МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«Сибирский федеральный университет»

На правах рукописи

Грузенкин Денис Владимирович

Модели описания и алгоритмы оценки диверсифицированности для принятия решений в мультиверсионных программных системах

Специальность 2.3.1 – Системный анализ, управление и обработка информации, статистика

Диссертация на соискание ученой степени кандидата технических наук

Научный руководитель:
доктор технических наук, профессор
Ковалёв Игорь Владимирович

Красноярск – 2026

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность работы. Одним из хорошо себя зарекомендовавших на практике направлений повышения надёжности программного обеспечения (ПО) информационных управляющих систем (ИУС) является мультиверсионное программирование, основанное на принципе программной избыточности и диверсифицированности версий, который позволяет обеспечить независимость сбоев в мультиверсиях. Однако, несмотря на признанную эффективность данного подхода, вопросы формализации и количественной оценки диверсифицированности мультиверсий, а также их влияние на процессы принятия решений и оптимизации в мультиверсионных программных системах, остаются недостаточно разработанными, хотя являются одними из наиболее важных, поскольку, чем больше различий между мультиверсиями, тем ниже вероятность возникновения связанных ошибок. Такие ошибки в нескольких версиях могут привести к получению идентичных неверных результатов, что значительно затрудняет процесс принятия решения о выборе действительно правильного ответа.

Для повышения надежности ПО в целом и ИУС в частности существует множество подходов. Все они могут быть разделены на 4 группы: предупреждение ошибок, обнаружение ошибок, исправление ошибок и обеспечение устойчивости к ошибкам. К последней группе относится методология мультиверсионного программирования.

Мультиверсионное программное обеспечение подобно аппаратной части вычислительной системы обеспечивает надёжность за счёт внедрения избыточности. Однако, простое дублирования одного и того же кода, где возникла ошибка, не является эффективным, поскольку отсутствует независимость сбоев, как в случае с аппаратной частью.

Успех мультиверсионного подхода обеспечивается следующими принципами его работы: применение программной избыточности, поддержание диверсифицированности мультиверсий и инкапсуляция.

Помимо перечисленных выше основных принципов также выделяют принцип использования контрольных точек. Его применение позволяет во время выполнения мультиверсии оценить корректность промежуточных результатов и сделать выводы о стабильности и корректности работы мультиверсий, а также в случае возникновения сбоя позволяет запустить версию или модуль с последней контрольной точки.

Таким образом, актуальность исследования обусловлена необходимостью создания формальных моделей и алгоритмов, позволяющих не только описывать и оценивать уровень разнообразия мультиверсий, но и использовать эти оценки для повышения эффективности и надёжности функционирования сложных систем управления и обработки информации.

Степень разработанности проблемы.

Приведённые выше принципы описаны в профессиональной литературе множеством авторов, работа по усовершенствованию мультиверсионного ПО в данных направлениях идёт непрерывно. Однако впервые вопросы диверсифицированности в рамках методологии мультиверсионного программирования рассмотрены в работах Альгирдаса Авижиенса. Автор постулировал, что версии одного модуля должны отличаться по следующим признакам (либо по одному из них): язык программирования, алгоритм, средства разработки или средства тестирования. Также все версии одного модуля должны разрабатываться различными изолированными друг от друга командами разработки. Совместно с Авижиенсом (Algirdas Avizienis) данное направление развивали такие учёные как Лиминг Чен (Liming Chen), Джон Келли (John PJ Kelly), Джин-Клауд Лаприе (Jean-Claude Laprie) и другие. Эти принципы рассматривались также в работках таких авторов, как Ковалёв И.В., Царёв Р.Ю., Морозов В.А. и другие.

Стоит отметить, что в своих работах, например, Морозов В.А., Царёв Р.С. , О. Берман (O. Berman), Н. Ашрафи (N. Ashrafi), Хо-Вон Джунг (Ho-Won Jung), Бьйоунгджу Чой (Byoungju Choi) и другие авторы приводили и иные модели мультиверсий, основанные на их характеристиках, таких как потребление различного рода ресурсов, стоимость разработки и сопровождения, а также их априорная надёжность. При использовании данного подхода появляется возможность не только определить на основе заданных характеристик оптимальный состав мультиверсионного модуля или всей системы целом, но и рассчитать надёжность модуля или мультиверсионной системы, как предлагали Ефимов С.Н. Терсков В.А., Ксяолин Тенг (Xiaolin Teng), Ноанг Фам (Hoang Pham), Ксянг Ли (Xiang Li), Ян Фу Ли (Yan Fu Li), Мин Ксие (Min Xie), Сцу Хуи Нг (Szu Hui Ng) и другие авторы. Однако среди всего множества характеристик, описывающих мультиверсии, мера их разнообразия авторами ранее не определялась. Ни для мультиверсионных систем, ни для их модулей ранее не определялась и не рассчитывалась степень разнообразия входящих в них версий.

Хотя, вне контекста мультиверсионного программирования работы по нахождению меры различия межу программами (их исходным кодом) всё же ведутся. Так многие авторы, например, Косьяненко И.А., Романов В.А., Иванов В.В., Горчаков А.В., Куртукова А.В., Романов А.С., Alina Petukhova, João P. Matos-Carvalho, Nuno Fachada, D Álvarez-Fidalgo, F Ortin, Gary A. McCully, John D. Hastings, Shengjie Xu, Adam Fortier в своих работах представляют программный код в виде эмбеддингов, т.е. векторов, с помощью технологий искусственного интеллекта и используют их для сравнения с заданными образцами или друг с другом. Такой подход применяется, например, для генерации описания кода или определения соответствия его стандарту кодирования, для верификации авторства или поиска уязвимостей и во многих других случаях. Однако в контексте мультиверсионного ПО это не применяется, хотя величина различия созданных на основе исходного кода программ векторов может быть включена в предложенную модель мультиверсии как ещё одна метрика разнообразия.

Таким образом, может быть сделан вывод о недостаточной проработке вопроса реализации базового принципа мультиверсионного ПО, связанного с разнообразием мультиверсий, то есть их диверсификацией. На сегодняшний день отсутствует формальный аппарат применения принципа разнообразия при принятии решений в мультиверсионных программных системах.

Проведенный анализ обуславливает необходимость разработки моделей и алгоритмов, описывающих и обеспечивающих анализ меры различия, т.е. уровень диверсифицированности мультиверсионного программного обеспечения.

Цель диссертационной работы состоит в повышение эффективности принятия решений на основе диверсифицированности при выборе корректного ответа в мультиверсионных программных системах.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Провести анализ моделей, методов и алгоритмов, используемых в процессе выбора верных результатов в мультиверсионных программных системах.
2. Разработать модели представления мультиверсий и модулей в мультиверсионных программных системах.
3. Разработать алгоритм оценки диверсифицированности для блоков принятия решений в мультиверсионных информационных управляющих системах, который позволяет определить группу наиболее различных между собой по набору заданных критериев мультиверсий.
4. Разработать обобщенную методику модификации алгоритмов голосования в мультиверсионных программных системах с учётом использования предложенных моделей представления мультиверсий и модулей.
5. Модифицировать по разработанной методике алгоритм голосования согласованным большинством для подтверждения её применимости и эффективности.
6. Применить модифицированный алгоритм голосования в реальной информационной управляющей системы для подтверждения на практике полученных теоретических результатов.

В диссертационной работе описана и решена актуальная научная задача повышения вероятности выбора верного ответа в ходе голосования в методологии мультиверсионного программного обеспечения.

Методы исследования. При выполнении работы использовались методы и подходы теории вероятностей, методы анализа и проектирования архитектуры информационно-управляющих и программных систем, системного анализа, теории голосования, мультиверсионного проектирования программного обеспечения отказоустойчивых систем, методы теории графов.

Тематика работы соответствует следующим пунктам паспорта специальности 2.3.1: п. №3 разработка критериев и моделей описания и оценки эффективности решения задач системного анализа, оптимизации, управления, принятия решений, обработки информации и искусственного интеллекта; п. №4 разработка методов и алгоритмов решения задач системного анализа, оптимизации, управления, принятия решений, обработки информации и искусственного интеллекта; п. №11 методы и алгоритмы прогнозирования и оценки эффективности, качества, надежности функционирования сложных систем управления и их элементов.

Научная новизна работы заключается в следующем:

1. Разработана новая комбинированная модель формального описания мультиверсий на основе метрики диверсифицированности, отличительной особенностью которой является возможность количественно оценивать разнообразие версий в мультиверсионном программном обеспечении для повышения надёжности функционирования ИУС, в отличие от существующих моделей представления мультиверсий, которые применяются для формирования оптимального состава мультиверсионных модулей, покрытия их тестами или обеспечения конкретных архитектурных характеристик.

2. Разработана новая графовая модель представления мультиверсионного программного модуля, отличительной особенностью которой является возможность формального сравнения групп мультиверсий посредством анализа изоморфизма графов, что позволило формализовать задачу оценки и сопоставления

структурных различий между версиями, в отличие от существующих моделей представления мультиверсионных программных модулей.

3. Впервые разработан алгоритм оценки диверсифицированности для поддержки принятия решений при выборе состава мультиверсионных модулей ИУС, результаты работы которого обеспечивают поддержку автоматизированного выбора групп мультиверсий с более высоким уровнем диверсифицированности, что способствует повышению отказоустойчивости ИУС, так как ранее степень различия версий не применялась при формировании состава мультиверсионного модуля.

4. Разработана новая методика модификации алгоритмов голосования с использованием модели описания мультиверсий, обеспечивающая повышение вероятности выбора корректного результата в условиях неопределённости и неоднородности поведения версий, позволившая получить новую модификацию алгоритма голосования согласованным большинством с интеграцией оценки диверсифицированности для принятия решений в ситуациях, когда классические процедуры голосования не обеспечивают достаточной устойчивости к связанным ошибкам из-за равенства голосов между группами версий.

Значение для теории состоит в разработке новой модели описания мультиверсий, основанной на применении метрик диверсифицированности, нового алгоритма оценки разнообразия версий модуля, а также в усовершенствованном алгоритме голосования, что в совокупности расширяет возможности оценки надёжности функционирования программного обеспечения ИУС.

Результаты, полученные в ходе выполнении диссертационной работы, создают теоретическую основу для разработки нового специального алгоритмического обеспечения среды исполнения мультиверсионного ПО ИУС, в частности блока голосования в таких системах. Благодаря предложенным модели и алгоритмам обеспечивается повышение эффективности процессов обработки информации и управления в высоконадёжных программных системах.

Практическая ценность. Предложенные в диссертации модели и алгоритмы позволяют рассчитать меру различия мультиверсий, что позволяет получить

оценку качества мультиверсионных модулей путём сравнения мультиверсий практически на любом из этапов жизненного цикла мультиверсионного программного обеспечения ИУС. Данное обстоятельство обеспечивает возможность оперативного принятия решений о модификации мультиверсионного программного комплекса, если это необходимо, а также позволяет оценивать его меру разнообразия, поскольку на аксиоме независимости сбоев в различных версиях, основанной на их диверсифицированности, и построен мультиверсионный подход. Это утверждение подтверждается множеством успешных практических применений методологии мультиверсионного программирования на практике, например, на железной дороге, в атомной энергетике, авиации и многих других сферах, что подтверждается большим количеством публикаций.

Предложенная методика модификации алгоритмов голосования позволяет повышать эффективность алгоритмов голосования, используемых в блоке принятия решений мультиверсионной информационной управляющей системы, что является важной практической задачей.

Модифицированный метод голосования согласованным большинством позволяет более эффективно производить выбор верного варианта решения в мультиверсионных ИУС за счёт введения показателя диверсифицированности, который помогает сделать выбор в условиях неопределённости.

Основные положения, выносимые на защиту:

1. Комбинированная модель формального описания мультиверсий программных модулей, основанная на метрике диверсифицированности, которая позволяет количественно оценивать разнообразие версий мультиверсионного программного обеспечения, что является необходимым условием для повышения надёжности функционирования информационных управляющих систем.

2. Графовая модель представления мультиверсионного программного модуля, которая позволяет производить сравнение программных модулей (групп мультиверсий) между собой с целью поддержки принятия решений при выборе верного ответа всего модуля или при формировании его состава.

3. Алгоритм оценки диверсифицированности на основе графовой модели представления программного модуля, позволяющей формально сравнивать группы версий, обеспечивает автоматизированный выбор групп с максимальным уровнем разнообразия для повышения отказоустойчивости ПО ИУС.

4. Методика модификации алгоритмов голосования на основе разработанной модели описания мультиверсий позволяет повысить устойчивость мультиверсионных ИУС к ошибкам, присущим каждой версии и модифицировать алгоритма голосования относительным большинством для повышения вероятности выбора верного ответа в условиях неопределённости.

Достоверность полученных результатов подтверждается результатами моделирования в имитационной среде, а также реализацией мультиверсионного программного модуля с применением описанных в диссертации результатов, внедрённого в лабораторную ИУС ОАО «Красцветмет». Противоречий между теоретическими расчётами и реальными данными, полученными в ходе моделирования и реализации описанных методов и алгоритмов, не выявлено.

Результаты внедрения. Созданный для лабораторной информационной управляющей системы ОАО «Красцветмет» мультиверсионный программный модуль обеспечил возможность безотказной работы ПО в ходе проведения анализов рентгеноспектральным методом вторичного сырья на основе золота.

Работа выполнена в рамках гранта РНФ № 25-11-20040, https://rscf.ru/project/25-11-20040/», гранта Красноярского краевого фонда науки.

Апробация работы. Основные положения и результаты работы прошли всестороннюю апробацию на следующих конференциях: международной научно-практической конференции «Новая наука: от идеи к результату» (г. Сургут, 2016 г.), на международной конференции «Математические и информационные технологии, MIT-2016» (г. Врнячка Баня, Сербия – г. Будва, Черногория, 2016), на международной конференции «Information Technologies in Business and Industry» (г. Новосибирск, 2019 г.), на VI Международной научной конференции "Информатизация образования и методика электронного обучения: цифровые технологии в образовании" (г. Красноярск, 2022 г.), на III Всероссийской научной

конференции с международным участием «Наука, технологии, общество: Экологический инжиниринг в интересах устойчивого развития территорий» (г. Красноярск, 2022 г.), на Международной школе-семинаре HMMOCS-III «Гибридные методы моделирования и оптимизации в сложных системах» (г. Красноярск, 2024 г.).

Публикации. По теме диссертации опубликовано 15 печатных работ (3 без соавторов), из них 7 статьи в журналах перечня ВАК и 4 в изданиях, индексируемых в международных базах цитирования Web of Science и/или Scopus.

В работах, опубликованных в соавторстве, которые приведены в конце автореферата, лично автором получены следующие результаты: [9] – определена метрика диверсифицированности мультиверсий на уровне алгоритмов; [16] – определена метрика диверсифицированности на уровне языков программирования; [14] – предложена и описана графовая модель производственного плана, основанная на метрике диверсифицированности на уровне алгоритмов, которая позволила определять дублирующиеся производственные планы; [55] – описан процесс создания мультиверсионного программного модуля с применением метрик диверсифицированности на уровне языков программирования и алгоритмов; [12] – обобщён подход t/(n-1)-вариантного программирования в рамках исследования алгоритмов голосования; [17] и [20] – определены критерии и проведено сравнение методологий мультиверсионного программирования и блоков восстановления; [15] – определена общая метрика диверсифицированности, основанная на нескольких частных метриках; [13] – проведён анализ моделей, используемых в программном обеспечении ИУС для принятия решений; [69], [70] – дано расширенное обоснование метрик диверсифицированности мультиверсий на уровне алгоритмов и языков программирования, соответственно.

1 Обзор предметной области

Приведённые выше принципы описаны в профессиональной литературе множеством авторов, работа по усовершенствованию мультиверсионного ПО (МВПО) в данных направлениях идёт непрерывно. Однако впервые вопросы диверсифицированности в рамках методологии мультиверсионного программирования рассмотрены в работах Альгирдаса Авижиенса. Автор постулировал, что версии одного модуля должны отличаться по следующим признакам (либо по одному из них):

- язык программирования;
- алгоритм работы;
- средства разработки;
- средства тестирования [28].

Также версии одного модуля считаются диверсифицированными, если разрабатываются различными изолированными друг от друга командами разработки.

Совместно с Авижиенсом (Algirdas Avizienis) данное направление развивали такие учёные как Лиминг Чен (Liming Chen), Джон Келли (John PJ Kelly), Джин-Клауд Лаприе (Jean-Claude Laprie) и другие. Эти принципы рассматривались также в работках таких авторов, как Ковалёв И.В., Царёв Р.Ю., Морозов В.А. и другие. Однако иных критериев определения разнообразия версий так и не было предложено.

Данными обстоятельствами обуславливается необходимость проведения обзора предметной области с целью выявления моделей представления мультиверсий и мультиверсионных программных модулей, которые позволяли бы определить меру различия мультиверсий модуля.

1.1 Модели представления мультиверсий

1.1.1 Модель «Чёрный ящик»

Когда версия мультиверсионного модуля представляется в виде «чёрного ящика», её внутренняя структура никак не учитывается [32].

В этой модели каждая версия описывается матрицей отказов или вероятностью отказа и матрицей совместных отказов (корреляция между версиями). Эта модель идеальна для задач оптимизации (какие N версий выбрать из M доступных), так как она абстрагируется от внутренней структуры кода и фокусируется только на входных/выходных данных [87].

В методологии мультиверсионного программного обеспечения модель «Чёрного ящика» является одним из фундаментальных абстрактных представлений мультиверсий и мультиверсионных модулей. В данной парадигме каждая программная версия не анализируется на уровне исходного кода или алгоритмической структуры. Вместо этого система рассматривается исключительно через её функциональную семантику: соответствие пар «входные данные – выходной результат» [96].

Математически мультиверсия в этой модели может характеризоваться своей априорной надежностью, количеством потребляемых ресурсов, стоимостью её создания и поддержки, а также иными характеристиками, учёт которых требуется для решения конкретной задачи. При этом мультиверсионный модуль представляется композицией таких характеристик [87].

Данная модель широко используется в задачах оптимизации состава мультиверсионных систем, как, например, в работах [23], [58], [66], [81]. Модель широко используется, когда существует набор из M доступных версий, требуется выбрать подмножество из N версий, которое окажется оптимальным для заданной переменной, по которой производится оптимизация, с учётом введённых ограничений.

Сильными сторонами описываемой модели являются:

– Вычислительная эффективность: оценка надежности производится на основе статистических данных, что позволяет проводить сложные оптимизационные расчеты в кратчайшие сроки.

– Независимость от реализации: модель применима к гетерогенным системам, где версии написаны на разных языках или разработаны разными командами, и структура которых неизвестна.

– Универсальность: позволяет формализовать задачу выбора оптимального набора версий в рамках комбинаторной оптимизации [77].

При этом слабыми сторонами являются:

– Игнорирование причин отказов: модель не различает структурные и логические ошибки, что делает невозможным выявление корневых причин корреляции отказов без дополнительных экспериментов.

– Проблема оценки наличия зависимых сбоев: поскольку корреляция часто оценивается эмпирически, риск недооценки системных сбоев, вызванных общей логической ошибкой, остается высоким.

– Невозможность динамической адаптации: модель не учитывает структуру входных данных, что ограничивает возможность создания адаптивных систем, выбирающих версию в зависимости от контекста задачи [77].

Применение модели «Чёрного ящика» является целесообразным в ситуациях, когда:

– Требуется быстрый синтез состава системы из большого набора готовых модулей.

– Версии являются гетерогенными и доступ к их внутреннему коду ограничен или невозможен.

– Акцент исследования сделан на количественной оценке надежности и статистическом анализе, а не на структурном анализе кода.

Напротив, использование данной модели нецелесообразно в случаях:

– Необходимости уменьшения корреляции отказов путем модификации алгоритмов (требует анализа «белого ящика»).

– Проектировании систем, где надежность зависит от последовательности операций или временных характеристик (требуется моделирование автоматов).

– Анализе специфических сценариев эксплуатации, где распределение входных данных крайне неравномерно, и усредненная вероятность отказа не отражает реального поведения системы [77].

1.1.2 Модель «Белый ящик»

В отличие от абстрактной модели «чёрного ящика», подход «Белого ящика» в теории мультиверсионного (*N*-версионного) программирования предполагает детальный анализ внутренней структуры программного обеспечения. В данной модели каждая версия рассматривается не как статистическая вероятность отказа, а как детерминированная функциональная структура, состоящая из переменных, логических условий, ветвей управления и потоков данных [84]. Основной целью такого представления является выявление и минимизация корреляций между версиями на уровне алгоритмической логики, а не только на уровне выходных данных [102].

Математически модель «Белого ящика» описывает пространство состояний и переходов, где отказ трактуется как достижение недопустимого состояния в графе путей исполнения. Каждая версия характеризуется набором логических формул [80].

Мультиверсионный модуль представляется как пересечение множеств путей исполнения всех версий. Критическим параметром здесь является степень перекрытия логических ветвей между версиями, что позволяет количественно оценить вероятность совместного отказа без проведения длительных статистических испытаний [102].

Данная модель находит широкое применение в задачах структурного синтеза мультиверсионных систем и генерации тестовых покрытий. Она используется при выборе версий не по принципу случайности, а на основе анализа кода. Это позволяет минимизировать риск общих сбоев до этапа эксплуатации. Также модель критически важна для верификации корректности механизмов голосования, анализа уязвимостей и создания адаптивных систем, где выбор версии зависит от текущего контекста выполнения, а не только от мажоритарного результата [94].

К сильным сторонам модели «Белый ящик» относятся:

– Глубокий анализ причин отказов: позволяет выявить корневые причины корреляции ошибок (например, использование одной и той же библиотеки или логической конструкции).

– Проактивное снижение рисков: возможность снизить вероятность совместного отказа еще на этапе проектирования, выбирая версии с минимальным структурным сходством.

– Обоснованность тестирования: позволяет создавать тестовые наборы, целенаправленно нацеленные на разветвления кода, где версии могут расхождения в логике [83].

При этом в качестве слабых сторон могут быть выделены:

– Высокая сложность: анализ кода требует значительных вычислительных ресурсов и времени, особенно для крупных систем с миллионами строк кода.

– Зависимость от доступности кода: модель неприменима, если версии являются проприетарными или их исходный код недоступен для анализа.

– Эвристичность оценки сходства: определение метрики структурного сходства само по себе является сложной задачей, требующей специализированных алгоритмов [83].

Таким образом, применение модели «Белого ящика» является целесообразным в ситуациях, когда:

– Требуется гарантировать отсутствие общих сбоев.

– Проводится синтез архитектуры системы с нуля, и есть возможность выбирать алгоритмы с разной логической структурой.

– Необходимо объяснить природу отказов в рамках статистического анализа (например, если частота отказов выше прогнозируемой, необходимо найти структурную причину).

Напротив, использование данной модели нецелесообразно в случаях:

– Необходимости быстрого отбора версий из большого пула готовых «черных ящиков» без доступа к исходному коду.

– Ограниченности ресурсов или времени на этапе проектирования, когда допустимо использование упрощенных вероятностных моделей.

– Работы с системами, где внутренняя логика не влияет на надежность (например, случайные алгоритмы, где структура не определяет выходной результат) [101].

1.1.3 Модели на основе конечных автоматов

В контексте повышения надежности программных систем, модели на основе конечных автоматов представляют собой формальный подход к описанию мультиверсионного программирования, где акцент смещается с единичных вход-выходных пар на траектории выполнения и временную эволюцию состояния системы [68]. В данной парадигме каждая программная версия формализуется как детерминированный или недетерминированный конечный автомат, состоящий из множества состояний, множества входных событий, функции переходов и функции выходов [75].

Мультиверсионный модуль в этой модели рассматривается не как набор параллельных независимых процессов, а как результат синхронной композиции нескольких автоматов. Система переходит из одного глобального состояния в другое только после того, как все или большинство версий согласованно обработают входное событие и выдадут выходное значение. Ошибка (отказ)

интерпретируется как попадание в специально выделенное множество «запрещенных» или «сбойных» состояний. Ключевой особенностью такой модели является способность отслеживать зависимость текущего результата не только от текущего ввода, но и от истории предшествующих состояний, что критически важно для систем реального времени [62].

Математическая модель надежности в рамках конечных автоматов описывается вероятностью нахождения системы в допустимом множестве состояний на определённом шаге. Вероятность отказа системы рассчитывается как сумма вероятностей нахождения системы в подмножестве отказывающих состояний, куда система переходит в результате несовпадения выходов версий при голосовании [75].

Модели на основе конечных автоматов находят широкое применение в проектировании систем с жесткими требованиями к временным характеристикам и безопасности [60]. Они используются для верификации корректности протоколов взаимодействия в распределенных мультиверсионных системах, где последовательность действий имеет решающее значение (например, в системах управления ядерными реакторами или авионике). Данная модель также полезна для анализа сценариев, где отказ одной версии может привести к «зависанию» или некорректной синхронизации, если механизм голосования не учитывает временные дедлайны [68]. Кроме того, модели на основе конечных автоматов позволяют формально доказать отсутствие определенных классов ошибок (например, недопустимых состояний) с помощью моделей проверок [31].

