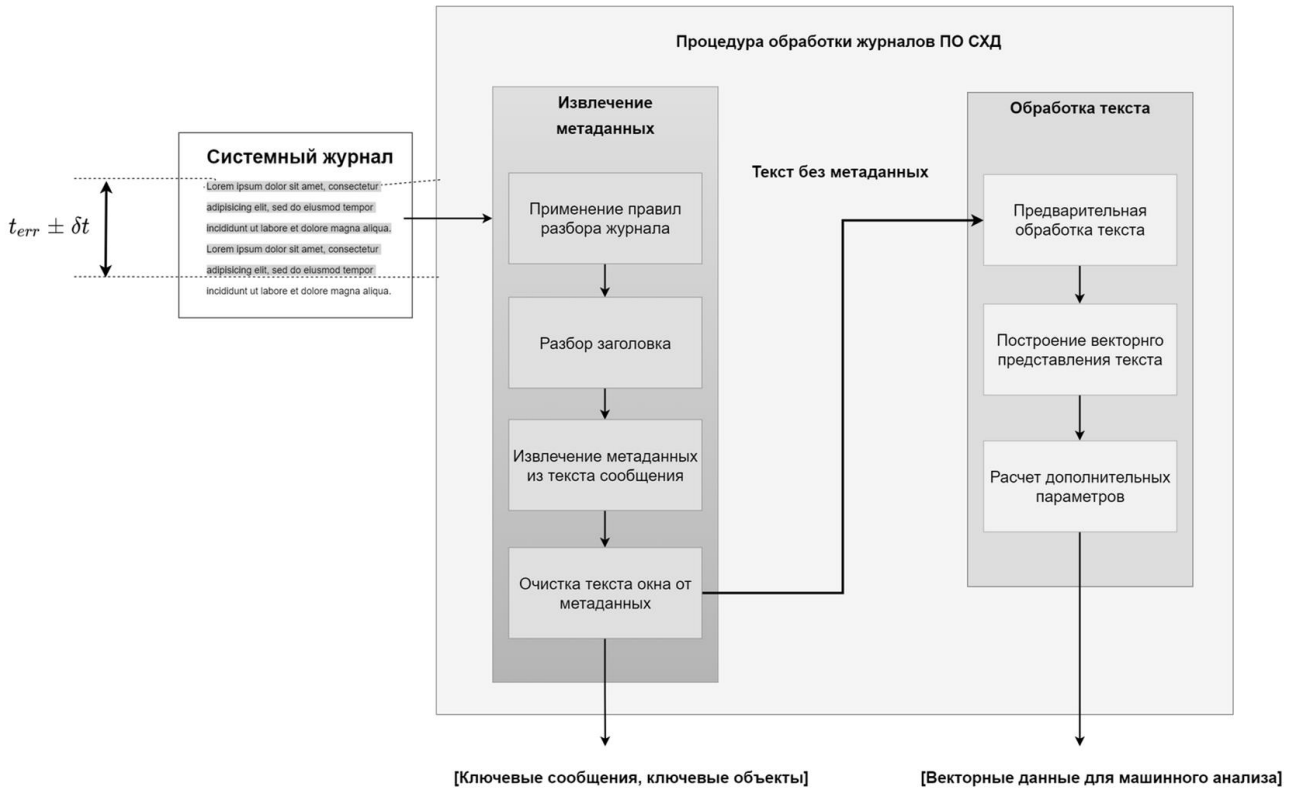


Рисунок 3.1 – Процедура обработки журналов ПО

Применяемые на текущий момент подходы к анализу системных журналов, как уже было указано ранее в разделе 1, чаще всего подразумевают использование набора правил или средств машинного обучения для систематизации структуры журналов и их дальнейшей обработки. В рамках использования результатов представленной на рисунке 3.1 процедуры предлагается применение комбинированного подхода, подразумевающее частичное структурирование

сообщений (в части заголовка), отсечение заголовка и работу с оставшейся частью сообщения как с неструктурированным текстовым блоком.

Схожий подход уже применялся некоторыми исследователями, например, в [16], однако он обладает некоторыми существенными недостатками с точки зрения обнаружения неисправностей:

- применяемый алгоритм направлен на выявление возникающих неисправностей не в момент их возникновения, а в процессе последующего за возникновением неисправности анализа журналов сбойного устройства;
- применяемый алгоритм работает исключительно с сообщениями журналов в формате syslog определенной структуры, что делает невозможным его использование в случае наличия журналов других форматов, или необходимо предварительно выполнить модификацию программных средств, их наполняющих;
- единый формат сообщений позволяет выполнять исключительно их базовую обработку без применения комплексных алгоритмов;
- следствием первых двух особенностей алгоритма является то, что в итоге векторное представление строится в два этапа – сначала для отдельных сообщений, а потом из них строится векторное представление журнала целиком.

3.2 Предварительная обработка журналов ПО

3.2.1 Постановка задачи предварительной обработки журналов ПО

Для обнаружения неисправностей на основании анализа журналов, размещающихся на контроллерах хранения СХД, в качестве источника входных данных могут рассматриваться системные журналы (auth, dmesg, messages и др.), журналы сторонних приложений (веб-серверы, кластерное ПО и др.), использующихся для организации работы СХД, журналы служб и сервисов управляющего ПО СХД (ПО для организации блочного, файлового и объектного доступа и др). Такое разнообразие приводит к необходимости одновременно

обрабатывать журналы разного формата и с различной структурой строк как для построения векторного представления, так и для обучения и применения классификатора обнаружения неисправностей.

Существует два основных подхода извлечения признаков для описания событий в журналах ПО, определенных в ходе анализа предметной области в главе 1 настоящего исследования:

- Подходы, основанные на извлечении из журналов набора шаблонов, соответствующих различным типам сообщений. Например, в [17] предлагается применение теории распознавания образов для выделения из журналов иерархических шаблонов. Извлекаемые по шаблонам в режиме реального времени сведения предоставляются для анализа администраторам или пользователям;
- Альтернативный вариант, подразумевающий использование индивидуального для каждого журнала подхода к выявлению признаков, (как, например, в подходе, описанном А. Панде и В. Ахуджа в [88]) представляется малоэффективным, так как требует недопустимо большого объема действий по предварительной ручной обработке исходных данных.

Полное исключение использования априорного знания о структуре сообщений журналов из процесса анализа системных журналов не представляется разумным, так как помимо собственно текстовой информации из сообщений полезно извлекать дополнительные классифицирующие признаки, такие, например, как временная метка и/или другие части заголовка. Для решения этой задачи в настоящей работе предлагается комбинированный подход, подразумевающий разделение сообщения на структурированную и случайную части, что позволяет работать с сообщением журнала, как с текстом, но при этом дополнительно извлекать из него необходимые метаданные. Для этого, сообщение обрабатывается в два этапа. На первом оно разделяется на заголовок и текстовую часть, для чего требуется априорное знание о его формате. После этого, формируется структура, состоящая из двух типов (метаданные заголовка такие, как

временная метка, и метаданные текста сообщения такие, как имена компонентов СХД, ip- и mac адреса и т.д.) и обработанного текстового блока.

3.2.2 Предварительная обработка и разбор заголовка сообщения

Для извлечения сведений из заголовка сообщения и его последующего удаления из текста сообщения необходимо использовать универсальную процедуру, подходящую для каждого сообщения журнала, по его разбору (парсингу).

Существует три основных способа построения комплексных разборщиков:

- Использование парсер-генератора (в различных источниках называется генератором парсеров, генератором нисходящих анализаторов, parser generator). Парсер генератор – это программный инструмент, предназначенный для синтаксического анализа, использующийся для генерации комплексного парсера для некоторого языка (как правило, компьютерного) на основании формально описанной грамматики этого языка, заданной в форме Бэкус-Наура или дополненной форме Бэкус-Наура [125];
- Использование парсер-комбинатора (комбинаторный парсер, parser combinatory) [126]. Парсер-комбинатор представляет собой объединение набора примитивных парсеров в единый парсер;
- Использование специально написанного парсера для решения конкретной задачи.

Так как заголовок сообщения с точки зрения разбора представляет собой фиксированный набор последовательно расположенных друг за другом полей, то большинство сильных сторон генератора парсеров, таких как приспособленность для решения проблемы левой рекурсии, выявления неоднозначностей и т.д., не представляют особого интереса. Построение специально написанных парсеров также не подходит, так как невозможно заранее предусмотреть все возможные

форматы журналов. Исходя из вышесказанного, наиболее оптимальным подходом к их анализу представляется использование комбинаторного парсинга.

Такой подход подразумевает разработку примитивных парсеров для обработки отдельных полей и построение комбинации из них для разбора заголовка целиком. В случае состоящих из фиксированного набора известных полей заголовков сообщений можно использовать единственный вариант комбинатора – bind [126, страница 5] определяемого в соответствии с формулами (6):

bind :: Parser a -> (a -> Parser b) -> Parser b (6)

p 'bind' f = \inp -> concat [f v inp' | (v,inp') <- p inp]

В таком случае разбор строки S набором парсеров P_i, $i \in [0, N]$ производится следующим образом: на вход P_i примитивного парсера подается строка, остаток строки подается на вход P_{i+1} примитивного парсера, и т.д. После выполнения P_N операция будет считаться завершенной. Процесс разбора строки заголовка сообщения $S = S_{Hi,i\epsilon[0,N]} + S_T$ представлен на рисунке 3.2. Результатом такого разбора является структура $(V(S_{Hi,i\epsilon[0,N]}), S_T)$, где V – словарь, в котором ключом является имя поля заголовка, а значением – результат разбора части строки S_{Hi}.

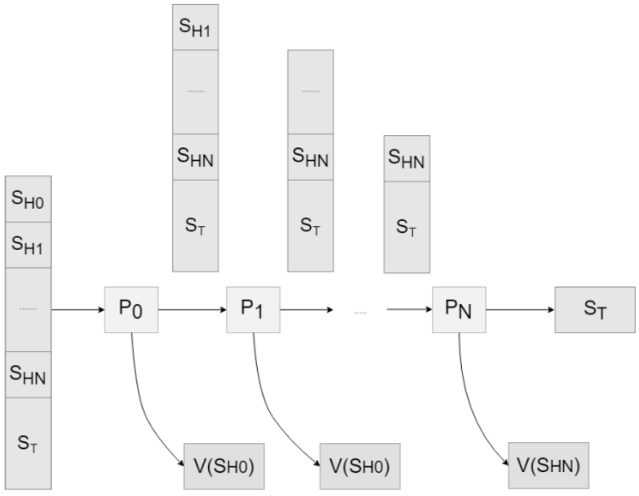


Рисунок 3.2 – Применение комбинаторного парсинга для разбора заголовков

Сведения для построения парсера заголовка сообщения берутся в режиме реального времени из онтологической модели. Предусмотрена двухуровневая система классов, где MessageHeaderFormatField определяет свойства поля как части заголовка конкретного журнала, а MessageHeaderFieldDatatype определяет

свойства объекта, который может быть связан с полями различных журналов, в котором описываются необходимые для разбора таких полей свойства. Таким образом, для разбора конкретного типа журнала строится комбинаторный парсер, обладающий информацией о перечне необходимых примитивных парсеров (связи parser_name и lowlevel_type), наименованиях соответствующих им полей (именованные сущности класса MessageHeaderFormatField) и порядке расположения примитивных парсеров (отношение parsing_order).

3.2.3 Предварительная обработка текстовой части сообщения

Предварительная обработка сообщений для решения задачи анализа сообщений журналов ПО с использованием методов машинного обучения предполагает выполнение определенных действий, направленных на устранение шума в анализируемых данных и снижения вычислительных затрат на выполнение анализа.

Так, например, приведенные Дж. Жу, С. Хе, Дж. Лю и др. в [127] результаты анализа функционирования 13 алгоритмов разбора журналов ПО для выявления шаблонов сообщений свидетельствуют о том, что почти каждый из рассмотренных алгоритмов существенно выигрывает в точности работы в случае его использования на предварительно обработанных журналах (кроме некоторых случаев, таких, как например алгоритм IPLoM [128], что в [87] П. Хе, Дж. Жу, С. Хе и др. объясняют тем, что он уже применяет свои собственные алгоритмы обработки сообщений журналов на первых ступенях выполнения). Предварительная обработка сообщений выполняется при помощи явным образом заданного набора правил, основанного на очевидных допущениях (таких как необходимость извлечения небуквенных токенов) или на знании предметной области приложения, формирующего журнал.

Так как предложенный подход к анализу сообщений журналов основывается на использовании методов, традиционных в NLP, а не использует в процессе работы алгоритмы определения шаблонов сообщений, то для предварительной

обработки сообщений журналов также разумно использовать методы, обычно применяемые для решения задач NLP. Операции, обычно выполняемые в рамках процедуры предварительной обработки текста, зависят от его характера (решаемой в результате обработки текста задачи, языка текста, прочих факторов), так как универсальной процедуры предварительной обработки текста общего назначения не существует. На основании анализа различных источников, например, [129], можно сделать вывод о том, что наиболее часто используются следующие возможные действия по предварительной обработке текста в рамках решения задач NLP:

1. Токенизация текста – разделение текста на отдельные элементы-токены, в простейшем случае это слова, но могут быть и более сложные варианты, такие как н-граммы или предложения;
2. Приведение токенов к единому регистру – в том случае, когда регистр не важен с точки зрения информационной важности токена, необходимо привести все токены к единому регистру, как правило, к нижнему;
3. Удаление знаков препинания – знаки препинания, особенно в случае сообщений из журналов, не имеют никакой собственной ценности;
4. Удаление небуквенных токенов – числа и служебные знаки, как правило, также имеет смысл исключить перед обработкой;
5. Удаление стоп-слов – для каждого языка существует определенный набор слов, не имеющих собственной смысловой нагрузки, исключение которых не повлияет на качество анализа текста;
6. Стемминг (приведение слова к основной форме) или лемматизация (приведение слова к нормальной форме).

Помимо стандартных действий, направленных на повышение эффективности обработки текста алгоритмами машинного обучения, для решения задач, поставленных в рамках настоящей работы, предлагается также выполнять процедуры по подготовке текста, специфичные для конкретного журнала и соответствующей ему предметной области – удалять из текста токены,

соответствующие объектам онтологии, описывающим компоненты СХД, различные идентификаторы, ключевые слова и т.д., причем делать это необходимо перед удалением небуквенных токенов. Для организации процесса такой предварительной обработки предназначен специальный раздел онтологической модели.

3.2.4 Общая структура конвейера для предварительной обработки и фильтрации текстов журналов СХД

Исходя из вышесказанного, а также из необходимости выделять из сообщений метаданные разного типа и строить векторное представление текстовой части, был предложен набор действий, позволяющий организовать конвейер по предварительной обработке текста. При возникновении необходимости повышения скорости обработки сообщений журналов ПО СХД данный конвейер может применяться параллельно, но в общем случае это не рекомендуется, чтобы не повышать нагрузку на вычислительные ресурсы контроллеров хранения.

Процесс обработки текста включает в себя следующие шаги:

1. В соответствии с типом журнала из онтологической модели получаются данные, необходимые для его обработки – формат сообщения и правила извлечения метаданных;
2. Текст разделяется на отдельные сообщения, для каждого из которых выполняются следующие действия:
    2.1. К тексту сообщений применяется набор правил, определяющий наличие в них метаданных, после чего выполняется их удаление из исходного текста;
    2.2. Токенизация текста – в качестве разделителя используются знаки табуляции и пробела. Так как текст в журналах разделен на предложения, то символы перевода строки в момент извлечения текста журнала заменяются на символ пробела;

2.3. Для каждого токена t_i выполняется операция приведения к нижнему регистру, извлечения и удаление метаданных, поиск и удаление небуквенных символов и удаление стоп-слов;

2.4. Для каждого токена применяется алгоритм стемминга (для экономии вычислительных ресурсов было принято решение отказаться от использования алгоритмов лемматизации, так как проведенные испытания показывают, что их использование не оказывает существенное влияние на итоговый результат. В качестве алгоритма стемминга используется алгоритм Портера, как один из наиболее часто используемых и обладающий относительно малым числом недостатков (см. обзорные статьи [130] А. Дживани и [131] М. Харуна). Ключевым преимуществом данного алгоритма является то, что в отличие от множества других он применяет набор правил по трансформации слов, не используя предварительно составленные базы основ слов, что делает его быстрым, но не всегда точным. В том случае, когда стемминг, во-первых, является всего лишь одним из шагов по обработке сообщений журналов ПО, а во-вторых, применяется к специфическому словарю, например, такому, из слов которого состоят журналы ПО, результаты, получаемые при применении более комплексных алгоритмов, не дают заметного преимущества перед алгоритмом Портера;

2.5. Обработанные токены заново объединяются в предложение, в качестве разделителя используется однократный символ пробела;

3. После предварительной обработки текста выполняется его фильтрация на уровне временного окна. На текущий момент добавлены следующие фильтры:

3.1. Фильтрация по частоте появления токена – исключаются уникальные токены, встречающиеся с частотой появления $Fti \leq Ft$, где Ft - минимально допустимая частота появления токена, определяемая для

каждого журнала и хранимая в онтологической модели. По умолчанию Ft = 1;

3.2. Фильтрация по длине сообщения - сообщение пропускается, если после шага 3.1 длина текстовой части сообщения ltext ≤ lmin, где lmin – минимальная длина сообщения, определяемая для каждого журнала и хранимая в онтологической модели. По умолчанию lmin = 0;

4. Выполняется построение векторного представления текстовой части;

5. Выполняется построение вектора дополнительных признаков;

6. В соответствии со списком метаданных и правил их извлечения, полученных из онтологической модели, заполняются значения актуальных для текущего журнала метаданных.

Данный конвейер применяется при считывании сообщений как в процессе формирования обучающих выборок, так и в реальном времени для решения задачи классификации.

3.2.5 Использование алгоритмов извлечения именованных сущностей для обработки текстовой части сообщений

После отделения структурированного заголовка сообщения оставшаяся его часть представляет собой, в общем случае, неупорядоченный набор слов. Тем не менее, этот набор слов может содержать дополнительные диагностические признаки, позволяющие отнести возникшую неисправность к классам онтологии, описывающим иерархическую структуру СХД. Процедура выделения таких признаков параллельно позволяет решить задачу исключения из текстовой части сообщений слов, соответствующих объектам, уникальным для конкретной ситуации (например, mac-адреса сетевых устройств, уникальные идентификаторы узлов уникальны для конкретного экземпляра СХД, уникальные идентификаторы блочных ресурсов вообще существуют на протяжении времени жизни этих ресурсов). Такая фильтрация позволит повысить точность обучения и работы алгоритма классификации. Кроме того, в некоторых случаях, применение

алгоритма выделения именованных сущностей (NER) позволит осуществлять определение компонентов СХД, ассоциированных с возникшей неисправностью.

Упрощенный пример работы такой процедуры (не отображены никакие этапы по обработке сообщений, кроме NER, выполнение которых обязательно в любом случае) представлен на рисунке 3.3.