К сильным сторонам моделей на основе конечных автоматов относятся:

– Формальная верификация: возможность использовать математические методы для доказательства отсутствия специфических ошибок проектирования, таких как тупики или живые тупики.

– Учет временной динамики: точное моделирование задержек, дедлайнов и последовательности операций, что делает модель реалистичной для встроенных систем.

– Выявление логических паттернов отказов: позволяет обнаруживать сценарии, при которых система попадает в недопустимое состояние не из-за ошибки вычислений, а из-за некорректной последовательности событий (например, потеря пакета данных, нарушающая синхронизацию голосований).

– Детальное описание состояний: позволяет дифференцировать виды отказов (например, сбой в вычислениях или сбой в управлении ресурсами) [92].

При этом к слабым сторонам относятся:

Проблема экспоненциального роста состояния: при увеличении количества версий и сложности их внутреннего автомата, размер составного пространства состояний растет экспоненциально, что делает полный анализ вычислительно невозможным для крупных систем.

– Сложность формализации: требуется высокий уровень квалификации для корректного построения автоматов, отражающих реальное поведение сложного программного кода.

– Отсутствие учёта статистической природы: традиционные модели на основе конечных автоматов часто являются детерминированными и могут не отражать вероятностный характер отказов, если не расширяются до вероятностных конечных автоматов, что усложняет анализ.

– Сложность интеграции: перевод существующих программных модулей в форму автоматов является трудоемким процессом, требующим глубокого анализа кода [68].

Применение модели на основе конечных автоматов является целесообразным в ситуациях, когда:

– Критична последовательность событий: система функционирует в условиях, где порядок выполнения операций и временные интервалы между ними напрямую влияют на безопасность и корректность работы.

– Требуется формальное доказательство надежности: необходимо строго доказать отсутствие определенных классов ошибок на этапе проектирования с помощью формальных методов.

– Моделируются системы реального времени: где задержки обработки и соблюдение дедлайнов являются частью логики отказоустойчивости (например, в системах управления движением транспорта).

– Система имеет сложное управление состоянием: где поведение программы определяется не только текущим вводом, но и внутренней историей выполнения.

Напротив, использование данной модели нецелесообразно в случаях:

– Анализе «бесконечных» или статических вычислений: для простых математических функций, где результат не зависит от истории и временных параметров, использование автоматов является избыточным усложнением.

– Ограниченных вычислительных ресурсах моделирования: когда количество версий велико, и полное построение пространства состояний приводит к экспоненциальному взрыву сложности, делающему анализ невозможным.

– Необходимости быстрой оценки надежности: если требуется оперативная оценка вероятности отказа без детального построения формальных моделей, модели «черного ящика» или «статистические» подходы являются более эффективными.

– Отсутствии четкой логики переходов: если поведение системы сложно формализовать в виде конечного множества состояний и переходов (например, из-за работы с неструктурированными данными или нейросетевыми моделями) [92].

1.1.4 Модели, учитывающие распределение данных

В классических моделях надежности *N*-версионного программирования (NVP) часто предполагается равномерное или детерминированное распределение входных данных, что позволяет оперировать усредненными вероятностями отказов. Однако в реальных эксплуатационных сценариях входные данные могут иметь сложное, неоднородное распределение, которое существенно влияет на вероятность сбоев конкретной версии. Модели, учитывающие распределение

данных, рассматривают надежность не как глобальную константу, а как функцию от плотности распределения входных переменных [78].

В данной модели каждая версия описывается как поле вероятностей отказов. Версия считается более надежной не на всем множестве входных данных, а в определенных его подмножествах. Мультиверсионный модуль моделируется как ансамбль функций, где итоговый результат формируется на основе мажоритарного голосования, взвешенного с учетом локальной плотности данных. Основным отличием от модели «Черного ящика» является переход от скалярной оценки надежности к функциональной зависимости, позволяющей выявлять «зоны уязвимости», где конкретная версия склонна к отказам при определенных условиях ввода [24].

Математически вероятность отказа системы в этом подходе вычисляется путем интегрирования по пространству входных данных с использованием плотности распределения [78].

Данная модель важна для адаптивного мультиверсионного программирования и динамического выбора версий. Она применяется в системах, где характер нагрузки предсказуем или меняется во времени [38]. Например, в системах управления беспилотными летательными аппаратами (БПЛА), где в условиях турбулентности (сложные входные данные) одна версия алгоритма стабилизации может работать хуже, чем в спокойной атмосфере, или в системах обработки телеметрической информации с малых космических аппаратов [24].

Модель позволяет:

– Синтезировать состав версий под конкретный профиль эксплуатации, выбирая те мультиверсии, которые наиболее надежны именно в «рабочих» зонах распределения данных, игнорируя их слабые места в маловероятных сценариях [24].

– Оптимизировать тестирование, целенаправленно генерируя тестовые данные, которые соответствуют пикам плотности, чтобы максимально точно оценить надежность в реальных условиях.

– Реализовывать гибридные стратегии, где для входных данных из области 1 приоритет отдается версии 1, а для области 2 – версии 2, что требует знания функции распределения входных данных [97].

К сильным сторонам модели можно отнести:

– Высокая точность оценки: позволяет получать реальную оценку надежности, специфичную для конкретного сценария использования, избегая ошибок усреднения.

– Возможность адаптации: открывает путь к созданию систем, которые могут динамически переключаться между версиями или менять веса голосования в зависимости от текущей нагрузки.

– Оптимизация ресурсов: позволяет не тратить ресурсы на обеспечение надежности в редких или незначимых сценариях, фокусируясь на критических областях распределения.

– Выявление специфических ошибок: помогает найти «слепые зоны» в логике версий, которые могут быть скрыты при равномерном тестировании, но проявляются на узких участках распределения [74].

При этом к слабым сторонам относятся:

– Сложность оценки плотности данных: требуется точное знание или надежная оценка функции распределения, что часто невозможно сделать априори без исторических данных.

– Вычислительная сложность интеграции: численное вычисление интегралов в многомерном пространстве входных данных требует значительных ресурсов и времени.

– Сложность моделирования функций отказов: построение точной функции для каждой версии требует обширных тестовых данных и сложных статистических методов аппроксимации.

– Риск смещения: если оценка функции неточна, то оптимизация состава может привести к ухудшению надежности в реальных условиях [74].

Применение модели, учитывающей распределение данных, является целесообразным в ситуациях, когда:

– Существует предсказуемый профиль нагрузки: система работает в условиях, где входные данные имеют четкое, устойчивое распределение (например, медицинские приборы, работающие в определенных диапазонах параметров пациента).

– Требуется адаптивность: необходимо создать систему, способную менять свое поведение в зависимости от контекста (динамический выбор версий).

– Существует гетерогенность версий: разные версии имеют различные сильные и слабые стороны в разных частях пространства входных данных, и их комбинация может дать значительный выигрыш в надежности по сравнению с выбором одной лучшей версии.

– Ресурсы на тестирование ограничены: необходимо получить максимально точную оценку надежности, используя минимальное количество тестов, сфокусированных на наиболее вероятных сценариях [97].

Напротив, использование данной модели нецелесообразно в случаях:

– Отсутствия информации о распределении данных: если входные данные являются полностью случайными и неизвестными, использование упрощенной модели «черного ящика» или равномерного распределения более оправдано.

– Отсутствия вычислительных ресурсов: если невозможно провести сложный численный анализ или сбор достаточного массива данных для построения функций.

– Необходимости быстрого принятия решений: в системах, где требуется мгновенный синтез состава, сложность интегрирования может быть избыточной.

– Однородных условий эксплуатации: если система работает в узком диапазоне данных, где все версии ведут себя схожим образом, детальное моделирование распределения не даст существенного преимущества [97].

1.1.5 Сравнительный анализ

Для проведения корректного сравнительного анализа было выделено пять ключевых критериев, характеризующих каждую из рассмотренных моделей («Чёрный ящик», «Белый ящик», конечные автоматы, модели с распределением данных):

– Уровень абстракции. Он определяет степень детализации внутренней структуры программного обеспечения, заложенной в модель.

– Способ учета корреляции отказов. Здесь оценивается, насколько глубоко модель способна выявлять и моделировать общие ошибки, возникающие из-за схожести алгоритмов или логики.

– Вычислительная сложность анализа. Она отражает ресурсы, необходимые для построения модели и проведения расчетов надежности, включая временные и вычислительные затраты.

– Адекватность учета временных и контекстных параметров. Этот показатель характеризует способность модели учитывать последовательность событий, временные задержки и влияние среды выполнения.

– Возможность оценки структурного разнообразия. Критично важный аспект, отражающий, способна ли модель количественно измерить степень различия между версиями на этапе формирования состава системы.

Сравнение базируется на принципе компромисса между детализацией модели и практической применимостью в прикладных задачах. Чем выше детализация (как в моделях «Белого ящика» или конечных автоматов), тем выше точность оценки, но тем сложнее и дороже получение модели. И наоборот, упрощенные модели («Чёрный ящик») обеспечивают быструю оценку, но могут скрывать фундаментальные причины потенциальной ненадежности. Результаты представлены в таблице 1.

Таблица 1 – Сводные результаты сравнительного анализа моделей мультиверсий

Критерий сравнения	**Чёрный ящик**	**Белый ящик**	**Конечные автоматы**	**Распределение данных**
Уровень абстракции	Высокий (только вход-выход)	Низкий (полная структура кода)	Средний/Низкий (состояния и переходы)	Средний (статистическая карта)
Учет корреляции отказов	Эмпирический (на основе тестов)	Структурный (анализ логики)	Временной и логический	Статистический (в области данных)
Вычислительная сложность	Низкая	Высокая	Экспоненциальная	Средняя/Высокая (интегрирование)
Учет временных параметров	Отсутствует	Частично (через логику)	Высокая (явное моделирование)	Опосредованно (через сценарии)
Оценка структурного разнообразия	Не позволяет	Позволяет (частично)	Позволяет (в аспекте путей)	Не позволяет

Анализ представленной таблицы и описания моделей свидетельствует о том, что каждая из рассмотренных моделей обладает строго очерченной областью применимости и не может считаться универсальным решением для всех задач повышения надежности.

Модель «Чёрного ящика» оптимальна для задач быстрой оптимизации состава из больших наборов версий, когда доступ к исходному коду ограничен или требуется статистическая оценка. Однако она не способна выявить структурные причины сбоев.

Модель «Белого ящика» полезна при проектировании систем, где критически важно минимизировать корреляцию ошибок на уровне алгоритма, но её применение затруднено высокой трудоемкостью и необходимостью полного доступа к коду.

Модели на основе конечных автоматов дают исчерпывающую информацию о динамике поведения системы и критичны для систем реального времени, но их масштабируемость ограничена экспоненциальным ростом пространства состояний.

Модели, учитывающие распределение данных, позволяют проводить точную оценку надежности в конкретных эксплуатационных сценариях, однако требуют достоверных статистических данных о нагрузке и сложны в реализации.

Несмотря на многообразие существующих подходов, анализ выявляет критический пробел в теории мультиверсионного программирования. Ни одна из описанных моделей не предлагает единого, общезначимого механизма для оценки меры различия между мультиверсиями, что является необходимым условием, как для повышения надежности модуля при формировании его состава, так и для принятия решения в ходе выбора верного ответа модуля алгоритмом голосования.

При этом, важно отметить, что модель «Белого ящика» позволяет выявлять зависимости и структурные совпадения, однако получаемая оценка различия носит локальный характер и зависит от конкретной задачи и выбранного алгоритма анализа. Для каждой новой группы версий необходимо заново разрабатывать метрику сравнения, что делает процесс более субъективным и менее масштабируемым.

Отсутствие единого подхода к определению численной меры различия, которая бы объективно характеризовала степень диверсификации мультиверсий и их устойчивости к общим и взаимозависимым ошибкам, сдерживает развитие эффективных методов синтеза мультиверсионных систем.

В связи с этим, актуальной научной задачей становится создание новой модели мультиверсии, способной в общем виде определять меру различия входящих в состав модулей версий. Такая модель должна не только оценивать различия между парами версий, но и предоставлять методологию для оценки общей меры диверсифицированности всего мультиверсионного модуля.

Причём важно, чтобы модель описания мультиверсии с учётом меры её диверсифицированности могла быть применима в процессе выбора верного ответа всего мультиверсионного модуля с целью реализации базового принципа мультиверсионного программирования: чем более различны мультиверсии, тем ниже вероятность зависимых ошибок и сбоев, чем ниже эта вероятность, тем надёжнее модуль.

На основании приведённых выводов был разработан дизайн исследования, представленный на рисунке 1.

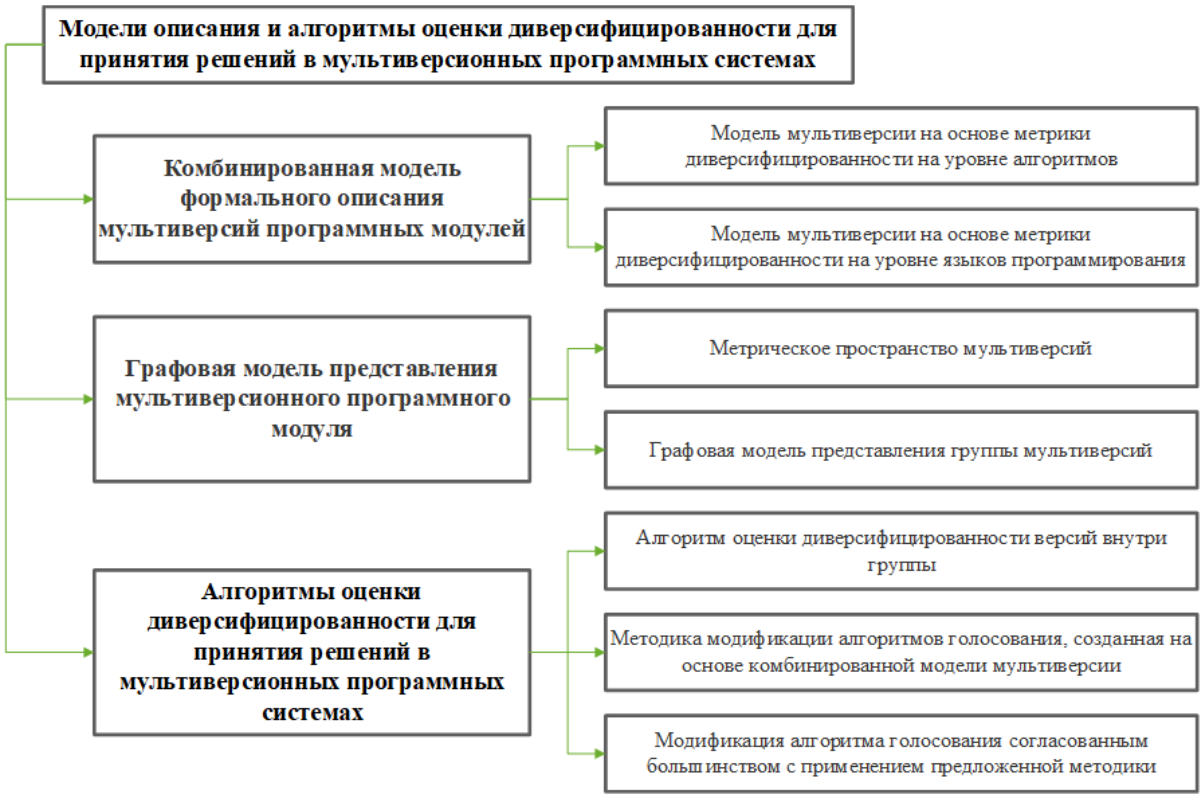

Рисунок 1 – Дизайн исследования

Ниже рассмотрены теории и подходы, методы которых могли бы быть использованы в качестве алгоритмов голосования в среде исполнения мультиверсионного программного обеспечения.

1.2 Подходы и методы выбора значения из множества альтернатив

На сегодняшний день сформировалось множество направлений и дисциплин, направленных на изучение методов оценки возможных решений с целью выявления наиболее благоприятного из них. Кроме того, вопросы выбора одной альтернативы из множества также часто встречаются в современном научно-техническом дискурсе, в том числе и в области мультиверсионного

программирования. Поэтому в данном разделе приведён обзор таких подходов и методов.

1.2.1 Методы поддержки принятия решений

Одним из самостоятельных направлений является теория принятия решений, начало формирования которого относят к середине XX века. Первоначально являлась ответвлением методологии системного анализа.

Данное направление науки занимается разработкой методов и средств, позволяющих выделить множество всех возможных решений поставленной задачи (проблемы) и выявить среди них наилучшие в соответствии с определенными требованиями, а также обосновать принятое решение [43]. Теория принятия решений играет ключевую роль не только в инженерных задачах, но и управленческих, экономических, политических и других задачах, решения которых могут повлечь за собой критически важные последствия [22].

Для приведения любой задачи к математическому виду необходимо избавиться от какой бы то ни было неопределенности, возникающей в рамках изучаемой предметной области. Для этого принято формулировать дополнительные условия, основанные на опыте и предположениях, которые «снимают» эту неопределенность и ограничивают множество потенциальных решений [95].

Формально задача принятия решения D может быть обобщенно поставлена как:

$$D = \langle F, A, X, G, P \rangle,$$

где F – формулировка задачи, A – совокупность возможных альтернатив, X – совокупность признаков, описывающих отличительные особенности вариантов; G – условия, ограничивающие область допустимых вариантов, P – предпочтения лиц, принимающих решение [22].

Среди методов теории принятия решений, сформировавшихся за время существования науки, многие успешно применяются при разработке сложных технических систем. Большинство методов ранее использовалось в разработке систем поддержки принятия решений (СППР) для определения наиболее «правильного» с экспертной точки зрения выбора в ходе работы программы [43].

Несмотря на то, что методы теории принятия решений в рамках оценки технического уровня систем недостаточно представлены в научно-технической литературе, некоторые из них могут быть применимы в том числе и в качестве алгоритмов голосования в среде исполнения мультиверсионного программного обеспечения (МВПО), например: метод весовых коэффициентов, метод идеальной точки и метод ЭЛЕКТРА [47].

Метод весовых коэффициентов определяет оценку полезности альтернативы как сумму произведений весовых коэффициентов критериев и их оценок [49]. При этом сумма весов всех альтернатив равняется единице. Данный метод подразумевает прохождение нескольких этапов:

1. Выявление ключевых характеристик успеха технической системы. Данные характеристики вариативны и напрямую зависят от назначения системы;

2. Оценка системы по каждому критерию, производимая по любой балльной шкале. Одним из наиболее популярных способов является метод ранжирования, подразумевающий оценку важности каждого критерия некоторой группой экспертов. Так, каждый из L экспертов расставляет m критериев в порядке, отражающим значимость каждого из них, где ранг 1 – наиболее важный, m - наименее важный. Далее происходит преобразование рангов таким образом, что первый ранг получает значение m, второй – $m-1$, последний – 1. Обозначить полученные оценки можно через r_{ik}, где i – номер эксперта, k – номер критерия. Результаты сводятся в таблицу, из L строк, где L+1 строка представляет собой сумму оценок каждого критерия в соответствии с его оценкой каждым экспертом. Сумма выражается формулой: $r_i = \sum_{j=1}^{L} r_{ji}\,, i = \overline{1,m}$;

3. Происходит определение весовых коэффициентов каждой характеристики, учитывая степень важности каждого критерия для достижения

корректности работы системы. Сумма весовых коэффициентов по всем критериям должна равняться единице. В случае, когда применяется метод ранжирования, определение весовых коэффициентов происходит по формуле: $\lambda_i = \frac{r_i}{\sum_{i=1}^{m} r_i}, i = \overline{1,m}$.

Критерий i, для которого значение λ_i является наибольшим, является наиболее важным, и наоборот;

4. Происходит определение ранга характеристик для каждой альтернативы, представленного взвешенной оценкой. Данная оценка вычисляется как произведение оценки альтернативы по каждому критерию на его вес;

5. Производится заключение о надежности каждой альтернативы или ее уязвимости в сравнении с другими. Для этого рассчитывается общая сумма оценок критериев для каждой альтернативы: $R_j = \sum_{i=1}^{m} R_{ij}$.

Данная оценка служит показателем надежности альтернативы, то есть чем больше ее надежность, тем устойчивее ее позиция в сравнении с остальными.

6. Составляется план по уменьшению уязвимости каждой альтернативы через улучшение показателей по наиболее слабым критериям [49].

Метод идеальной точки, также упомянутый выше, заключается в определении некоторой идеальной точки, в которой все критерии достигают своих идеальных значений. Необходимо на множестве всех решений найти точку, максимально приближенную к идеальному значению [46].

1. Определяется обобщенный критерий $a_i = maxF_i(X), i = \overline{1,m}$ так, что a_i является максимально (или минимально) возможным значением по критерию i. Положим $a = (a_1, a_2, \dots, a_m)$, тогда точка a считается идеальной. Такие точки являются оптимальными сразу по всему набору критериев, что означает невозможность получения большего (меньшего) значения ни по одному из критериев.

2. Как правило, точка $a \neq Y_D$. Для всех точек $Y \subseteq Y_D$задается функция, являющаяся евклидовым расстоянием между точками Y и a: $\rho(y,a) = [\sum_{i=1}^{m}(a_i - y_i)^2]^{\frac{1}{2}}$.

3. За обобщенный критерий берут функцию: $f(X) = \sum_{i=1}^{m} \lambda_i [a_i - F_i(X)]^2$, где λ_i- весовые коэффициенты.

4. Тогда задача оптимизации формулируется как: $min \sum_{i=1}^{m} \lambda_i [a_i - F_i(X)]^2$, $min \sum_{i=1}^{m} \lambda_i \left[\frac{a_i - F_i(X)}{F_i^0}\right]^2$ с учетом нормирования, где принцип оптимальности выражается функцией выбора, определяемой близостью к идеальной точке [25].

Метод ЭЛЕКТРА (ELECTRE), также предназначенный для решения многокритериальных задач, предполагает выдвижение гипотезы о том, что в каждой паре альтернатив одна является лучше другой. Для каждой из пары альтернатив затем находится индекс согласия, подтверждающий гипотезу о превосходстве одной альтернативы над другой, и несогласия, опровергающий гипотезу. На основе анализа данных индексов затем выбирается одна или несколько лучших альтернатив. Обобщенный алгоритм может быть описан следующим образом [1]:

1. Пусть даны две альтернативы *A* и *B*, для каждой из которых существуют оценки по *N* критериям. Пусть также для каждого из *N* критериев имеется дискретная шкала численных оценок;

2. Множество критериев разбивается на подмножества, где I^+ – оценки альтернативы *A* лучше, чем альтернативы B; $I^=$ – оценки альтернатив равны; I^- – оценки *B* лучше, чем оценки *A*;

3.Пусть выдвигается гипотеза о том, что альтернатива *A* лучше *B*. Тогда индекс согласия (конкорданса) рассчитывается по формуле: $C(A,B) = \sum_{i \in I^+, I^=} w_i \,/\, \sum_{i=1}^{N} w_i$, где w_i – вес *i*-го критерия, w_i – целое число;

4.Критерии, принадлежащие множеству I^-, выражают несогласие с выдвинутой гипотезой. Для них индекс несогласия (дискорданса) определяется по формуле: $d(A,B) = max_{i \in I^-}(x_i^B - x_i^A)/m_i$, где x_i^B, x_i^A – оценки альтернатив *A* и *B* по *i*-му критерию из подмножества I^-, m_i – длина шкалы *i*-го критерия;

5.Индексы *C* и *d* изменяются в диапазоне [0, 1]. Пусть p является близким к единице, а q – близким к 0. Данные числа выступают пороговыми значениями.

Положим, что альтернатива A превосходит B тогда и только тогда, когда индекс согласия не меньше p: $C(A,B) \geq p$, а индекс несогласия не превосходит q: $d(A,B) \leq q$.

Введенное бинарное отношение позволяет построить граф *G(p,q)*, где направленная дуга означает превосходство одной альтернативы над другой, отсутствие дуги – отсутствие возможности сравнить две альтернативы по данному бинарному отношению;

6. Граф *G(p,q)* обладает свойствами:

6.1. Если $p \leq p'$ и $q \geq q'$, то граф *G(p,q)* содержит граф *G(p′,q′)* в качестве частичного графа;

6.2. Если $p < 1$ и $q > 0$, то граф *G(p,q)* не обязательно является транзитивным (превосходства *A* над *B* и *B* над *C* недостаточно, чтобы заявить о превосходстве *A* над *C*). Дополнительно, в графе могут быть циклы, тогда альтернативы, входящие в них, принято считать эквивалентными;

7. С уменьшением p и увеличением q количество число сравнимых альтернатив увеличивается. В случае, когда альтернатив больше двух, необходимо из всего множества выделить наиболее предпочтительные. В таком случае, если некоторая альтернатива *A* превосходит альтернативу *B* при определенных пороговых значениях, *B* может быть удалена из рассмотрения. Тогда обозначается подмножество *S* оставляемых альтернатив и подмножество $E \setminus S$ исключаемых альтернатив. Подмножество *S* должно удовлетворять условиям:

7.1. Для любой из исключенных альтернатив имеется хотя бы одна, превосходящая ее, среди оставляемых;

7.2. Никакая из оставляемых альтернатив не превосходится другой, также относящейся к подмножеству *S*;

8. Подмножества графа *G(p,q)*, удовлетворяющие обоим условиям, называются ядрами. Если в графе имеются нетранзитивности, то требуется удовлетворение одного, второго, условия [1].

Основной проблемой использования вышеуказанных методов для оценки сложных технических систем является потенциальное несовпадение результатов,

полученных каждым отдельным методом, связанное с их ограничениями. Более того, возникает проблема подбора корректных критериев для их применения совместно с различными способами вычисления интегральных оценок [57]. Это говорит о методах теории принятия решения как о мало подходящих для решения задач голосования.

Тем не менее, вышеупомянутые методы могут применяться для анализа мультиверсий и определения их слабых мест с целью дальнейшего повышения надежности каждой из них на этапах разработки и тестирования.

1.2.2 Классификация

Многие из задач, связанных с принятием решений, попадают под категорию классификации – классической задачи статистического анализа данных. В зависимости от цели задачи, для классификационного деления могут приниматься различные признаки. При этом в основу класса всегда закладывается наиболее важный признак, отвечающий цели классификации. Сам же процесс предполагает разделение и группировку некоторых входных данных на два и более класса в соответствии с выделенными признаками [37].

Задачи данного типа предполагают наличие классификатора, алгоритм которого строится на основании некоторого конечного набора данных (обучающей выборки), где для каждого из объектов заранее известна принадлежность тому или иному классу [53].

В общем виде математическая постановка задачи классификации звучит следующим образом. Имеется множество *M* объектов *ω*, для которых характерны описания [26]:

$$I(\omega) = (x_1(\omega), x_2(\omega), \ldots, x_N(\omega)) .$$

Задан критерий *K(I((ω)))*, позволяющий различать объекты на основании определенного критерия. В соответствии с заданным критерием строится разбиение множества *M* на классы $\Omega_i : M = \cup_{i=0}^{m} \Omega_i$, количество которых

определяется по результатам выполнения классификации. Тогда действие критерия определяется формулой [26]:

$$\forall i,j(i,j=1,\ldots,m) \quad i \neq j \Leftrightarrow K(I(\omega \in \Omega_i)) \neq K(I(\omega \in \Omega_j)).$$

В основе алгоритмов классификации может лежать разнообразие методов, относящихся к статистическому распознаванию образов, нейронным сетям и методам машинного обучения [3].

Одним из наиболее простых и популярных методов классификации, относящихся к категории методов машинного обучения, является метод *k*-ближайших соседей. Он предполагает, что объект принадлежит тому классу, к которому относится большинство его ближайших соседей. Степень родства объектов определяется евклидовым расстоянием между ними, тогда как расположение на плоскости определяется значениями различных критериев [40].

Поскольку критерии могут оцениваться по собственным шкалам, тем самым существенно влияя на точность классификации, необходимо прибегать к нормализации - переходу от абсолютных значений признаков к относительным. Наиболее популярными подходами к нормализации являются:

1. $x_i \equiv \frac{x_i - x_{min}}{x_{max} - x_{min}}$

Осуществляется переход от абсолютных значений признаков к относительным, принимающим значения от 0 до 1 или от 0 до 100 в процентном соотношении [40].

2. $x_i \equiv \frac{x_i - \overline{x}}{s}$

Здесь $\overline{x}$ – выборочное среднее, s – выборочное среднеквадратичное отклонение [40].

Точность классификации зависит не только от правильности выбора критериев, но и от k – количества рассматриваемых ближайших соседей. Подбор оптимальной величины параметра k позволяет минимизировать число неверных ответов. На практике значение параметра часто полагают равным $\sqrt{N}$, где N – число классифицируемых объектов [5].

В общем виде алгоритм определения принадлежности произвольного объекта z при фиксированном количестве соседей k может быть описан следующим образом [51]:

1. Вычисляется расстояние $d(z_i, z)$ от объекта z до каждого из объектов z_i, классовая принадлежность которых известна. При этом наиболее часто используется евклидова мера расстояния, вычисляемая по формуле:

$$d(z_i, z) = (\sum_{i=1}^{q}(z_i^l - z^l)^2)^{\frac{1}{2}},$$

где q – число характеристик объектов z_i и z.

2. Выполняется упорядочение вычисленных расстояний по возрастанию их значений.

3. Выбирается k объектов z_i, наиболее близко расположенных к объекту z.

4. Выявляется классовая принадлежность каждого из k ближайших соседей объекта z. Класс, преобладающий среди k выбранных соседей данного объекта z принимается за класс этого объекта. Для определения наиболее вероятного класса объекта зачастую прибегают к методам голосования, таким как простое невзвешенное голосование и взвешенное голосование. Также могут использоваться и другие алгоритмы голосования, подходящие под ограничения решаемой задачи [51].

Говоря же о статистическом анализе данных, целесообразно выделить достаточно известный метод линейного дискриминантного анализа, позволяющего оценивать различия между классами объектов по нескольким параметрам одновременно. В его основе лежат две статистические процедуры, такие как интерпретация групповых различий и классификация [103].

Основой дискриминантного анализа является гипотеза, что объекты каждого k-го класса представлены реализацией многомерной случайной величины, распределены по нормальному закону $N_m(\mu_k; \Sigma_k)$ со средними μ_kи ковариационной матрицей, где индекс m указывает на размерность признакового пространства [103]:

$$C_k = \frac{1}{n_k - 1} \sum_{i=1}^{n_k} (x_{ik} - \mu_k)^T (x_{ik} - \mu_k \quad)$$

В случае двух классов (в задачах голосования это могут быть классы «Верно» и «Неверно» в соответствии с результатом работы мультиверсии) задача дискриминантного анализа сводится к проведению дополнительной оси *z* таким образом, чтобы проекции точек на неё обеспечивали наилучшую разделяемость на группы (классы). Положение оси *z* задается линейной дискриминантной функцией с весовыми коэффициентами β_j, определяющими вклад каждой исходной переменной x_j [103]:

$$z(x) = \beta\_1\ x\_1 + \beta\_2\ x\_2 + \cdots + \beta\_m\ x\_m.$$

Если предположить, что $C_1 = C_2$, то вектор коэффициентов β_1, … ,β_m линейного дискриминанта $z(x)$ может быть вычислен по формуле $\beta = C^{-1}(\mu_1 - \mu_2)$, где C^{-1}– матрица, обратная ковариационной, μ_k – вектор средних *k*-го класса. Полученная ось совпадает с уравнением прямой, проходящей через центроиды двух групп объектов классов, а обобщенное расстояние Махаланобиса, равное дистанции между ними в многомерном пространстве признаков, оценивается как [103]:

$$D\hat{}2 = \beta(\mu\_1 - \mu\_2).$$

В результате, кроме предположения о нормальности распределения данных в каждом классе, которое на практике выполняется довольно редко, выдвигается еще и более серьезное предположение о статистическом равенстве внутригрупповых матриц дисперсий и корреляций. Если между ними нет серьезных отличий, их объединяют в расчетную ковариационную матрицу. После чего происходит проверка всех гипотез и строится модель, для которой затем необходимо составить функцию вывода показателей оценки ее качества. Построенная функция позволяет оценить эффективность дискриминантной модели прогнозирования по каждому показателю [41].

В ходе анализа может оказаться, что некоторые параметры не являются информативными, в таком случае применяются методы вычисления параметров

качества моделей-претендентов, позволяющие сократить количество показателей, оставив только значимые. В результате сокращения числа переменных до двух становится возможным построение частного двумерного графика, в свою очередь интерпретирующего процесс классификации [41].

Альтернативой статистическому распознаванию образов могут служить нейронные сети, которые являются непараметрическими моделями и не требуют выдвижения предположений о вероятности распределения данных. Они служат компромиссом между параметрическими и метрическими методами и, как следствие, являются универсальными классификаторами [48].

Сами по себе искусственные нейронные сети представляют собой упрощенные модели биологических нейронных сетей. Данная технология позволяет решать трудно формализуемые и неформализуемые задачи, до недавнего времени поддающиеся решению исключительно человеческому мозгу. К таким задачам также относятся распознавание образов, глубокий анализ данных, рисование [72].

Искусственная нейронная сеть представлена набором «нейронов», связанных друг с другом. Каждый нейрон имеет определенное количество входов, куда поступают сигналы, которые суммируются с учетом значимости (веса) каждого входа. Вес каждого узла может быть как положительным, так и отрицательным, и характеризует силу связи между нейронами. Нейроны объединяются в слои, представленные входным слоем, скрытым слоем и выходным слоем [56].

Входной слой нейронов получает данные из «внешнего мира», тогда как выходные нейроны возвращают преобразованные данные, обработанные скрытым слоем нейронов. При этом скрытый слой фактически может состоять из нескольких слоев. Так, обработанные одним слоем данные передаются на следующий слой, при этом каждый отдельный нейрон усиливает или ослабляет сигнал в ходе обработки. Обработка сигналов каждым нейроном происходит по единому алгоритму (функции). В результате, к моменту получения выходного сигнала вероятность ошибочных вычислений минимальна [33].

Из наиболее простых разновидностей нейронных сетей выделяют сети прямого и обратного распространения ошибки. При прямом распространении происходит передача входных значений и получение выходных данных, которые называются прогнозируемым значением [44]. После получения прогнозируемого значения, необходимо вычислить ошибку, для чего производится сравнение прогнозируемого значения с фактическим с использованием функции потери, что является задачей обратного распространения [42]. Происходит вычисление производной от значения ошибки (градиента) по каждому весу в нейросети. Значение градиента вычитается из значения веса, тем самым корректируя соединения нейронов и уменьшая значение ошибки [42].

Для интеллектуального анализа данных задача классификации является одной из ключевых, а алгоритмы и методы для ее реализации достаточно просты. Определенных сетевых архитектур, направленных конкретно на классификацию, не выделяется, однако на практике достаточно популярно применение сетей прямого распространения с многослойными персептронами [29].

Так, на входные нейроны поступает информация о значениях признаков классифицируемого объекта, которые затем распределяются по нейронам первого скрытого слоя нейросети. Размерность задачи изменяется, что повышает точность вычислений, и в случае правильного подбора конфигурации и параметров нейронной сети формируемая на выходе метка (или цифровой код) класса является верной. Однако подход с использованием нейронных сетей имеет и свои недостатки, например, конфигурация сети для правильной классификации данных заранее неизвестна, подбирается экспериментальным путем и, как следствие, может быть очень затратна по времени. Более того, если для классификации используется сложная функция, размерность сети может стать слишком большой, существенно влияя на время работы алгоритма [64].

Это ведет к необходимости предварительной обработки входных данных. Она включает в себя выделение действительно значимых признаков, а также определение их оптимального количества и нормализацию данных в диапазоне от 0 до 1. Так, увеличение числа признаков увеличивает время вычисления и

размерность сети, но повышает точность, а снижение их числа ухудшают различимость [73].

Если же говорить о нормализации, то она позволяет уменьшить разброс значений признаков с разной природой происхождения. Более того, необходимо, чтобы данные были «чистыми» – для этого исключаются противоречия и дубликаты, заполняются пропуски, подавляются выбросы и аномальные значения [73].

Являясь моделью, базирующейся на обучении с учителем, нейронная сеть требуют присвоения метки класса каждому объекту обучающей выборки. При этом для обучения сети как правило используется алгоритм обратного распространения [42].

Таким образом, можно сделать вывод, что методы классификации данных довольно разнообразны и значительно различаются между собой, поэтому определение наиболее подходящего из них в каждом конкретном случае напрямую зависит от поставленной задачи, рассматриваемых исходных данных, возможности построения точной математической модели, исходя из них, и множества других аспектов. И хотя все представленные методы являются трудоемкими в реализации и в контексте МВПО напрямую зависят не только от поставленной задачи, но и от разрабатываемой программной системы, с теоретической точки зрения они могут быть применимы для решения задач голосования и давать точные результаты.

1.2.3 Теория голосования

Еще одним подходом к принятию «правильного» решения является теория голосования, математическое исследование которой берет свое начало в 17 веке [89]. Изначально теория голосования появилась в результате необходимости принятия коллективного решения, которое учитывало бы мнение большинства и являлось бы наиболее объективным.

Вычислительная теория голосования является более узким направлением вычислительного социального выбора (Computational Social Choice – CSC). Вычислительный социальный выбор включает в себя исследование проектирования и анализа методов коллективного принятия решений и является сферой деятельности математиков и экономистов [89].

Автоматизированные же групповые решения являются важным аспектом в задачах искусственного интеллекта, где независимые агенты должны прийти к общему решению, для определения которого используются процедуры голосования [90].

В широком смысле математическая модель голосования включает в себя набор $N = \{1, \ldots, n\}$ агентов, которым необходимо прийти к коллективному решению в ходе рассмотрения числа $A, |A| = m$ альтернатив. Альтернативы могут выступать кандидатами на политических выборах, однако в вычислительных условиях это, как правило, какие-либо убеждения, совместные планы, рекомендации или другие математически описываемые вопросы. Каждый агент производит оценку всех альтернатив, а конечный исход определяется функцией «социального выбора» [86].

Наиболее распространенной является функция относительного большинства, используемая во многих политических выборах, где каждый агент отдает условный балл за наилучшую альтернативу, в результате чего побеждает альтернатива с наибольшим числом баллов. [89].

Среди классических методов теории голосования, помимо вышеупомянутого метода относительного большинства, существуют и другие. К ним также относятся метод абсолютного большинства, голосование с последовательным исключением, голосование по принципу Кондорсе, голосование по правилу Борда, голосование при помощи подсчета очков, голосования по правилам Компленда и Симпсона и другие [99].

Сами по себе процедуры голосования представлены множеством форм и могут использоваться различным образом в зависимости от системы, в которой

находят применение. Как следствие, классические методы голосования были адаптированы и для процесса оценки правильности выходов мультиверсий [98].

Рассматривая совокупность методов голосования в мультиверсионном программном обеспечении укрупнённо, можно отметить, что все они могут быть разделены на алгоритмы, основанные на сравнении выходных данных и алгоритмы с принятием решения вне зависимости от схожести выходных данных [35].

Первая группа имеет свои особенности применения. Так, в случае получения одинаковых ошибочных результатов разными мультиверсиями ошибки считаются идентичными, и выдвигается предположение о том, что вероятность их пренебрежимо мала; в результате работы алгоритмов всё множество выходных данных версий делится на подмножество «корректных» выходов и подмножество «некорректных», а выбор множества «корректных» ответов основан на сравнении всего множества ответов; при сравнении значений выходов мультиверсий используется понятие эквивалентности выходов; в качестве корректного результата положено принимать набор эквивалентных выходов, выбираемый из всего множества выходов версий; выбор корректного набора выходных данных осуществляется либо с использованием матрицы согласования, либо с использованием подмножеств согласных версий [35].

К алгоритмам, основанных на сравнении выходных данных принято относить NVP-MV (N-version programming with majority voting, алгоритм голосования абсолютным большинством, АГАБ), NVP-CV (N-version programming with consensus voting, алгоритм голосования согласованным большинством или АГСБ), Fuzzy CV (Fuzzy consensus voting, нечёткий алгоритм голосования согласованным большинством), Fuzzy MV (Fuzzy majority voting, нечёткий алгоритм голосования абсолютным большинством), minMV (алгоритм голосования абсолютным большинством с минимизацией), minCV (алгоритм голосования согласованным большинством с минимизацией), FMV (Formalized majority voting, формализованный алгоритм голосования абсолютным большинством) и FCV (Formalized consensus voting, формализованный алгоритм голосования согласованным большинством) [35].

В классификации, представленной в [35], также предполагает разбиение перечисленных алгоритмов на формализованные и неформализованные.

Так, в неформализованных алгоритмах (NVP-MV, NVP-CV, Fuzzy CV, Fuzzy MV, minMV, minCV) происходит разбиение множества выходных данных мультиверсий на непересекающиеся подмножества. В формализованных же (FMV, FCV) подмножества могут пересекаться, то есть выходы могут принадлежать более чем одному подмножеству [58].

Вторая же группа отличается тем, что результат работы алгоритмов представлен только одним значением той же структуры, что и у выходов мультиверсий, и не зависит от их схожести. К данной категории относятся алгоритмы MLV (Maximum likelihood voting, «максимально вероятное» голосование) и усредненное голосование [82].

Алгоритмы голосования абсолютным и относительным большинством относятся к категории «классических» для задач голосования в МВПО и обладают некоторыми отличительными особенностями [35].

Так, основой принятия решения о выборе корректных выходов служит матрица согласования (булева матрица размерности N x N, где N – число версий), которая отражает эквивалентность каждого выхода другим выходам. Каждый элемент вычисляется по принципу:

$$r_{ij} = \begin{cases} 1, \text{если } |x_i - x_j| \leq \varepsilon, \\ 0, \text{если } |x_i - x_j| > \varepsilon, \end{cases}$$

где r_{ij} – элемент матрицы согласования в i-той строке и j-ом столбце, x_iи x_j – выходы, проверяемые на эквивалентность [35].

К матрице согласования предъявляется следующее дополнительное требование: на матрице согласования R должно быть выполнено отношение эквивалентности [35]:

- (рефлексивность) $r_{ij} = 1 \ \forall i,$
- (симметричность) $r_{ij} = r_{ji} \ \forall i \neq j$
- (транзитивность) если $r_{ik} = 1$и $r_{ki} = 1$, то $r_{ij} = 1 \ \forall i, j$

Выполнение такого требования необходимо для решения проблемы несовместимости разбиения.

В случае невыполнимости отношения эквивалентности к матрице согласования должны применяться булевы композиции r_{ij}, и происходит это до тех пор, пока отношение не будет удовлетворено. Отношение, на котором выполнены только свойства рефлексивности и симметричности, называется допустимым отношением. В таком случае матрица согласования, удовлетворяющая отношению, будет получена не более чем за *N*-1 булевых композиций [35].

Чтобы на матрице согласования *R* выполнялось отношение эквивалентности необходимо последовательное выполнение булевых композиций *R* самой с собой по следующему принципу:

$$C = A^{\circ}B, \text{где } c_{ij} = \bigoplus_{k=1}^{N} (a_{ik} \otimes b_{ik}),$$

где все элементы матриц *A* и *B* равны 0 или 1, $\bigoplus$ – логическое «или», $\otimes$ – логическое «и» [35].

$$E = R^1 \cup R^2 \cup R^3 \cup \ldots \cup R^Q \ (1 \leq Q \leq N-1),$$

где *E* – матрица согласования, на которой выполнено отношение эквивалентности, *Q* – количество последовательных булевых композиций, *N* – число версий, R^1=R, R^2 = R○R, R^3 = R○R○R, …[35]

Целесообразно рассмотреть схему алгоритма голосования абсолютным большинством. Пусть дан некоторый модуль, реализованный *N* версиями, а $x_1, x_2, \ldots, x_N$ – конкретные выходные значения, выданные каждой из *N* версий [79]. Положим заданным допустимое отклонение *e*. Тогда:

1. Строится матрица согласования *R*;
2. Выполняется проверка отношения эквивалентности на матрице согласования. В случае его невыполнения, матрицу согласования необходимо

изменить с использованием булевых композиций – перейти к шагу 3. Иначе – к шагу 4;

3. Булевы композиции выполняются до тех пор, пока отношение эквивалентности не будет удовлетворено;

4. Определяется набор корректных выходов. Рассматривается полученная матрица согласования, в каждой строке которой подсчитывается количество единиц. Пусть количество единиц в *i*-той строке – Y_i. Если существует такая строка i, для которой выполнено $Y_i \geq \left[\frac{N+1}{2}\right]$, то набор корректных версий формируется из тех версий, которым соответствуют единицы в строке *i* [79].

Алгоритм голосования согласованным большинством несколько отличается. Пусть по аналогии с предыдущим алгоритмом дан некоторый модуль, реализованный N версиями, где $x_1, x_2, \ldots, x_N$ – конкретные выходные значения, которые выдают версии [79]. Пусть также задано допустимое отклонение e. Тогда:

1. Строится матрица согласования R.

2. Происходит проверка эквивалентности на матрице согласования. Если отношение эквивалентности не выполнено, то матрицу согласования необходимо изменить с использованием булевых композиций – перейти к шагу 3. Иначе – к шагу 4.

3. Булевы композиции выполняются до тех пор, пока отношение эквивалентности не будет удовлетворено.

4. Определяется набор корректных выходов. В каждой строке матрицы согласования подсчитывается количество единиц. Пусть количество единиц в *i*-той строке – Y_i. Далее выбирается строка, в которой Y_i максимально. Набор корректных версий формируется из тех версий, которым соответствуют единицы в строке i. Если матрица согласования содержит более одной строки, в которых количество единиц максимально, то строка выбирается случайным образом [79].

Алгоритм голосования согласованным большинством выдает результат даже в том случае, если «согласных» версий нет – тогда возвращается выход, выбранный случайным образом [79].

Несмотря на популярность методов голосования для оценки выходов мультиверсий, они также не являются универсальными и сильно зависят от специфики программного обеспечения, формата выходов и особенностей среды исполнения мультиверсионного ПО. Более того, данные алгоритмы не предлагают приемлемого решения для случаев существования равного количества голосов за различные значения выходов мультиверсий, что не позволяет полностью полагаться на их корректность.

1.2.4 Функциональные методы голосования

В ходе выполнения мультиверсионного программного модуля может сложиться ситуация, когда выходы всех мультиверсий являются ошибочными. Это исключает возможность применения методов теории голосования, и требует рассмотрения альтернативных алгоритмов – ими служат функциональные методы голосования. Эти алгоритмы обычно применяются в случаях, когда произвести прямое сравнение результатов мультиверсий затруднительно [30]. Например, когда мультиверсионная система используется для поиска оптимальных направлений. В такой ситуации напрямую сравнивать векторы не имеет смысла, так как они как минимум могут оказаться разной длины. Так, результатом работы всего модуля принимается результат обработки всех выходов версий некоторой функцией [50].

Наиболее распространенным примером такого алгоритма является медианное голосование. Суть метода заключается в нахождении медианы всех выходов мультиверсий, которая и становится результатом работы модуля. При этом считается, что все версии равноправны и в равной степени влияют на выходной результат модуля [50].

В случае, если версии модуля могут в различной степени влиять на выход системы, то применяется алгоритм взвешенного медианного голосования. В этом случае необходимо учитывать значения весовых коэффициентов α_i [50]:

$$r = \frac{1}{n}\sum_{i=1}^{n} \alpha_i \cdot x_i.$$

И хотя данный метод является наиболее популярным при ошибочных значениях всех версий, допустимо также применять другие подходы к усреднению, используемые в математической статистике. Так, если веса всех версий одинаковы, то есть одинаково их влияние на выход системы, возможно использование функции средней гармонической. Тогда вычисление будет осуществляться по формуле [6]:

$$r = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}.$$

Иначе целесообразно вычисление средней гармонической взвешенной [6]:

$$r = \frac{\sum_{i=1}^{n} \alpha_i}{\sum_{i=1}^{n} \frac{1}{x_i}\alpha_i}.$$

Еще одним способом усреднения значений выходов является средняя геометрическая. И хотя в статистике она наиболее часто применяется для определения средних темпов роста, ее использование также возможно для вычисления среднего между максимальным и минимальным значениями признака (выходов). Так, формула средней геометрической в качестве функционального метода голосования приобретает вид [6]:

$$r = \sqrt[n]{nx_i},$$

а взвешенная средняя геометрическая [6]:

$$r = \sqrt[\sum_{i=1}^{n} \alpha_i]{nx_i^{\alpha_i}}.$$

Для измерения вариации выходов в совокупности также допустимо использование функции средней квадратичной, которая в случае равенства весовых коэффициентов имеет вид [52]:

$$r = \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}},$$

а в случае различия весовых коэффициентов [52]:

$$r = \sqrt{\frac{\sum_{i=1}^{n} x_i^2 \alpha_i}{\sum_{i=1}^{n} \alpha_i}}$$

Еще одной альтернативой перечисленным выше методам функционального голосования может служить функция определения моды. Для дискретного ряда значений модой является число, наиболее часто встречающееся на множестве выходных значений. Однако в случаях, когда выходные значения версий оказываются неидентичными (или не полностью идентичными), целесообразно производить расчет моды в интервальном вариационном ряду. Для этого необходимо выделить интервалы на основании полученного множества значений. Интервал, в который входит наибольшее число значений, используется для поиска моды. При этом формула имеет вид [45]:

$$r = x_{\text{м}0} + i_{\text{м}0} \frac{(f_{\text{м}0} - f_{\text{м}0-1})}{(f_{\text{м}0} - f_{\text{м}0-1}) + (f_{\text{м}0} - f_{\text{м}0+1})},$$

где $x_{\text{м}0}$ – нижняя граница модального интервала, $i_{\text{м}0}$ – величина (разница между верхней и нижней границей) модального интервала, $f_{\text{м}0}$ – частота модального интервала, $f_{\text{м}0-1}$ и $f_{\text{м}0+1}$ – частоты интервалов, предшествующего и следующего за модальным соответственно [45].

И хотя данные методы являются базовыми, исходя из основ математической статистики, можно сделать вывод, что они все направлены на усреднение выходных значений мультиверсий, и, теоретически, область функциональных методов голосования ими не ограничивается. Это означает, что для вычисления наиболее правильного значения среди выходов мультиверсий может использоваться любая функция, удовлетворяющая требованиям мультиверсионной среды исполнения, а также специфике программного модуля.