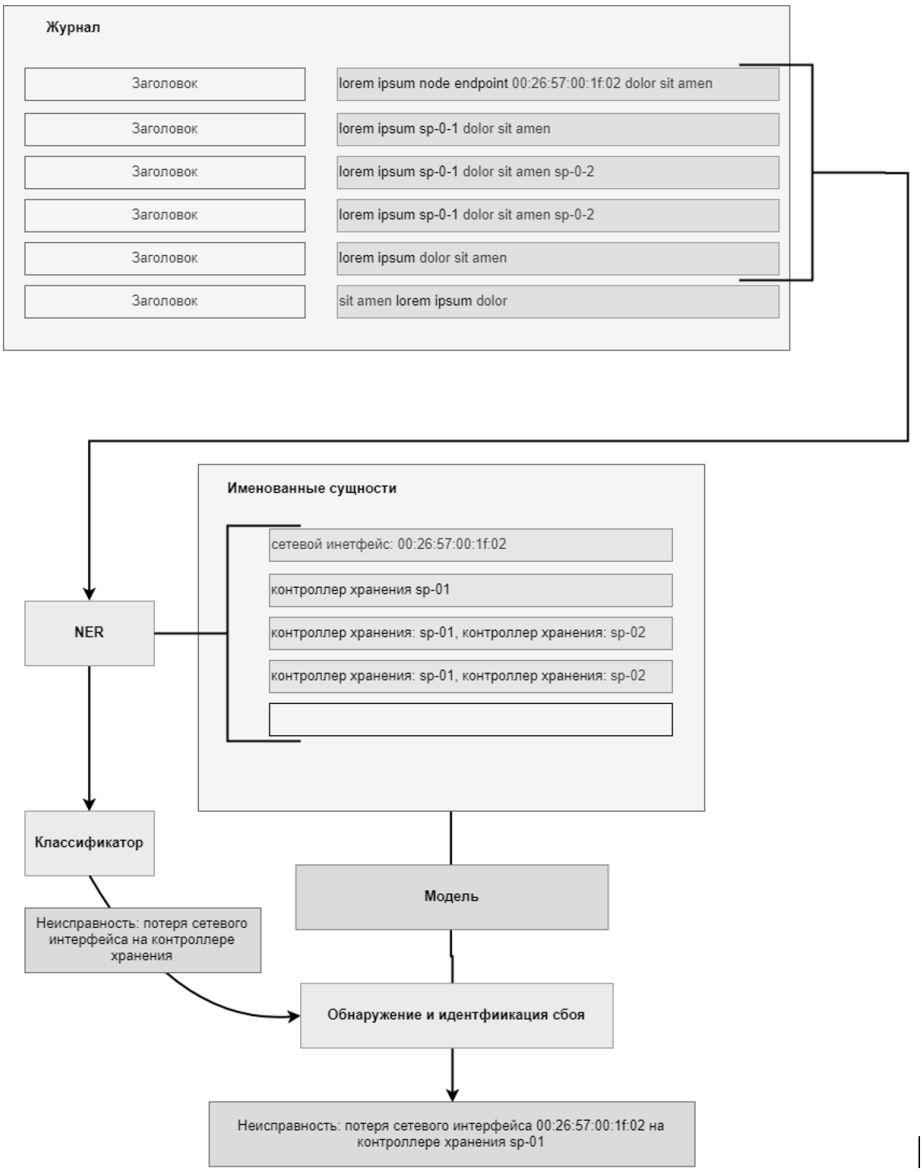


Рисунок 3.3 – Выделение именованных сущностей

Алгоритм NER выделяет поля, соответствующие именованным сущностям, при этом их типы являются классами онтологической модели, после чего текстовая часть сообщения с удаленными именованными сущностями передается для выполнения процедуры классификации неисправностей. В том случае, если

классификатор обнаруживает неисправность F_i, из онтологической модели извлекается перечень классов компонентов C_j, соответствующих данной неисправности. Из перечня именованных сущностей N_k, выявленных NER, определяются конкретные объекты данных классов, после чего выполняется построение итогового объекта, описывающего неисправность (7):

$$F_i' = \{F_i, C(N_j)\}. \quad (7)$$

Надо отметить, что k всегда больше или равно j, так как не все выявленные при помощи NER именованные сущности имеют значение с точки зрения идентификации возникшей неисправности. Обнаруженные в ходе работы NER объекты $N_{k,\,k\geq j}$ удаляются из текста сообщения.

Алгоритмы NER обучаются на предварительно размеченной обучающей выборке отдельно для каждого из журналов, анализ которых требуется выполнять, так как обучение одного алгоритма для всех журналов, очевидно, приводит к снижению точности обнаружения именованных сущностей из-за разного набора обнаруживаемых элементов и контекста.

Использование алгоритмов выделения именованных сущностей в общем случае является избыточным для решения задачи извлечения, фильтрации и классификации диагностических признаков из текстовой части сообщений журналов ПО. На первый взгляд, наиболее очевидным способом решения такого типа задачи является создание набора парсеров (например, на основе регулярных выражений) и правил для выделения именованных сущностей, но реализация такой схемы на базе онтологической модели делает эту модель слишком запутанной, а реализация на уровне программного кода негативно сказывается на универсальности решения. Исходя из этого, было принято решение попробовать применить для решения данной задачи алгоритм выделения именованных сущностей, основанный на статистических методах или средствах машинного обучения. В рамках настоящей работы не ставилась задача разработки специализированного алгоритма, так как на текущий момент представлено

множество реализаций таких алгоритмов, основанных на различных принципах работы и эффективных для различных типов задач.

Несмотря на то, что наиболее подходящим с точки зрения унификации подходов вариантом реализации является использование алгоритмов, основанных на знаниях (knowledge-based), было решено от них отказаться, так как в настоящее время такой подход практически не применяется. Это объясняется тем, что он накладывает даже больше ограничений, чем вариант, основанный на регулярных выражениях и правилах. В частности, проблему представляет идентификация сущностей, не присутствующих в базе знаний (см., например, работу В. Ядава и С. Бетарда - [132]), что делает данный метод неприменимым, например, для идентификации ip-адресов. Исходя из вышесказанного, принято решение использовать один из существующих алгоритмов с использованием средств машинного обучения, что позволяет использовать в процессе работы алгоритма обнаружения неисправностей заранее обученные модели.

Выбор алгоритма выполнялся с учетом следующих предпосылок:

- Все выделяемые именованные сущности отличаются от прочих токенов в тексте содержанием небуквенных символов (“.”, “/”, “-“), что должно существенно упрощать решаемую задачу;
- Значительную сложность при выделении именованных сущностей представляет необходимость решения задачи неоднозначности токенов и идентификации пограничных случаев. По понятным причинам, данные факторы также не актуальны для проблем, решаемых в настоящей работе;
- Не требуется выделения частей речи, позиции в предложении и т.д.;
- Алгоритм применяется к тексту сообщений из журналов ПО, поэтому не должен зависеть от грамматики конкретного языка;
- Из-за значительного числа журналов процесс обучения алгоритма должен быть максимально простым.

В соответствии с этим, приоритет при выборе алгоритма должен отдаваться не комплексным моделям и алгоритмам с большим количеством возможностей, а

более простым, но производительным и не требующим или имеющим максимально упрощенный процесс обучения на небольшом объеме обучающей выборки.

3.3 Применение алгоритмов обработки текста для анализа сообщений

В настоящей работе алгоритмы машинной обработки естественного языка применяются для решения двух задач: обнаружения именованных сущностей в задаче локализации неисправностей и для классификации текста сообщений.

Для выбора **классификатора текста** проведена оценка качества классификации разными алгоритмами на имеющихся в наличии размеченных данных. В качестве базовой модели для сравнения алгоритмов рассматривался классический, но на настоящий момент несколько устаревший, алгоритм логистической регрессии [133].

В качестве альтернативных вариантов рассмотрены алгоритмы «случайный лес» (random forest) [134] и градиентного бустинга (gradient boosting) [135], как обеспечивающие наиболее качественную классификацию при относительно невысоком потреблении ресурсов.

Алгоритм «случайный лес» основывается на построении ансамбля решающих деревьев с голосованием по большинству (majority voting ensemble) в случае применения в задачах классификации.

В общем виде алгоритм градиентного бустинга подразумевает итеративное построение из ансамбля «слабых» предсказывающих моделей (weak prediction models) результирующей «сильной» модели предсказания путем минимизации функции потерь. Если предположить, что имеется набор признаков X и набор целевых значений, определяемых как $y = f(x)$, то задача получения целевых значений сводится к аппроксимации $f(x)$ и определению наиболее успешной аппроксимации путем использования функции потерь $L(y,f)$, которая подвергается минимизации [136] – формула (8):

$$y \approx f\hat{}(x), f\hat{}(x) = arg \min_{f(x)} L(y, f(x)). \quad (8)$$

Для частного вида этой же формулы на конкретном семействе функций и текущим набором анализируемых данных получается (9), (10):

$$\widehat{F} = \sum_1^M \widehat{F}_i, ; \quad (9)$$

$$L_F(\widehat{F}) = \sum_1^N L\left(y_i, f\left(x_i, \widehat{F}\right)\right). \quad (10)$$

Для решения задачи последнее выражение итеративно минимизируется с применением алгоритма градиентного спуска.

Традиционным подходом для обработки (выделение частей речи, NER, классификация и т.д.) текста является использование нейронных сетей с использованием модулей долгой краткосрочной памяти. Так применение таких сетей для решения задач NER впервые описано в [136]. LSTM – это вариант реализации рекуррентной нейронной сети, специально разработанный для того, чтобы избежать проблемы длительных временных разрывов между событиями, что достигается путем замены скрытых узлов в архитектуре RNN на специально разработанные узлы-ячейки памяти.

В [136] приводится структура отдельной ячейки памяти LSTM, см. рисунок 3.4:

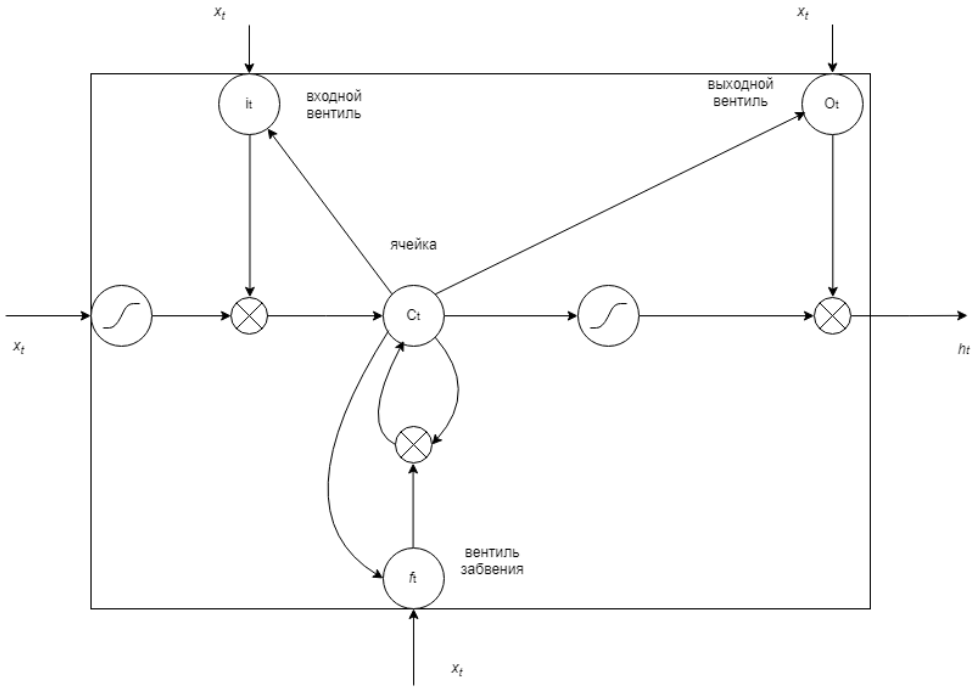


Рисунок 3.4 – Ячейка памяти LSTM с тремя вентилями

Данная ячейка памяти реализуется следующим образом (11):

$$\begin{gathered} i_t = \sigma(W_{xi}x_t + W_{hi}x_{t-1} + W_{ci}c_{t-1} + b_i), \\ f_t = \sigma\big(W_{xf}x_t + W_{hf}x_{t-1} + W_{cf}c_{t-1} + b_f\big), \\ c_t = f_t C_{t-1} + itanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \\ c_t = \sigma(W_{xO}x_t + W_{hO}x_{t-1} + W_{cO}c_{t-1} + b_O), \\ h_t = \ O_t \tanh(c_t). \end{gathered} \tag{11}$$

Здесь σ – функции активации, i,f,O,c – векторы входного вентиля, вентиля забвения, выходного вентиля и состояния, размерности которых равны размерности скрытого вектора h, W – матрицы весов, в которых индексы соответствуют названиям векторов.

Помимо уже упоминавшихся ранее классификаторов на основе алгоритмов решающих деревьев, градиентного бустинга и нейронных сетей на базе долгой краткосрочной памяти для классификации текстовой части сообщений также используются еще два типа нейронных сетей: нейронные сети на основе долгой краткосрочной памяти с механизмом внимания и нейронные сети на базе управляемого рекуррентного блока.

Нейронные сети на базе управляемого рекуррентного блока, применяемые в решении задачи классификации, например в работе Т. Ле, Дж. Ким, Х. Ким - [137], представляют собой некоторый упрощенный вариант долгой краткосрочной памяти, обладая всего двумя вентилями вместо трех (отсутствует выходной вентиль). Работа такого нейрона описывается следующими уравнениями (12):

$$\begin{gathered} z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \\ r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r), \\ h_t = z_t h_{t-1} + (1 - z_t)\sigma_h(W_h x_t + U_h(r_t h_{t-1}) + b_h), \end{gathered} \tag{12}$$

где x_t, h_t – входные и выходные вектора,

z_t и r_t – векторы вентилей сброса и обновления,

W, U, b – матрицы и вектор параметров,

σ_g и σ_h – функции активации.

Нейронные сети на базе долгой краткосрочной памяти с механизмом внимания (рассматривается, например, в работе Г. Лао, Дж. Гуо – [138]**)** впервые применены как раз в задачах NLP, для решения проблемы машинного перевода. Реализация механизма внимания подразумевает выделение некоторых частей текста, которые анализируются особенно тщательно, путем формирования матрицы весов важности. Построение матрицы осуществляется с использованием метода обратного распространения ошибки (error back progagation).

3.3.1 Определение метрик для сравнения результатов разных подходов

В настоящее время фактическим стандартом для оценки качества работы алгоритмов машинного обучения считается использование метрик: точность (precision), полнота (recall) и f-мера (f-score). Предлагаемые в более поздних источниках, например, при решении задач именованных сущностей, более сложные меры, такие, как Correct (COR)/ Incorrect (INC)/ Partial (PAR)/ Missing (MIS)/ Spurius (SPU) [139] или шестиступенчатая процедура оценки, предложенная в [140], не получили широкого распространения и являются очевидно избыточными для решаемой задачи.

Точность и полнота вычисляются по следующим формулам (13), (14):

$$\text{точность} = \frac{TP}{TP+FP}, \tag{13}$$

$$\text{полнота} = \frac{TP}{TP+FN}. \tag{14}$$

В этих формулах TP – число истинно-положительных, FP – число ложноположительных, а FN – число ложноотрицательных срабатываний алгоритма.

F-мера предназначена для построения единого критерия на базе точности и полноты и представляет собой их отношение, вычисляемое по следующей формуле (15):

$$f - \text{мера} = \frac{2*\text{точность}*\text{полнота}}{\text{точость}+\text{полнота}}. \tag{15}$$

При этом для различных задач NLP данные метрики могут определяться немного по-разному. Особенность алгоритмов NER заключается в том, что в зависимости о решаемой задачи и особенностей применения алгоритма разные авторы по-разному формулируют, каким образом определяются истинно-положительные, ложноположительные и ложноотрицательные результаты его работы.

3.4 Анализ исходных данных

Объектом исследования является пакет файлов журналов системного ПО СХД, размещаемый в процессе работы СХД на контроллерах хранения и фабрике. В соответствии с критериями, предъявляемыми в разделе 1, из всего списка отобран следующий набор из 33 журналов, в дальнейшем именующихся slg0 – slg32. В этот список входят как системные журналы, так и журналы, соответствующие специфическому программному обеспечению СХД. Некоторое программное обеспечение может в процессе работы вести сразу несколько журналов, таким образом в пакет входит от 99 до 305 файлов журналов.

Общая выборка содержит 5904 пакета журналов (что составляет порядка 350Гб файлов журналов), размещаемых на контроллерах хранения СХД, в том числе 1574 соответствующих возникавшим когда-либо в процессе эксплуатации СХД неисправностям, с указанием примерного временного диапазона возникновения неисправностей. Всего в пакетах идентифицирован 41 тип различных неисправностей. Полный перечень возникавших типов неисправностей приведен в таблице 10.

Таблица 10 – Полный перечень идентифицированных неисправностей

Код типа неисправности	Категория неисправности в соответствии с источником происхождения
1	Построение логического тома из носителей информации
2	Схема резервирования
3	Применение схемы резервирования
4	Построение пула хранения носителей информации
5	Построение пула хранения носителей информации

Продолжение таблицы 10

6	Построение пула хранения носителей информации
7	Проверка состояния аппаратуры
8	Маршрутизация данных
9	Построение пула хранения носителей информации
10	Построение пула хранения носителей информации
11	Восстановление носителей информации
12	Взаимодействие между узлами СХД
13	Построение пула хранения носителей информации
14	Маршрутизация данных
15	Взаимодействие между узлами СХД
16	Взаимодействие между узлами СХД
17	Взаимодействие между узлами СХД
18	Взаимодействие между узлами СХД
19	Удаление пула хранения
20	Взаимодействие логических томов
21	Проверка доступности устройств
22	Проверка доступности устройств
23	Удаление пула хранения
24	Проверка доступности устройств
25	Проверка доступности устройств
26	Неисправность управления пулом хранения
27	Неисправность управления пулом хранения
28	Неисправность управления пулом хранения
29	Неисправность управления пулом хранения
30	Неисправность управления пулом хранения
31	Неисправность управления пулом хранения
32	Неисправность управления пулом хранения
33	Неисправность управления пулом хранения
34	Неисправность управления пулом хранения
35	Проверка ограничений
36	Проверка доступности устройств
37	Работа системы уведомления
38	Работа системы уведомления
39	Удаление пула хранения
40	Удаление пула хранения
41	Работа системы уведомления

Сравнительная частота отдельных неисправностей от общего количества пакетов со сбоями представлена на рисунке 3.5. Как видно из получившейся диаграммы, исходная выборка достаточно хорошо сбалансирована, разница между

появлением самого частого сбоя и самого редкого сбоя составляет менее одного порядка.