С одной стороны, такой подход добавляет гибкости функциональным методам голосования, делая их неплохой альтернативой классическим алгоритмам. С другой стороны – никакой речи об универсальности данного подхода идти не может, поскольку одна и та же функция может давать отличные результаты для одного модуля, и быть совершенно не применимой к другому.

1.2.5 Сравнительный анализ

Проанализировав все представленные в главе методы, отражающие основную специфику рассматриваемых подходов, можно сказать, что на первый взгляд все они в той или иной мере подходят для выбора верного ответа среди всех выходов мультиверсий в результате работы МВПО. Однако при более детальном рассмотрении становится очевидно, что представленные алгоритмы либо являются слишком общими, либо возможность их применения сильно зависит от поставленной задачи.

В связи с большим разнообразием задач, потенциально решаемых МВПО, невозможно выделить универсальный подход на основании рассмотренных альтернатив. Так, обращаясь к методам теории принятия решений, можно сделать вывод об их специфичности. Данные алгоритмы с большей вероятностью подошли бы для оценки качества работы отдельной мультиверсии и определения ее уязвимостей, нежели для сравнения результатов работы всех версий. Представленные алгоритмы принятия решений также могут служить подспорьем для определения априорного показателя надежности каждой мультиверсии по ряду признаков. Однако их использование в качестве алгоритмов голосования не представляется возможным.

Методы классификации по сравнению с ними подходят в большей степени: они предполагают наличие классов, которые в контексте МВПО могут быть представлены множествами «верных» и «неверных» выходов мультиверсий. Как говорилось в пункте 2 данной главы, предполагается наличие классификатора, определяющего принадлежность полученных значений к тому или иному классу в соответствии со значениями некоторых параметров или метрик. При этом метрические алгоритмы, такие как *k*-ближайших соседей, предполагают наличие обучающей выборки, как и в случае применения искусственных нейронных сетей, что существенно затрудняет использование указанных методов для задач голосования в рамках МВПО.

Данные трудности обусловлены тем, что мультиверсии во время работы зачастую решают именно вычислительные задачи. Более того, результат работы программы отличается в каждой итерации. Тем не менее, сбор достаточной для обучения выборки, теоретически, возможен, однако это сопровождается значительными временными и трудовыми затратами. Дополнительную сложность привносит необходимость повторной генерации выборки в случае изменений программных алгоритмов, что опровергает универсальность данных подходов и подтверждает нецелесообразность их использования.

Немаловажно также и то, что при оценке результатов работы мультиверсий не используются критерии, необходимые для решения задачи классификации параметрическими методами, поэтому единственной опорой для принятия решения остаются значения выходов мультиверсий.

Функциональные методы голосования, хотя и обладают гибкостью и практической эффективностью, на практике находят применение относительно редко. Это связано с тем, что алгоритмы, такие как медианное голосование, используются только в том случае, если все выходы мультиверсий считаются заведомо неверными, а также в случае, если выходы представлены в форме, неподходящей для их сравнения. Более того, как упоминалось выше, подбор функции для обработки результатов вычислений мультиверсионного модуля в значительной степени зависит от предназначения программного обеспечения, ожидаемого результата работы программы, а также особенностей мультиверсионной среды исполнения. Это делает функциональные методы голосования не универсальным инструментом, а средством, которое требует экспертного контроля, а также доработок в случае изменений в алгоритмах мультиверсий.

На контрасте с требованиями, предъявляемыми теорией принятия решений, классификацией и функциональными методами, становится очевидно, что алгоритмы голосования, описанные выше, наилучшим образом подходят для решения задачи выбора одного значения из множества выходов мультиверсий. Данные методы позволяют выбрать одну наиболее приемлемую для большинства

агентов альтернативу из получившегося множества вариантов, тем самым принимая ее за наиболее верную. Это объясняет активное применение данных методов на практике [67].

Алгоритмы теории голосования популярны за счет своей простоты и эффективности. И хотя применение алгоритмов классификации «в чистом виде» в качестве функций голосования нецелесообразно, на практике иногда принято совмещать с ними методы голосования. Примером таких обстоятельств может служить ситуация, когда выходы каких-либо двух мультиверсий считаются одинаково корректными на основании равенства количества голосов, отданных агентами за эти значения. В таком случае необходимо провести классификацию по набору признаков, что позволит принять окончательное решение.

В случаях, когда две или более групп мультиверсий одинаковой численности выдали разные результаты (например, две группы по две версии выдали разные результаты, и одна версия – третья – выдала свой собственный результат, который не подлежит рассмотрению). В классическом варианте алгоритма голосования согласованным большинством один из этих вариантов будет принят корректным случайным образом. В приведенном случае это снижает вероятность правильного ответа до 50%, а с увеличением числа «равноправных» значений эта вероятность уменьшается соответствующим образом. Не поддается сомнению, что данный подход значительно снижает надежность МВПО, не позволяя в полной мере полагаться на правильность принятого решения.

Один из способов решения данной проблемы предложен в работе [36] и представляет собой модификацию алгоритма голосования согласованным большинством. Так, в случае если в матрице согласования существует несколько строк с максимальным количеством единиц, набор корректных результатов определяется по позициям единиц в строке, и работа алгоритма завершается. Если же существует несколько наборов различных строк, необходимо проведение дополнительной проверки - вычисления надежности каждого набора, состоящего из соответствующих единицам строки версий. Вычисление производится по формуле [36]:

$$R_{k,\dots,l} = l - \prod_{i=k}^{l}(l - p_i),$$

где p_i - надежность i-ой версии. По результатам вычисления надежности каждого из наборов версий наиболее правильным принимается тот, надежность которого максимальна [36].

И хотя этот подход более эффективен по сравнению с классическим алгоритмом голосования согласованным большинством, он также обладает своими недостатками. Так, необходимо знать априорное значение надежности всех исполняемых версий. Соответственно, не учитываются особенности исполнения этих версий в мультиверсионной среде [27].

Более того, решение принимается на основании одного критерия, что не всегда является достаточным. Например, может возникнуть ситуация равенства надежности двух наборов мультиверсий, что снова приводит к необходимости выбора одного из них в качестве наиболее верного случайным образом.

Все это обуславливает необходимость поиска такого алгоритма голосования, который в случае существования равнозначных вариантов ответов мультиверсий мог бы использовать более одного критерия (или один составной) для принятия решения. При этом значения хотя бы некоторых критериев должны определяться апостериорно для учета особенностей исполнения мультиверсий в среде.

Таким образом был получен вывод, что введение дополнительных критериев для оценки уровня разнообразия мультиверсий с целью повышения надёжности алгоритмов голосования в условиях неопределённости является актуальным.

1.3 1.3 Выводы

В главе 1 проведён анализ моделей представления мультиверсий и мультиверсионных программных модулей, в ходе которого был сделан вывод об отсутствии формального аппарата представления, как мультиверсий, так и мультиверсионных модулей, позволяющего определить их меру различия.

На основании проведённого сравнительного анализа был разработан дизайн исследования, определяющий ряд необходимых направлений работы для решения поставленных проблем.

В результате проведенного анализа предметной области мультиверсионного программного обеспечения были систематизированы и обобщены результаты изучения существующих моделей представления мультиверсий и методов выбора верного ответа мультиверсионного модуля.

Обзор предметной области показал, что в современной теории МВПО уже сформирован круг широко применяющихся на практике моделей представления мультиверсий, включающий следующие модели: модель «Черного ящика», модель «Белого ящика», модели на основе конечных автоматов и модели, учитывающие распределение входных данных. Каждая из этих моделей обладает специфической областью применения и ограничениями, определяющими её эффективность в конкретных условиях эксплуатации.

Детальный анализ показал, что существующие модели представления мультиверсий, несмотря на их разнообразие и развитый математический аппарат, не обеспечивают поддержку базового принципа разнообразия МВПО – количественной оценки диверсифицированности программных версий.

В ходе исследования методов выбора итогового решения были рассмотрены три основных направления: теория принятия решений, методы классификации и теория голосования. Анализ показал, что:

– Методы теории принятия решений требуют предварительного обучения или экспертной оценки;

– Классификационные методы ограничены необходимостью наличия обучающей выборки;

– Алгоритмы теории голосования остаются основным инструментом, но демонстрируют недостаточную эффективность в ситуациях в ситуациях неопределённости (выбор может производиться случайным образом).

На основании проведенного анализа выявлены следующие проблемы:

1. Отсутствие единого формализованного механизма для количественной оценки меры различия между версиями;

2. Недостаточная эффективность существующих алгоритмов голосования в условиях неопределенности;

3. Отсутствие комплексной методологии оценки разнообразия версий для минимизации вероятности совместных сбоев.

Сформулированные выводы определяют направления дальнейших исследований:

1. Разработка новой модели представления мультиверсий, интегрирующей механизм оценки диверсифицированности;

2. Создание или модификация алгоритмов голосования, использующих комплексную оценку разнообразия версий.

3. Формирование способов оценки меры диверсифицированности мультиверсионных модулей на основе количественной меры различия между версиями.

Таким образом, проведенное исследование выявило критический разрыв в теории и практике МВПО, связанный с отсутствием эффективных инструментов оценки диверсифицированности мультиверсий, что делает актуальной задачу разработки новых методологических подходов в данной области.

2 Комбинированная модель мультиверсии и графовая модель мультиверсионного модуля

Как уже было сказано выше, для обеспечения высокого уровня надёжности программного обеспечения мультиверсионных программных систем версии модулей должны быть диверсифицированы, хотя бы на одном из приведённых ниже уровней диверсификации:

1. уровень языков программирования;
2. уровень алгоритмов;
3. уровень средств разработки;
4. уровень средств тестирования [28].

Мультиверсии программных модулей мультиверсионного программного обеспечения считаются диверсифицированными, по меньшей мере, на одном из уровней, если они были созданы различными командами разработчиков, изолированными друг от друга. Однако это может быть не всегда так. Поскольку до сих пор мера различия версий программных модулей мультиверсионного программного обеспечения не имела возможности быть выражена конкретным числовым значением. В связи с чем, в работе предлагается ввести метрику диверсифицированности мультиверсий программных модулей.

В данной главе описывается комбинированная модель мультиверсии, основанная на метриках диверсифицированности.

2.1 Метрика диверсифицированности версий на уровне алгоритмов

2.1.1 Общее описание

Так как зачастую версии модулей мультиверсионного программного обеспечения выполняют какие-либо вычислительные операции, выдвинута

гипотеза о том, что определение меры различия версий на уровне алгоритмов может осуществляться путём сравнения изменений данных, обрабатываемых модулем. Хотя модули могут оказывать и побочные эффекты на среду исполнения, к которым относятся такие операции как изменение файлов или объектов среды исполнения, взаимодействие с датчиками различного рода и т.д. [65], было решено пренебречь наличием подобных побочных эффектов при определении меры различия версий на уровне алгоритмов, так как описанные выше возможные варианты взаимодействия ПО со средой исполнения реализуются путём применения специализированных средств разработки и слабо зависят от применяемых алгоритмов.

Если говорить о выдвинутой гипотезе более простым языком, то было принято решение подать идентичные наборы данных на вход различным версиям одного модуля и отслеживать их изменение в процессе работы каждой версии. Таким образом, становится возможным не только учёт синхронности изменений входных параметров, но и отслеживание последовательных изменений, производимых программами во время выполнения [9].

Зачастую поведение вычислительных математических алгоритмов не инвариантно, то есть один и тот же алгоритм может вести себя по-разному при подаче ему на вход различных по какому-либо признаку данных. То есть для обеспечения чистоты и корректности эксперимента необходимо было обеспечить систематический обход различных путей выполнения каждого сравниваемого программно реализованного алгоритма.

Для удовлетворения данного требования принято решение об использовании того же подхода, что используется при автоматизированной генерации модульных тестов программного обеспечения. Область мультиверсионного ПО является новой для применения подхода автоматизированной генерации тестов. Благодаря его применению становится возможным прохождение всех ветвей кода каждого алгоритма.

Результаты работы по созданию генератора тестов, проделанной в данном эксперименте, описаны в статье [9].

Для введения меры различия (метрики) между алгоритмами предметная область была формализована следующим образом:

- Набор (массив) обрабатываемых данных на каждом шаге работы алгоритма – это точка в многомерном пространстве;
- Мерность пространства определяется наибольшей длиной массива входных данных за всё время работы алгоритма;
- Если на каком-либо шаге алгоритма количество элементов в массиве входных данных меньше мерности пространства, то недостающие координаты измерений заполняются нулями;
- Каждые 2 точки в многомерном пространстве образуют вектор;
- Совокупность изменений значений входного массива образует трассу;
- Если в течение нескольких шагов алгоритма значение входного набора данных остаётся неизменным, то точка остаётся на месте, благодаря чему в геометрическом смысле не искажается длина трассы алгоритма [9].

Как уже было сказано выше, в общем случае фиксируемый на каждом шаге (или в каждой контрольной точке) набор данных может содержать и различное количество значений, поэтому для определения описываемой метрики в качестве мерности метрического пространства принимается наибольшее количество элементов, зафиксированных на каком-либо шаге любого из алгоритмов, а недостающие координаты в других наборах данных считаются заполненными нулями [14].

Это справедливо, поскольку если мерность пространства одной точки больше, чем у другой, это значит, что все точки пространства с меньшей мерностью имеют одну и ту же координату в пространстве с большей мерностью. Так, например, если переводить две точки на плоскости в трёхмерное пространство, может быть добавлено нулевое значение третьей координаты в Декартовом пространстве, поскольку они в действительности лежат в одной плоскости и удалены по оси Z на одно и то же расстояние от начала координат по вертикали [14]. Для однозначности и обеспечения простоты понимания это расстояние определяется как нулевое.

Метрика различия мультиверсий в мультиверсионном программном обеспечении на уровне алгоритмов позволяет определять следующие показатели для описания полученных трасс:

1. отношение мерности пространства точек к количеству точек (узлов) в трассе, то есть условная скорость прохождения трассы:

$\upsilon = \frac{|S|}{|steps|}$, где где |S| – мерность пространства точки, |steps| – количество точек в трассе;

2. отношение количества общих отрезков у двух трасс к количеству ребер (отрезков) одной из трасс:

$S(V) = \frac{|SV|}{|V|}$, где |V| – количество рёбер трассы 1, |SV| – количество рёбер трассы 1, совпадающих с рёбрами трассы 2;

3. длина трасс (сумма расстояний между точками в Евклидовом пространстве):

$l = \sum_{i=1}^{|steps|-1} \sqrt{\sum_{k=1}^{n}(p_i^k - p_{i+1}^k)^2}$, где l – длина всей трассы, $|steps|$ – количество точек в трассе, p_i – текущая точка трассы, p_{i+1} – следующая после текущей точка трассы, n – мерность точек, k – индекс измерения соответствующей точки;

4. отношение длины прямого пути от начальной до конечной точки к длине всей трассы:

$I = \frac{l_{forward}}{l}$, где $l_{forward}$ – длина прямого пути от первой до последней точки трассы, l – длина трассы [9];

5. отношение количества общих точек у двух трасс (по аналогии с пунктом 2) к количеству точек в одной из трасс:

$S(E) = \frac{|SE|}{|E|}$, где $|E|$ – количество точек трассы 1, $|SV|$ – количество точек трассы 1, совпадающих с точками трассы 2;

6. количество точек (узлов) в трассе;

7. количество отрезков (рёбер) в трассе;

8. длина совпадающих отрезков двух трасс;

9. отношение длины совпадающих отрезков двух трасс к длине одной из них: $S(Vl) = \frac{l_{CV}}{l_i}$, l_{CV} - суммарная длина совпадающих рёбер трасс 1 и 2, l_i– длина i-й трассы, $i = 1,2$.

Показатели 1 (условная скорость прохождения трассы), 3 (длина трассы), 4 (отношение длины прямого пути от начальной до конечной точки к длине всей трассы), 6 (количество точек в трассе) и 7 (количество отрезков в трассе) определяют индивидуальные характеристики трасс, которые могут относиться лишь к каждой конкретной трассе в отдельности. Они могут быть применимы для сравнения трасс алгоритмов, однако в случае наличия различий в них (трассах) данные показатели не смогут определить, насколько велики указанные различия.

Показатели 2 (отношение количества общих отрезков у двух сравниваемых трасс к количеству ребер одной из этих трасс), 5 (отношение количества общих точек у двух трасс к количеству точек в одной из трасс), 8 (длина совпадающих отрезков двух трасс) и 9 (отношение длины совпадающих отрезков двух трасс к длине одной из них) определяют меру схожести двух трасс, причём показатель 8 (длина совпадающих отрезков у двух трасс) является абсолютным. Если использовать его и только его для сравнения трасс, нельзя в точности утверждать, насколько в действительности схожи трассы, как в случае с показателями 1, 3, 4, 6, 7.

Показатели 2, 5 и 9 при этом отражают именно степень схожести двух сравниваемых трасс друг относительно друга. В случае нахождения такого отношения значения данных показателей будут лежать в промежутке от 0 до 1, благодаря чему, их легко перевести в процентный показатель, численно отражающий меру схожести двух трасс. Для нахождения меры различия полученный показатель схожести необходимо вычесть или из единицы:

$$D_i = 1 - S_i, \tag{2.1}$$

где D_i – мера диверсифицированности алгоритмов по i-му показателю, S_i – мера схожести алгоритмов, определяемая на основании i-го показателя трасс алгоритмов из набора: $\{S(V), S(E), S(Vl)\}$.

Несмотря на то, что эти показатели подходят для решения прикладной задачи нахождения клонов алгоритмов в конкретном наборе мультиверсий конкретного программного модуля [11], для определения полноценной метрики различия алгоритмов в общем виде их всё-таки недостаточно.

Пусть расстояние между точками трасс будет определяться на множестве X. Для того чтобы можно было говорить о наличии метрического пространства, двум элементам a и b множества X должно быть поставлено в соответствие неотрицательное число $\rho(a, b)$, называемое «расстоянием» между ними [4]. При этом должны выполняться три следующих условия (аксиомы):

1. $\rho(a, b) = 0$ тогда и только тогда, когда $a = b$;
2. $\rho(a, b) = \rho(b, a)$ для любых двух элементов a и b из множества X – «аксиома симметрии»;
3. $\rho(a, c) \leq \rho(a, b) + \rho(b, c)$ для любых трёх элементов a, b и c из множества X – «аксиома треугольника» [4].

Таким образом, можно отметить, что предложенная в данной главе метрика соответствует всем приведённым выше аксиомам, кроме второй, поэтому многомерное пространство X, в котором она определена, не является метрическим пространством. Поскольку при использовании показателей 2, 5 и 9 аксиома симметрии нарушается, так как длина, количество отрезков (рёбер) и количество точек (узлов) у двух сравниваемых трасс может быть различным, поэтому знаменатель у всех этих показателей будет различаться, в зависимости от порядка, в котором мы сравниваем трассы (когда сравнивается трасса 1 с трассой 2 и с трасса 2 с трассой 1).

Таким образом, для определения полноценной метрики диверсифицированности алгоритмов необходимо определить точку начала координат, чтобы находить удалённость каждого показателя трассы в метрическом пространстве относительно начала координат, а не относительно аналогичных

показателей иной трассы. Данное обстоятельство обуславливает необходимость определения такой трассы гипотетического алгоритма, которая имела бы что-то общее с каждой из сгенерированных трасс алгоритмов, чтобы каждую из трасс алгоритмов можно было сравнивать с ней.

В качестве такой трассы, с которой будет сравниваться каждая из трасс алгоритмов мультиверсий, было решено принять минимальное остовное дерево [2], поскольку для его построения по имеющимся точкам разработано достаточно большое количество численных и аналитических методов, а также их программных реализаций. Такое дерево будет содержать все точки всех трасс исследуемых алгоритмов. Благодаря этому, показатели трасс 2, 5 и 9 могут быть применены не для сравнения трасс двух конкретных алгоритмов, а для сравнения каждой трассы алгоритма из набора мультиверсий с минимальным деревом.

Таким образом при использовании минимального дерева для сравнения с ним трасс алгоритмов, формулы показателей 2, 5 и 9, соответственно, приобретают следующий вид:

2. отношение количества общих отрезков (рёбер) у сравниваемой трассы с минимальным деревом к количеству ребер в минимальном дереве:

$S(V) = \frac{|SV|}{|V_{ST}|}$, где $|V_{ST}|$ – количество рёбер в минимальном дереве, $|SV|$ – количество рёбер трассы, совпадающих с рёбрами минимального дерева;

5. отношение количества общих точек (узлов) у сравниваемой трассы и минимальным деревом к количеству узлов в минимальном остовном дереве:

$S(E) = \frac{|SE|}{|E_{ST}|}$, где $|E_{ST}|$ – количество точек в минимальном остовном дереве, $|SE|$ – количество точек сравниваемой трассы, совпадающих с точками в минимальном дереве;

9. отношение длины совпадающих отрезков у сравниваемой трассы и минимального дерева к длине минимального дерева:

$S(Vl) = \frac{l_{CV}}{l_{ST}}$, l_{CV} – суммарная длина совпадающих рёбер трассы, сравниваемой с минимальным деревом, l_{ST} – длина минимального дерева.

Теперь, применяя измененные показатели, становится возможным определить удалённость трассы от минимального дерева, как от начала координат, сразу по трём координатам. Таким образом, может быть определено трёхмерное метрическое пространство, в котором соблюдаются все аксиомы метрических пространств, в том числе и аксиома симметрии [4]. Соответственно, в таком пространстве уже может быть определена метрика схожести и диверсифицированности алгоритмов. Кроме того, в данное метрическое пространство могут быть добавлены и остальные, перечисленные выше показатели, в качестве дополнительных координат, по которым также может производиться сравнение. Таким образом, получается девятимерное метрическое пространство, в котором мера различия алгоритмов может быть выражена численно.

При его применении на практике в представленном виде могут возникать неточности в случаях, когда сравниваемые алгоритмы имеют трассы одинаковой длины и с одинаковым количеством точек (см. рисунок 2).

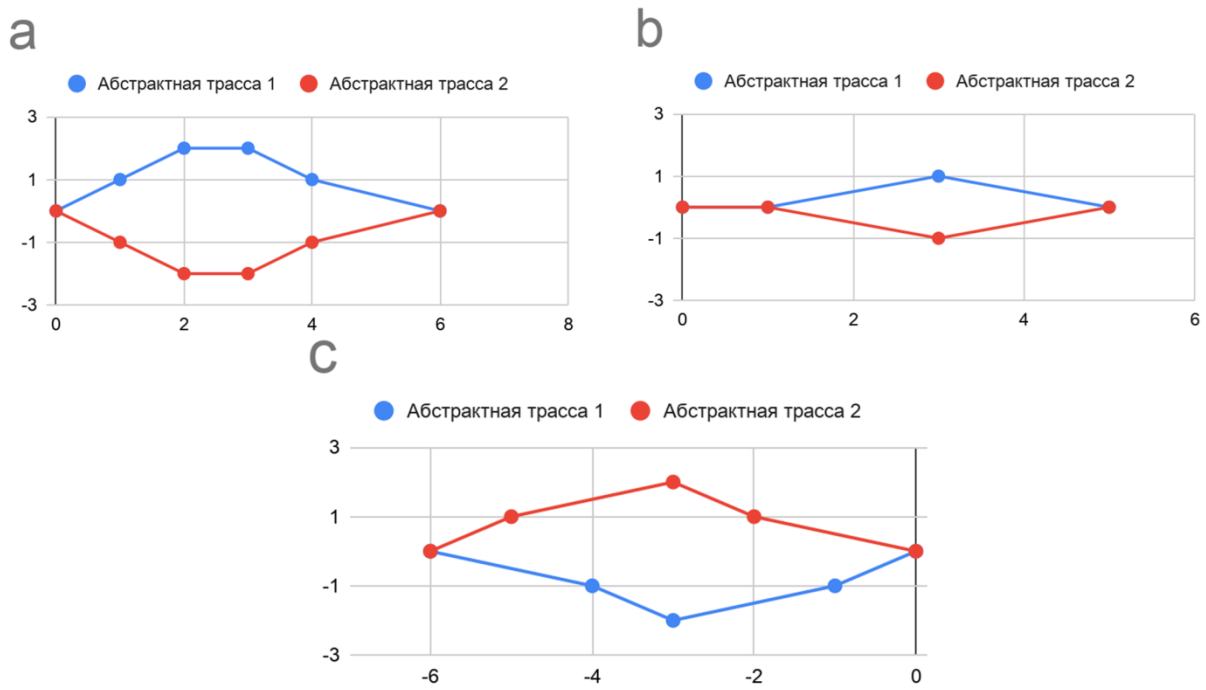

Рисунок 2 – Трассы гипотетических алгоритмов в двумерном пространстве, которые схожи по всем показателям, но фактически не являются клонами

Так, например, на рисунках 2a, 2b и 2c приведены абстрактные гипотетические трассы прохождения алгоритмов в двумерном пространстве, которые по всем указанным выше параметрам будут считаться клонами, однако фактически ими не являются [11].

Для исключения подобных неточностей целесообразно ввести (добавить к уже имеющимся девяти) несколько дополнительных показателей, определяющих характеристики трасс:

10. отношение количества рёбер сравниваемой трассы, совпадающих с рёбрами всех других сравниваемых трасс, к среднему количеству ребер во всех сравниваемых трассах:

$S(V_{Avg}) = \frac{|SV_j|}{|V_{AVE}|}$, где $|V_{AVE}|$ – среднее количество рёбер во всех сравниваемых трассах, $|SV_j|$ – количество рёбер сравниваемой трассы, совпадающих с рёбрами всех остальных сравниваемых трасс, $j = 1,2$, m – количество сравниваемых трасс;

11. отношение количества точек (узлов) сравниваемой трассы, совпадающих с точками всех других сравниваемых трасс, к среднему количеству узлов во всех сравниваемых трассах:

$S(E_{Avg}) = \frac{|SE_j|-2}{|E_{AVE}|}$, где $|E_{AVE}|$ – среднее количество точек во всех сравниваемых трассах, $|SE_j|$ – количество точек сравниваемой трассы, совпадающих с точками всех других сравниваемых, $j = 1, 2$, m – количество сравниваемых трасс, 2 – количество точек, которые должны совпадать во всех трассах (точка начала и конца);

12. отношение длины совпадающих отрезков сравниваемой трассы и всех остальных сравниваемых трасс к средней длине всех сравниваемых трасс:

$S(L_{Avg}) = \frac{l_j^S}{l_{AVE}}$, l^S_j – суммарная длина совпадающих рёбер сравниваемой трассы с рёбрами всех остальных сравниваемых трасс, l_{AVE} - средняя длина всех сравниваемых трасс.

Преимущество этих трёх показателей перед описанными выше, заключается в том, что в данном случае с началом координат соотносится именно та часть

сравниваемой трассы, которая является общей для всех трасс алгоритмов мультиверсий модуля. Очевидно, что с помощью данных показателей можно сравнивать на различимость любое количество алгоритмов, однако наиболее точные результаты будут давать при сравнении только двух алгоритмов.

Таким образом, итоговая мера дтверсифицированности двух алгоритмов определяется как Евклидово расстояние между точками в многомерном пространстве. Такими точками являются меры диверсифицированности по каждому из показателей трасс алгоритмов, вычисленные по формуле (2.1):

$$D_A = \sqrt{\sum_{i=1}^{n}(D_i^1 - D_i^2)^2}, \quad (2.2)$$

где $D^1{}_i$ – мера диверсифицированности трассы 1 по i-му показателю, $D^2{}_i$ – мера диверсифицированности трассы 2 по i-му показателю.

2.1.2 Результаты эксперимента

Для апробации предложенной метрики диверсифицированности и проверки её эффективности был проведён эксперимент, в ходе которого были исследованы три алгоритма сортировки: алгоритма сортировки вставкой, групповой сортировки и гномьей сортировки [93].

Для генерации программных реализаций указанных алгоритмов была применена модель двухфазной трансляции кода мультиверсий программных модулей [18].

После выполнения данных алгоритмов были получены три трассы в шестимерном пространстве, состоящие из значений, которые определяют координаты точки. Полученные трассы представлены в таблице 2.

Таблица 2 – Трассы алгоритмов сортировок

Шаг	**Сортировка вставкой**	**Групповая сортировка**	**Гномья сортировка**
№	Трасса A	Трасса B	Трасса C
0	{5, 0, 2, 4, 1, 3}	{5, 0, 2, 4, 1, 3}	{5, 0, 2, 4, 1, 3}
1	{0, 5, 2, 4, 1, 3}	{0, 5, 2, 4, 1, 3}	{0, 5, 2, 4, 1, 3}
2	{0, 2, 5, 4, 1, 3}	{0, 2, 5, 4, 1, 3}	{0, 2, 5, 4, 1, 3}
3	{0, 2, 4, 5, 1, 3}	{0, 2, 4, 5, 1, 3}	{0, 2, 4, 5, 1, 3}
4	{0, 1, 2, 4, 5, 3}	{0, 2, 4, 1, 5, 3}	{0, 2, 4, 1, 5, 3}
5	{0, 1, 2, 3, 4, 5}	{0, 2, 1, 4, 5, 3}	{0, 2, 1, 4, 5, 3}
6		{0, 1, 2, 4, 5, 3}	{0, 1, 2, 4, 5, 3}
7		{0, 1, 2, 4, 3, 5}	{0, 1, 2, 4, 3, 5}
8		{0, 1, 2, 3, 4, 5}	{0, 1, 2, 3, 4, 5}

Значения показателей 1, 3, 4, 6, 7 и 8 трасс алгоритмов могли бы при необходимости быть включены в метрическое пространство, как уже было сказано выше, для определения меры схожести, а на её основе и меры различия алгоритмов. Фактически же они представляют собой те компоненты, из которых строятся показатели, определяющие меру схожести алгоритмов относительно начала координат. В рамках предложенного подхода для определения меры различия будут применяться только показатели трасс 2, 5, 9, 10, 11 и 12.

Для краткости трассы алгоритмов сортировки вставкой, групповой сортировки и гномьей сортировки далее обозначаются как трассы A, B и C, соответственно.

Для составления сравнительных характеристик между наборами точек, составляющих эти три трассы, требуется найти количество точек, отрезков и длины этих трасс. Последние рассчитываются как суммы евклидовых расстояний между парами следующих друг за другом точек. Таким образом, для трасс A, B и C получаем соответствующие длины:

$$l_A = \sqrt{50} + \sqrt{18} + \sqrt{2} + \sqrt{22} + \sqrt{6} \approx 19.8678;$$

$$l_B = \sqrt{50} + \sqrt{18} + \sqrt{2} + \sqrt{32} + \sqrt{18} + \sqrt{2} + \sqrt{8} + \sqrt{2} \approx 28.2843;$$

$$l_C = \sqrt{50} + \sqrt{18} + \sqrt{2} + \sqrt{32} + \sqrt{18} + \sqrt{2} + \sqrt{8} + \sqrt{2} \approx 28.2843.$$

Также для последующего сравнения трасс было построено минимальное дерево с помощью инструмента ESMT-Smith [91].

Перечисленные данные трасс и минимального остовного дерева приведены в таблице 3.

Таблица 3 – Значения некоторых индивидуальных параметров сравниваемых трасс и минимального остовного дерева

Характеристика	**трасса сортировки вставкой**	**трасса групповой сортировки**	**трасса гномьей сортировки**	**минимальное остовное дерево**
количество точек (вершин)	6	9	9	20
количество отрезков (рёбер)	5	8	8	19
суммарная длина	19.8678	28.2843	28.2843	21.9351

Затем были рассчитаны неиндивидуальные характеристики, основанные на непосредственном сравнении трасс между собой и минимальным остовным деревом: средняя длина трасс, среднее количество точек и отрезков, количество общих точек и отрезков, длина общих отрезков.

Средняя длина трасс была получена как среднее арифметическое длин трасс:

$$l_{cp} = \frac{19.8678 + 28.2843 + 28.2843}{3} \approx 25.4788.$$

Аналогичным образом были рассчитаны средние количества точек и отрезков: 8 и 7 соответственно.

Для расчёта 12 показателя была найдена суммарная длина отрезков, общих для всех трёх трасс:

$$l_j^S = \sqrt{50} + \sqrt{18} + \sqrt{2} \approx 12.7279.$$

Некоторые сравнительные показатели трасс, фигурирующие в дальнейших расчётах, представлены в таблице 4.

Таблица 4 – Некоторые сравнительные показатели трасс

Характеристика	**сортировка вставкой**	**групповая сортировка**	**гномья сортировка**
Средняя длина трасс	25.4788	25.4788	25.4788
Среднее количество точек	8	8	8
Среднее количество отрезков	7	7	7
Количество точек, общих для всех трасс	4	4	4
Количество отрезков, общих для всех трасс	3	3	3
Суммарная длина отрезков, общих для всех трасс	12.7279	12.7279	12.7279
Количество точек, общих с минимальным остовным деревом	6	9	9
Количество отрезков, общих с минимальным остовным деревом	0	0	0

На основании полученных данных для определения меры различия были установлены вышеперечисленные показатели трасс.

Стоит отметить, что для трасс трех данных алгоритмов не существует отрезков, общих с минимальным остовным деревом. Это объясняется относительно небольшим количеством точек в трассах, взятых для эксперимента. Отсюда же следует, что показатели 2 и 9, представляющие собой отношения, в которых в качестве числителей выступают длина и количество общих с минимальным остовным деревом отрезков, будут равны нулю для трасс небольшого размера.

Что касается общих точек у сравниваемой трассы и минимального остовного дерева, то их число всегда будет равно числу точек сравниваемой трассы по определению самой задачи построения минимального остовного дерева. Показатель отношения их числа к числу точек в минимальном остовном дереве для трасс A, B и C будет соответственно равен:

$$S_A(E) = \frac{6}{20} = 0.3;$$

$$S_B(E) = \frac{9}{20} = 0.45;$$

$$S_C(E) = \frac{9}{20} = 0.45.$$

Характеристики трасс 10, 11 и 12 используют общие для всех трёх трасс значения, поэтому также являются общими для них. Их значения соответственно равны:

$$S(V_{Avg}) = \frac{3}{7} \approx 0.4286;$$

$$S(E_{Avg}) = \frac{4-2}{8} = 0.25;$$

$$S(L_{Avg}) = \frac{12.7279}{25.4788} \approx 0.4995.$$

В результате измерений были получены значения вышеперечисленных характеристик для каждой из трасс. Результаты приведены в таблице 5.

Таблица 5 – Неиндивидуальные характеристики сравниваемых трасс

Характеристика	**сортировка вставкой**	**групповая сортировка**	**гномья сортировка**
2. Отношение количества общих отрезков у сравниваемой трассы с минимальным остовным деревом к количеству ребер в минимальном остовном дереве	0	0	0

Характеристика	**сортировка вставкой**	**групповая сортировка**	**гномья сортировка**
5. Отношение количества общих точек у сравниваемой трассы с минимальным остовным деревом к количеству узлов в минимальном остовном дереве	0.3	0.45	0.45
9. Отношение длины совпадающих отрезков у сравниваемой трассы и минимального остовного дерева к длине минимального остовного дерева	0	0	0
10. Отношение количества рёбер сравниваемой трассы, совпадающих с рёбрами всех других сравниваемых трасс, к среднему количеству ребер во всех сравниваемых трассах	0.4286	0.4286	0.4286
11. Отношение количества точек сравниваемой трассы, совпадающих с точками всех других сравниваемых трасс, к среднему количеству узлов во всех сравниваемых трассах	0.25	0.25	0.25
12. Отношение длины совпадающих отрезков сравниваемой трассы и всех остальных сравниваемых трасс к средней длине всех сравниваемых трасс	0.4995	0.4995	0.4995

Полученные результаты показывают, что алгоритмы гномьей и групповой сортировок равны по всем характеристикам. Очевидно, что мера диверсифицированности этих алгоритмов равна нулю, так как, согласно формуле (2.2), это то же самое, что евклидово расстояние между двумя совпадающими

точками. Мера диверсифицированности для трассы сортировки вставкой относительно каждой из двух других трасс была найдена как:

$$D_A = \sqrt{(0.3 - 0.45)^2} = 0.15.$$

В получившемся подкоренном выражении осталось лишь одно слагаемое, так как диверсифицированность общих характеристик всегда будет равна нулю, а все остальные рассматриваемые характеристики, кроме пятой, оказались равны нулю из-за небольшого размера трасс.

Итоговые значения позволяют заключить, что алгоритмы гномьей и групповой сортировок являются клонами.

Проведенный эксперимент подтвердил, что описанная в работе метрика может использоваться для определения диверсифицированности алгоритмов, используемых в мультиверсионном программном обеспечении.

2.2 Метрика диверсифицированности версий на уровне языков программирования

2.2.1 Общее описание

Определение меры различия версий на уровне языков программирования может осуществляться путем сравнения языков программирования по заданному набору признаков.

На сегодняшний день не существует единой общепринятой таксономии языков программирования. Все множество языков программирования можно разделить на классы по определенному набору общих признаков.

В данной главе для исследования были отобраны следующие признаки языков программирования: парадигма, типизация, управление памятью, управление потоком вычислений. Поскольку именно они определяют наиболее значимые различия между программным кодом, написанным на этих языках

программирования [76]. Однако для решения практических задач могут быть использованы любый признаки, подходящие для описания свойств языков программирования в каждом конкретном случае. Краткое описание каждого из признаков приведено ниже на примере нескольких наиболее популярных на сегодняшний день языков программирования.

Парадигма программирования

Парадигма программирования – это совокупность идей и понятий, определяющих стиль написания программного кода [8]. На сегодняшний день существует множество парадигм программирования, однако среди них принято выделять несколько основных, к которым относятся: императивная, функциональная, логическая, объектно-ориентированная, рефлексивная, обобщенное программирование, распределенная, декларативная [85].

Парадигма программирования зачастую не может быть однозначно определена языком программирования, поскольку большинство современных языков программирования так или иначе допускают использование различных парадигм [85]. В то же время существуют и языки, ориентированные на реализацию лишь одной парадигмы [61]. Так в таблице 6 приведён список наиболее популярных языков программирования и их парадигм.

Здесь и далее, во всех остальных таблицах, знак «+» означает, что указанное значение признака присутствует у данного языка, «-» – указанное значение признака отсутствует у языка, «+/-» – указанное значение признака поддерживается не полностью данным языком программирования, и «-/+» – указанное значение признака имеет значительные ограничения использования в данном языке программирования, «N/A» – указанное значение признака не поддерживается данным языком программирования.

Таблица 6 – Парадигмы языков программирования

	C	**C++**	**C#**	**Java**	**Python**	**Ruby**	**Delphi**
Императивная	+	+	+	+	+	+	+

	C	C++	C#	Java	Python	Ruby	Delphi
Объектно-ориентированная	-/+	+	+	+	+	+	+
Функциональная	+/-	-/+	+/-	-/+	+	+	-/+
Рефлексивная	-	-/+	-/+	-/+	+	+	-/+
Обобщенное программирование	+	-/+	+	+	+	+	+
Логическая	-	-	-	-	-	-	-
Декларативная	-	-	-/+	-	+	+	-
Распределенная	+/-	+/-	-/+	+	-/+	-/+	-

Типизация

Типизация – свойство языка программирования различать типы данных [59]. Языки программирования по признаку типизации могут быть разделены на следующие классы: со статической и динамической типизацией, с явной и неявной типизацией, строгой и нестрогой типизацией [59].

В языках со статической типизации переменная связывается с типом в момент своего объявления (или инициализации) и впоследствии не может изменять свой тип. В языках же с динамической типизацией переменный могу изменять свой тип данных в ходе работы программы, при этом в языках с явной типизацией тип переменной задаётся при её объявлении, а в языках с неявной типизацией тип переменной определяется в момент её инициализации [88].

Основным отличием явно типизированных языков программирования является необходимость однозначного (явного) задания типов новых переменных / функций / аргументов функций. При этом языки с неявной типизацией перекладывают решение данной задачи с программиста на транслятор [88].

Стоит отметить, что трансляторы некоторых языков программирования поддерживают возможность самостоятельного логического вывода типов значений выражений. Вывод типа может происходить при инициализации переменных, в

момент установки значений по умолчанию, а также вовремя определения типов возвращаемых функциями значений [59].

Список возможных вариантов типизации для некоторых популярных на сегодняшний день языков программирования приведен в таблице 7.

Таблица 7 – Варианты типизации в языках программирования

	C	**C++**	**C#**	**Java**	**Python**	**Ruby**	**Delphi**
Статическая типизация	+	+	+	+	-	-	+
Динамическая типизация	-	-	+	-	+	+	-/+
Явная типизация	+	+	+	+	+/-	-	+
Неявная типизация	-	-/+	-/+	-	+	+	-
Неявное приведение типов без потери данных	+	+	+	+	+	+	+
Неявное приведение типов с потерей данных	+	+	-	-	-	-	+
Неявное приведение типов в неоднозначных ситуациях	+	+	+	-	-	-	-
Алиасы типов	+	+	+	-	N/A	N/A	+
Вывод типов переменных из инициализатора	-	+/-	+	-	N/A	N/A	-
Вывод типов переменных из использования	-	+/-	-	-	N/A	N/A	-
Вывод типов-аргументов при вызове метода	-	+	+	+	N/A	N/A	-
Вывод сигнатуры для локальных функций	-	+/-	-	-	N/A	N/A	-
Параметрический полиморфизм	N/A	-	+	+	N/A	N/A	-
Параметрический полиморфизм с ковариантностью	N/A	-	+/-	+	N/A	N/A	-
Параметрический	N/A	-	-	-	N/A	N/A	-

	C	C++	C#	Java	Python	Ruby	Delphi
полиморфизм высших порядков							
Информация о типах в runtime	-	-/+	+	+	+	+	+
Информация о типах-параметрах в runtime	-	-/+	+	-	+	+	+

Управление памятью

Задача управления памятью является одной из наиболее фундаментальных в программировании [76]. Во многих скриптовых языках, разработчикам не нужно заботиться об управлении памятью (так как за это отвечает «движок» языка) [59], но это не делает проблему управления памятью менее важной. В целях эффективной разработки критически важны знания возможностей и ограничений используемого менеджера памяти.

Большинство системных языков в свою очередь не поддерживает автоматического управления памятью. Для создания объекта в динамической памяти необходимо явно вызывать команду выделения памяти. Для доступа к объекту программа обращается по указателю на выделенную область памяти. После окончания работы с объектом необходимо вызвать команду освобождения памяти.

В таблице 8 представлены виды управления памятью в некоторых популярных на сегодняшний день языках программирования.

Таблица 8 – Виды управления памятью в языках программирования

	C	C++	C#	Java	Python	Ruby	Delphi
Создание объектов на стеке	+	+	+	-	-	-	-/+
Неуправляемые указатели	+	+	+	-	-	-	+
Ручное управление памятью	+	+	+	-	-	-	+
Сборка мусора	-	-/+	+	+	+	+	-

Управление потоком вычислений

Еще одной важной задачей в программировании является управление потоком вычислений. Процесс выполнения последовательности операторов может происходить как непрерывно, а может прерываться при определенных условиях. Прерывание происходит в том случае, если в потоке вычислений будут обнаружены соответствующие операторы. В случае прерывания управление будет передаваться в другое место [59].

Нормальное выполнение оператора может быть прервано также при возникновении исключительных ситуаций. Возбуждение исключительной ситуации прерывает нормальное выполнение оператора.

В некоторых языках программирования поддерживается стратегия вычисления, согласно которой вычисления следует откладывать до тех пор, пока не понадобится их результат [39].

Список возможных вариантов управления потоком вычислений для некоторых популярных на сегодняшний день языков программирования приведен в таблице 9.

Таблица 9 – Варианты управления потоком вычислений в языках программирования

	C	**C++**	**C#**	**Java**	**Python**	**Ruby**	**Delphi**
Инструкция goto	+	+	+	-	-	-/+	+
Инструкция break без метки	+	+	+	+	+	+	+
Инструкция break с меткой	-	-	-	+	-	+	-
Поддержка try/catch	-	+	+	+	+	+	+
Блок finally	-	-	+	+	+	+	+
Блок else	-	-	-	+	+	+	+
Перезапуски	-	?	-	?	?	+	?
Ленивые вычисления	-	-/+	-/+	-	+	-/+	-
Continuations	-/+	?	+	?	-	+	?

	C	C++	C#	Java	Python	Ruby	Delphi
Легковесные процессы (coroutines)	-	-	-	+/-	+/-	+	-

Многие рассматриваемые языки имеют множество значений одного признака, то есть относятся сразу к нескольким классам. Так, например, если в качестве признака взять парадигму программирования, а за описываемый язык – Python, то можно сказать, что на этом языке могут быть написаны программы в функциональном, декларативном, объектно-ориентированном и других стилях программирования.

Определение метрики

Для определения метрики диверсифицированности мультиверсий на уровне языков программирования каждому признаку каждого языка программирования может быть поставлено в соответствие некоторое непустое множество значений. Что по своей сути является формализацией задачи сравнения языков программирования между собой по некоторому набору признаков. То есть путём нахождения мощности множества, которое образовалось на пересечении двух множеств значений определенного признака двух конкретных языков программирования, может быть получено количество сходных значений данного признака. При делении этого значения на общее число значений данного признака для конкретного языка может быть найдено соотношение схожести одного языка с другим [16].

Стоит сразу отметить, что числовая мера различия мультиверсий при использовании данного подхода является вероятностной, поскольку сравниваются именно языки программирования, а не написанные с их помощью исходные коды программ. Так, например, при использовании одного языка для написания двух мультиверсий могут быть использованы его различные возможности (например, код будет написан в декларативном и функциональном стилях), верно и обратное утверждение, когда разные языки программирования используют схожие

возможности (например, языки C# и Iron Python могут использовать схожие возможности общеязыковой среды выполнения CLR).

Исходя из приведённых выше заключений, степень схожести двух языков программирования по *k*-му признаку определяется как:

$$S_{ij}{}^{k} = \frac{|set_i(k) \cap set_j(k)|}{|set_i(k)|}, i = 1,2, j = 1,2, i \neq j, \quad (2.3)$$

где i и j – индексы языков программирования, k – номер признака из всего множества признаков, $set_i(k)$ – множество значений k-го признака i-го языка программирования, $set_j(k)$ – множество значений k-го признака j-го языка программирования [16].

А для расчета степени схожести двух языков программирования с учетом всех признаков используется следующая формула:

$$S_{ij} = \frac{\sum_{k=1}^{m} S_{ij}^{k}}{m}, \quad (2.4)$$

где i и j – индексы языков программирования, k – номер признака из всего множества признаков, m – число признаков, $S_{ij}{}^{k}$– степень схожести двух языков программирования по *k*-му признаку, вычисленное по формуле (2.3) [16].

Однако, как и для метрики диверсифицированности мультиверсий на уровне алгоритмов, на основании данного математического аппарата всё ещё невозможность ввести полноценную метрику диверсифицированности языков программирования, поскольку не выполняется одна из аксиом метрического пространства: ρ(a, b) = ρ(b, a) для любых двух элементов a и b из множества X – «аксиома симметрии» [4].

Приведённая выше аксиома не выполняется, когда мощности возможных значений у одного признака различны для сравниваемых языков. В таких, случаях при сравнении одних и тех же языков программирования, знаменатель в формуле (2.4) будет различным в зависимости от того, для какого языка будет находиться отношение схожести с другим, для первого или для второго.

Поэтому для ввода полноценной метрики диверсифицированности языков программирования необходима начальная точка, с которой бы сравнивались все

остальные значения. То есть при расчётах меры схожести конкретного языка с ней в метрическом пространстве определялось бы расстояние от этой точки до точки, соответствующей сравниваемому с ней языку программирования.

Такой начальной точкой, с которой будут сравниваться все остальные языки программирования, станет абстрактный несуществующий язык программирования, который имеет все признаки сравниваемых языков программирования, а каждый его признак может принимать все значения этого признака, содержащиеся во всех сравниваемых языках программирования [16].

Таким образом для определения метрики диверсифицированности двух мультиверсий одного модуля мультиверсионного программного обеспечения на уровне языков программирования необходимо сравнить языки, на которых они были написаны, с «идеальным» языком, то есть найти их удаленность от него в метрическом пространстве. После чего с помощью вычитания одного значения из другого может быть получена мера схожести мультиверсии в метрическом пространстве, поскольку изначально производился расчёт количества совпадающих значений признаков. Описанное логическое заключение приводит формулу (2.3) к следующему виду:

$$S_{ij} = \frac{\sum_{k=1}^{m}({S_{ip}}^{k} - {S_{jp}}^{k})}{m}, \quad (2.5)$$

где i и j – индексы сравниваемых языков программирования, p – индекс для обозначения «идеального» языка программирования, k – номер признака из всего множества признаков, m – число признаков, ${S_{ip}}^{k}$ и ${S_{jp}}^{k}$ – степени схожести i-го и j-го языков программирования, соответственно, по k-му признаку с «идеальным» языком программирования (p), который является началом координат в метрическом пространстве [16].

В итоге метрика диверсифицированности двух мультиверсий одного модуля мультиверсионного программного обеспечения на уровне языков программирования (D_L) определяется формулой [16]:

$$D_L = 1 - S_{ij}. \quad (2.6).$$

2.2.2 Результаты эксперимента

Для проведения эксперимента были отобраны 4 языка программирования: C, Java, Python, Delphi.

По каждому признаку были рассчитаны степени схожести между отобранными языками программирования и «идеальным» языком программирования. Полученные результаты представлены в таблицах 10 и 11.