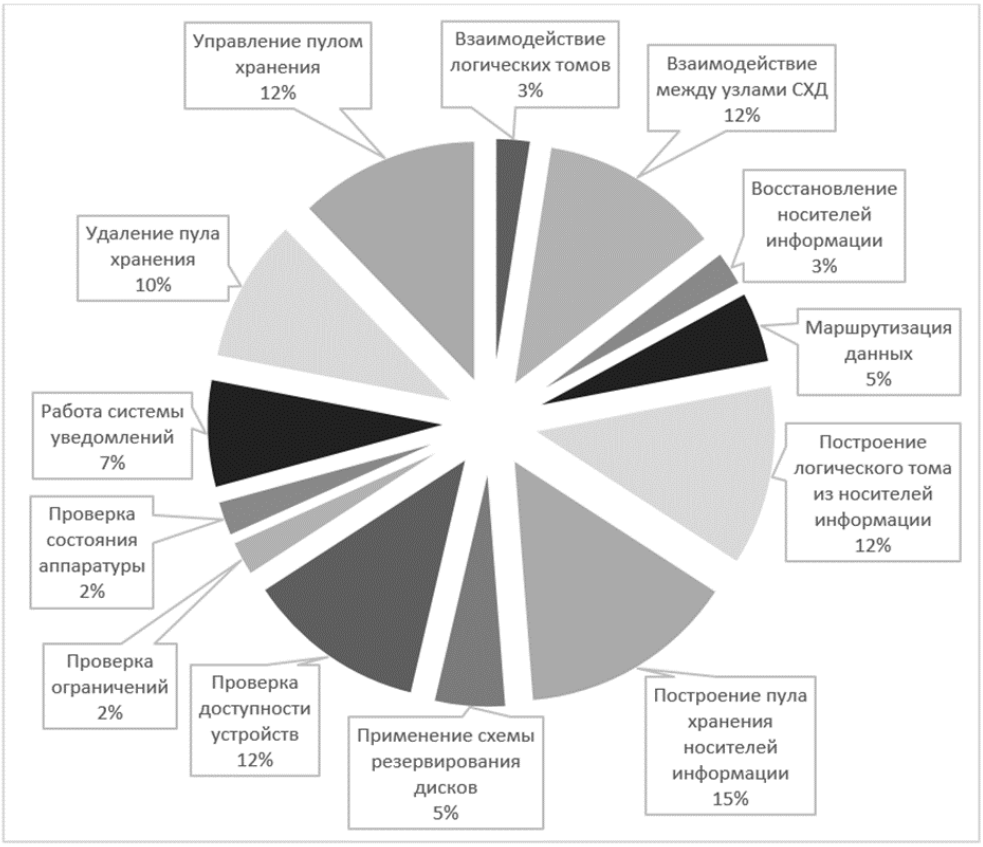


Рисунок 3.5 – Сравнительная частота возникновения неисправностей

В качестве меры сбалансированности ***B*** использовалось отношение энтропии по Шэннону к двоичному логарифму числа классов [141] (16):

$$B = \frac{-\sum_{i=1}^{k} \frac{c_i}{n} \log_2 \frac{c_i}{n}}{\log_2 k}, \quad (16)$$

где k – число различных классов,

c_i – размер класса,

n – размер выборки.

Данная мера принимает значение 0, когда набор полностью не сбалансирован (0, когда есть всего один большой класс), и 1, когда набор состоит из классов одинакового размера. Значение меры Шэнннона для данного набора классов

составляет 0,974, что означает, что набор данных достаточно хорошо сбалансирован.

Для построения обучающих и тестовых выборок выполнена следующая последовательность действий:

- На первом шаге выполняется распаковка упакованных журналов, объединение фрагментов однотипных журналов, подвергшихся ротации, удаление неиспользуемых журналов из пакетов;
- На втором этапе из журнала фиксации неисправностей извлекаются сведения о наличии или отсутствии неисправностей в данном пакете журналов, временные интервалы начала и окончания неисправностей;
- На третьем шаге выполняется обработка журналов в соответствии с правилами онтологической модели – фильтрация, отделение заголовка от текстовой части, токенизация текстовой части, разбор временной метки;
- На четвертом шаге выполняется извлечение из разобранных журналов диапазонов сообщений, соответствующих временным окнам возникновения неисправностей.

Для всех извлеченных таким образом временных окон выполняется построение векторов диагностических признаков, после чего полученные результаты анализируются с использованием выбранного классификатора. На рисунке 3.6 приведено построение объектов классов неисправностей в двумерном пространстве, выполненное при помощи алгоритма T-SNE (широко применяется для визуализации данных, например в работе М. Баламурали и А. Мелкумяна – [142]) по дополнительным координатам. Из общего вида рисунка можно сделать вывод о том, что классы неисправностей достаточно сильно перемешаны и для эффективной классификации необходимо использовать не только дополнительные параметры текстов журналов, но и их векторные представления, что резко увеличивает размерность параметрического пространства.

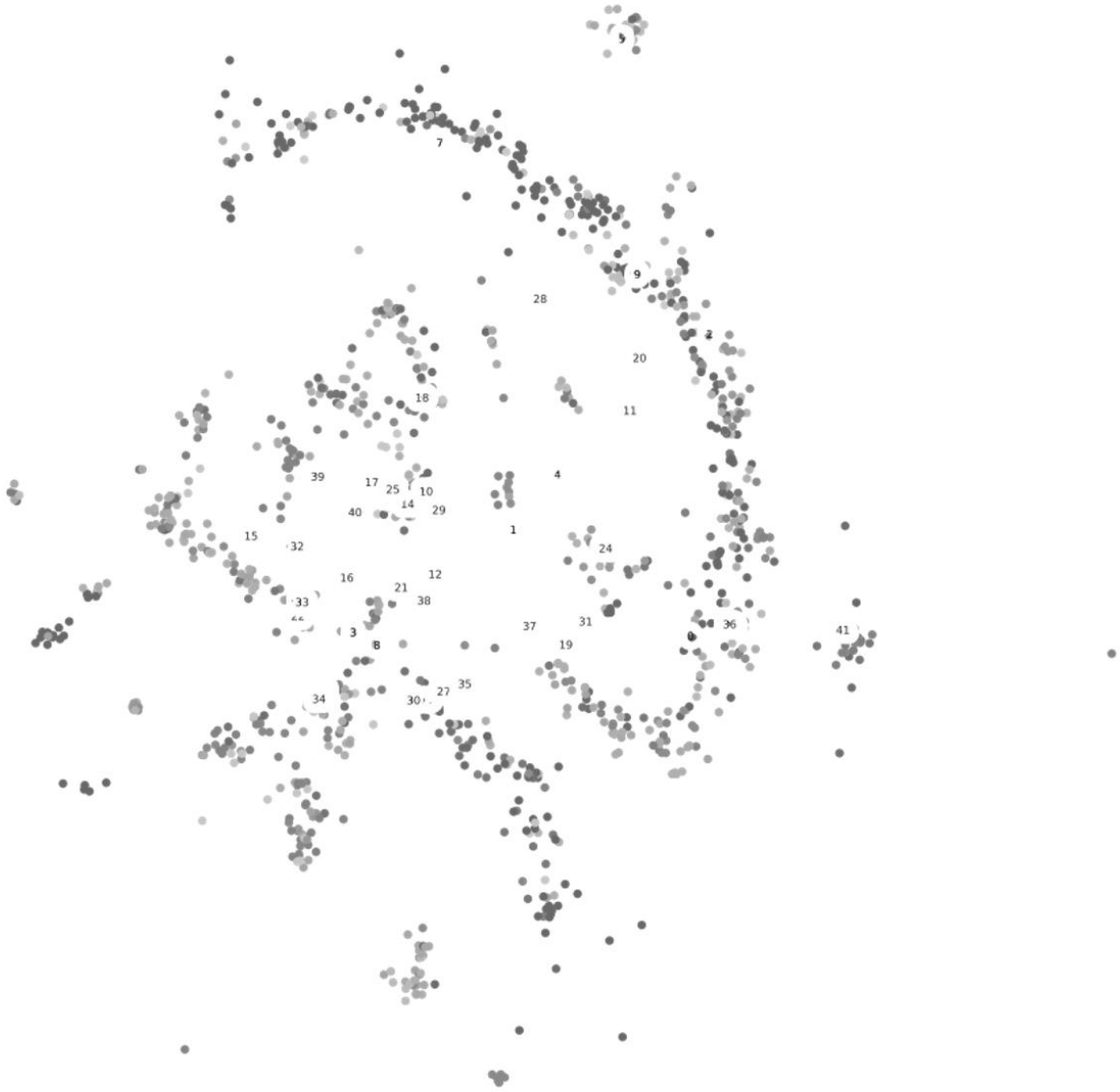


Рисунок 3.6 – Двумерная визуализация классов неисправностей

Оценка близости классов при помощи алгоритма аггломеративной иерархической кластеризации [143], при котором последовательно выполняется попарное объединение ближайших с точки зрения геометрического расстояния объектов до тех пор, пока вся выборка не объединится в один класс, подтверждает сделанные ранее выводы – объекты из объединенной выборки, состоящей из объектов обучающей и тестовой выборок, объединяются случайным образом, см. рисунок В.1 приложения В. Несмотря на то, что на рисунке можно визуально

выделить крупные кластеры, это объединение не имеет никакого отношения к фактической классификации объектов.

Исходя из вышеприведенного анализа набора исходных данных, можно утверждать, что для его классификации затруднительно использовать традиционные математические методы из-за высокой размерности параметрического пространства. Для решения данной проблемы в настоящей работе применяются методы, основанные на использовании машинного обучения, способные обнаруживать более сложные связи между объектами.

3.5 Формирование параметрического пространства для описания текстовой части журнала

Так как математические методы, предназначенные для классификации текста, не могут принимать на вход непосредственно текстовые данные, для того, чтобы обеспечить возможность их применения, необходимо представить текст в виде набора векторов чисел. Для решения этой задачи в настоящей работе используются комбинация двух различных подходов:

- построение векторного представления непосредственно текстов журналов V_t;
- расчет дополнительных параметров, описывающих текст журнала X_m.

Значения данных признаков определяются по каждому журналу в рамках временного окна, после чего они объединяются в единый вектор $X_u = \{V_t, X_m\}$ и подаются на вход классификатора.

3.5.1 Построение векторного представления текстовой текста журнала

Построение матрицы значений, соответствующих каждому некоторому элементу текста, называется построением векторного представления текста. На текущий момент существует множество подходов к построению векторного представления, которые можно классифицировать по различным признакам:

- по уровню применения алгоритма векторизации: на уровне отдельных символов (например, Fasttext, предложенный в [144] А. Жуленом, Э. Гравом и П. Бояновским и др.), на уровне слов (например, word2vec, предложенный в [145] Т. Миколовым, К. Ченом, Г. Коррадо и др.; ELMO, предложенный в [146] М. Петерсом, М. Ньюманом, К., Гарднером и др.; Google BERT [147] Дж. Девлина, М. Чана, К. Ли и др.; ALBERT, описанный Л. Женжоном, М. Ченом, С. Гудменом и др. в [148]), на уровне документов (doc2vec, предложенный К. Ле и Т. Миколовым в [149]) или отдельных предложений (sentence2vec см. также К. Ле и Т. Миколов, [150]);
- по способу реализации: от самого простого, One Hot Encoding, где подсчитывается частота вхождения каждого токена, до глубоких нейронных сетей в ранее упоминавшихся новейших моделях типа BERT, ALBERT, ELMO и т.д.
- по зависимости от контекста: в зависимости от сложности модели, могут быть полностью контексто-независимые (разные кодировщики), большинство сложных моделей в той или иной степени строят вектор слова (символа) в зависимости от окружающих его слов.

Несмотря на то, что на текущий момент наилучшие результаты в решении задач NLP показывают сложные модели, такие как BERT, ELMO и др., в настоящей работе решено отказаться от их использования по ряду причин, в том числе:

- Большой размер моделей. Как правило, данные модели предварительно обучены на выборках огромного размера, таких, например, как полный текст Википедии;
- Сравнительно высокая потребность в вычислительных ресурсах при построении векторного представления.

Так как использование таких моделей предполагается для решения задачи обнаружения неисправностей в составе встроенного программного обеспечения СХД, необходимо ориентироваться на минимальное потребление вычислительных ресурсов. Кроме того, словарь, используемый при построении сообщений

журналов программного обеспечения, существенно отличается от общеупотребимой лексики, а структура сообщений отличается от структуры предложений естественного языка, что приводит к снижению эффективности таких моделей. Другой особенностью текста журналов ПО СХД является то, что контекстная зависимость между одними и теми же токенами может быть совершенно разной для разных типов системных журналов, что вызывает сложность при формировании закономерностей.

В целом, решение задачи разработки наиболее эффективной модели для построения векторного представления сообщений журналов программного обучения, хотя и представляет значительный интерес, но выходит за рамки настоящего исследования.

Для построения векторного представления документа в целом, как правило, применяется усреднение векторов отдельных слов/символов на уровне документа (в случае настоящего исследования – временного окна), либо применяются уже готовые методы, такие как doc2vec/sentence2vec. Для повышения эффективности построения векторного представления текста в настоящем исследовании предложен подход, предполагающий совместное использование двух типов коэффициентов:

- Весовой коэффициент $W_{tf\text{-}idf}$, характеризующий важность слова в рамках контекста и представляющий собой отношение частоты появления слова в документе к частоте употребления слова во всех документах из выборки;
- Весовой коэффициент W_{err}, характеризующий важность слова для идентификации неисправности.

Коэффициент W_{err} вводится в настоящей работе для того, чтобы компенсировать дискретность размера временного окна, используемого для построения векторного представления сообщений. Так, при размере временного окна длиной в 1 минуту сообщения, зарегистрированные в начале временного окна, очевидно имеют меньшую важность для идентификации возникающих

неисправностей, чем сообщения, зафиксированные ближе к концу временного окна.

Таким образом, итоговый вектор V_w, описывающий каждое слово, имеет следующий вид (17):

$$V_w = W_{tf-idf} W_{err}[V]. \quad (17)$$

Соответственно, вектор V_t, описывающий текущее временное окно, должен быть определен как (18):

$$V_t = \frac{\sum_{i=1}^{n} W_{tf-idf} W_{err}[V_i]}{n}, \quad (18)$$

где n – общее число слов в выборке.

На рисунке 3.7 представлен алгоритм построения векторной модели текстовой части сообщений:

Рисунок 3.7 – Алгоритм построения векторной модели текстовой части сообщения

В рамках настоящего исследования рассмотрено два способа векторизации текста. В первом случае тексты всех журналов объединялись в рамках временного окна, и выполнялось построение векторного представления для текста V_t (19):

$$V_t = V\left(\sum_{k=1}^{n} V(T_k)\right). \quad (19)$$

Во втором случае векторное представление определяется для каждого журнала в отдельности и, соответственно, представляет собой набор векторов V_{ti}, каждый из которых является отдельным признаком для использования в алгоритмах машинного обучения (20):

$$V_t = \{V_k(T_k)\}_{k=1,n}. \quad (20)$$

Для сравнения вышеуказанных подходов выполнено обучение и проверка точности классификации для разных типов построения векторного представления для всех анализируемых классификаторов без учета дополнительных признаков. Исходя из результатов данной проверки, можно утверждать, что вариант, указанный в (3) дает существенное (порядка 40%) повышение точности классификации.

3.5.2 Определение дополнительных признаков для описания текста журналов

Для решения задачи классификации текстов значительную, а иногда даже и более значимую роль, чем собственно векторизация текста, могут играть дополнительные признаки, косвенно характеризующие документ целиком. Это могут быть лексические (количество определенных слов в тексте) и синтаксические признаки текста (количество определенных частей речи в тексте), частотные характеристики (частота сообщений за временное окно) и т.д. Для случая системных журналов использование синтаксических признаков не пригодно, так как сообщения в необработанном виде содержат существенное количество элементов, затрудняющих или делающих невозможным их определение, а при обработке сообщения в общем случае могут терять свою

синтаксическую структуру. В ходе настоящего исследования предложены и рассмотрены следующие дополнительные признаки, представленные в таблице 11, использование которых позволило повысить точность классификации текста.

Таблица 11 – Дополнительные признаки текстовой части сообщений

Название	**Значение признака**	**Описание параметра**
Количество слов в i-ом журнале	Целочисленное	Подсчитывается общее количество слов во временном окне, соответствующем каждому журналу
Количество сообщений в i-ом журнале	Целочисленное	Подсчитывается общее количество сообщений во временном окне, соответствующем каждому журналу
Средняя длина сообщения в i-ом журнале	С плавающей точкой	Подсчитывается средняя длина, по каждому сообщению во временном окне, соответствующем каждому журналу
Наличие сообщений в i-ом журнале	Целочисленное	Фиксируется наличие сообщений по каждому журналу в процессе построения временного окна (наличию соответствует 0, отсутствию – 1)
Среднее количество слов в секунду в i-ом журнале	С плавающей точкой	Фиксируется количество слов в секунду для каждого журнала
Среднее количество сообщений в секунду в i-ом журнале	С плавающей точкой	Фиксируется среднее количество сообщений за одну секунду временного окна

Значения дополнительных признаков определяется автоматически в процессе построения временного окна.

Для анализа полезности данных признаков их полный набор вычислен на обучающей выборке, сформированной из имеющихся в наличии размеченных исходных данных. Для полученных таким образом векторов координат обучающей выборки определена оценка их важности при помощи классификатора «случайный лес».

Оценка важности признаков выполняется в процессе работы классификатора на каждом этапе построения узла дерева на основании определения наилучшего из набора случайных признаков путем вычисления критерия Джини [151], см. формула (21):

$$G(k) = \sum_{i=0}^{J} \mathrm{P(i)} * \big(1 - \mathrm{P(i)}\big), \tag{21}$$

где $P(i) -$ вероятность классификации i для признака k.

Для проверки данного предположения выполнен альтернативный отбор признаков при помощи распространенного в сфере машинного обучения подхода, основанного на методах дисперсионного анализа (Analysis of Variance, ANOVA) [152], а именно, путем определения F-критерия для множества признаков. Фактически, F-критерий представляет собой отношение межгрупповой дисперсии к внутригрупповой дисперсии (22):

$$Var_b = \frac{\sum_i n_i(\bar{Y}i - \overline{Y})^2}{K-1}, \tag{22}$$

где K — количество групп,

$\bar{Y}i$ – среднее значение для i-ой группы,

n_i - число наблюдений в i-ой группе,

$\bar{Y}$ – среднее по всей выборке (23):

$$Var_w = \frac{\sum_{ij} n_i(\bar{Y}ij - \overline{Yi})^2}{N-K}, \tag{23}$$

где $\bar{Y}ij$ – это j-ое наблюдение в i-ой из K групп,

N — общий объем выборки.

Тогда значение – F-критерия определяется как (24):

$$\mathrm{F} = \frac{Var_b}{Var_w}. \tag{24}$$

Полученное с использованием коэффициента Джини распределение оценок важности отдельных признаков приведено на рисунке 3.8.

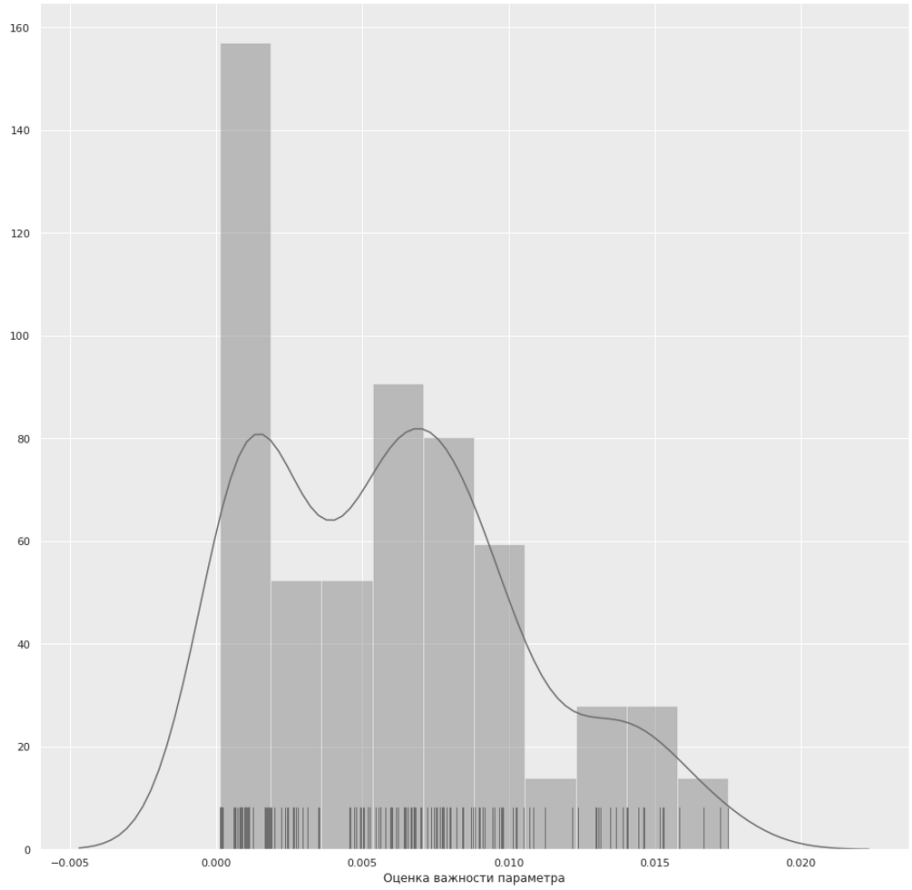


Рисунок 3.8 – Распределение оценок важностей признаков

В таблице 12 приведен отсортированный по убыванию значения дополнительного признака набор параметров, у которых значение признака больше, чем пороговое значение 0.01, а название журнала заменено на обозначение log N.

Таблица 12 – Оценка важности дополнительных признаков

№п.п.	Признак	Оценка
1	slg 8 число слов	0,0175
2	slg 8 число сообщений	0,01723
3	slg 28 длина сообщения	0,01668
4	slg 8 длина сообщения	0,01582
5	slg 27 число сообщений	0,01529

Продолжение таблицы 12

№п.п.	Признак	Оценка
6	slg 19 число слов	0,01527
7	slg 8 токенов./сек	0,01516
8	slg 27 число слов	0,01465
9	slg 8 сообщ./сек	0,01461
10	slg 19 токенов./сек	0,01445
11	slg 19 число сообщений	0,01408
12	slg 19 длина сообщения	0,01403
13	slg 19 сообщ./сек	0,01391
14	slg 3 число слов	0,01366
15	slg 27 токенов./сек	0,01347
16	slg 27 сообщ./сек	0,01314
17	slg 27 длина сообщения	0,01308
18	slg 3 число сообщений	0,01301
19	slg 0 длина сообщения	0,01296
20	slg 15 длина сообщения	0,01237
21	slg 30 длина сообщения	0,01216
22	slg 5 длина сообщения	0,01123
23	slg 12 длина сообщения	0,01086
24	slg 31 длина сообщения	0,01072
25	slg 28 число сообщений	0,01052
26	slg 0 токенов./сек	0,01028
27	slg 28 число слов	0,01023
28	slg 26 длина сообщения	0,01016

Как видно из данной таблицы, в списке наиболее важных признаков встречаются все признаки из таблицы 11, кроме признака «Наличие сообщений в i-ом журнале». Таким образом, можно сделать предположение о том, что все

перечисленные признаки, кроме признака «Наличие сообщений в i-ом журнале» необходимо включить в итоговый список дополнительных признаков.

Полученное распределение признаков при помощи F-меры в целом повторяет сделанные ранее выводы. Распределение значений признаков приведено на рисунке 3.9:

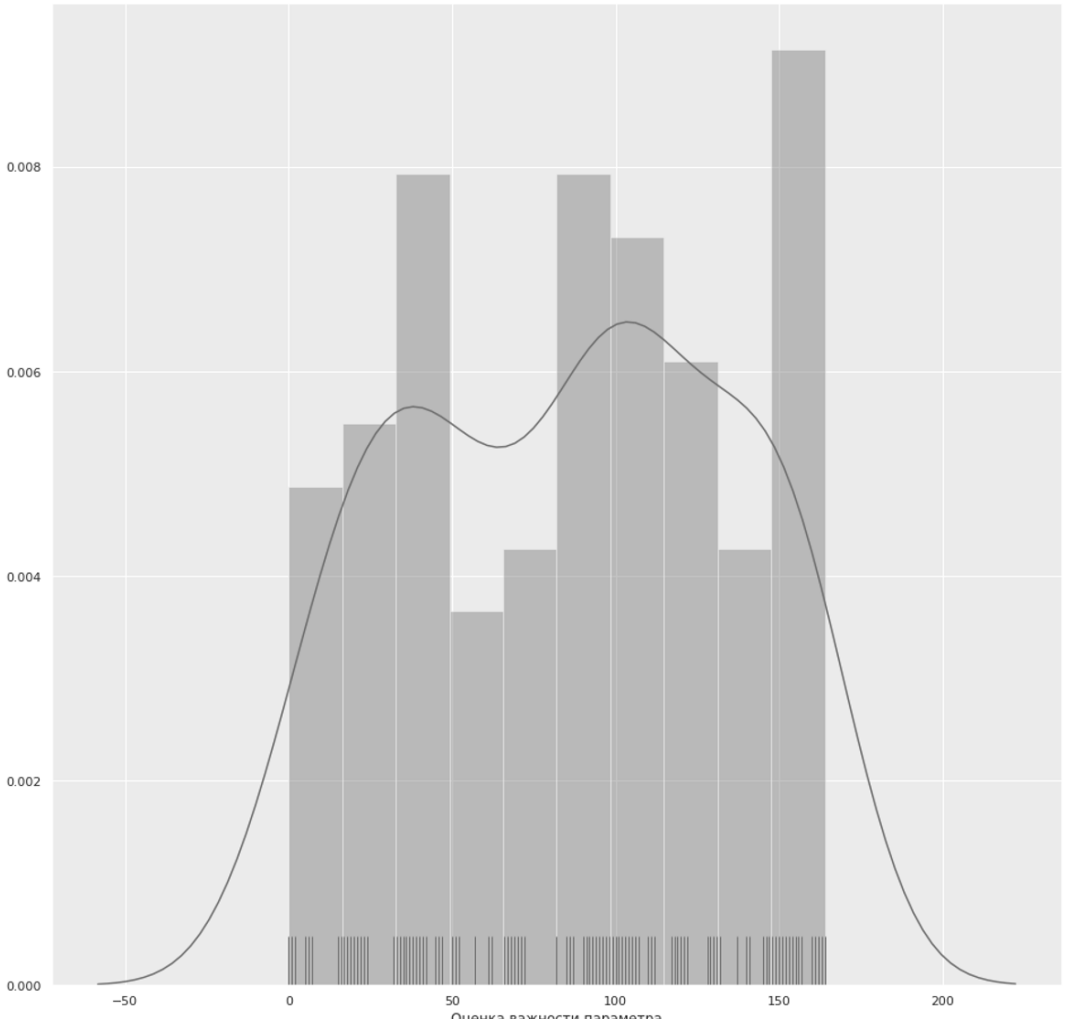


Рисунок 3.9 – Оценка важности параметра

Для сравнения с полученными ранее результатами взяты 30 признаков с наивысшей оценкой, см. таблица 13.

Как видно из таблицы, при таком определении важности признаков также представлены все имеющиеся типы признаков, кроме признака «Наличие сообщений в i-ом журнале». Исходя из этого, можно сделать вывод о том, что предположение о полезности всех признаков, кроме вышеуказанного, можно считать подтвержденным, и при классификации текстов необходимо учитывать все

дополнительные признаки. При этом, так как перечень журналов, описываемых данными признаками, практически полностью не совпадает с полученным ранее, для расчета признаков необходимо использовать полный список имеющихся журналов.

Таблица 13 – признаки с максимальным значением F-критерия

№пп	Параметр	Значение
0	slg 27 число сообщений	120
1	slg 27 длина сообщения	121
2	slg 27 токенов/сек	122
3	slg 27 сообщ./сек	128
4	slg 30 число токенов	129
5	slg 30 число сообщений	130
6	slg 30 длина сообщения	131
7	slg 30 токенов/сек	132
8	slg 30 сообщ./сек	137
9	slg 28 число токенов	140
10	slg 28 число сообщений	141
11	slg 28 длина сообщения	145
12	slg 28 токенов/сек	146
13	slg 28 сообщ./сек	147
14	slg 15 число токенов	148
15	slg 15 число сообщений	149
16	slg 15 длина сообщения	150
17	slg 15 токенов/сек	151
18	slg 15 сообщ./сек	152
19	slg 3 число токенов	153
20	slg 3 число сообщений	154
21	slg 3 длина сообщения	155
22	slg 3 токенов/сек	156
23	slg 3 сообщ./сек	157
24	slg 20 число токенов	160
25	slg 20 число сообщений	161

Продолжение таблицы 13

26	slg 20 длина сообщения	162
27	slg 20 токенов/сек	163
28	slg 20 сообщ./сек	164

3.6 Выводы по главе 3

В главе 3 решены следующие задачи:

1) Разработан метод и алгоритм анализа, трансформации и обработки текстовой информации, получаемой в процессе мониторинга СХД из журналов ПО СХД, позволяющий обнаруживать неисправности без детального анализа структуры данных мониторинга, формата и последовательности текстовых сообщений.
2) Выполнен анализ существующих размеченных данных с применением модели T-SNE и алгоритма иерархической кластеризации, сформирован массив данных для построения обучающих и тестовых выборок, определена сбалансированность массива данных с использованием энтропии Шэннона.
3) Обоснован и описан порядок и состав процедур предварительной обработки текстов журналов, предложены инструменты для разбора текста, например, онтологическое описание комбинаторного парсинга и т.д.
4) Определен и обоснован набор диагностических признаков для решения задачи классификации, включающий в себя, в том числе, векторное представление текстовой части сообщений, количественные и временные характеристики текста. Предложено использование временных коэффициентов для уменьшения влияния более ранних сообщений из временного окна на качество обнаружения неисправностей.
5) Определен набор метрик для сравнения алгоритмов, описаны применяемые для классификации алгоритмы обработки текста для выбора оптимального в рамках экспериментальной проверки их эффективности, определен базовый алгоритм для дальнейшего сравнения.

4 ОБНАРУЖЕНИЕ И ЛОКАЛИЗАЦИЯ НЕИСПРАВНОСТЕЙ В СХД С ИСПОЛЬЗОВАНИЕМ РАЗРАБОТАННЫХ МОДЕЛЕЙ, МЕТОДОВ И АЛГОРИТМОВ

4.1 Построение раздела онтологической модели, предназначенной для предварительной обработки журналов

Онтологическая модель может использоваться не только для описания элементов, используемых в диагностической процедуре, но и для хранения различной диагностической информации, в том числе, правил предварительной обработки входных сигналов, получаемых из различных источников, в том числе из журналов СПО СХД.

Так, в онтологической модели предусмотрен набор классов, объектов и свойств, предназначенный для предварительной обработки текста и заголовков сообщений журналов СПО СХД. Граф соответствующего раздела онтологической модели, включающий в себя именованные сущности, предназначенные для разбора заголовка журнала messages, представлен на рисунке 4.1.

Данный раздел онтологии включает в себя:

1. Корневой класс для каждого журнала;
2. Набор классов и свойств для описания заголовка сообщений журнала;
3. Набор классов и свойств для описания признаков, по которым выполняется фильтрация токенов и сообщений.

С этой целью в текущей онтологии предусмотрен служебный класс формата заголовков сообщений MessageHeaderFormat, объединяющий описание полей заголовка сообщений и методов их извлечения. Исходя из того, что журналы могут иметь разный уровень структурированности, в онтологию для экземпляров SoftwareLog добавляется отношение данных «has_unstructured_text_part» имеющее значение булевого типа и позволяющее определить журналы, для которых не требуется работа с текстовой частью.

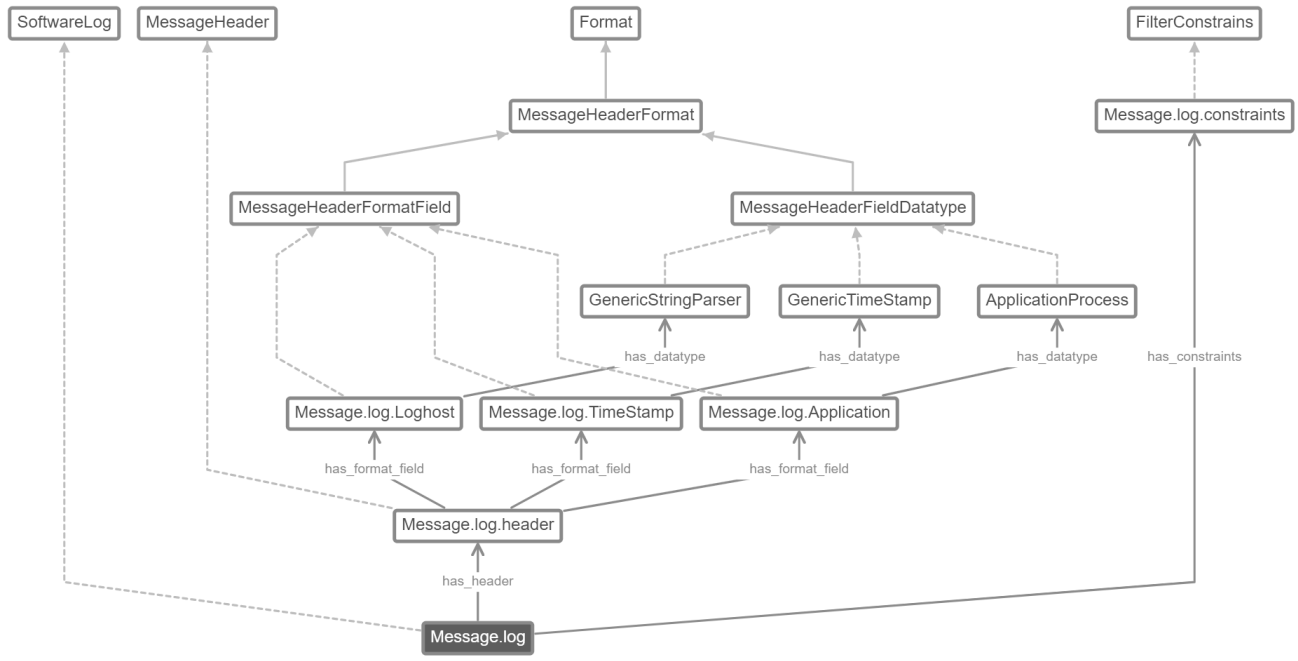


Рисунок 4.1 – Раздел онтологии, предназначенный для обработки журналов

Соответствующие этим понятиям классы и связи, реализованные в текущем разделе онтологии представлены в таблицах 14 – 15.

Таблица 14 – Описание классов онтологии для разбора журналов

Класс	Родительский класс	Описание
FilterConstraints	-	Содержит описание ограничений, применяемых для пропуска токенов и сообщений
Format	-	Корневой класс для всех классов, относящихся к форматированию
MessageHeader Format	Format	Класс, описывающий поля заголовков сообщений журнала SoftwareLog
MessageHeader FieldDataType	MessageHeaderFormat	Класс, содержащий набор низкоуровневых описаний типов данных для форматов полей
MessageHeader FormatField	MessageHeaderFormat	Класс, содержащий набор полей, соответствующих заголовку журнала ПО
MessageTemplate	-	Класс, описывающий форматы текстовой части сообщений журнала ПО
SoftwareLog	-	Класс, соответствующий журналу ПО. Связывает данный раздел онтологии с прочими разделами
MessageHeader	-	Класс, описывающий заголовок сообщения журнала ПО, связан с классом SoftwareLog

Соотношение между классами раздела задается при помощи соотношений, заданных в таблице 15.

Таблица 15 – Соотношение между классами раздела онтологии

Название	**Тип связи**	**Domains**	**Ranges**	**Примечание**
has_constraints	объект, «1-1»	SoftwareLog	FilterConstrains	Задает ограничения для журнала
textpart				
has_datatypes	объект, «1-1»	MessageHeader FormatField	MessageHeader FieldDatatype	Задает низкоуровневое описание типа данных для поля заголовка
has_format_field	объект, «1-N»	MessageHeader	MessageHeader FormatField	Задает набор полей для заголовка сообщения
has_header	объект, «1-1»	SoftwareLog	MessageHeader	Задает заголовок сообщения для журнала
header_delimiter	данные	MessageHeader	xsd:string	Задает разделитель для полей заголовка
header_length	данные	MessageHeader	xsd:int	Задает число полей заголовка
lowlevel_type	данные	MessageHeader FieldDatatype	xsd:string	Эта связь объединяет тип данных онтологии с фактическим типом данных, который будет применяться в языке программирования
min_message_len	данные	FilterContrains	xsd:int	Минимально допустимая длина сообщения в журнале
min_token_freq	данные	FilterContrains	xsd:int	Минимально допустимая частота наблюдения токена в журнале
parser_name	данные	MessageHeader FieldDatatype	xsd:string	Имя парсера, который нужно применить для разбора поля
parsing_order	данные	MessageHeader FormatField	xsd:int	Порядок появления поля в заголовке

Представленный на рисунке 4.1 участок раздела онтологии включает в себя набор именованных сущностей, соответствующих журналу messages.log (см. ссылку). Объект Message.log класса SoftwareLog объединяет все соответствующие данному журналу сущности. Через сущность Message.log.constrains с использованием связей minimum_token_freq (актуальное значение 5) и min_message_len (актуальное значение 3) заданы фильтры минимально допустимой частоты появления токена и минимальной длины сообщения.

Для построения заголовка для данного типа журналов (объект Message.log.header класса MessageHeader) использовалось априорное знание о структуре его сообщений, см. рисунок 4.2.

```
2019-08-05T09:32:11.816527+03:00 linux kernel: Console: colour dummy device 80x25
2019-08-05T09:32:11.816529+03:00 linux kernel: pid_max: default: 32768 minimum: 301
2019-08-05T09:32:11.816531+03:00 linux kernel: Security Framework initialized
2019-08-05T09:32:11.816533+03:00 linux kernel: AppArmor: AppArmor initialized
2019-08-05T09:32:11.816535+03:00 linux kernel: Dentry cache hash table entries: 524288 (order: 6, 4194304 bytes)
```

Рисунок 4.2 – Пример сообщений журнала messages.log

На основании этих данных выделены стабильные поля из заголовка журнала:

1. Поле временной метки (объект Message.log.Timestamp класса MessageHeaderFormatField) – для получения временной метки. Соответствующий ему низкоуровневый тип данных (MessageHeaderFieldDataType) - GenericTimeStamp, так как в программном обеспечении предусмотрен универсальный парсер временных меток, позволяющий обеспечить их разбор практически в любом возможном формате.
2. Поле хоста сообщения, соответствующее полю host формата syslog (объект Message.log.Loghost класса MessageHeaderFormatField). Соответствующий ему низкоуровневый тип данных (MessageHeaderFieldDataType) – GenericStringParser, выдающий значение переданного поля как строку произвольного размера.
3. Поле «автора» сообщения, соответствующее полю Application/Process формата syslog (объект Message.log.ApplicationProcess класса MessageHeaderFormatField). Соответствующий ему низкоуровневый тип

(MessageHeaderFieldDataType) – ApplicationStringParser, отсекающий идентификатор процесса у переданного ему фрагмента строки.

Этим полям присваивается соотношение «parsing_order» со значениями от 0 до 2 в том порядке, в котором планируется их разбор. Поля соотносятся с объектом «Message.log.header» через связь «has_format_field».

4.2 Построение раздела онтологической модели, предназначенного для локализации неисправностей по данным СПО СХД

Связь между параметрами и состоянием, описанная при помощи алгоритмов машинного обучения, в общем случае (например, если она задается при помощи журналов СПО) не обязательно позволяет определить, к каким именованным сущностям относится обнаруженная неисправность.