Таблица 10 – Степень схожести анализируемых языков программирования с «идеальным» языком программирования по парадигме и типизации

	Парадигма программирования				**Типизация**			
	C	**Java**	**Python**	**Delphi**	**C**	**Java**	**Python**	**Delphi**
C	-	0	0.25	0.125	-	0.059	0	0.059
Java	0	-	0.25	0.125	0.059	-	0.059	0
Python	0.25	0.25	-	0.375	0	0.059	-	0.059
Delphi	0.125	0.125	0.375	-	0.059	0	0.059	-

Таблица 11 – Степень схожести анализируемых языков программирования с «идеальным» языком программирования по управлению памятью и управлению потоком вычислений

	Управление памятью				**Управление потоком вычислений**			
	C	**Java**	**Python**	**Delphi**	**C**	**Java**	**Python**	**Delphi**
C	-	0.5	0.5	0.25	-	0.4	0.4	0.3
Java	0.5	-	0	0.25	0.4	-	0	0.1
Python	0.5	0	-	0.25	0.4	0	-	0.1
Delphi	0.25	0.25	0.25	-	0.3	0.1	0.1	-

Затем по формуле (2.5) были рассчитаны меры схожести каждого исследуемого языка с каждым. Результаты этих расчетов приведены в таблице 12.

Таблица 12 – Меры схожести исследуемых языков между собой

	C	**Java**	**Python**	**Delphi**
C	-	0.76025	0.7125	0.8165
Java	0.76025	-	0.92275	0.88125
Python	0.7125	0.92275	-	0.804
Delphi	0.8165	0.88125	0.804	-

Далее по формуле (2.6) был осуществлён расчёт самой метрики диверсифицированности версий одного программного модуля мультиверсионного программного обеспечения на уровне языков программирования для каждого языка программирования, из рассматриваемых в этом эксперименте. Результаты данного расчета приведены в таблице 13.

Таблица 13 – Результат эксперимента – мера диверсифицированности анализируемых языков программирования

	C	**Java**	**Python**	**Delphi**
C	-	0.23975	0.2875	0.1835
Java	0.23975	-	0.07725	0.11875
Python	0.2875	0.07725	-	0.196
Delphi	0.1835	0.11875	0.196	-

Таким образом, из результатов эксперимента можно сделать вывод, что все сравниваемые языки имеют небольшие различия между собой при сравнении их по определенным в данном эксперименте критериям. Такие результаты вполне объяснимы тем, что все сравниваемые языки являются современными

высокоуровневыми языками программирования, и каждый из них подходит в качестве инструмента для решения широкого круга задач.

Рассматриваемая метрика по своей сути является неким правилом сравнения языков программирования по заданным критериям. Сравнивая языки между собой, можно установить как их степень схожести, так и величину различия.

Данная метрика выражает величину различия двух языков программирования в вероятностном соотношении. При значении, близком к нулевому, сравниваемые языки будут полностью совпадать по заданным критериям. В свою очередь, при значении, стремящемся к единице, рассматриваемые языки будут максимально различаться. Величина различия будем максимальной только в случае использования тех возможностей языка, которые отсутствуют в сравниваемом с ним языке.

На основе получаемой величины различия языков программирования можно делать вывод и о степени различия мультиверсий, реализованных на них. Величина данной метрики эквивалентна вероятности диверсифицированности версий, реализованных на сравниваемых языках программирования.

2.3 Комбинированная модель мультиверсии

В предыдущих разделах данной главы описаны метрики диверсифицированности на уровне алгоритмов и языков программирования. Каждую из них можно использовать как самостоятельную меру различия между мультиверсиями или даже просто функционально эквивалентными программами (не входящим в состав мультверсионной программной системы).

Однако данные метрики целесообразно рассматривать совместно, поскольку главным постулатом мультиверсионной методологии является утверждение, что чем более различны мультиверсии между собой, тем меньше вероятность возникновения в них похожих или зависимых ошибок, благодаря чему повышается надёжность модуля, в который они входят. Таким образом, обуславливается

необходимость ввода новой комбинированной модели мультиверсии, основанной на описанных метриках диверсифицированности.

При использовании любой из описанных метрик диверсифицированности для каждой из мультиверсий определяется её расстояние от начала координат, будь то «идеальный» язык программирования или минимальное остовное дерево. Причём каждая из описанных метрик определяется в диапазоне от 0 до 1. Благодаря чему, данные метрики могут быть использованы как измерения в *N*-мерном пространстве, где *N* – количество различного рода метрик диверсифицированности.

Таким образом, предлагаемая модель определяет мультиверсию как точку в многомерном пространстве, где каждое измерение определяется одной (частной) метрикой диверсифицированности.

Общая (интегральная) метрика диверсифицированности двух мультиверсий в рамках данной модели определяется как Евклидово расстояние между точками в многомерном пространстве. Такими точками являются меры диверсифицированности на каждом из уровней диверсификации (на уровне алгоритмов и языков программирования), вычисленные по формулам (2.1) и (2.6):

$$D = \sqrt{\sum_{n=1}^{N}(D_n^1 - D_n^2)^2}, \quad (2.7)$$

где $D^1{}_n$ – мера диверсифицированности мультиверсии 1 по *n*-ой метрике, $D^2{}_n$ – мера диверсифицированности мультиверсии 2 по той же, *n*-ой, метрике, $n = \overline{1, N}$, *N* – количество метрик диверсифицированности, входящих в комбинированную модель мультиверсии.

В контексте данной работы рассматривались только две метрики диверсифицированности: на уровне алгоритмов и на уровне языков программирования. Поэтому далее все объяснения будут приводиться на примере двумерного метрического пространства с использованием указанных метрик. Однако для решения каждой конкретной прикладной задачи (или класса задач) могут быть отобраны иные метрики, отражающие меру различия мультиверсий, из-за чего может меняться не только способ их вычисления, но и мерность

метрического пространства в комбинированной модели мультиверсии. В случае совместного использования метрик с различными шкалами потребуется их нормализация.

В случае описании мультиверсии с помощью вышеизложенных метрик диверсифицированности (на уровне алгоритмов и языков программирования) её формульное представление выглядит следующим образом: $V(D_A, D_L)$ – это **модель представления мультиверсии в виде точки в двумерном пространстве** [15].

А расстояние между мультиверсиями в случае использования метрик на уровне языков программирования и алгоритмов, то есть мера их разнообразия, представляется формулой [15]:

$$D = \sqrt{(D_L^1 - D_L^2)^2 + (D_A^1 - D_A^2)^2}.$$

С помощью описанной выше модели мультиверсии мультиверсионный модуль может быть представлен как взвешенный граф, вершинами которого являются мультиверсии, то есть точки в *N*-мерном пространстве, а весами рёбер – значения расстояний между версиями в этом метрическом пространстве.

Комбинированная модель мультиверсии объединяет несколько разнородных метрик воедино, благодаря чему мера различия мультиверсий может быть вычислена наиболее точно (по сравнению с использованием только одной или нескольких метрик по отдельности).

Поскольку основополагающей аксиомой парадигмы мультиверсионного программирования является утверждение, что чем более различны версии, входящие в модуль, тем более надёжен этот модуль (как было сказано выше), предложенная модель мультиверсионного модуля позволяет оценить надёжность модуля не только на основании априорных данных, но и с использованием апостериорных (с помощью использования метрики диверсифицированности на уровне алгоритмов).

Таким образом может оцениваться уровень надёжности мультиверсионных модулей и отдельных групп мультиверсий, входящих в них, поскольку чем более различны версии между собой, тем меньше вероятность возникновения в них схожих или зависимых ошибок и сбоев. **Поэтому предложенные модели**

представления мультиверсии и мультиверсионного модуля целесообразно применять в работе алгоритмов голосования в спорных ситуациях, или для проведения сравнения мультиверсий по определенным критериям.

2.4 Выводы

В ходе исследования были разработаны и обоснованы две метрики диверсифицированности мультиверсионных программных модулей: на уровне алгоритмов и на уровне языков программирования.

Метрика диверсифицированности на уровне алгоритмов основана на анализе трасс выполнения программ в многомерном пространстве. Предложенный подход позволяет количественно оценивать различия между версиями через анализ изменения обрабатываемых данных на каждом шаге выполнения алгоритма. Для того, чтобы эта оценка действительно являлась метрикой, было решено использовать минимальное остовное дерева в качестве базовой точки отсчёта, то есть начала координат.

Метрика диверсифицированности на уровне языков программирования основывается на тех признаках языка программирования, которые являются наиболее критичным в каждом конкретном случае применения метрики. В качестве таких признаков могут использоваться, например, парадигма программирования, типизация, управление памятью или управление потоком вычислений.

Метрика диверсифицированности на уровне языков программирования основана на определении степени схожести их характеристик, то есть значений выбранных признаков, с абстрактным «идеальным» языком программирования, который включает в себя все значения всех выбранных признаков сравниваемых языков программирования. Такой подход позволяет получить вероятностную оценку различий между версиями на этом уровне, поскольку сравниваются именно языки программирования, а не конкретные реализации мультиверсий.

На основе разработанных метрик создана комбинированная модель мультиверсии, представляющая собой точку в многомерном пространстве, где каждое измерение определяется отдельной метрикой диверсифицированности. Это позволяет получить комплексную оценку различий между версиями программного модуля.

Кроме того, была предложена графовая модель мультиверсионного модуля, где вершины графа представляют отдельные версии, а веса рёбер отражают расстояния между ними в метрическом пространстве. Такой подход даёт возможность оценивать меру разнообразия мультиверсий, входящих в модуль, как на этапе проектирования, так и в процессе его эксплуатации.

Проведённые эксперименты на примере различных алгоритмов сортировки и набора популярных языков программирования подтвердили работоспособность и практическую применимость разработанных метрик.

На основании проведённого в данной главе исследования можно сделать следующие выводы:

1. Разработанные метрики позволяют объективно оценивать степень диверсифицированности мультиверсионных программных модулей на различных уровнях реализации, что ранее могло быть оценено только качественно, но не количественно;

2. Созданные модели представления мультиверсии и мультиверсионного модуля могут быть использованы как на этапе проектирования мультиверсионных систем для формирования оптимального состава модулей, так и в процессе эксплуатации для оценки их надёжности;

3. Разработанные модели могут быть внедрены в алгоритмы голосования в качестве дополнительного критерия, который может быть использован в случаях неопределённости, то есть, когда у нескольких версий в модули возникли взаимозависимые ошибки.

Таким образом, в данной главе предложены новые модели описания мультиверсий и мультиверсионных программных модулей на основе метрики диверсифицированности, которые позволили формализовать процесс

обеспечения разнообразия мультиверсионного программного обеспечения ИУС, а также управлять процессом выбора наиболее разнообразных групп мультиверсий с целью повышения надёжности мультиверсионной системы в целом.

3 Оценка диверсифицированности и её применение в МВПО

Поскольку предложенные модели представления мультиверсий и мультиверсионных программных модулей оперируют расстояниями между версиями в пространстве и сформированными графами из групп мультиверсий, эти расстояния и графы могут сравниваться между собой с целью формирования оценки диверсифицированности групп мультиверсий. Такая оценка может служить значимым критерием в процессе голосования за верное решение, возвращаемое мультиверсионным модулем, в условиях неопределённости. Данное обстоятельство позволяет формализовать один из базовых принципов мультиверсионной методологии – принцип разнообразия.

Однако мало просто ввести новую модель представления мультиверсий, чтобы обеспечить формализацию принципа разнообразия мультиверсий – необходим алгоритм оценки разнообразия групп мультиверсий.

3.1 Алгоритм оценки диверсифицированности

На основе представленной модели мультиверсии был разработан алгоритм оценки диверсифицированности для блоков принятия решений в мультиверсионных ИУС, который позволяет определить группу наиболее различных по набору заданных критериев мультиверсий. Блок-схема алгоритма приведена на рисунке 3.

Данный алгоритм включает в себя следующие этапы:

1. Определить набор частных метрик диверсифицированности, которые отражают наиболее значимые различия мультиверсий для решения поставленной задачи;

2. Рассчитать до запуска модуля (или системы в целом) те метрики диверсифицированности, которые не меняются после исполнения мультиверсий (как метрика диверсифицированности на уровне языков программирования);

3. Обеспечить расчёт метрик, пересчитываемых после каждого выполнения мультиверсий (как метрика диверсифицированности на уровне алгоритмов);

4. Обеспечить расчёт общей метрики диверсифицированности для каждой мультиверсии;

5. Теперь, когда каждая из мультиверсий представлена точкой в многомерном пространстве, находится Евклидово расстояние между всеми мультиверсиями.

6. Если блоку принятия решений необходимо сравнить между собой несколько групп мультиверсий по уровню их диверсифицированности, то необходимо определить, какая величина будет взята за уровень диверсифицированности группы версий. Исходя из специфики конкретной задачи необходимо составить ранжированный по приоритету список величин разнообразия. Например, максимальное расстояние между двумя мультиверсиями входящими в одну группу, среднее или медианное расстояние между всеми версиями в группе или иная подходящая величина;

7. В качестве результата, то есть группы наиболее разнообразных мультиверсий, выбирается та, в которой мера разнообразия версий оказалась выше, если были получены различные значения метрик диверсифицированности для сравниваемых групп версий – алгоритм завершён. Иначе, переход к шагу 8.

8. Если значения метры разнообразия у сравниваемых групп мультиверсий оказались эквиваленты, то проверяется, не исчерпан ли список уровней диверсифицированности, определённый на шаге 6. Если не исчерпан, то переход к шагу 6 для выбора новой величины сравнения. Иначе – шаг 9.

9. Применяется модель представления мультиверсионной группы версий для проверки графов сравниваемых групп мультиверсий на изоморфизм, результаты сравнения интерпретируются следующим образом:

a. Если сравниваемые остовные деревья неизоморфны, то делается вывод, что сравниваемые группы мультиверсий различны, однако их диверсифицированность находится на одном уровне для определённых на шаге 6 величин. Алгоритм завершён.

b. Если сравниваемые остовные деревья изоморфны, то делается вывод, что сравнение таких групп мультиверсий (возможно модулей) бессмысленно, поскольку они являются клонами. Алгоритм завершён.

Таким образом, описанный алгоритм принимает на вход заранее известные метрики диверсифицированности мультиверсий, такие как метрика диверсифицированности на уровне языков программирования, после чего исполняет мультиверсии из сравниваемых наборов для получения метрик диверсифицированности, вычисляемых после исполнения мультиверсий, таких как метрика диверсифицированности на уровне алгоритмов. Далее определяется величина уровня диверсифицированности (какая-то агрегирующая функция), которая позволит все расстояния между версиями внутри одной группы привести к единому значению, которое можно будет с аналогичным значением из другой группы, например, среднее, минимальное или максимальное расстояние.

На следующем этапе происходит вычисление выбранной величины уровня диверсифицированности для каждой из сравниваемых групп. Если группа с наибольшим уровнем разнообразия найдена, то алгоритм завершается, иначе – выбирается другая величина уровня диверсифицированности из набора доступных величин, и алгоритм повторяется уже с её применением. Если группу наиболее различных версий не удалось найти, то эти группы, представленные в виде графов, проверяются на изоморфизм, после чего делается вывод, являются ли они полностью идентичными, то есть клонами, или нет.

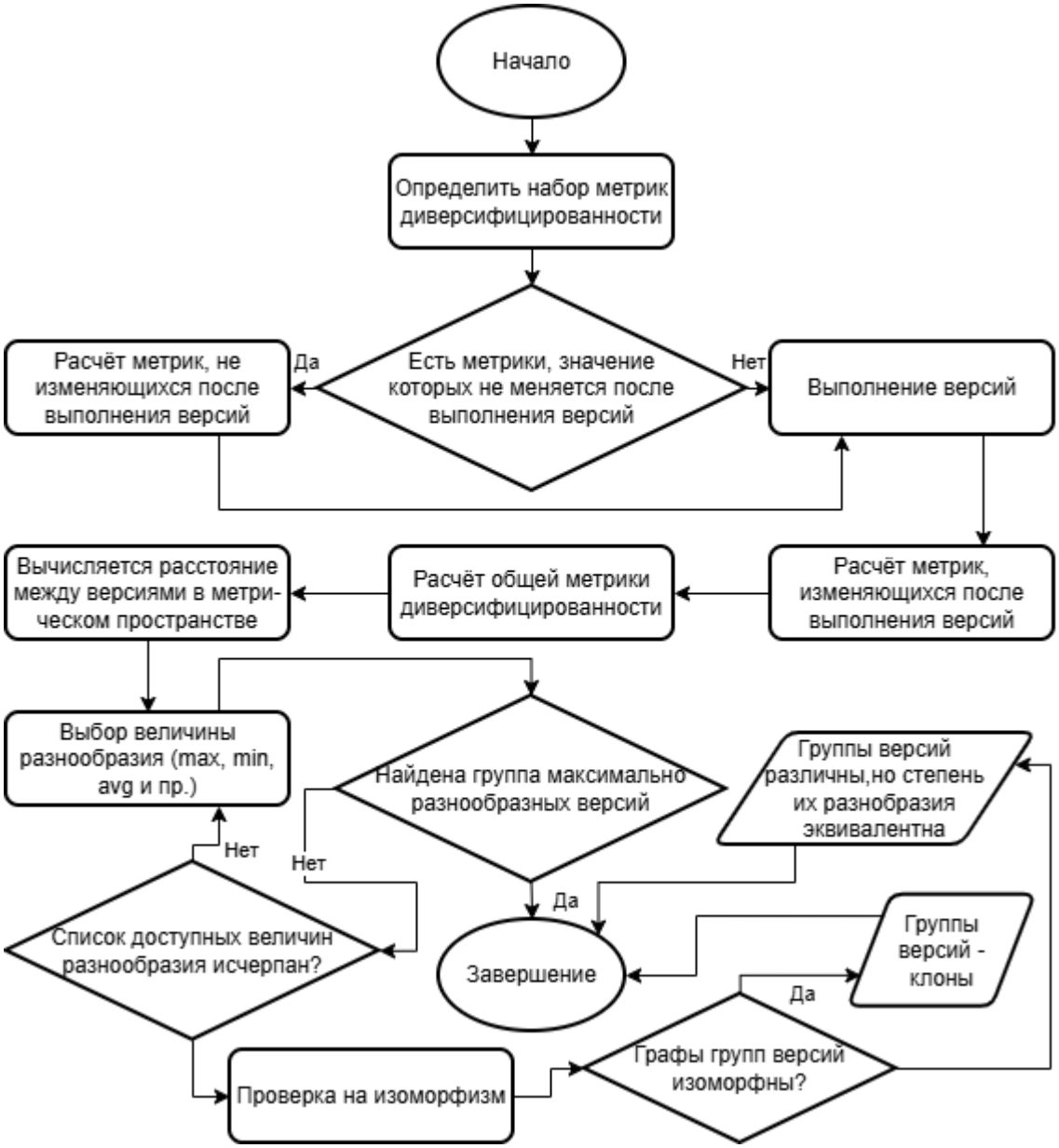

Рисунок 3 – Алгоритм оценки диверсифицированности для блоков принятия решений в мультиверсионных ИУС

Описанный алгоритм может применяться, как для оценки уровня разнообразия мультиверсионных программных модулей в ходе их проектирования и эксплуатации, так и в качестве дополнительного критерия, используемого при принятии решений алгоритмами голосования в ситуациях неопределённости.

Разумеется, не во всех алгоритмах голосования могут возникать ситуации неопределённости, например, в функциональных алгоритмах голосования, таких как функции среднего или медианы, это попросту невозможно. Однако существует и ряд алгоритмов голосования, в которых решение может в конечном счёт

приниматься случайным образом, когда такие ситуации неопределённости возникают, например, алгоритм голосования согласованным большинством.

Таким образом, был разработан алгоритм оценки диверсифицированности для блоков принятия решений в мультиверсионных информационных управляющих системах, который позволяет определить группу наиболее различных между собой по набору заданных критериев мультиверсий или найти одинаковые группы мультиверсий (клоны).

3.2 Методика модификации алгоритмов голосования

Для модификации алгоритмов голосования с применением описанной модели мультиверсии и алгоритма оценки диверсифицированности с целью повышения вероятности выбора действительно верного варианта ответа в условиях неопределённости была разработана следующая методика (Блок-схема методики приведена на рисунке 4):

1. Выполняются действия алгоритма оценки диверсифицированности для блоков принятия решений в мультиверсионных информационных управляющих системах. Если в рамках выполнения этого алгоритма не производился запуск мультиверсий, то переход к шагу 3, иначе – к шагу 2.
2. Выполнить мультиверсии для получения результатов их вычислений.
3. Результаты выполнения мультиверсий сравниваются между собой по заданному алгоритму голосования для нахождения единственного верного ответа. Если единственный результат алгоритма голосования найден, то он принимается за результат работы модуля. Если алгоритм голосования обнаружил, что несколько групп мультиверсий, куда входит одинаковое количество версий, выдали разные результаты, то есть возникла ситуация неопределённости, то переход к шагу 4;
4. Производится выбор верного варианта ответа с использованием описанной модели мультиверсии и представленного алгоритма оценки диверсифицированности групп мультиверсий – в качестве верного принимается

ответ от той группы мультиверсий, уровень диверсифицированности которой оказался выше, поскольку именно высокий уровень разнообразия в основе самой методологии мультиверсионного программирования определяет независимость сбоев.

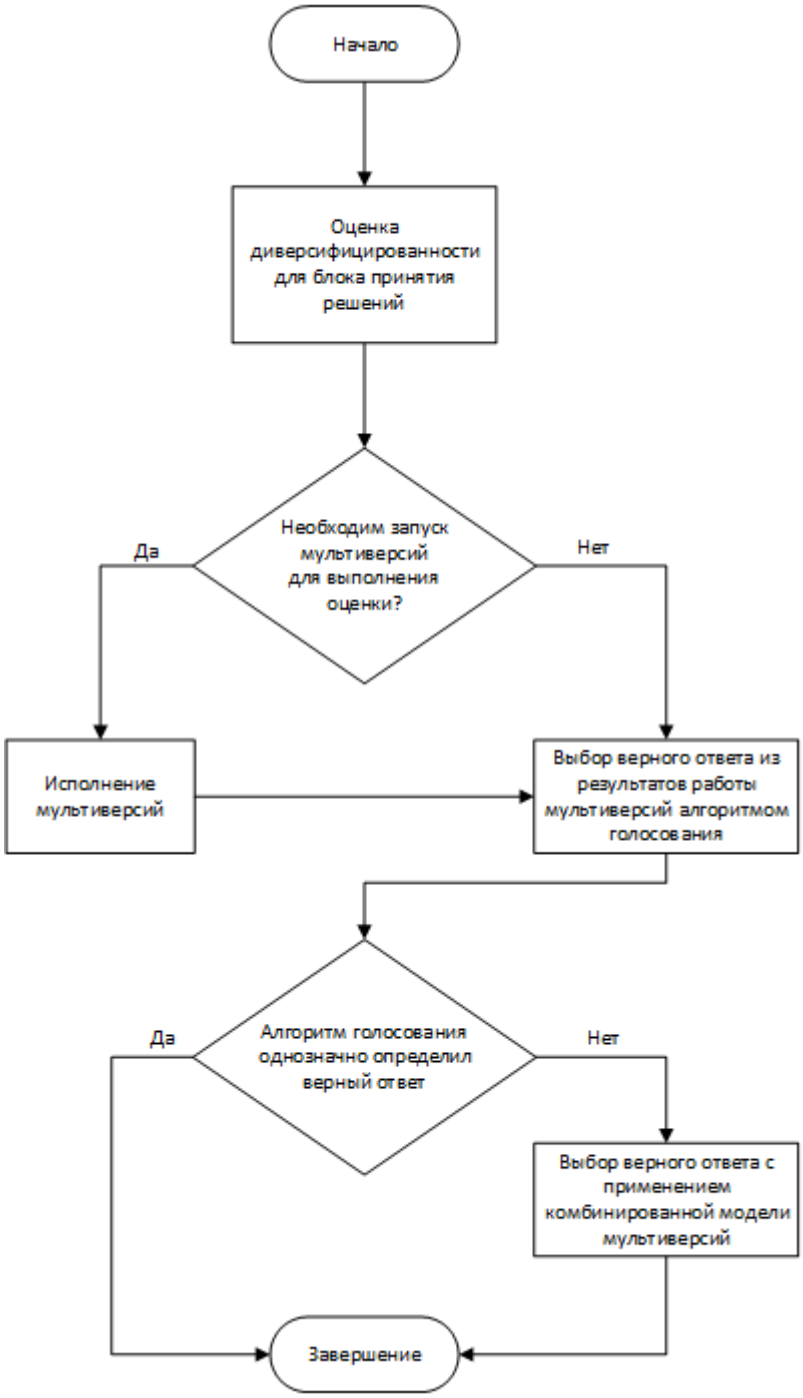

Рисунок 4 – Методика модификации алгоритмов голосования с применением общей метрики диверсифицированности

Как уже упоминалось выше, область применения описанной методики ограничена набором тех алгоритмов голосования, в которых могут возникать ситуации неопределённости выбора. Именно в таких ситуациях уровень разнообразия мультиверсий и служит дополнительным критерием.

Для проверки работоспособности разработанной методики необходимо с её помощью модифицировать подходящий для этого алгоритм голосования.

Таким образом, была разработана методика модификации алгоритмов голосования с применением новой модели описания мультиверсий, которая позволяет повысить вероятность выбора корректного результата в условиях неопределённости.