Раздел онтологической модели, предназначенной для локализации таких неисправностей, содержит экспертные сведения, описывающие типы выделяемых именованных сущностей, соответствующих конкретным журналам и соответствующие им типы меток. Реализация раздела онтологии для журнала warn.log представлен на рисунке 4.3.

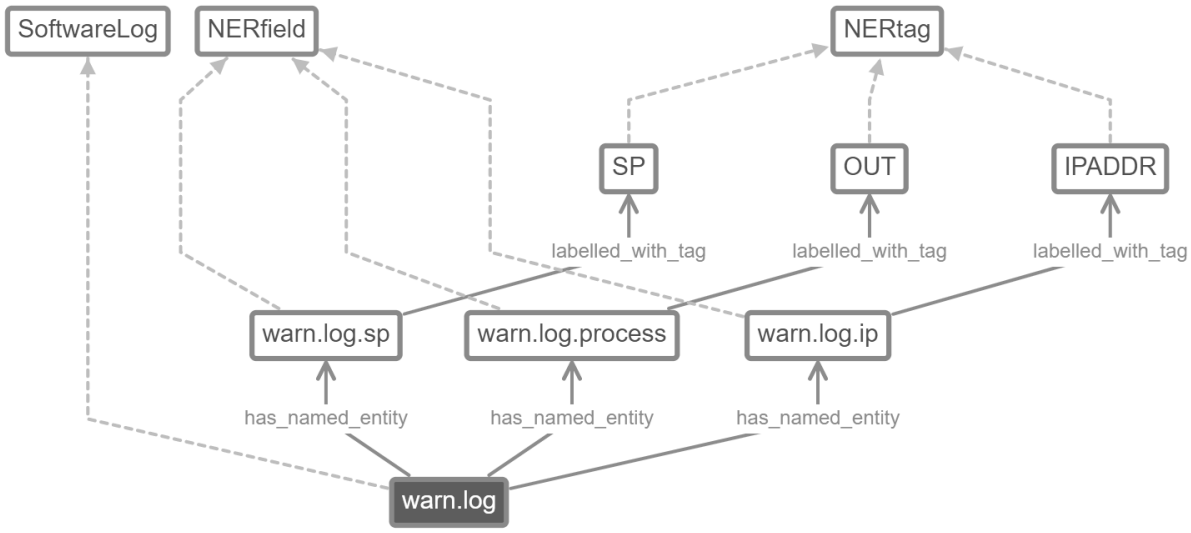


Рисунок 4.3 – Заполнение раздела онтологии для журнала warn.log

Данный раздел включает в себя относительно небольшое число классов (2), так что в основном раздел состоит из их экземпляров и связей между ними. Классы раздела приведены в таблице 16, описание связей - в таблице 17.

Таблица 16 – Описание классов онтологии для NER

Класс	**Родительский класс**	**Описание**
NERfield	-	Объект-локализатор – что именно должно быть выделено из сообщений конкретного журнала
Класс	Родительский класс	Описание
NERtag	-	Метка для аннотирования объекта NER

Таблица 17 – Соотношение между классами раздела онтологии

Название	**Тип связи**	**Domains**	**Ranges**	**Примечание**
has_named_entity	объект, «1-N»	SoftwareLog	NERField	Перечисление NERField, соответствующих конкретному журналу
labelled_with_tag	объект, «N-1»	NERField	NERTag	Метка NER, соответствующая полю NER
reflects_on_topology	данные	NERfield	xsd:int	1 – выделенная метка должна отображаться на топологии СХД, 0 – не должна
corresponds_with	аннотация	NERtag	Component	Строковое имя класса компонента

Для журнала warn.log выделяется несколько сущностей:

1. Сущность корневого класса SoftwareLog – warn.log;

2. Поля NER – warn.log.ip - ip адреса, warn.log.process – процессы, сообщающие об ошибках, warn.log.sp – контроллеры хранения;

3. Объекты тэгов NER – они не уникальны для экземпляров класса NERfield конкретного журнала, а могут подходить разным журналам. Для warn.log нужны три тэга – IPADDR, OUT и SP, соответствующие ip-адресам, прочим тэгам и контроллерам хранения;

4. Экземпляр класса NERtag SP имеет заданную связь “corresponds_with” с классом StorageProcessor.

Поле warn.log.sp имеет флаг reflects_on_topology, подразумевающий, что в случае обнаружения данной именованной сущности в блоке данных, соответствующих неисправности, данный компонент будет отображен в списке затронутых данной неисправностью.

4.3 Порядок применения онтологической модели при решении задачи обнаружения неисправностей

Порядок применения онтологической модели при решении задачи обнаружения неисправностей в процессе функционирования СХД описывается на рисунке 4.4:

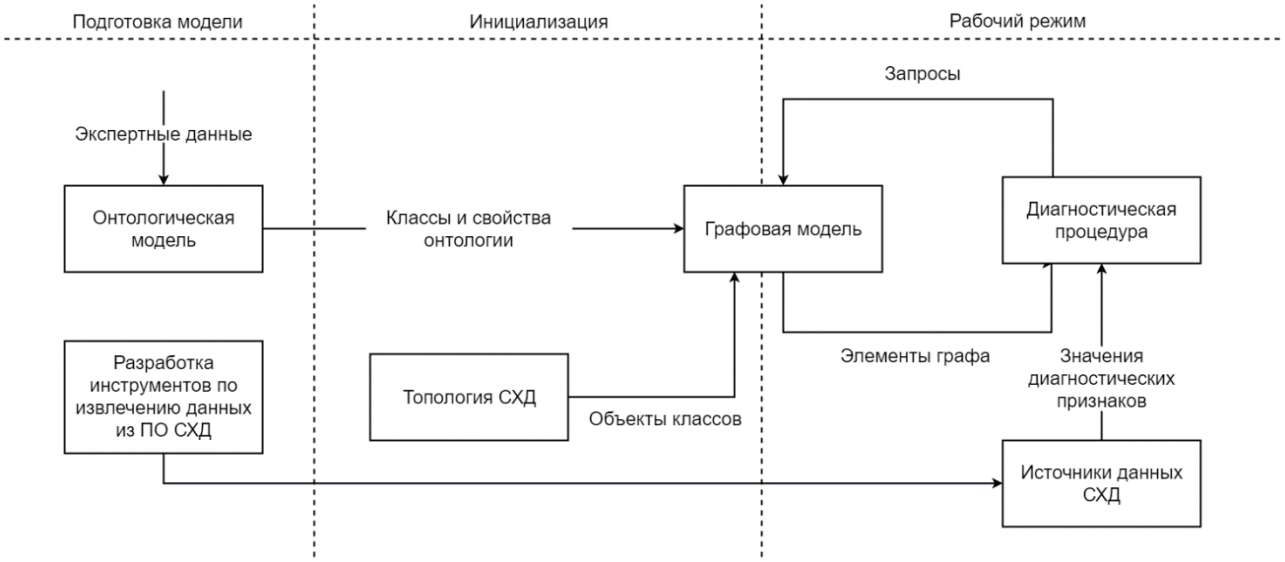


Рисунок 4.4 – Порядок применения онтологической модели

На первом этапе выполняется подготовка онтологической модели путем наполнения ее экспертными данными в сторонних средствах разработки (Stanford Protégé). Далее на этапе инициализации ПО СХД модель преобразуется к графовому виду, причем структура модели, то есть ее классы и свойства (classes, properties), получается путем преобразования онтологической модели, а объекты классов (named individuals) – путем обращения к системному программному обеспечению СХД. Сформированная таким образом модель системы помещается в

графовую базу данных и используется в рамках диагностической процедуры для получения диагностических сведений о взаимосвязи значений диагностических признаков со сбоями в СХД.

Главным преимуществом использования такого процесса разработки модели является то, что он позволяет обеспечить достаточно высокую универсальность программной реализации процедуры диагностики – для адаптации решения к другим типам диагностируемых СХД достаточно разработать инструменты для получения информации о топологии СХД и настроить взаимодействие с системой мониторинга СХД.

Исходя из вышесказанного, диагностическая процедура включает в себя следующие основные шаги:

1. Сбор данных мониторинга;

2. Для каждого компонента выполняется определение его детерминированного состояния путем подстановки значений параметров в правила определения симптомов (как на уровне структурных связей, так и определяемых через связь «solves_with») (26):

$$Sc_i = f([P_i]), i = [0, N], \quad (26)$$

где $P_i -$ вектор параметров, соответствующий компоненту C_i,

$N -$ число компонентов;

3. Для каждого компонента выполняется определение состояния связанных подсистем (27)

$$Ss_j = f([Sc_i]), j = [0, M], \quad (27)$$

где $Sc_j -$ вектор состояний компонентов,

соответствующих подсистеме Ss_j,

M – число подсистем;

4. Для СХД в целом определяется состояние, как функция от состояний его подсистем (28):

$$S = f([Ss_j]), \quad (28)$$

Здесь Ss_k – вектор подсистем, состояния которых влияют на состояние СХД в целом.

Шаги 1-4 повторяются в соответствии с заданным интервалом запуска процедуры обнаружения неисправностей. Данный интервал не может быть меньше интервала опроса подсистемы мониторинга.

4.4 Описание ПО, реализующего эффективности применения онтологической модели при решении задачи обнаружения неисправностей

Упрощенное графовое представление модели и соответствующий алгоритм диагностики входят в состав модуля диагностирования, применяемого в составе комплекса предотвращения неисправностей СПО СХД в соответствии со схемой, представленной на рисунке 4.5. Модуль диагностирования получает текущие значения параметров СХД, в том числе тексты временных окон из журналов СПО СХД от модуля выдачи данных, который получает их из базы данных, формируемой модулем мониторинга и сбора данных.

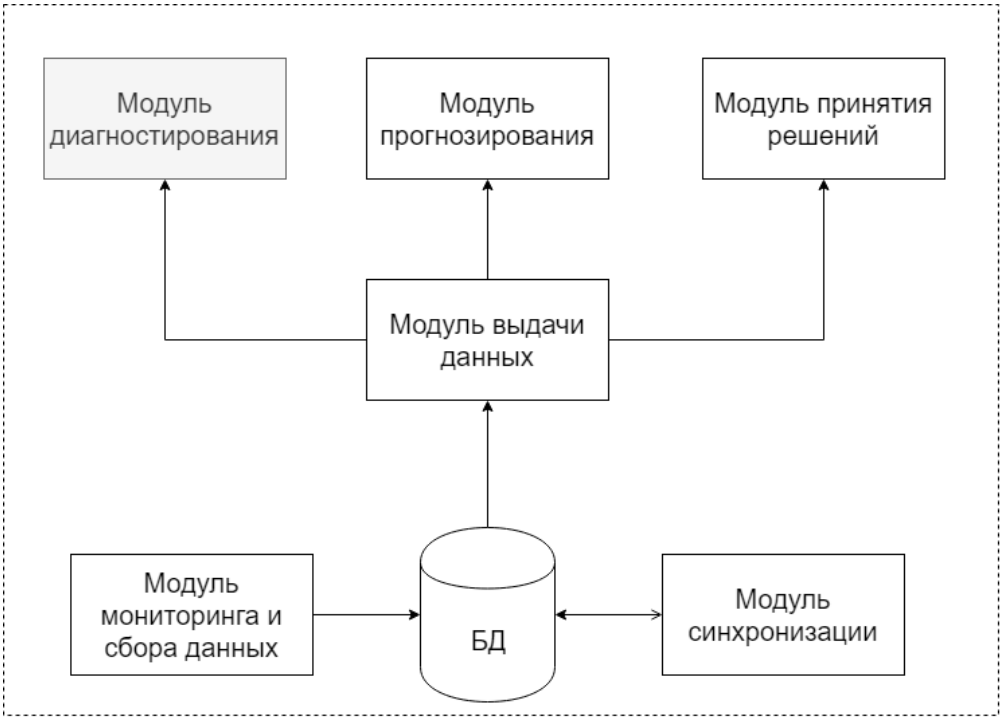


Рисунок 4.5 Архитектура программного комплекса предотвращения неисправностей

Экземпляр такого диагностического ПО запускается менеджером кластера на одном из контроллеров хранения, входящих в кластер, и определяет состояние системы в целом, ее отдельных подсистем и компонентов.

Схема работы модуля диагностики представлена на рисунке 4.6. Классы и свойства онтологической модели передаются в модуль из внешнего редактора онтологий перед формированием установочного пакета ПО. Именованные сущности, соответствующие данным классам и связанные данными свойствами, формируются автоматически, на основании данных путем опроса источников данных в СХД. Полученная модель преобразуется в графовый вид и применяется в рамках модуля определения состояний для классификации данных мониторинга.

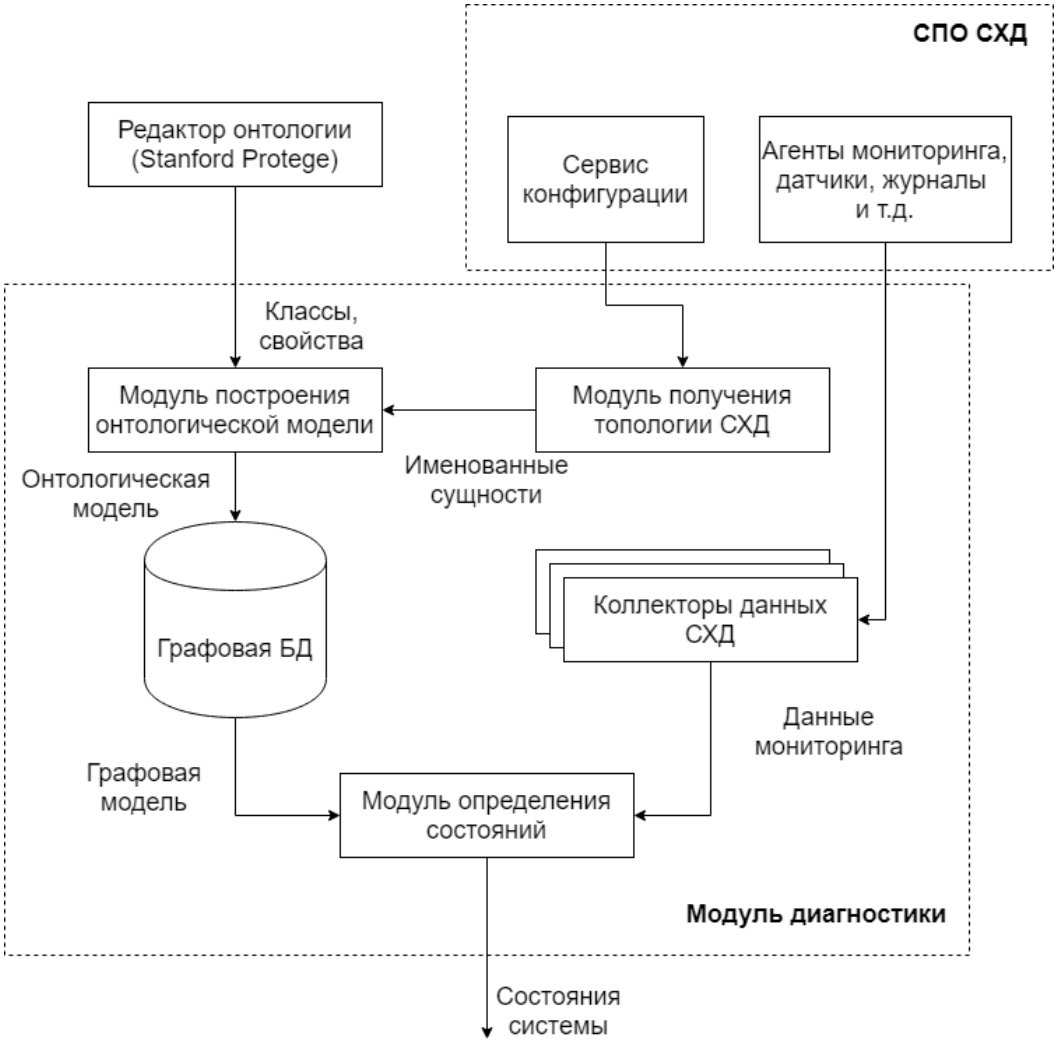


Рисунок 4.6 – Схема работы модуля диагностики

Алгоритм работы модуля диагностики состоит из следующих, циклически повторяющихся шагов:

1. Запуск диагностической процедуры.

Диагностическая процедура запускается по расписанию с определенным временным шагом (данное ограничение обосновано необходимостью экономить вычислительные ресурсы контроллера СХД);

2. Перестроение входных последовательностей для алгоритма, для параметров, основанных на использовании журналов СПО СХД:

2.1. Определяются все журналы, изменившие свой размер с момента предыдущего запуска алгоритма;

2.2. Для каждого изменившегося журнала формируется временное окно фиксированного размера, соответствующего временному шагу запуска алгоритма;

2.3. Для каждого неизменного журнала временное окно остается прежним (на практике, такая оптимизация часто оказывается излишней, так как для экономии ресурсов системы целесообразно использование временного окна достаточного большого размера);

2.4. Для каждого временного окна выполняется процедура предварительной обработки текста;

2.5. Для каждого предварительно обработанного временного окна выполняется двухступенчатая процедура для каждого вектора [X*i*], где $Xi = \{V_{ti,} X_{pi}\}$, объединение векторного представления текста временного окна V_{ti} и дополнительных признаков X_{pi} для *i-ого* журнала для текущего временного окна;

3. Обнаружение неисправностей выполняется в два основных этапа:

3.1. Применение классификатора для классификации векторного представления окна как одной из известных неисправностей;

3.2. В случае, если удалось классифицировать временное окно как неисправность, производится локализация возникшей неисправности относительно топологии СХД путем анализа извлеченных метаданных с использованием онтологической модели СХД;

4. Сохранение вектора последних сообщений для каждого журнала для использования в шаге 2.

4.5 Выводы по главе 4

В главе 4 решены следующие задачи:

1) Разработан комплексный метод обнаружения неисправностей в СХД, основанный на совместном использовании диагностической модели СХД и метода обработки текстовых данных мониторинга СХД с использованием алгоритмов машинного обучения.
2) Разработаны разделы онтологической модели, содержащие правила обработки текстовых частей сообщений журналов СПО СХД и алгоритмов выделения именованных сущностей, приведены примеры наполнения таких разделов для отдельных журналов СХД.
3) Определен порядок использования онтологической и графовой моделей от момента формирования модели до интеграции и автоматического применения в составе диагностического модуля.
4) Рассмотрена архитектура разработанного программного обеспечения для выполнения процедуры с указанием положения диагностического модуля в общей архитектуре ПО предотвращения неисправностей.
5) Описан алгоритм работы модуля диагностики для случая извлечения исходных данных из журналов ПО СХД, включающий в себя этапы обнаружения изменения входных данных, построение входных последовательностей, обнаружение и локализацию неисправностей.

5 ПРОВЕДЕНИЕ ЭКСПЕРИМЕНТАЛЬНЫХ ИССЛЕДОВАНИЙ С ЦЕЛЬЮ ОЦЕНКИ ЭФФЕКТИВНОСТИ РАЗРАБОТАННЫХ МЕТОДОВ И СРЕДСТВ

5.1 Эффективность обнаружения неисправностей в зависимости от выбранного типа векторного представления данных и типа классификатора

Для проведения исследовательских испытаний выбрано четыре способа построения векторного представления данных:

- Матрица токенов;
- Матрица токенов с весовыми коэффициентами;
- Усредненный FastText с весовыми коэффициентами;
- Усредненный Word2Vec с весовыми коэффициентами.

Такая подборка вариантов векторного представления позволяет охватить как наиболее простые способы, так и более сложные, учитывающие контекст слова. Как уже было сказано ранее, более комплексные модели в рамках настоящего исследования не рассматривались, так как при пробных запусках демонстрировали сочетание слабой производительности с неудовлетворительными результатами.

Классификация векторизованных данных выполнялась с использованием набора классификаторов, описанных в предыдущем разделе. Проведено сравнение эффективности классификации при помощи алгоритмов Random Forest, XGBoost, рекуррентных нейронных сетей: на основании LSTM, GRU и LSTM с механизмом внимания.

Для первичной оценки при сравнении эффективности классификации применялась комплексная метрика «Доля правильных результатов» (Accuracy), для более комплексной оценки наиболее эффективного алгоритма определялись взвешенные метрики «Точность», «Полнота» и «F-мера».

Для проверки эффективности различных способов классификации применялись следующие шаги:

1. Определение оптимального состава вектора признаков

1.1. Применение алгоритма классификации к входному вектору, состоящему из векторного представления текста без дополнительных параметров;

1.2. Применение алгоритма классификации к входному вектору, состоящему только из дополнительных параметров;

1.3. Применение алгоритма классификации к входному вектору, состоящему из векторного представления текста совместно с дополнительными параметрами;

1.4. Применение алгоритма классификации к входному вектору, состоящему из векторного представления текста без разбиения по отдельным журналам совместно с дополнительными параметрами;

1.5. Применение алгоритма классификации к входному вектору, состоящему из векторного представления текста и разбиением по отдельным журналам совместно с дополнительными параметрами при построении векторного представления, на основе усредненного word2vec;

1.6. Применение алгоритма классификации к входному вектору, состоящему из векторного представления текста и разбиением по отдельным журналам совместно с дополнительными параметрами при построении векторного представления, на основе усредненного fasttext;

2. Оптимизация классификатора

2.1. Определение оптимального классификатора;

2.2. Оптимизация параметров классификатора с использованием метода случайного поиска;

2.3. Уточнение полученных параметров с использованием полного перебора.

5.2 Определение оптимального состава вектора признаков

Для совместного использования числовых и текстовых признаков применялся механизм объединения признаков (feature union), в работе которого

каждый признак обрабатывается соответствующей функцией, а результаты обработки объединяются в общий вектор. Для проведения испытаний выполнялось пятикратное разбиение массива исходных данных на тестовую и обучающую выборки с итоговым усреднением оценок качества классификации.

В результате проведения испытаний по п.1.1 получены следующие результаты, см. таблица 18.

Таблица 18 – Входной вектор без дополнительных параметров

Классификатор	**Точность**	**Полнота**	**F-мера**	**$t_{обуч, с}$**
Random Forest	0,73	0,73	0,69	206 с
Наивный Байес	0,43	0,44	0,37	161 с
KNN	0,66	0,64	0,63	221 с
Логистическая регрессия	0,65	0,65	0,63	590 с
Метод опорных векторов	0,07	0,27	0,12	319 с
XGBoost	0,49	0,50	0,48	188 с

В результате проведения испытаний по п.1.2 получены следующие результаты, см. таблица 19.

Таблица 19 – Входной вектор только с дополнительными параметрами

Классификатор	**Точность**	**Полнота**	**F-мера**	**$t_{обуч, с}$**
Random Forest	0,68	0,62	0,63	0,180 с
Наивный Байес	0,45	0,24	0,25	0,102 с
KNN	0,40	0,41	0,39	0,119 с
Логистическая регрессия	0,39	0,4	0,36	2,4 с
Метод опорных векторов	0,11	0,28	0,13	1,5 с
XGBoost	0,58	0,56	0,54	12 с

В результате проведения испытаний по п.1.3 получены следующие результаты, см. таблица 20.

Таблица 20 – Входной вектор с дополнительными и текстовыми парметрами

Классификатор	**Точность**	**Полнота**	**F-мера**	**$t_{обуч, с}$**
Random Forest	0,76	0.76	0.74	212 с
Наивный Байес	0,45	0,27	0,22	205 с
KNN	0,40	0,41	0,39	166 с
Логистическая регрессия	0,38	0,40	036	550 с
Метод опорных векторов	0,11	0,28	0,13	430 с
XGBoost	0,75	0,75	0,73	Более 3600 с

По итогам испытаний по пунктам 1.1-1.3 можно сделать вывод о том, что эффективность использования в качестве входных данных векторного представления, дополнительных параметров или их комбинации зависит от используемого алгоритма классификации. Наилучшие результаты получены с алгоритмами Random Forest и XGBoost при комбинированном наборе входных данных. При этом скорость обучения Random Forest более чем на порядок превышает скорость обучения XGBoost, поэтому именно данный классификатор будет использоваться в последующих тестах.

Испытания по п.п.1.4. предназначены для определения эффективности разбиения текстов отдельных журналов по отдельным признакам. Изначально можно предполагать, что такое разбиение позволит уменьшить влияние на параметрическую оценку классов тех журналов, которые в меньшей степени относятся к данному классу.

В результате проведения испытаний по п.2.4 получены следующие результаты, см. таблица 21.

Таблица 21 – Объединение журналов

Классификатор	**Точность**	**Полнота**	**F-мера**	**$t_{обуч, с}$**
Random Forest	0,72	0,70	0,69	218 с
Наивный Байес	0,45	0,24	0,25	227 с
KNN	0,40	0,41	0,39	239 с

Продолжение таблицы 22

Логистическая регрессия	0,39	0,40	0,36	509 с
Метод опорных векторов	0,11	0,28	0,13	935 с
XGBoost	0,60	0,58	0,57	262 с

Из полученных результатов видно, что, как и предполагалось, эффективность классификации при использовании разбиения журналов по отдельным признакам выше, чем в случае, когда текст всех журналов объединяется в один признак.

Испытания по п.п 1.1-1.4 проводились с использованием векторного представления, построенного по методу “матрица токенов с весовыми коэффициентами”, п.п. 1.5 и 1.6 предназначены для сравнения эффективности использования различных способов построения векторного представления, отсечения заголовка и временных коэффициентов.

В результате проведения испытаний по п.1.5, 1.6 получены следующие результаты, см. таблица 23

Таблица 23 – Усредненный word2vec

№п.п	Точность	Полнота	F-мера	$t_{обуч, с}$
Token matrix	0,66	0,68	0,66	73 с
Token matrix w.coeff.	0,73	0.73	0.71	212 с
Fasttext	0,71	0,67	0,65	310 с
Word2Vec	0,68	0,66	0,64	290 с

Использование максимально сложной схемы построения вектора входных параметров (с разделением по журналам, совместное использование векторного представления и дополнительных параметров, с отсечением заголовков и применением весовых коэффициентов) является оптимальным с точки зрения точности классификации, но при этом наиболее требовательно к системным ресурсам.

Выбор метода построения векторного пространства не оказывает существенного влияния на точность классификации, вероятнее всего потому, что

семантическая структура сообщений журналов достаточно примитивна и контекст не оказывает влияние на эффективность классификации.

Исходя из полученных результатов, можно сделать вывод о том, что наиболее эффективным способом построения входного вектора признаков является построение отдельных векторных представлений для каждого журнала с применением способа векторизации на основе матрицы токенов и весовых коэффициентов, с использованием дополнительных параметров.

5.3 Определение оптимального классификатора

Оптимальный классификатор определяется путем запуска описанных в главе 3 методов обработки естественного текста с использованием оптимального набора диагностических признаков. Результаты данного эксперимента, усредненные для десятикратного случайного разбиения исходных данных на тестовую и обучающую выборки, приведены в таблице 24.

Таблица 24 – Сравнительная оценка алгоритмов классификации

Классификатор	Средняя точность	Средняя полнота	Средняя f-мера	Среднее время обучения модели
Random Forest	0,76	0.76	0.74	212 с
XGBoost	0,75	0,75	0,73	Более 3600 с
LSTM	0,47	0,44	0,45	~4 часа
GRU	0,24	0,22	0,23	~7 часов
LSTM с механизмом внимания	0, 42	0,39	0,40	~ 6 часов

Исходя из представленных в таблице результатов, можно сделать вывод о том, что наилучшим алгоритмом классификации на текущем наборе данных является алгоритм Random Forest. Из рисунка 5.1, на котором приведены графики обучения нейронных сетей, можно сделать вывод о том, что нейронные сети GRU с задачей классификации не справились, а нейронные сети LSTM демонстрируют признаки недообученности, что может быть устранено за счет увеличения объема размеченных исходных данных.

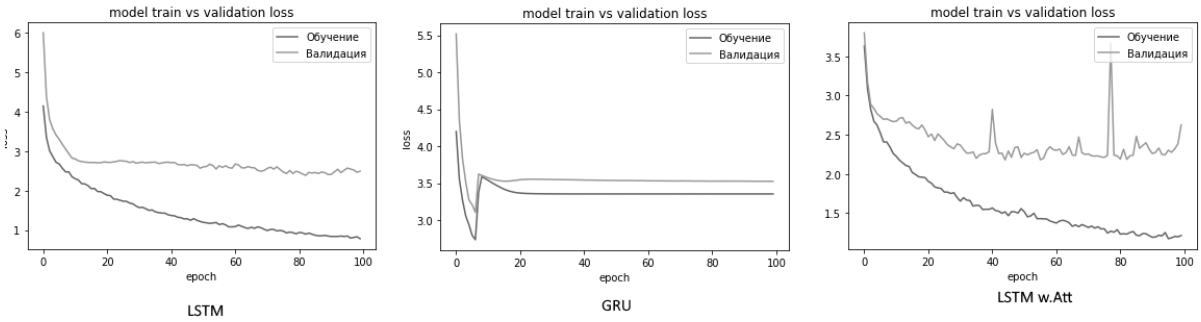


Рисунок 5.1 – Графики обучения нейронных сетей

В таблице 25 представлена зависимость качества классификации от объема обучающей выборки.

Таблица 25 – Зависимость качества классификации от объема выборки

N, % от набора	Средняя точность	Средняя полнота	Средняя F-мера
50	0,68	0,66	0,64
60	0,73	0,73	0,71
70	0,76	0.76	0.74
80	0,76	0.76	0.74

5.4 Оценка влияния оптимизации гиперпараметров классификатора на эффективность обнаружения неисправностей

Оптимизация гиперпараметров классификатора позволяет добиться более высокого качества классификации за счет изменения настроек, регулирующих его функционирование. Для выбранного классификатора «случайный лес» рекомендуется оптимизировать следующие параметры [153]:

- **N_estimators** – число деревьев леса. Чем выше значение, тем качественнее классификация и тем дольше время обучения;
- **max_depth** – максимальная глубина дерева. Повышение значения приводит к более качественному описанию особенностей модели данных, при этом слишком большое значение может приводить к переобучению;
- **min_samples_split** – минимальное число объектов для выполнения расщепления. Слишком большое число может приводить к переобученности. При выборе параметра рекомендуется руководствоваться объемом выборки;

- **min_samples_leaf** – минимальное число объектов в листе дерева. Слишком большое число может приводить к переобученности;

- **max_features** – максимальное число признаков для расщепления. С одной стороны, повышение значения может приводить к увеличению точности классификации, с другой – деревья при этом становятся более однообразными, что приводит к переобучению.

Итоговая матрица для подбора параметров представлена в таблице 26.

Таблица 26 – Матрица для подбора параметров

Параметр	Диапазон	Шаг	Дополнительно
N_estimators	200, 2000	10	Шаг 10
max_features	Auto, Sqrt	Sqrt	
max_depth	10, 110	10	значение None
min_samples_split	2, 5, 10		
min_samples_leaf	1, 2, 4		
bootstrap	Истина/Ложь		

Для определения оптимальных значений параметров классификаторов выполнялся двухступенчатый поиск с использованием двух алгоритмов подбора – на первом этапе выполняется случайный перебор комбинаций параметров в заданных диапазонах с кросс-валидацией, на основании чего определяется оптимальная комбинация параметров; на втором – полный перебор значений в окрестности оптимальных параметров, определенных на предыдущем шаге.

Результирующее вектор параметров имеет следующий вид:

- n_estimators=600;
- min_samples_split=2;
- min_samples_leaf=2;
- max_features='auto';
- max_depth=60;
- bootstrap=False;
- class_weight="balanced".

В результате проведения испытаний определена оптимальная матрица параметров для выбранного классификатора на базе алгоритма Random Forest.

Стоит отметить, что данный классификатор является не только самым точным, но также и одним из самых эффективных с точки зрения использования вычислительных ресурсов. Увеличение точности классификации относительно значений параметров из начала диапазона составило 13,148%.

5.5 Экспериментальная оценка эффективности применения разработанных методов и средств для обнаружения неисправностей в СХД

Оценка эффективности всего подхода выполнялась как на реальном объекте исследования, так и с использованием средств имитации неисправностей, поскольку частота возникновения неисправностей в процессе штатного функционирования СХД слишком мала, чтобы иметь возможность наблюдать достаточное количество неисправностей за время проведения эксперимента.

При испытании на реальном объекте, были проанализированы результаты запусков диагностической процедуры с периодичностью в 1 минуту на протяжении 200 часов реального времени, при этом на данном интервале не было зафиксировано реальных неисправностей. По результатам эксперимента получены следующие показатели:

- для неисправностей, определяемых в результате применения правил обнаружения, заданных детерминированными онтологическими связями, не зафиксировано ложных срабатываний;
- для неисправностей, определяемых в результате применения правил, заданных условными онтологическими связями, отсутствие неисправностей зафиксировано с вероятностью 0,98.

Для проведения эксперимента с имитацией неисправностей применена схема с подключением к модулю мониторинга и сбора данных программного комплекса предотвращения неисправностей (см. рисунок 4.5) таким образом, чтобы на вход данного модуля подавались нарезанные с интервалом в 1 минуту журналы, соответствующие имеющимся размеченным данным, см. схему на рисунке 5.2.

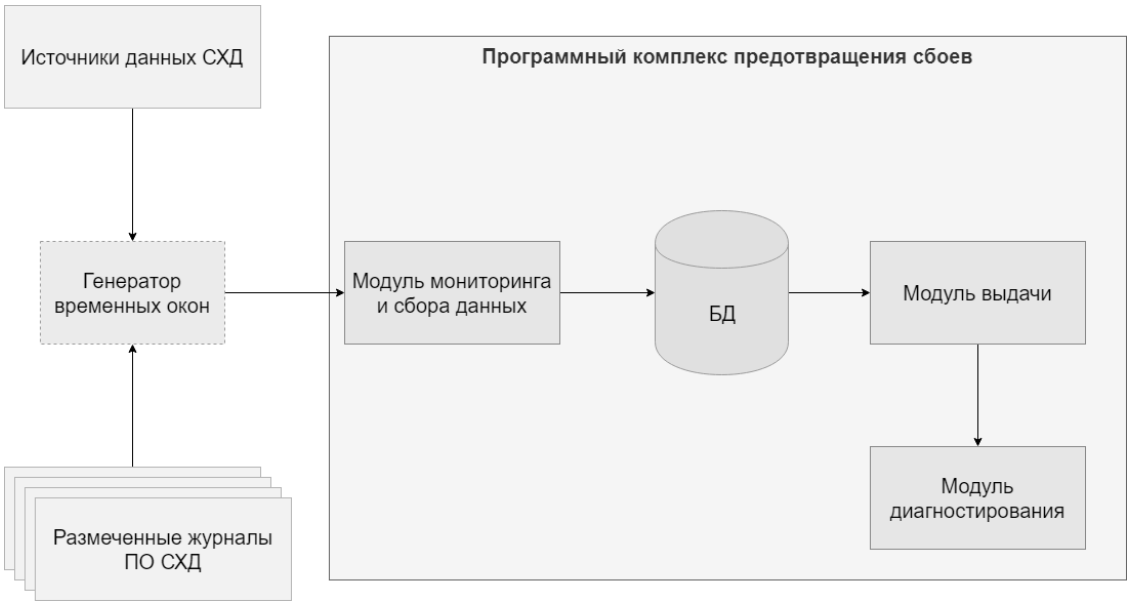


Рисунок 5.2 – Включение имитатора в схему сбора данных

По результатам эксперимента получены следующие результаты:

- для неисправностей, определяемых в результате применения правил обнаружения, заданных детерминированными онтологическими связями, доля идентифицированных составляет 0.99 на массиве данных в 10000 неисправностей;
- для неисправностей, определяемых в результате применения правил, заданных условными онтологическими связями, доля идентифицированных составляет 0.73 на массиве данных в 10000 неисправностей.

Прямое сравнение эффективности алгоритма обнаружения неисправностей на основании классификации текстов журналов ПО затруднительно, так как отсутствуют прямые аналоги, реализующие ту же функциональность. Так, схожий по базовому принципу метод, предложенный в [154], не способен решать аналогичные задачи, так как ориентирован на конкретный формат журнала. Практическое сравнение с методами, основанными на автоматизированном извлечении шаблонов сообщений из журналов ПО может быть выполнено только косвенно, на основании имеющихся данных об их средней точности. Необходимо учитывать, что помимо собственно извлечения шаблонов для обнаружения неисправностей требуется также обеспечивать их классификацию в соответствии с типами возникающих неисправностей. В [127] приводятся усредненные данные по

13 различным алгоритмам автоматического извлечения шаблонов сообщений, со средней точностью обнаружения шаблона в 0.68 (максимальная точность 0.867), при этом только два алгоритма имеют среднюю точность выше 0.75. Следовательно, можно утверждать, что предложенный в настоящей работе алгоритм, дающий сравнимую точность уже непосредственно обнаружения неисправностей является более предпочтительным.

5.6 Выводы по главе 5

В главе 5 решены следующие задачи:

1) Экспериментально определен оптимальный состав и структура параметров, описывающих журналы ПО СХД и их влияние на процедуру обнаружения неисправностей. Проведены испытания для проверки влияния на качество классификации:
   - наличия или отсутствия векторного представления текста;
   - наличия или отсутствия дополнительных признаков;
   - построения диагностических признаков по всем журналам сразу;
   - построения диагностических признаков по каждому журналу отдельно;
   - применения весовых коэффициентов и методов отсечения заголовка сообщений;
   - способа построения векторного представления.

   По результатам проведения эксперимента можно сделать вывод о том, что наибольшую эффективность показывает самая сложная модель, при этом влияние типа построения векторного представления на качество классификации минимально.
2) Определен оптимальный классификатор, путем сравнения набора алгоритмов классификации, основанных на использовании различных алгоритмов машинного обучения, в том числе нейронных сетей глубокого обучения. По результатам экспериментальной проверки можно сделать вывод о том, что большинство алгоритмов не смогли удовлетворительно

решить поставленную задачу. В числе показавших наилучшее качество классификации - алгоритмы Random Forest, XGBoost и нейронные сети на базе LSTM. Наилучший результат у Random Forest.

3) Выполнена экспериментальная проверка влияния оптимизации гиперпараметров алгоритма классификации на процедуру обнаружения неисправностей. Оптимизация гиперпараметров алгоритма Random Forest позволила увеличить качество классификации на 13,148%.
4) Проведена экспериментальная проверка эффективности разработанных методов на реальном объекте, и, в связи с невозможностью получить достаточное количество неисправностей, на имитаторе, которая показала, что качество диагностирования в составе программного комплекса незначительно отличается от результатов, полученных при моделировании.

ЗАКЛЮЧЕНИЕ

В диссертационной работе решена актуальная научно-техническая задача по повышению эффективности обнаружения неисправностей в СХД за счет применения методов диагностирования, основанных на совместном использовании диагностической модели и алгоритмов, основанных на методах машинного обучения для анализа диагностических параметров.