3.3 Модифицированный алгоритм голосования согласованным большинством

3.3.1 Общее описание

Для модификации по описанной выше методике подходят алгоритмы, позволяющие сравнивать в том числе одинаковые по количеству группы версий, например, t/(n-1)-вариантное программирование, а также алгоритм голосования согласованным большинством. Алгоритмы голосования, которые в любом случае выдают результат, не используя случайный выбор в случаях неопределённости, модифицировать нецелесообразно.

Поскольку, как уже было определено в первой главе данной работы, алгоритмы теории голосования наилучшим образом подходят для определения конечного результата работы мультиверсионного модуля на основании выходов его мультиверсий, что подтверждается практическими примерами [100], для применения разработанной методики был выбран один из таких алгоритмов.

Алгоритм голосования согласованным большинством на практике хорошо зарекомендовал себя, как для применения в средах исполнения мультиверсионного программного обеспечения, так и для использования в политических выборах. Это связано с тем, что данный алгоритм достаточно прост в реализации, при его использовании считается, что есть по меньшей мере одна альтернатива, удовлетворяющая большинство избирателей, и он достаточно эффективен при решении широкого класса задач. Однако в ходе работы данного алгоритма могут возникать такие ситуации, когда две или более альтернатив получили одинаковое количество голосов. В таких случаях классический алгоритм голосования принимает в качестве верного варианта решения один из этих результатов, выбранный случайным образом [30].

Потенциальная возможность выбора неверной альтернативы в случае наличия нескольких альтернатив с одинаковым максимальным числом голосов значительно снижает надёжность работы данного алгоритма голосования в мультиверсионных ИУС, поскольку, чем больше окажется равнозначных групп, тем ниже будет вероятность выбора именно верного ответа. В лучшем случае вероятность выбора верной альтернативы будет составлять 0,5 (если выбор происходит между двух альтернатив).

Однако указанный недостаток может быть нивелирован с помощью модификации алгоритма голосования согласованным большинством путём введения критериев, позволяющих сравнивать между собой группы альтернатив (мультиверсий) и однозначно производить выбор между ними. В качестве такого критерия вводится мера диверсифицированности мультиверсий, входящих в программный модуль. Причём, чем больше частных метрик будет в себе содержать эта мера диверсифицированности, тем более точным окажется результат сравнения мультиверсий [10].

Стоит отметить, что попытки модификации алгоритма голосования согласованным большинством для решения описанной проблемы путём ввода дополнительных критериев уже производились, например в работе [36], где в качестве критерия используется надёжность каждой из групп мультиверсий. В

данном подходе надёжность (вероятность безотказной работы) вычисляется на основании априорных показателей надёжности каждой версии, что не всегда может соответствовать реальной надёжности мультиверсий, поскольку в таких расчётах зачастую не учитываются потенциальные межверсионные зависимости, возникающие в процессе исполнения программного обеспечения [71].

Таким образом, если применить описанную методику к модификации алгоритма голосования согласованным большинством, то этапы модифицированного алгоритма будут выглядеть так, как показано ниже:

1. Сбор результатов работы всех мультиверсий;
2. Объединение в группы совпадающих результатов;
3. Подсчёт количества результатов в каждой группе;
4. Если найдена группа, количество совпадающих результатов в которой превышает количество совпадающих результатов в других группах, то результат, этой группы принимается как верный – завершение работы алгоритма. Если найдено несколько групп с равным максимальным количеством результатов, то переход к шагу 5;
5. Производится выбор варианта ответа с использованием описанной модели мультиверсии – в качестве верного варианта ответа выбирается ответ из той группы, в которую входят наиболее различные между собой мультиверсии, поскольку вероятность возникновения в них схожих или зависимых ошибок ниже. Если меру разнообразия определить не удалось (группы версия являются клонами), или значения мер идентичны, то предположительно верный ответ выбирается случайным образом [10].

Кроме того, в графическом виде модифицированный алгоритм представлен на рисунке 5. Добавленные в классический алгоритм шаги выделены пунктирными линиями.

Таким образом, был модифицирован алгоритм голосования относительным большинством путём применения в ходе голосования алгоритма оценки диверсифицированности с целью обеспечения его более надёжной работы в условиях неопределённости.

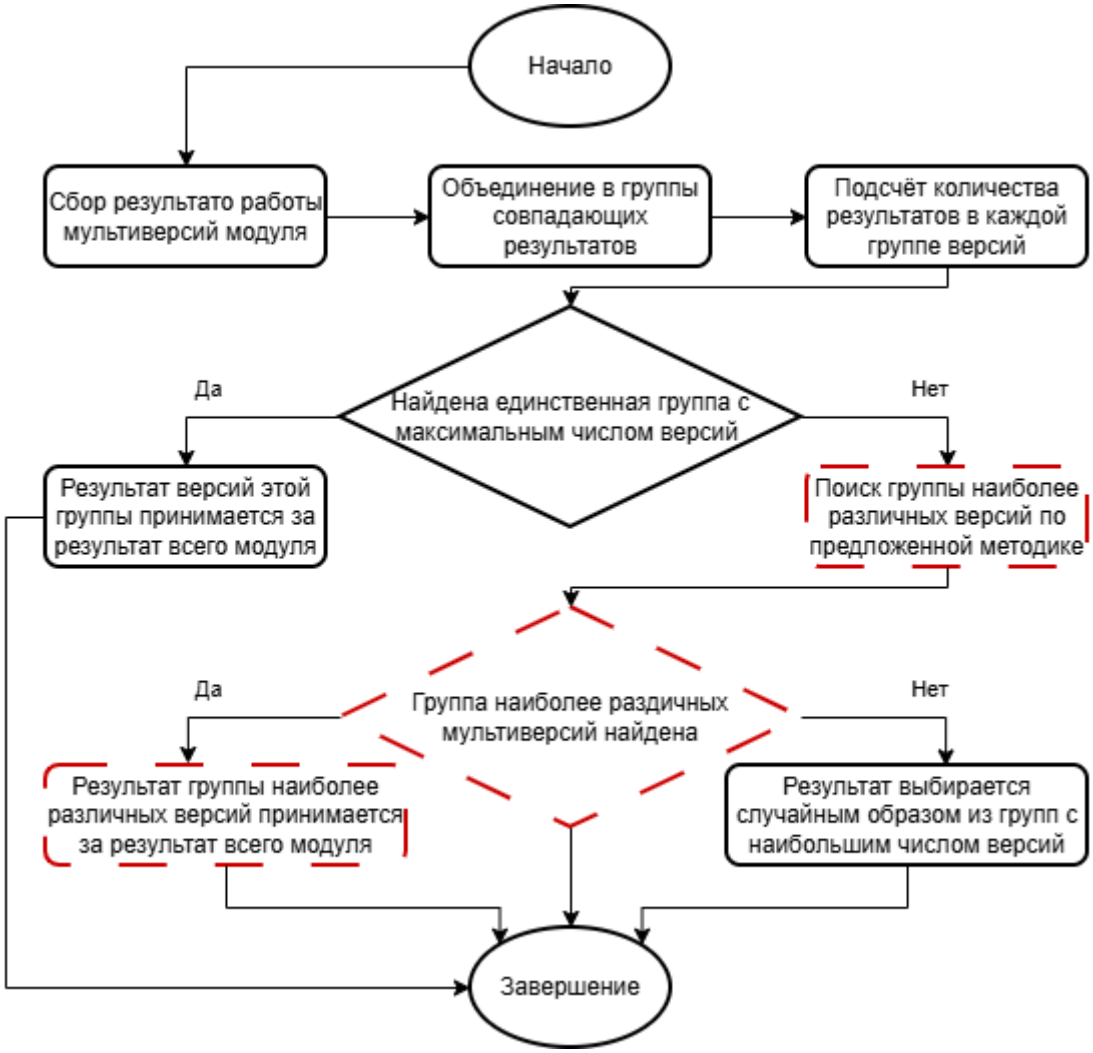

Рисунок 5 – Модифицированный алгоритм голосования согласованным большинством

3.3.2 Результаты эксперимента

Для проверки модифицированного алгоритма был проведён модельный эксперимент. Для его постановки была реализована программа, имитирующая работу мультиверсионной ИУС, которая по заданным правилам генерировала результаты работы мультиверсий для определённого числа запусков. После чего запускались алгоритмы голосования для выбора верного решения.

В имитирующей программе было определено 7 мультиверсий, из которых составлялись программные модули. Версии определялись следующим набором

атрибутов: априорная надёжность, диапазон возвращаемых значений, диапазон изменения метрики диверсифицированности на уровне алгоритмов и значение метрики диверсифицированности на уровне языков программирования, которая известна ещё до запуска мультиверсий – все эти атрибуты использовались для имитации работы мультиверсий, то есть для генерации их выходов (ответов).

Мультиверсии, помещённые в модули фактически подготавливали (генерировали) данные для проведения эксперимента, то есть исследования алгоритмов голосования.

Для имитации работы мультиверсионного модуля на каждом шаге его работы случайным образом генерировалось значение, которое принималось за верный ответ. Далее мультиверсии генерировали свои ответы. Заданная априорная надёжность мультиверсии определяла вероятность того, что мультиверсия выдаст верный ответ. Если ответ версии оказывался верным, то в качестве её выхода бралось сгенерированное ранее значение, иначе – производилась генерация неверного ответа.

Для генерации неверного ответа использовался следующий алгоритм: если это первый случай на текущей итерации работы мультиверсионного модуля, когда версия выдаёт неверный ответ, то её выходное значение генерируется случайным образом. Если текущая версия уже не первая, кто выдаёт неверный ответ, то вероятность её сбоя (её априорная надёжность вычитается из единицы) поочерёдно умножается на вероятность сбоя каждой из версий, уже выдавших неверный ответ. После каждого такого перемножения генерируется случайным образом число в диапазоне от 0 до 1. Если полученное число оказывается не больше полученного при перемножении вероятностей отказов значения, то сбой считается взаимозависимым – тогда за выход текущей версии принимается неверное значение той версии, с которой произошёл такой взаимозависимый сбой. Если ни с одной из уже выдавших ошибочное значение версий взаимозависимый сбой не случился, то неверное значение генерируется случайным образом.

В результате для проведения эксперимента было составлено 3 мультиверсионных модуля:

1. Модуль 3 (содержит 3 мультиверсии);
2. Модуль 5 (содержит 5 мультиверсий);
3. Модуль 7 (содержит 7 мультиверсий).

Состав каждого из перечисленных модулей приведён в таблице 14.

Таблица 14 – Состав мультиверсионных модулей и характеристики входящих в них мультиверсий

Название версии	D_L	D_A	**Априорная надёжность**	**Модуль, содержащий версию**
V1	0,0	0,0237874	0,99	Модуль 3
V2	0,4	0,1758233	0,98	Модуль 3
V3	0,5	0,5477165	0,87	Модуль 3
V1	0,0	0,0370791	0,99	Модуль 5
V2	0,4	0,0525644	0,98	Модуль 5
V3	0,5	0,4929281	0,87	Модуль 5
V4	0,2	0,9258402	0,999	Модуль 5
V5	0,55	0,6253112	0,8	Модуль 5
V1	0,0	0,037391	0,99	Модуль 7
V2	0,4	0,1893516	0,98	Модуль 7
V3	0,5	0,4944692	0,87	Модуль 7
V4	0,2	0,8562333	0,999	Модуль 7
V5	0,55	0,5844127	0,8	Модуль 7
V6	0,8	0,1103062	0,95	Модуль 7
V7	0,55	0,7963656	0,77	Модуль 7

Важно отметить, что вместо диапазонов возможных значений метрики диверсифицированности на уровне алгоритмов (D_A) для каждой из мультиверсий приведено среднее значение этой метрики для всего набора её запусков в рамках указанного модуля. Это сделано для обеспечения большей наглядности.

Для проверки модифицированного алгоритма голосования и сравнения результатов его работы с другими алгоритмами голосования каждым составленным модулем было сгенерировано по 4 набора данных. Каждый из наборов содержит разное число запусков (итераций) мультиверсионного модуля:

1. Набор 1 – 10 запусков;
2. Набор 2 – 100 запусков

3. Набор 3 – 1000 запусков;
4. Набор 4 – 10000 запусков.

После имитации запусков каждого из модулей были получены метрики диверсифицированности на уровне алгоритмов для каждой из версий, входящих в модуль. На основании этих метрик были посчитаны расстояния между мультиверсиями модуля, а результаты расчёта помещены в матрицы смежности, которые приведены в таблицах 15, 16 и 17 для «Модуля 3», «Модуля 5» и «Модуля 7», соответственно.

Таблица 15 – Матрица смежности версий в метрическом пространстве для «Модуля 3»

	V1	**V2**	**V3**
V1	0	0,59	1,00
V2	0,590865	0	0,531745
V3	1	0,531745	0

Таблица 16 – Матрица смежности версий в метрическом пространстве для «Модуля 5»

	V1	**V2**	**V3**	**V4**	**V5**
V1	0	0,439413	0,74272	1	0,883993
V2	0,439413	0	0,495699	0,983423	0,649914
V3	0,74272	0,495699	0	0,578164	0,155338
V4	1	0,983423	0,578164	0	0,506398
V5	0,883993	0,649914	0,155338	0,506398	0

Таблица 17 – Матрица смежности версий в метрическом пространстве для «Модуля 7»

	V1	**V2**	**V3**	**V4**	**V5**	**V6**	**V7**
V1	0	0,44698	0,70766	0,88052	0,81032	0,83915	0,97912
V2	0,44698	0	0,33541	0,72729	0,44143	0,42593	0,65317
V3	0,70766	0,33541	0	0,49094	0,10750	0,50917	0,31966
V4	0,88052	0,72729	0,49094	0	0,46293	1,00000	0,37092
V5	0,81032	0,44143	0,10750	0,46293	0	0,55989	0,22141
V6	0,83915	0,42593	0,50917	1,00000	0,55989	0	0,76277
V7	0,97912	0,65317	0,31966	0,37092	0,22141	0,76277	0

Также на основе приведённых выше матриц смежности был определён уровень диверсифицированности указанных модулей по нескольким показателям: минимальное значение, максимальное значение, среднее и медиана. Результаты этого расчёта приведены в таблице 18.

Таблица 18 – Меры диверсифицированности каждого модуля

	Минимум	**Максимум**	**Среднее**	**Медиана**
Модуль 3	0,531745	1,000000	0,707537	0,590865
Модуль 5	0,155338	1,000000	0,643506	0,614039
Модуль 7	0,107498	1,000000	0,573913	0,509169

Важно отметить, что минимальное и среднее значение уровня разнообразия мультиверсионного модуля снижается с увеличением числа версий в модуле. Это обусловлено тем, что с увеличением числа версий плотность их размещения в ограниченном интервале (от 0 до 1) увеличивается – сокращаются минимальные и средние расстояния между версиями. Поэтому при сравнении уровней диверсифицированности модулей важно учитывать количество версий в их составе.

Для каждого из наборов данных, сгенерированных каждым из мультиверсионных модулей, были запущены следующие алгоритмы голосования:

1. Голосование согласованным большинством;
2. Голосование согласованным большинством (модифицированная версия);
3. Медианное голосование;
4. Функция усреднения всех полученных результатов;
5. t/(n-1)-вариантный подход.

Результаты проведённого эксперимента приведены в таблице 19. Их отображение в графическом виде представлено на рисунке 6.

Таблица 19 – Сравнение эффективности работы алгоритмов голосования

	Верных ответов на число запусков				
Число запусков	**Среднее**	**АГСБ**	**АГСБ модифицированный**	**Медиана**	**t/(N-1)**
10000 (N = 7)	5109	9998	10000	9998	9996
1000 (N = 7)	516	1000	1000	999	999
100 (N = 7)	52	100	100	100	100
10 (N = 7)	5	10	10	10	10
10000 (N = 5)	6710	9995	10000	9994	9990
1000 (N = 5)	653	999	1000	1000	997
100 (N = 5)	61	100	100	100	100
10 (N = 5)	6	10	10	10	10
10000 (N = 3)	8456	9976	9976	9972	9965
1000 (N = 3)	873	1000	1000	1000	999
100 (N = 3)	84	100	100	99	99
10 (N = 3)	7	10	10	10	10

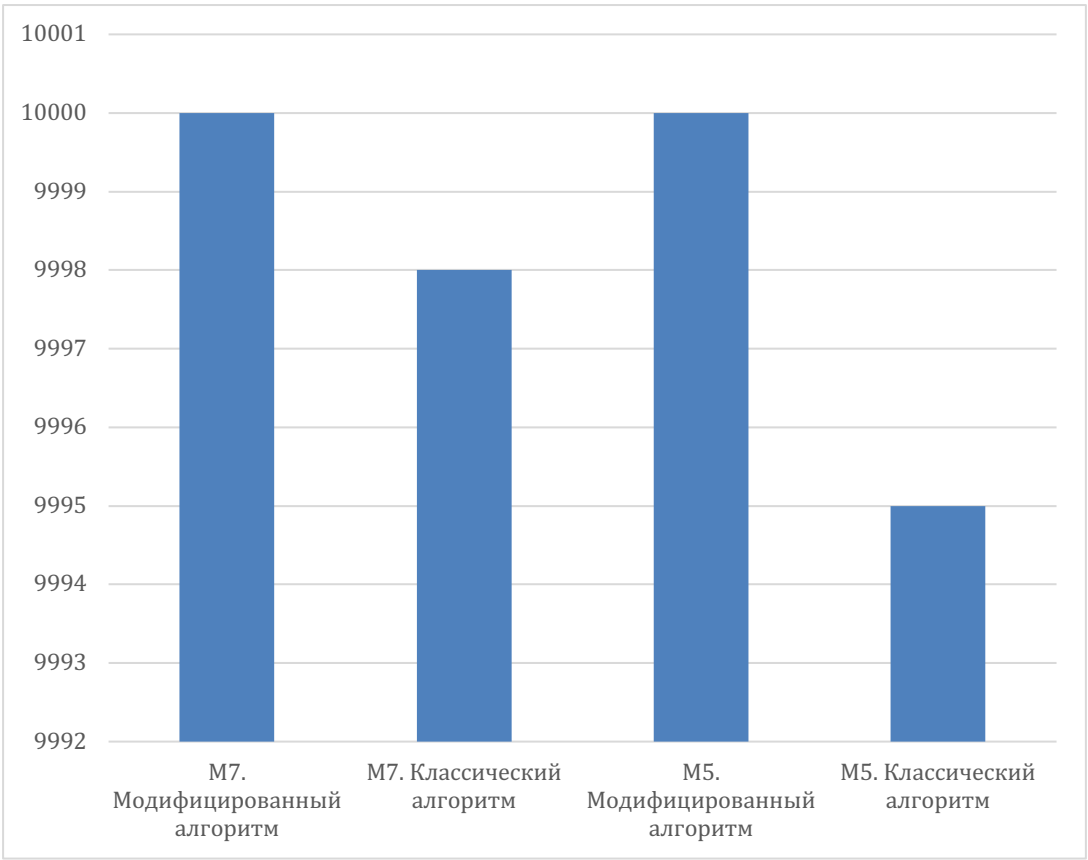

Рисунок 6 – Сравнение эффективности работы модифицированного и немодифицированного алгоритма голосования согласованным большинством

Модифицированный алгоритм голосования согласованным большинством обеспечил бóльшую надёжность при принятии решений в сравнении с немодифицированной версией. Например, в модуле из 5 версий на выборке из 10000 запусков модуля, в 5 случаях возникала ситуация неопределённости, корректно обработанная модифицированным алгоритмом, благодаря чему были выбраны действительно верные результаты. На 10000 запусках модуля, состоящего из 7 версий, подобная ситуация неопределённости возникла лишь 2 раза, что логично, поскольку вероятность выдачи эквивалентных некорректных ответов группой версий снижается по мере увеличения числа версий в группе. В данном случае модифицированный алгоритм также корректно отработал в ситуациях неопределённости и выбрал действительно верные варианты ответов.

Для подтверждения значимости полученных результатов был рассчитан критерий Уилкоксона. Для модифицированной и немодифицированной версий АГСБ сравнивались пары значений из выборок, содержащих данных о десяти тысячах запусков модулей, состоящих из 5 и 7 мультиверсий. В качестве значений выборок использовались 0 и 1, где 1 обозначает совпадение выбранного алгоритмом голосования ответа с верным ответом, который был известен заранее, 0 – несовпадение. Более формально это может быть выражено так: $M = \{m_1, m_2, \dots, m_{10000}\}$, $A = \{a_1, a_2, \dots, a_{10000}\}$, $C = \{c_1, c_2, \dots, c_{10000}\}$, $m_i \in M, c_i \in C, a_i \in A, i = \overline{1, 10000}$, $m_i = \begin{cases} 1, & a_i = c_i \\ 0, & a_i \neq c_i \end{cases}, i = \overline{1, 10000}$, где M – множество, хранящее отметки о том, был ли выбран корректный ответ алгоритмом голосования, A – множество ответов модуля, определённых алгоритмом голосования, C – множество заранее известных корректных ответов, ожидаемых от модуля на каждой итерации.

В множество M записывается 1, если алгоритм голосования выдал корректный ответ на i-й итерации, 0 – в противном случае.

В ходе такого преобразования в сформированной выборке для немодифицированного алгоритма голосования согласованным большинством присутствовали 5 и 2 нуля для модулей, состоящих из 5 и 7 версий, соответственно.

То есть алгоритм голосования согласованным большинством выбрал неверный результат в 5 случаях, когда модуль состоял из 5 мультиверсий, и в 2 случаях, когда модуль состоял из 7 мультиверсий. Остальные значение были равны единице, то есть в остальных случаях были выбраны верные ответы.

В ходе проведения расчётов были получены следующие результаты: $W_{M7} = 0.025$, $W_{M5} = 0.157$, где W_{M5} – значение вероятности критерия Уилкоксона для модуля, состоящего из 5 мультиверсий (немодифицированный алгоритм допустил ошибки в 5 случаях из 10000), W_{M7} – значение вероятности критерия Уилкоксона для модуля, состоящего из 7 мультиверсий (немодифицированный алгоритм допустил ошибки в 2 случаях из 10000). При пороговом значении в 5%, необходимом для принятия гипотезы, что модифицированный алгоритм голосования согласованным большинством более надёжен, в случае наличия 5 ситуаций неопределённости различия в выборках статистически значимы, в отличие от выборки с 2 ситуациями неопределённости.

Однако полученные после модификации алгоритма голосования согласованным большинством результаты являются значимыми, поскольку ИУС с мультиверсионным ПО применяются в таких сферах науки и техники, где негативный эффект от единственной ошибки, сбоя или отказа превышает полезный эффект от работы программного обеспечения за всё время использования. К таким негативным эффектам относятся гибель людей и значительные экономические потери. Поэтому повышение вероятности выбора верного ответа даже в одном или двух случаях из многотысячной выборки является значимым результатом.

Важно отметить, что эффективность модифицированного алгоритма голосования снижается по мере сокращения количества версий в модуле. Это видно на примере модуля из трёх мультиверсий, когда модифицированный алгоритм работает аналогично по сравнению с не модифицированным. В случае неопределённости образуется 3 равномощных множества, содержащих по одному ответу. В связи с чем применение метрики диверсифицированности теряет смысл, поскольку не предоставляет дополнительных критериев для сравнения мультиверсий. Таким образом, для повышения эффективности выбора верного

ответа модифицированному алгоритму голосования согласованным большинством необходимо не менее четырёх версий.

При этом, интересно отметить, что с уменьшением количества версий в модуле растёт эффективность функции усреднения для выбора верного ответа.

Кроме того, важно рассмотреть и сами случаи неопределённости, в которых немодифицированный алгоритм голосования дал сбой.

Так модулем из 5 мультиверсий было сгенерировано на 10000 запусков 5 случаев неопределённости. А модулем из 7 мультиверсий на 10000 запусков – 2 подобных случая. Эти случаи неопределённости приведены в таблицах 20 и 21 для «Модуля 5» и «Модуля 7», соответственно. Рассматривать случаи неопределённости для модуля, состоящего из трёх мультиверсий, не имеет смысла по описанным выше причинам.

Таблица 20 – Случаи неопределённости, возникшие в ходе работы «Модуля 5»

Верный ответ	**Ответ V1**	**Ответ V2**	**Ответ V3**	**Ответ V4**	**Ответ V5**
379,6396	252,58656	379,6396	121,48467	379,6396	121,48467
981,8401	981,8401	241,09443	314,18883	981,8401	314,18883
393,17931	393,17931	352,88335	125,81738	393,17931	125,81738
631,73672	631,73672	52,85979	202,15575	631,73672	202,15575
837,10697	837,10697	993,70873	267,87423	837,10697	267,87423

Таблица 21 – Случаи неопределённости, возникшие в ходе работы «Модуля 7»

Верный ответ	**Ответ V1**	**Ответ V2**	**Ответ V3**	**Ответ V4**	**Ответ V5**	**Ответ V6**	**Ответ V7**
727,11	727,11	727,11	579,032	727,11	312,657	312,657	312,657
92,102	92,102	92,102	707,456	92,102	39,604	39,604	39,604

Как видно из таблицы 20, в первом случае версии V2 и V4 выдали действительно верный ответ, когда у версий V3 и V5 произошёл зависимый сбой. В остальных случаях работы «Модуля 5» зависимые сбои происходили также у версий V3 и V5, однако верные ответы выдавались уже версиями V1 и V4. В то же время в ходе работы «Модуля 7» возникло 2 случая неопределённости,

приведённых в таблице 21. В обоих случаях версии V1, V2 и V4 возвращали верные ответы, а у версий V5, V6 и V7 наблюдались зависимые сбои.