Получены следующие результаты:

1) Выполнен анализ систем класса СХД как объекта диагностики, определены требования к методам обнаружения неисправностей в СХД и реализующему их ПО. Выполнен анализ существующих научных работ и прикладных инструментов в области диагностики вычислительных систем, сравнительный анализ их характеристик.

2) Разработан метод построения диагностической модели для систем класса СХД, определяющей отношения между параметрами и возможными состояниями СХД и ее элементов, где отношения могут быть заданы и как детерминированные связи между диагностическими сущностями, и как функции машинного обучения.

3) Разработаны методы и программные средства преобразования диагностической модели к упрощенному графовому виду для её применения в составе диагностического ПО.

4) Разработаны методы и инструментальные средства обнаружения неисправностей в СХД, основанные на анализе текстовой информации, получаемой в процессе мониторинга ПО методами классификации текста с использованием машинного обучения.

5) Выполнена экспериментальная проверка разработанных моделей, методов и средств на целевой платформе СХД.

В качестве рекомендаций и перспектив дальнейшей разработки темы можно указать возможность исследования моделей векторного представления текстов журналов ПО СХД, основанных на применении нейронных сетей, и определение

возможности совместного использования разных алгоритмов для решения задачи классификации.

Полученные результаты соответствуют пунктам 4 «Разработка методов и алгоритмов решения задач системного анализа, оптимизации, управления, принятия решений и обработки информации.», 5 «5. Разработка специального математического и алгоритмического обеспечения систем анализа, оптимизации, управления, принятия решений и обработки информации», 11 «Методы и алгоритмы прогнозирования и оценки эффективности, качества и надежности сложных систем.», 12 «Методы получения, анализа и обработки экспертной информации» паспорта специальности 05.13.01 Системный анализ, управление и обработка информации (по отраслям).

6 СПИСОК ЛИТЕРАТУРЫ

1. The Digitization of the World From Edge to Core [Электронный ресурс]. - Режим доступа: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf.

2. Cheng W. Ranking causal anomalies via temporal and dynamical analysis on vanishing correlations / W. Cheng, K. Zhang, H. Chen, G. Jiang, Z. Chen, W. Wang // KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining – 2016. – P. 805–814.

3. Queiroz L.P. Fault Detection in Hard Disk Drives Based on a Semi Parametric Model and Statistical Estimators / L.P. Queiroz, J.P.P Gomes., F.C.M. Rodrigues, F.T. Brito, I.C. Chaves, L.G.M. Leite, J.C. Machado // New Generation Computing – 2018. – Vol. 36 – № 1 – P. 5–19.

4. Ding J. An anomaly detection approach for multiple monitoring data series based on latent correlation probabilistic model / J. Ding, Y. Liu, L. Zhang, J. Wang, Y. Liu // Applied Intelligence – 2016. – Vol. 44 – № 2 – P. 340–361.

5. Wang G. An Anomaly Detection Framework for Detecting Anomalous Virtual Machines under Cloud Computing Environment / G. Wang, J. Wang // International Journal of Security and Its Applications – 2016. – Vol. 10 – № 1 – P. 75–86.

6. Dasgupta D. An immunity-based technique to characterize intrusions in computer networks / D. Dasgupta, F. González // IEEE Transactions on Evolutionary Computation – 2002. – Vol. 6 – № 3 – C. 281–291.

7. Li D. Negative selection algorithm with constant detectors for anomaly detection / D. Li, S. Liu, H. Zhang // Applied Soft Computing Journal – 2015. – Vol. 36 – P. 618–632.

8. González F. An evolutionary approach to generate fuzzy anomaly (attack) / F. Gonzalez, J. Gomez, K. Kaniganti, D. Dasgupta // IEEE Systems, Man and Cybernetics Society Information Assurance Workshop – 2003. – P. 251–259.

9. Eiras-Franco C. Large scale anomaly detection in mixed numerical and categorical

input spaces / C. Eiras-Franco, D. Martínez-Rego, B. Guijarro-Berdiñas, A. Alonso-Betanzos, A. Bahamonde // Information Sciences – 2019. – Vol. 487 – P. 115–127.

10. Slimani A. Fusion of Model-based and Data-based Fault Diagnosis Approaches / A. Slimani, P. Ribot, E. Chanthery, N. Rachedi // IFAC – 2018. – Vol. 51 – № 24 – P. 1205–1211.

11. Luo J. Integrated model-based and data-driven diagnosis of automotive antilock braking systems / J. Luo, M. Namburu, K.R. Pattipati, L. Qiao, S. Chigusa // IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans – 2010. – Vol. 40 – № 2 – P. 321–336.

12. Jung D. A Combined Data-Driven and Model-Based Residual Selection Algorithm for Fault Detection and Isolation / D. Jung, C. Sundstrom // IEEE Transactions on Control Systems Technology – 2019. – Vol. 27 – № 2 – P. 616–630.

13. ГОСТ 20911-89 ТЕХНИЧЕСКАЯ ДИАГНОСТИКА. Термины и определения. МЕЖГОСУДАРСТВЕННЫЙ СТАНДАРТ. - М.: Стандартинформ, 2009. - 22 с.

14. Narayanan I. SSD Failures in Datacenters: What? When? and Why? / I. Narayanan, D. Wang, M. Jeon, B. Sharma, L. Caulfield, A. Sivasubramaniam, B. Cutler, J. Liu, B. Khessib, K. Vaid // Proceedings of the 9th ACM International on Systems and Storage Conference – 2016. – Vol. 1 – №1 – P. 1-11.

15. Mahdisoltani F. Improving storage system reliability with proactive error prediction / F. Mahdisoltani, I. Stefanovici, B. Schroeder // Proceedings of the 2017 USENIX Conference on Usenix Annual Technical Conference – 2017. – Vol. 1 – №.1 – P. 391-402.

16. Vaarandi R. A data clustering algorithm for mining patterns from event logs // Proceedings of the 3rd IEEE Workshop on IP Operations & Management – 2003. – C.119–126.

17. Qianng F. Where Do Developers Log? An Empirical Study on Logging Practices in Industry / F. Qiang, J. Zhu, W. Hu, J. Lou, R. Ding, Q. Lin, D. Zhang, T. Xie // Proc. of

International Conference on Software Engineering (ICSE) – 2014. – Vol. 2014 – № 1 – P. 24-33.

18. Ming Jiang Z. An automated approach for abstracting execution logs to execution events / Z. Ming Jiang, A. Hassan, G. Hamann, P. Flora // Journal of software maintenance and evolution: research and practice – 2008. – Vol. 20 – № 4 – P. 249-267.

19. Bertero C. Experience Report: Log Mining Using Natural Language Processing and Application to Anomaly Detection / C. Bertero, M. Roy, C. Sauvanaud and G. Tredan // 2017 IEEE 28th International Symposium on Software Reliability Engineering – 2017. – 351–360c.

20. Messaoudi S. A search-based approach for accurate identification of log message formats / S. Messaoudi, A. Panichella, D. Bianculli, L. Briand, R. Sasnauskas // ICPC '18: Proceedings of the 26th Conference on Program Comprehension – 2018. – Vol. 1 – №1 – P. 167 – 177.

21. Troppens U.Storage Networks Explained: Basics and Application of Fibre Channel SAN, NAS, iSCSI,InfiniBand and FCoE / U. Troppens, R. Erkens, W. Mueller-Friedt, R. Wolafka, N. Haustein –Willey, 2009 – 583 p.

22. TATLIN Storage [Электронный ресурс]. – Режим доступа: https://yadro.com/tatlin.

23. Егоров С.И. Коррекция ошибок в информационных каналах периферийных устройств ЭВМ: монография / С. И. Егоров – КГТУ, 2008. – 251с.

24. VESNIN Server [Электронный ресурс]. – Режим доступа: https://yadro.com/vesnin.

25. Overview: Volumes in Storage Pools [Электронный ресурс]. – Режим доступа: https://publib.boulder.ibm.com/tividd/td/TSMCW/GC32-0782-00/en_US/HTML/anrwgd92.htm.

26. GOLDSZMIDT, M. Finding soon-to-fail disks in a haystack. // Proceedings of the 4th USENIX Conference on Hot Topics in Storage and File Systems – 2012 – Vol. 1 – №1 –

P. 8.

27. Agarwal V. Discovering rules from disk events for predicting hard drive failures. / V. Agarwal, C. Bhattacharyya, T. Niranjan, S. Susarla // IEEE International Conference on Machine Learning and Applications'09. – 2009. – Vol. 1 – №1 – P. 782–786.

28. Botezatu M. Predicting Disk Replacement towards Reliable Data Centers / M. Botezatu, I. Giurgiu, J. Bogojeska, D. Wiesmann // Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining – 2016. – Vol. 1 – №1 – P. 39–48.

29. Xinpu J. A Proactive Fault Tolerance Scheme for Large Scale Storage Systems / J. Xinpu, M. Yuxiang, P. Li, J. Ma, G. Wang, X. Liu, Z. Li // International Conference on Algorithms and Architectures for Parallel Processing – 2015. – Vol. 1 – №1 – P. 337-350.

30. Li, J. Hard drive failure prediction using classification and regression trees / J. Li, X. Ji, Y. JIA, B. ZHU и др. // 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks – 2014. – P. 383–394.

31. Higley L. Storage Analytics: Can We Put Any More Lipstick On That Pig? [Электронный ресурс]. – Режим доступа: URL: https://cloud.kapos tcontent.net/pub/3da21605-fc17-4712-991a-1c49dc77b871/mfx131e-pc-mon-130-higleyl.pdf?kui=xxPHjA087 0Nzv0HTEjftEw.

32. Gopisetty S. Evolution of storage management: Transforming raw data into information / S. Gopisetty, S. Agarwala, E. Butler, D. Jadav и др. // IBM Journal of Research and Development – 2008. – Vol. 52 – № 4–5 – P. 341–352.

33. 100 Top Server & Application Performance Monitoring (APM) Solutions [Электронный ресурс]. – Режим доступа: https://haydenjames.io/20-top-server-monitoring-application-performance-monitoring-apm-solutions/.

34. Флеш-платформа прогнозной аналитики с ИИ HPE InfoSight | HPE Россия [Электронный ресурс]. – Режим доступа:

https://www.hpe.com/ru/ru/solutions/infosight.html.

35. HPE Integrated Lights Out — инструменты удаленного управления сервером (iLO) | HPE Россия [Электронный ресурс]. – Режим доступа: https://www.hpe.com/ru/ru/servers/integrated-lights-out-ilo.html.

36. Решение Storage Analytics на базе модели «ПО как услуга» — Dell EMC Unity | Dell Technologies Russia [Электронный ресурс]. – Режим доступа: https://www.delltechnologies.com/ru-ru/storage/cloudiq.htm.

37. Active IQ: AI-Powered Insights [Электронный ресурс]. – Режим доступа: https://mysupport.netapp.com/info/web/AboutAIQ.html.

38. IBM Storage Insights for IBM Spectrum Control [Электронный ресурс]. – Режим доступа: https://www.ibm.com/support/knowledgecenter/SS5R93_5.3.2/com.ibm.spectrum.sc.doc/prd_storage_monitoring_cloud.html.

39. Hitachi Storage Advisor 3.3. User guide [Электронный ресурс]. – Режим доступа: https://download.hitachivantara.com/download/epcra/HSA_v3_3_Documentation_PDL-00HL409-12.zip.

40. Data Sheet FUJITSU ETERNUS SF MA V8.5 Storage Management Software Storage Management Software [Электронный ресурс]. – Режим доступа: www.fujitsu.com/eternus.

41. Storage Performance Monitoring Tools | Infrastructure | AppDynamics | AppDynamics [Электронный ресурс]. – Режим доступа: https://www.appdynamics.com/product/application-performance-management/infrastructure-visibility/storage-performance-monitoring.

42. IntelliMagic Vision for SAN - Vendor-Neutral Storage Monitor [Электронный ресурс]. – Режим доступа: https://www.intellimagic.com/products/intellimagic-vision-for-san/.

43. Storage Monitoring Software: Tools for Storage Management | SevOne

[Электронный ресурс]. – Режим доступа: https://www.sevone.com/network-storage-monitoring.

44. Storage Management Software | SolarWinds [Электронный ресурс]. – Режим доступа: https://www.solarwinds.com/storage-resource-monitor.

45. Storage Monitoring | Sentry Software [Электронный ресурс]. – Режим доступа: https://www.sentrysoftware.com/solutions/storage-monitoring.asp.

46. Storage Monitoring Tools & SAN Performance Monitoring [Электронный ресурс]. – Режим доступа: https://www.eginnovations.com/storage-monitoring.

47. VirtualWisdom: Modernize IT Operations for Infrastructure Visibility [Электронный ресурс]. – Режим доступа: https://www.virtana.com/products/virtualwisdom/.

48. Elastic Stack Overview [7.5] | Elastic [Электронный ресурс]. – Режим доступа: https://www.elastic.co/guide/en/elastic-stack-overview/current/index.html.

49. Nagios - The Industry Standard In IT Infrastructure Monitoring [Электронный ресурс]. – Режим доступа: https://www.nagios.org/.

50. Real-time performance monitoring, done right! [Электронный ресурс]. – Режим доступа: https://github.com/netdata/netdata.

51. okmeter.io – server monitoring ready-made for you [Электронный ресурс]. – Режим доступа: https://okmeter.io/.

52. Server monitoring [Электронный ресурс]. – Режим доступа: https://www.zabbix.com/ru/server_monitoring.

53. Anomaly Detection and Monitoring Service [Электронный ресурс]. – Режим доступа: https://anomaly.io/.

54. datastream.io: An open-source framework for real-time anomaly detection using Python, ElasticSearch and Kibana [Электронный ресурс]. – Режим доступа: https://github.com/MentatInnovations/datastream.io.

55. Bosun [Электронный ресурс]. – Режим доступа: https://bosun.org/.

56. Cacti® - The Complete RRDTool-based Graphing Solution [Электронный ресурс]. – Режим доступа: https://www.cacti.net/what_is_cacti.php.

57. collectd – The system statistics collection daemon [Электронный ресурс]. – Режим доступа: https://collectd.org/.

58. Grafana: The open observability platform | Grafana Labs [Электронный ресурс]. – Режим доступа: https://grafana.com/.

59. Graphite [Электронный ресурс]. – Режим доступа: https://graphiteapp.org/.

60. Introduction to InfluxData’s InfluxDB and TICK Stack | InfluxData [Электронный ресурс]. – Режим доступа: https://www.influxdata.com/blog/introduction-to-influxdatas-influxdb-and-tick-stack/.

61. statsd: Daemon for easy but powerful stats aggregation [Электронный ресурс]. – Режим доступа: https://github.com/statsd/statsd.

62. Prometheus - Monitoring system & time series database [Электронный ресурс]. – Режим доступа: https://prometheus.io/.

63. Sensu | Home [Электронный ресурс]. – Режим доступа: https://sensu.io/.

64. Michelle Tan How to Streamline Infrastructure Monitoring with Sensu, InfluxDB, and Grafana | Grafana Labs [Электронный ресурс]. – Режим доступа: https://grafana.com/blog/2019/05/14/how-to-streamline-infrastructure-monitoring-with-sensu-influxdb-and-grafana/.

65. Ellingwood J. An Introduction to Tracking Statistics with Graphite, StatsD, and CollectD | DigitalOcean [Электронный ресурс]. – Режим доступа: https://www.digitalocean.com/community/tutorials/an-introduction-to-tracking-statistics-with-graphite-statsd-and-collectd.

66. RFC 1157 - Simple Network Management Protocol (SNMP) [Электронный ресурс]. – Режим доступа: https://tools.ietf.org/html/rfc1157.

67. Storage Management Initiative Specification (SMI-S) Releases | SNIA [Электронный ресурс]. – Режим доступа: https://www.snia.org/tech_activities/standards/curr_standards/smi.

68. Scheifler R.W.X Window System : the complete reference to Xlib, X Protocol, ICCCM, XLFD / R. W. Scheifler, J. Gettys – Digital Press, 1992.– 1000с.

69. Использование Windows Management Instrumentation для диагностики - WCF | Microsoft Docs [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/framework/wcf/diagnostics/wmi/.

70. RFC 854 - Telnet Protocol Specification [Электронный ресурс]. – Режим доступа: https://tools.ietf.org/html/rfc854.

71. Costa B.S.J. Real-time fault detection using recursive density estimation / B.S.J. Costa, P.P. Angelov, L.A. Guedes // Journal of Control, Automation and Electrical Systems – 2014. – Vol. 25 – № 4 – P. 428–437.

72. Zhang L. Adaptive kernel density-based anomaly detection for nonlinear systems / L. Zhang, J. Lin, R. Karim // Knowledge-Based Systems – 2018. – Vol. 139 – P.50–63.

73. Kim J. Multivariate network traffic analysis using clustered patterns / J. Kim, A. Sim, B. Tierney, S. Suh и др. // Computing – 2019. – Vol. 101 – № 4 – P. 339–361.

74. Alkasem A. Improving fault diagnosis performance using hadoop mapreduce for efficient classification and analysis of large data sets / A. Alkasem, H. Liu, M. Shafiq // Journal of Computers (Taiwan) – 2018. – Vol. 29 – № 4 – P. 185–202.

75. Yu L. A Scalable, Non-Parametric Method for Detecting Performance Anomaly in Large Scale Computing / L. Yu, Z. Lan // IEEE Transactions on Parallel and Distributed Systems – 2016. – Vol. 27 – № 7 – P. 1902–1914.

76. Lan Z. Toward automated anomaly identification in large-scale systems / Z. Lan, Z. Zheng, Y. Li // IEEE Transactions on Parallel and Distributed Systems – 2010. – Vol. 21 – № 2 – P. 174–187.

77. Wang X. Fault localization using disparities of dynamic invariants / X. Wang, Y. Liu

// Journal of Systems and Software – 2016. – Vol. 122 – P. 144–154.

78. Yin S. Fault detection based on a robust one class support vector machine / S. Yin, X. Zhu, C. Jing // Neurocomputing – 2014. – Vol. 145 – P. 263–268.

79. Costa Silva G. Artificial immune systems applied to fault detection and isolation: A brief review of immune response-based approaches and a case study / G. Costa Silva, W.M. Caminhas R.M., R.M. Palhares // Applied Soft Computing Journal – 2017. – Vol. 57 – P. 118–131.

80. Shapiro J.M. An evolutionary algorithm to generate ellipsoid network intrusion detectors / J. M. Shapiro, G. B. Lamont, G. L. Peterson // Proceedings of the 7th annual workshop on Genetic and evolutionary computation – 2005. – P. 178–180.

81. Balachandran S. A framework for evolving multi-shaped detectors in negative selection / S. Balachandran, D. Dasgupta, F. Nino, D. Garrett // 2007 IEEE Symposium on Foundations of Computational Intelligence – 2007. – C. 401–408.

82. Chen X. Fault Detection and Backtrace Based on Graphical Probability Model / X. Chen, J. Wang, J. Zhou // 2018 Prognostics and System Health Management Conference – 2018. – C.584–590.

83. Plakias S. Exploiting the generative adversarial framework for one-class multi-dimensional fault detection / S. Plakias, Y.S. Boutalis // Neurocomputing – 2019. – Vol. 332 – P. 396–405.

84. Goodfellow I.J. Generative adversarial nets Neural information processing systems foundation / I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu и др. // Proceedings of the 27th International Conference on Neural Information Processing Systems – 2014. – Vol. 2 – P. 2672–2680.

85. Mack D.L.C. Combining expert knowledge and unsupervised learning techniques for anomaly detection in aircraft flight data / D.L.C. Mack, G. Biswas, H. Khorasgani, D. Mylaraswamy, R. Bharadwaj // At-Automatisierungstechnik – 2018. – Vol. 66 – № 4 – P. 291–307.

86. Xu W. Detecting large-scale system problems by mining console logs New York, New York, USA: ACM Press, 2009. – 117p.

87. He P. Towards Automated Log Parsing for Large-Scale Log Data Analysis / P. He, J. Zhu, S. He, J. Li, M.R. Lyu // IEEE Transactions on Dependable and Secure Computing – 2018. – Vol. 15 – № 6 – P. 931–944.

88. Pande A. WEAC: Word embeddings for anomaly classification from event logs / A. Pande, V. Ahuja // IEEE International Conference on Big Data – 2017. – P. 1095–1100.

89. Cinque M. Event Logs for the Analysis of Software Failures: A Rule-Based Approach / M. Cinque, D. Cotroneo, A. Pecchia // IEEE Transactions on Software Engineering – 2013. – Vol. 39 – № 6 – P. 806–821.

90. Lu S. Detecting anomaly in big data system logs using convolutional neural network / S. Lu, X. Wei, Y. Li, L. Wang // 2018 IEEE 16th International Conference on Dependable, Autonomic and Secure Computing – 2018. – C. 159–165.

91. Yamanishi K. Dynamic syslog mining for network failure monitoring New York, New York, USA: ACM Press, 2005. – 499p.

92. Yagoub I. IT Equipment Monitoring and Analyzing System for Forecasting and Detecting Anomalies in Log Files Utilizing Machine Learning Techniques / I. Yagoub, M. A. Khan, L. Jiyun // 2018 International Conference on Advances in Big Data – 2018. – P. 1–6.

93. Lopes F. Automating orthogonal defect classification using machine learning algorithms / F. Lopes, J. Agnelo, C.A. Teixeira, N. Laranjeiro и др. // Future Generation Computer Systems – 2020. – Vol. 102 – P. 932–947.

94. RFC 5424 - The Syslog Protocol [Электронный ресурс]. – Режим доступа: https://tools.ietf.org/html/rfc5424.

95. Berners-Lee T. The semantic web // Sci. Am. – 2001. – Т. 284. – № 5. – С.34–43.

96. Минский М. Фреймы для представления знаний / М. Минский – Москва: Энергия, 1979. – 152с.

97. Новик И.Б. Введение в информационный мир / И. Б. Новик, А. Ш. Абдуллаев – Наука, 1991. – 232 с.

98. Камынина М. Построение фреймовой модели представления знаний в интеллектуальной системе поддержки принятия решений системы управления сетью тактовой сетевой синхронизации / М. Камынина, А. Канаев, Е. Опарин // Бюллетень результатов научных исследований – 2012. – Т. 1 – С. 59–68.

99. The 2012 ACM Computing Classification System [Электронный ресурс]. – Режим доступа: https://www.acm.org/publications/class-2012.

100. Mercier D. Unified access to heterogeneous data sources using an ontology / D. Mercier, H. Cheong, C. Tapaswi // Joint International Semantic Technology Conference – 2018. – С.104–118.

101. SCSI Commands Reference Manual [Электронный ресурс]. – Режим доступа: https://www.seagate.com/files/staticfiles/support/docs/manual/Interface%20manuals/100293068j.pdf.

102. Hard disk drive specifications HGST Ultrastar 7K6000 [Электронный ресурс]. – Режим доступа: https://documents.westerndigital.com/content/dam/doc-library/en_us/assets/public/western-digital/product/data-center-drives/ultrastar-sata-series/product-manual-ultrastar-7k6000-sata-oem-spec.pdf.

103. Михайлов А. Сигнатурный анализатор для многофункционального аппаратно-программного комплекса КДК / А. Михайлов, О. Иванов, М. Успенский // Вопросы радиоэлектроники – 2014. – Т. 1 – № 2 – С.106–112.

104. Иванов О. Подсистема подготовки тестовых проектов для контрольно-диагностических комплексов / О. Иванов, Е. Михайлов, В. Пустоветов, М. Успенский // Вопросы радиоэлектроники – 2013. – Т. 1 – № 1 – С.99–105.

105. What are Linux Logs? Code Examples, Tutorials & More [Электронный ресурс]. – Режим доступа: https://stackify.com/linux-logs/.

106. 12 Critical Linux Log Files You Must be Monitoring - [Электронный ресурс]. –

Режим доступа: https://www.eurovps.com/blog/important-linux-log-files-you-must-be-monitoring/.

107. Linux Logging Basics - The Ultimate Guide To Logging [Электронный ресурс]. – Режим доступа: https://www.loggly.com/ultimate-guide/linux-logging-basics/.

108. LinuxLogFiles - Community Help Wiki [Электронный ресурс]. – Режим доступа: https://help.ubuntu.com/community/LinuxLogFiles.

109. RFC 3164 - The BSD Syslog Protocol [Электронный ресурс]. – Режим доступа: https://tools.ietf.org/html/rfc3164.

110. RFC 6872 - The Common Log Format (CLF) for the Session Initiation Protocol (SIP): Framework and Information Model [Электронный ресурс]. – Режим доступа: https://tools.ietf.org/html/rfc6872.

111. RFC 5427 - Textual Conventions for Syslog Management [Электронный ресурс]. – Режим доступа: https://tools.ietf.org/html/rfc5427.

112. RFC 1413 - Identification Protocol [Электронный ресурс]. – Режим доступа: https://tools.ietf.org/html/rfc1413.

113. Protégé [Электронный ресурс]. – Режим доступа: https://protege.stanford.edu/.

114. Vigo M. Comparing ontology authoring workflows with Protégé: In the laboratory, in the tutorial and in the ‘wild’ / M. Vigo, N. Matentzoglu, C. Jay, R. Stevens // Journal of Web Semantics – 2019. – T. 57 – P. 1–16.

115. OWL - Semantic Web Standards [Электронный ресурс]. – Режим доступа: https://www.w3.org/OWL/.

116. RDF 1.1 N-Quads [Электронный ресурс]. – Режим доступа: https://www.w3.org/TR/n-quads/.

117. Dgraph — A Distributed, Fast Graph Database [Электронный ресурс]. – Режим доступа: https://dgraph.io/.

118. Query language — Dgraph Doc v1.2.0 [Электронный ресурс]. – Режим доступа:

https://docs.dgraph.io/query-language/.

119. GraphQL | A query language for your API [Электронный ресурс]. – Режим доступа: https://graphql.org/.

120. SPARQL Query Language for RDF [Электронный ресурс]. – Режим доступа: https://www.w3.org/TR/rdf-sparql-query/.

121. Holzschuher F. Querying a graph database - Language selection and performance considerations / F. Holzschuher, R. Peinl // Journal of Computer and System Sciences – 2016. – Vol. 82 – № 1 – P. 45–68.

122. Pat. WO2015086448 International, Internet-based Diagnostic Assistant For Device Analysis [Text] / M. Jungmayr, S. Kim, J. Barth – № WO 2015/086448, 18.06.2015.

123. Pat. EP3497527 International, Generation Of Failure Models For Embedded Analytics And Diagnostic/prognostic Reasoning [Text] / Justinian R., Lamparter S. – № WO 2018/052433 A1 – 16.09.2016.

124. Pat. US7603458B1 USA, System And Methods For Processing And Displaying Aggregate Status Events For Remote Nodes [Text] / J. Sexton, P. Bradley, H. Yehuda, D. Barta – № US 7603458 – 13.10.2011.

125. RFC 5234 - Augmented BNF for Syntax Specifications: ABNF [Электронный ресурс]. – Режим доступа: https://tools.ietf.org/html/rfc5234.

126. Hutton G. Monadic Parser Combinators / G. Hutton, E. Meijer – Nottingham, 1996.– 38 p.

127. Zhu J. Tools and Benchmarks for Automated Log Parsing / J. Zhu, S. He, J. Liu, P. He и др., // Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice – 2019 – P. 121–130.

128. Makanju A. Clustering event logs using iterative partitioning. / A. Makanju, N. Zincir-Heywood, E. Milios // Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining – 2009. – P. 1255–1264.

129. Preparing Text for Natural Language Processing - Data Driven Investor - Medium [Электронный ресурс]. – Режим доступа: https://medium.com/datadriveninvestor/preparing-text-for-natural-language-processing-f93b11bfb3fe.

130. Jivani A. A Comparative Study of Stemming Algorithms / A. Jivani // International Journal of Computer Technology and Applications – 2011. – Vol. 2 – № 6 – P. 1930–1938.

131. Haroon M.M. Comparative Analysis of Stemming Algorithms for Web Text Mining / M.M. Haroon // Modern Education and Computer Science – 2018. – Vol. 9 – P. 20–25.

132. Yadav V. A Survey on Recent Advances in Named Entity Recognition from Deep Learning models [Электронный ресурс]/ V. Yadav, S. Bethard – 2019. – Режим доступа: https://arxiv.org/pdf/1910.11470v1.pdf.

133. Pranckevičius T. Comparison of Naive Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification / T. Pranckevičius, V. Marcinkevičius // Baltic Journal of Modern Computing – 2017. – Vol. 5 – № 2 – P. 221–232.

134. Breiman L. Random forests / Breiman L. // Machine Learning – 2001. – Vol. 45 – № 1 – P. 5–32.

135. Mason L. Boosting Algorithms as Gradient Descent. / L. Mason, J. Baxter, P. Bartlett, M. Frean // Proceedings of the 12th International Conference on Neural Information Processing Systems – 1999. – С.512–518.

136. Huang Z. Bidirectional LSTM-CRF Models for Sequence Tagging [Электронный ресурс]/ Z. Huang, W. Xu, K. Yu – 2015. – Режим доступа: https://arxiv.org/pdf/1508.01991.pdf.

137. Le T. Classification performance using gated recurrent unit Recurrent Neural Network on energy disaggregation / T. Le, J. Kim, H. Kim // 2016 International Conference on Machine Learning and Cybernetics – 2016. – С.105–110.

138. Liu G. Bidirectional LSTM with attention mechanism and convolutional layer for text classification / G. Liu, J. Guo // Neurocomputing – 2019. – Vol. 337 – P. 325–338.

139. Chinchor N. {MUC}-5 Evaluation Metrics , 1993 [Электронный ресурс]. – Режим доступа: https://www.aclweb.org/anthology/M93-1007.pdf.

140. Batista D. Named-Entity evaluation metrics based on entity-level [Электронный ресурс]. – Режим доступа: http://www.davidsbatista.net/blog/2018/05/09/Named_Entity_Evaluation/.

141. Лопез П.М. Машинное обучение: алгоритмы для бизнеса / П. М. Лопез – «Издательский дом «Питер» 2019. – 432с.

142. Balamurali M. t-SNE Based Visualisation and Clustering of Geological Domain / M. Balamurali, A. Melkumyan // International Conference on Neural Information Processing – 2016. – P. 565–572.

143. Fillbrunn A. Diversity-driven Widening of Hierarchical Agglomerative Clustering / A. Fillbrunn, M. R. Berthold // International Symposium on Intelligent Data Analysis – 2015. – P. 84–94.

144. Joulin A. Bag of Tricks for Efficient Text Classification / A. Joulin, E. Grave, P. Bojanowski, T. Mikolov // Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics – 2016. – Vol. 2 – P. 427– 431.

145. Mikolov T. Distributed Representations of Words and Phrases and their Compositionality / T. Mikolov, K. Chen, G. Corrado, J. Dean. и др. // Proceedings of the 26th International Conference on Neural Information Processing Systems – 2013. – Vol. 2. – P. 3111–3119.

146. Peters M.E. Deep contextualized word representations / M. E. Peters, M. Neumann, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer. // Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies – 2018. – Vol. 1 – P. 2227–2237.

147. Devlin J. BERT: Pre-training of Deep Bidirectional Transformers for Language

Understanding 2018 [Электронный ресурс]/ J. Devlin, M.-W. Chang, K. Lee, K. Toutanova – Режим доступа: https://arxiv.org/pdf/1810.04805.pdf.

148. Lan Z. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations [Электронный ресурс]/ Z. Lan, M. Chen, S. Goodman, K. Gimpel и др. – 2019. – Режим доступа: https://arxiv.org/pdf/1909.11942.pdf.

149. Le Q. Distributed Representations of Sentences and Documents / Q. Le, T. Mikolov. // ICML'14: Proceedings of the 31st International Conference on International Conference on Machine Learning – 2014. – Vol. 32. – P. 1188–1196.

150. Radford A. Language Models are Unsupervised Multitask Learners [Электронный ресурс]/ A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever. – Режим доступа: https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

151. Jaiswal J.K. Application of Random Forest Algorithm on Feature Subset Selection and Classification and Regression / J. K. Jaiswal, R. Samikannu // 2017 World Congress on Computing and Communication Technologies – 2017. – P. 65–68.

152. Rutherford A.ANOVA and ANCOVA: A GLM Approach: Second Edition / A. Rutherford – Wiley Blackwell, 2013.– P. 344.

153. Дьяконов А. Случайный лес (Random Forest) | Анализ малых данных [Электронный ресурс]. – Режим доступа: https://dyakonov.org/2016/11/14/случайный-лес-random-forest/.

154. Aussel N. Improving performances of log mining for anomaly prediction through NLP-based log parsing / N. Aussel, Y. Petetin, S. Chabridon // 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems – 2018. – P. 237–243.

ПРИЛОЖЕНИЕ А СОСТАВ КОНФИГУРАЦИЙ СХД

Таблица А.1 – Состав конфигураций СХД платформы TATLIN

№ пп	Наименование компонента	Кол-во				
		Конфигурация 1	Конфигурация 2	Конфигурация 3	Конфигурация 4	Конфигурация 5
1	Контроллер хранения	2	2	2	4	-
2	Контроллер фабрики	1	1	1	1	1
3	Дисковые модули расширения	4	4	4	6	-
4	Дисковые модули расширения NVMe	-	-	-	-	1
5	HDD 12 ТБ 7200 об/мин SAS 3.5 дюйма	96	128	128	96	-
6	SSD NVMe 8 Тб U.2	-	-	-	-	64
7	блоки электропитания	2	-	-	4	3
8	источник бесперебойного питания	-	1	1	-	-
9	комплект сетевых коммутаторов FibreChannel	2	-	2	2	-
10	коммутатор сети Ethernet 1 Гбит, 4 порта 1000BASE-T	-	-	-	-	2
11	СПО СХД	1	1	1	1	1
12	комплект периферийных устройств для управления функционированием СХД	1	1	1	1	1

ПРИЛОЖЕНИЕ Б РЕЗУЛЬТАТЫ ПРИМЕНЕНИЯ АЛГОРИТМА ИЕРАРХИЧЕСКОЙ КЛАСТЕРИЗАЦИИ

Рисунок В1 – Результаты применения иерархической кластеризации

ПРИЛОЖЕНИЕ В КОПИИ АКТОВ ВНЕДРЕНИЯ РЕЗУЛЬТАТОВ ДИССЕРТАЦИОННОЙ РАБОТЫ

ООО «КНС Групп»
123376, г. Москва, ул. Рочдельская, д.15 с.15
ИНН 7701411241 / КПП 770301001
T: +7 495 540 50 55 / E: info@yadro.com

Акт
О внедрении результатов кандидатской диссертационной работы

Настоящим актом подтверждается, что основной результат диссертационной работы «Разработка и исследование методов и моделей обработки диагностической информации для обнаружения и локализации неисправностей в системах хранения данных», выполненной Успенским Михаилом Борисовичем в части разработанного нового подхода к обнаружению неисправностей в системах хранения данных, основывающемся на совместном использовании онтологической модели системы хранения данных и алгоритмов машинного обучения, был внедрен в ООО «КНС Групп» в рамках разработки программного комплекса предотвращения сбоев в работе системы хранения данных.

Практическим результатом на текущем этапе внедрения является использование предложенного подхода для диагностирования состояний отдельных компонентов системы хранения данных и системы хранения данных в целом. В рамках созданного прототипа системы хранения данных, полностью повторяющего оригинальный серийный продукт компании ООО «КНС Групп», использование данного метода позволило существенно повысить надежность хранения данных и снизить трудоёмкость решения задачи обнаружения неисправностей.

Генеральный директор
ООО «КНС Групп» _____ / Морозов Н.Л.

М.П.

МИНОБРНАУКИ РОССИИ
федеральное государственное автономное
образовательное учреждение высшего образования
«Санкт-Петербургский политехнический
университет Петра Великого»
(ФГАОУ ВО «СПбПУ»)

ИНН 7804040077, ОГРН 1027802505279,
ОКПО 02068574
Политехническая ул., 29, Санкт-Петербург, 195251
тел.: +7(812)297 2095, факс: +7(812)552 6080
office@spbstu.ru

УТВЕРЖДАЮ
Проректор по научной работе
В.В. Сергеев
04 марта 2020 г.

Акт реализации результатов диссертационной работы Успенского Михаила Борисовича, выполненной на тему «Разработка и исследование методов и моделей обработки диагностической информации для обнаружения и локализации неисправностей в системах хранения данных» и подготовленной для соискания ученой степени кандидата технических наук по специальности 05.13.01 «Системный анализ, управление и обработка информации (технические системы)»

Настоящий акт составлен о том, что следующие результаты диссертационной работы Успенского М.Б.:

1) Метод построения диагностических моделей СХД, отличающихся конфигурацией аппаратных средств, составом программных средств и параметрами использованных схем избыточности, позволяющий обнаруживать большее число типов неисправностей относительно существующих решений за счёт обеспечения возможности совместного использования диагностических параметров разного рода.

2) Метод и алгоритм анализа, трансформации и обработки текстовой информации, получаемой в процессе мониторинга СХД, позволяющий, в отличие от существующих решений, обнаруживать неисправности без детального анализа структуры данных мониторинга, формата и последовательности текстовых сообщений.

3) Комплексный метод обнаружения неисправностей в СХД, основанный на совместном использовании диагностической модели СХД и метода обработки текстовых данных мониторинга СХД с использованием алгоритмов машинного обучения, позволяющий масштабировать онтологическую модель СХД и обеспечивающий увеличение числа обнаруживаемых типов неисправностей относительно существующих средств.

Внедрены в следующем научном проекте, выполняемом в СПбПУ:

- ФЦП, 14.581.21.0023, Разработка аппаратно-программного комплекса для прогнозирования сбоев в работе системы хранения данных с целью предотвращения критических ситуаций, в том числе деградации производительности, отказа сервиса записи/чтения данных и потери данных», 2017-2020

Полученные Успенским М.Б. результаты используются в качестве алгоритмического и методического обеспечения для разработки программного комплекса, предназначенного для обнаружения, прогнозирования и предотвращения сбоев в системах хранения данных. Разработанный с использованием результатов диссертационной работы Успенского М.Б. программный комплекс позволяет обеспечить повышение надежности хранения данных и предотвращение деградации производительности системы хранения данных за счёт своевременного обнаружения внештатных режимов её работы, соответствующих неисправностями в программных, аппаратных компонентах, а также в процессе их взаимодействия. Применяемая диагностическая модель позволяет локализовать возникающие неисправности в рамках архитектуры системы и расширить число диагностируемых состояний, а алгоритм анализа журналов программного обеспечения – увеличить число доступных для анализа диагностических признаков.

Научный руководитель работ Болсуновская М.В.