Таким образом, на основании полученных данных из таблиц 16, 17, 20 и 21 были посчитаны меры различия тех групп мультиверсий внутри указанных модулей, которые выдали верный ответ, и которые подверглись совестному сбою.

Мера различия мультиверсий из «Модуля 5», выдавших верный ответ, близка к единице (то есть к максимальному значению) и составляет 1 для версий V1, V4 и 0,983423 – для версий V2, V4. При этом расстояние между версиями V3, V5 составляет 0,155338. Рассчитывать средние, медианные и прочие значения не имеет смысла в данном случае, поскольку все эти значения будут совпадать, т.к. мера разнообразия определяется только для двух версий.

Для «Модуля 7» в качестве меры различия было принято считать среднее расстояние между мультиверсиями. В этом случае мера диверсифицированности между версиями, выдавшими верный ответ (V1, V2, V4) составляет 0,68493. А метрика диверсифицированности для группы версий, подвергшихся совместному (созависимому) сбою (V5, V6, V7) составляет 0,51469.

На основании вышеизложенного может быть сделан вывод, что принцип диверсифицированности подтверждён экспериментально – более высокий уровень диверсифицированности группы мультиверсий определяет её более высокую надёжность по сравнению с менее диверсифицированной группой с таким же числом мультиверсий.

Таким образом, проведённый эксперимент подтверждает эффективность алгоритма голосования согласованным большинством, модифицированного с помощью методики, основанной на применении комбинированной модели представления мультиверсий.

3.4 Выводы

В третьей главе описана разработка алгоритм оценки диверсифицированности мультиверсионных программных модулей, а также методика его применения для модификации алгоритмов голосования. Кроме этого, в данной главе приведена модификация алгоритма голосования согласованным большинством, описан эксперимент по оценке эффективности работы модифицированного алгоритма.

Алгоритм оценки диверсифицированности позволяет определять наиболее различные по набору критериев группы мультиверсий. Алгоритм включает комплексный анализ частных метрик диверсифицированности, формирование общей метрики диверсифицированности для каждой мультиверсии. На основе полученных данных производится вычисление евклидовых расстояний между мультиверсиями в метрическом пространстве, на основании чего вычисляется уровень диверсифицированности групп мультиверсий. В качестве значения диверсифицированности группы мультиверсий может быть взято среднее, медианное, максимальное, минимальное или любое другое расстояние между версиями в группе, которое удовлетворяем потребностям текущей задачи.

Методика модификации алгоритмов голосования, описанная в данной главе, позволяет использовать оценку диверсифицированности как дополнительный критерий при принятии решений в условиях неопределённости. Применение данной методики целесообразно в алгоритмах, где возможны ситуации равного распределения голосов, например, в алгоритме голосования согласованным большинством. Именно этот алгоритм голосования и был модифицирован с применением разработанной методики.

Модифицированный алгоритм в ситуации неопределённости (равного количества голосов у нескольких групп мультиверсий) в качестве дополнительного критерия использовал уровень диверсифицированности мультиверсий в сравниваемых группах, поскольку согласно базовому принципу МВПО, чем более

различны версии, тем ниже вероятность у них совместных сбоев. Поэтому модифицированный алгоритм принимал в качестве верного ответа значение от той группы мультиверсий, которая имеет более высокий уровень диверсифицированности.

Экспериментальная проверка предложенных решений проводилась на модели мультиверсионной информационной управляющей системы с различным количеством версий. Для проведения экспериментов в имитационной среде были сформированы три мультиверсионных модуля, содержащие 3, 5 и 7 версий соответственно. Для каждого модуля проводилось по 4 серии испытаний с различным количеством запусков: от 10, 100, 1000 и 10000 запусков.

Результаты проведённого эксперимента показали эффективность предложенного подхода. Модифицированный алгоритм голосования согласованным большинством продемонстрировал более высокую надёжность по сравнению с базовой версией, то есть более высокую вероятность выбора верного ответа. Преимущество было достигнуто при работе с модулями, содержащими 5 и 7 версий. При этом было установлено, что для эффективного применения модифицированного алгоритма необходимо наличие как минимум четырёх версий в модуле.

Проведённый анализ случаев неопределённости подтвердил практическую ценность использования оценки диверсифицированности. Группы мультиверсий с более высоким уровнем разнообразия демонстрировали более высокую надёжность и меньшую вероятность возникновения совместных сбоев.

На основании проведённого в данной главе исследования можно сделать следующие выводы:

1. Разработанный алгоритм оценки диверсифицированности позволяет эффективно определять наиболее различные группы мультиверсий и может применяться как на этапе проектирования, так и в процессе эксплуатации мультиверсионных систем;
2. Методика модификации алгоритмов голосования с использованием оценки диверсифицированности повышает вероятность выбора верного ответа в

условиях неопределённости для модулей, состоящих более чем из трёх мультиверсий.

3. Результаты эксперимента подтверждают на практике приведённые теоретические выкладки.
4. Подтверждён базовый принцип мультиверсионного ПО, что группы мультиверсий с более высоким уровнем диверсифицированности обладают большей надёжностью.

Таким образом, был разработан алгоритм оценки диверсифицированности для блоков принятия решений в мультиверсионных ИУС, который находит группу наиболее различных по набору заданных критериев мультиверсий или идентичные группы мультиверсий (клоны), р**азработана методика модификации алгоритмов голосования с применением новой модели описания мультиверсий, которая позволяет повысить вероятность выбора корректного результата в условиях неопределённости**, **модифицирован алгоритм голосования согласованным большинством путём применения в ходе голосования алгоритма оценки диверсифицированности с целью обеспечения его более надёжной работы в условиях неопределённости. Кроме того, был проведённый эксперимент подтверждает эффективность алгоритма голосования согласованным большинством, модифицированного с помощью методики, основанной на применении комбинированной модели представления мультиверсий.**

4 Практическое применение МВПО в ИУС и анализ полученных результатов

Описанные в работе результаты были применены в реальной лабораторной информационной управляющей системе в аналитическом центре ОАО «Красцветмет».

4.1 Общее описание процесса анализа на базе лабораторной ИУС

В целом, лабораторные исследования могут проводиться с различными целями. Например, для первичного анализа сырья, поступившего на производство, или для определения технологического процесса, по которому оно пойдёт (в зависимости от его состава), или для расчёта с поставщиком данного сырья. Анализ готовой продукции производится для установления соответствия параметров полученного продукта требованиям нормативных документов (ГОСТу, техническому условию, спецификации и т.д.). Анализ какого-либо промежуточного изделия или материала может производиться прямо во время прохождения техпроцесса для оперативного управления данным процессом. Кроме того, получение результата анализа может быть самой конечной целью проведения этого анализа, как, например, бывает в медицинских или научных лабораториях.

Деятельность любой лаборатории строго регламентирована набором нормативных документов, среди которых могут быть, как локальные стандарты для аттестации готовой продукции и методики проведения анализов, так и государственные нормативные документы того же назначения. То, документами какого уровня регламентируется работа лаборатории, зависит от наличия у лаборатории международной или национальной аккредитации. В случае наличия у лаборатории аккредитации, методики анализа, которые в ней используются, должны быть зарегистрированы в национальном или международном органе аккредитации. В России Федеральная служба по аккредитации (Росаккредитация)

является федеральным органом исполнительной власти, осуществляющим функции национального органа Российской Федерации по аккредитации [54].

Анализы от аккредитованных лабораторий признаются легитимными во многих государственных и международных организациях, а готовая продукция предприятия, проанализированная в аккредитованных лабораториях, как правило, ценится выше и имеет более широкий рынок сбыта. Стоимость проведения анализа в аккредитованных лабораториях зачастую выше, чем в неаккредитованных, из-за чего потеря аккредитации может принести значительные экономические потери предприятию. Поэтому, в том числе все методики анализа, входящие в область аккредитации, должны быть надёжны в плане получения результатов и легко воспроизводимы, как на уже проанализированных образцах, так и на новых.

Для автоматизации рутинных процессов, а также более лёгкого прохождения аккредитаций в лабораториях зачастую используются системы класса LIMS или ЛИУС (Лабораторная Информационная Управляющая Система). Системы класса LIMS обычно позволяют покрыть если не все, то большинство процессов лаборатории, включая и процесс создания и использования методик проведения анализов.

Стоит отметить, что, если говорить укрупнённо, все методы анализа можно условно разделить на два класса: измерительные и расчётные. Измерительные методы анализа предполагают получение конечного результата сразу после проведения каких-либо измерений. Такие анализы могут проводиться несколькими лаборантами, каждый из которых выполняет несколько параллельных измерений. Конечным результатом считается среднее арифметическое всех измерений, их медиана или ещё какая-либо обобщающая математическая функция. Расчётные методы анализа предполагают проведение более сложных расчётов, чем применение обобщающих функций, для получения конечного результата.

Если надёжность работы (правильность получаемых результатов) измерительных методов анализа определяется в большей степени исправностью, точностью, состоянием (пройдены ли поверка и калибровка) и другими параметрами средств измерения, а также квалификацией персонала, проводящего

анализ, то надёжность расчётных методов анализа во многом зависит от надёжности программной реализации алгоритмов обработки входных данных и расчёта конечного значения, а также от целесообразности применения этих алгоритмов для решения каждой конкретной задачи.

Как раз для повышения надёжности расчётных методов анализа может быть применена хорошо себя зарекомендовавшая на практике и одна из наиболее известных методологий повышения надёжности – методология мультиверсионного программирования.

Информационная надёжность результатов расчётных методов анализа, т.е. их корректность, в некоторых ситуациях бывает важна критически. Например, при анализе крупной партии сырья (несколько тонн), содержащего драгоценные металлы, некорректное определение содержаний элементов (пусть даже разница с реальными значениями будет менее чем 1%) может привести к значительным экономическим и репутационным потерям. Корректность результатов расчётных анализов становится особенно важна, когда, например, анализ подобных крупных партий сырья производится не единоразово, а на регулярной основе. Данное обстоятельство обуславливает необходимость повышения надёжности программных реализаций расчётных методов анализа.

4.2 Описание объекта применения результатов

С ОАО «Красцветмет» в части аффинажа драгоценных металлов (ДМ) сотрудничает достаточно большое количество контрагентов, которые поставляют своё сырьё на переработку. В целях повышения конкурентоспособности и клиентоориентированности компания постоянно стремится к сокращению сроков расчёта с поставщиками, что достигается в том числе сокращением сроков анализа за счёт развития экспрессных методов анализа, без снижения точности. Каждый контрагент, поставляющий на предприятие ювелирные сплавы, желает получить быстрый и точный расчет за содержащиеся в материале драгоценные металлы [55].

Классическая высокоточная методика пробирного анализа золота занимает порядка 8 часов. Применяемая методика рентгеноспектрального анализа платины, палладия и серебра в сплавах на основе золота позволяет получить результаты анализа не более чем за 1 час, точность которых сопоставима с точностью результатов пробирного анализа [55].

Объектом анализа описываемой методики являются, принятые у физических лиц (в своём большинстве) ювелирные изделия, состоящие из сплавов на основе золота. Химический состав этих сплавов определён ГОСТ 30649-99 «Сплавы на основе благородных металлов ювелирные. Марки». В поступающем на переработку вторичном сырье в произвольных соотношениях содержатся изделия разной пробности (375, 500, 585, 750, 958 и 999,9), а также разных марок. Описываемая методика предназначена для анализа сплавов с содержанием золота от 56.0% до 63.0% [55].

Аналитическая проба поступившего в переработку вторичного сырья отбирается вакуумным способом после приемной плавки. Далее проводятся процедуры, предусмотренные схемой опробования, после чего на рентгеноспектральный анализ поступает проба, состоящая из двух стержней диаметром 6 мм и длиной 12-15 мм. Измерения интенсивностей аналитических линий производятся на рентгеновском спектрометре Axios фирмы PANalytical (Нидерланды) с Mo-анодом [55].

4.3 Описание методики рентгенофлуоресцентного анализа

В данном пункте приведено описание методики, для реализации которой были применены результаты диссертационного исследования.

Сложный химический состав анализируемого материала, а также широкий диапазон изменений содержаний элементов, входящих в него, обуславливают необходимость применения в описываемой методике уравнения связи для учета

межэлементных взаимодействий. В качестве уравнения связи использована линейная регрессия вида [55]:

$$C_{Au} = a_0 + \sum_{n=1}^{N} a_n I_n, \quad (3.1)$$

где $N = |M| = 8$ – число элементов в пробе;

$M = \{Au, Cu, Ag, Zn, Ni, Pd, Pt, Cd\}$ – множество элементов, содержащихся в пробе;

C_{Au} – содержание золота в пробе;

I_n – интенсивность текущего элемента (из множества M) в пробе;

a_0 и a_n – коэффициенты регрессии.

Стоит отметить, что кадмий (*Cd*) и платина (*Pt*) в более чем в 90% проб отсутствуют, поскольку лишь в одной из из 27 марок сплавов, указанных в ГОСТ 30649-99, присутствует платина, а кадмий может содержаться только в 2-х марках сплава. Смешение различных марок сплавов ювелирных изделий, принятых у населения, ещё больше снижает содержание этих элементов в конечном материале. Поэтому влияние *Pt* и *Cd* на точность результатов анализа становится пренебрежимо мало. Поэтому интенсивности данных компонентов были исключены из уравнения линейной регрессии. А пробы с интенсивностью платины или кадмия, превышающими заданное значение, направлять на определение содержания золота пробирным методом анализа. Соответственно, выбранное уравнение связи интенсивностей излучения компонентов и содержания золота в пробе приняло следующий вид [55]:

$$C_{Au} = a_0 + a_{Au} I_{Au} + a_{Ag} I_{Ag} + a_{Cu} I_{Cu} + a_{Zn} I_{Zn} + a_{Ni} I_{Ni} + a_{Pd} I_{Pd}. \quad (3.2)$$

Поскольку химический состав, поступающего от поставщиков вторичного сырья, формируется путём смешивания случайным образом различных марок ювелирных сплавов на основе благородных металлов, подбор подходящих по составу градуировочных образцов для расчета коэффициентов уравнения регрессии по формуле (3.2) оказывается практически невозможным. Поэтому в качестве градуировочных образцов используются пробы материалы, анализ которых проводится в рамках текущей работы [55].

При проведении такого анализа информацию о пробах фиксируется в базе данных (БД) лабораторной информационной управляющей системы предприятия. К фиксируемым данным относятся: номер пробы, содержание золота, полученное пробирным методом анализа, и интенсивности рентгеновского излучения элементов, входящих в химический состав пробы. Содержание золота в пробах определяется пробирным методом анализа с погрешностью 0,09% абс. во всем диапазоне содержаний. Измерение интенсивностей рентгеновского излучения образцов, также проводятся с высокой достоверностью на спектрометре Axios фирмы PANalytical (Нидерланды), у которого относительное среднеквадратическое отклонение выходного сигнала не превышает 1%. В связи с тем, что материал проанализированной таким образом пробы физически не хранится (регулярно ссыпается в производство), пиведённую информацию о пробе, которая хранится в базе данных лабораторной ИУС (ЛИУС), можно считать виртуальным градуировочным образцом (ВГО) [55].

Для формирования системы линейных алгебраических уравнений, решение которой позволит найти коэффициенты регрессии, производится выборка данных ВГО из базы данных ЛИУС. Для выборки к интенсивности излучения каждого компонента анализируемой пробы применяются фильтры (чтобы найти подобные пробы в уже имеющейся статистике). Так, например, выбираются строки из БД, интенсивности золота у которых отличаются от сравниваемой на величину $\pm\Delta^{Au}$ (называемую фильтром), то есть значения в промежутке от $I_{Au} - \Delta^{Au}$ до $I_{Au} + \Delta^{Au}$, где I_{Au} – интенсивность линии золота в анализируемой пробе. Подобным образом применяются фильтры и для остальных элементов, входящих в состав анализируемого образца. Выборка данных с применением всех фильтров производится с применением оператора логического «И», то есть проба из накопленного набора статистических данных включается в конечную матрицу при условии соответствия всем фильтрам [55].

Данные из полученной выборки составляют матрицу, являющуюся системой линейных алгебраических уравнений (СЛАУ), корнями которой являются коэффициенты зависимости. После приведения матрицы к виду, более пригодному

для использования методов решения СЛАУ (транспонирование, получение квадратной матрицы, формирование столбца свободных членов [7]) производится расчёт коэффициентов регрессии, то есть решение СЛАУ. Далее полученные коэффициенты регрессии (a_0 и a_n) подставляются в формулу (3.2), умножаются на соответствующие интенсивности и складываются между собой, что позволяет получить значение содержания золота в анализируемой пробе [55].

4.4 Мультиверсионная программная реализация методики рентгенофлуоресцентного анализа

В ходе реализации и внедрения результатов исследования на производстве был реализован процесс, описанный ниже. Его графическое представление приведено на рисунке 7.

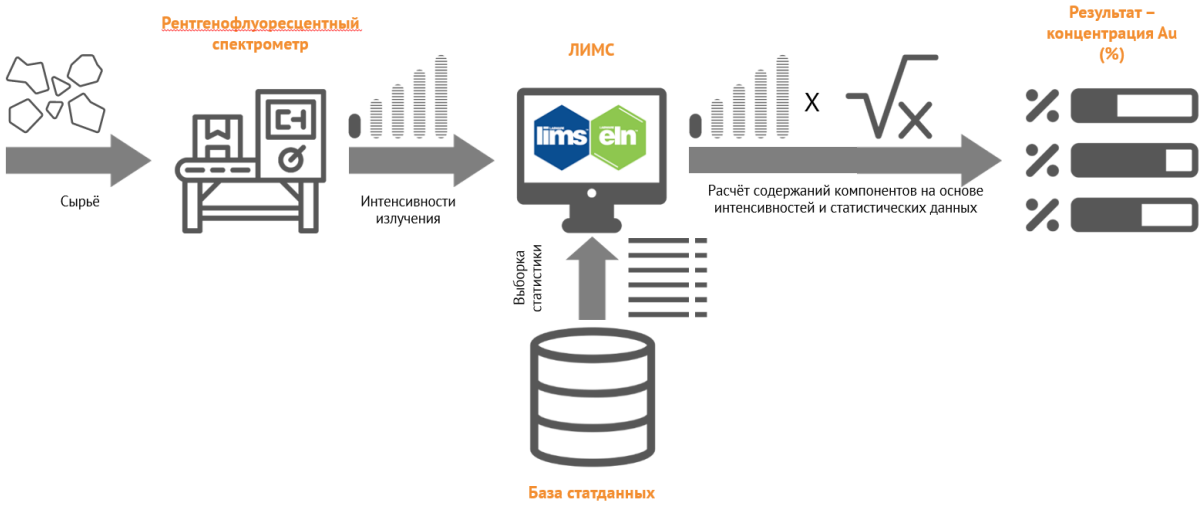

Рисунок 7 – Процесс проведения анализа рентгеноспектральным методом

Интенсивности аналитических линий элементов, снятые на рентгеновском спектрометре Axios, передаются по локальной сети предприятия в лабораторную информационную управляющую систему LIMS LabWare, где на основании этих сырых данных выполняется расчёт содержания золота в пробе [55]. Контекст, в

котором работает лабораторная информационная управляющая система, представлен на диаграмме контекста C4 на рисунке 8.

В ходе внедрения описанной методики были выявлены ситуации, в которых возникают сбои в процессе расчёта, или выдаётся некорректный результат. Оказалось, что такие ситуации возникают, когда на анализ попадает проба материала, химический состав которого по различным причинам не соответствует требованиям ГОСТ 30649-99 [55].

Для таких проб программа не может найти в БД необходимое для расчета коэффициентов количество аналогичных ВГО и сформировать конечную матрицу минимум из шести строк (по количеству неизвестных в СЛАУ). Кроме того, быстрые численные методы решения СЛАУ иногда оказывались недостаточно точны, также, метод Якоби, например, имеет ограничение, позволяющее решать только СЛАУ, матрицы которых имеют диагональное преобладание [34].

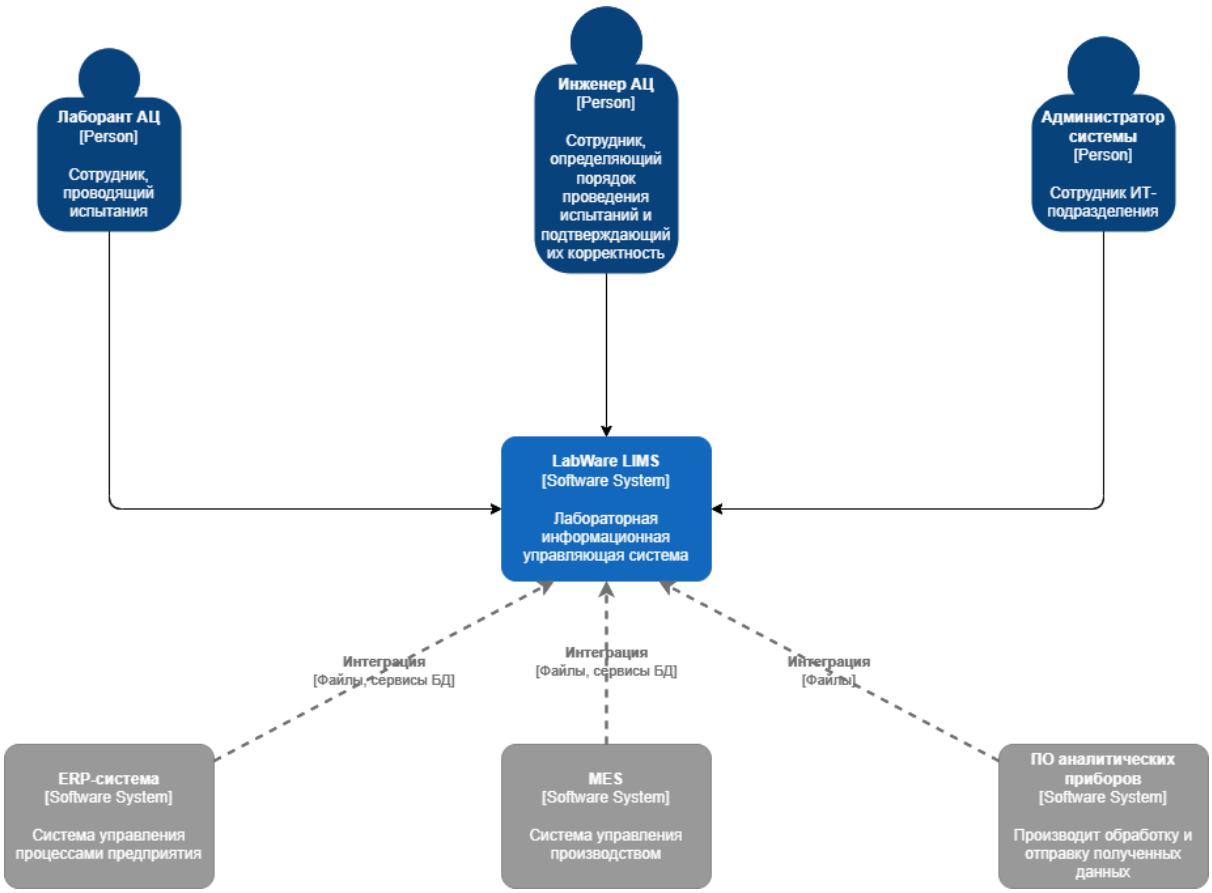

Рисунок 8 – Диаграмма контекста лабораторной ИУС

Мультиверсионный модуль и среда исполнения были реализованы в лабораторной ИУС, поскольку именно в ней концентрируются относящиеся к анализам и пробам данные с аналитических приборов и смежных систем, кроме того, она является единой точкой входа для всех пользователей, работающих в аналитическом центре (АЦ) [55].

Сбои в лабораторной информационной системе недопустимы, поскольку время на восстановление работоспособности после их возникновения является критичным для работы аналитического центра из-за достаточно короткого срока выполнения анализа, продолжительность которого регламентирована на уровне руководства предприятия. Данное обстоятельство обусловило необходимость использования высоконадёжной программной реализации методики анализа в существующей информационной системе [55].

Среди прочих подходов повышения надёжности программного обеспечения [21], [20], [63] была выбрана методология мультиверсионного программирования, как один из наиболее популярных, детально изученных и хорошо себя зарекомендовавших подходов к повышению надёжности программных систем.

Концепция применения мультиверсионного программного обеспечения довольно хорошо вписывается в специфику решаемой задачи, поскольку системы линейных алгебраических уравнений, корни которых должны быть найдены, могут быть сформированы разных видов из-за разнородности поступающих на анализ материалов, как от одного, так и от различных поставщиков, поэтому использование различных алгоритмов для их решения оказалось целесообразным.

Состав разработанной системы анализа и обработки рентгеноспектральной информации в рамках лабораторной ИУС в виде UML-диаграммы компонентов представлен на рисунке 9. Описание реализованной системы приведено ниже.

Для формирования оптимального состава мультиверсий программного модуля была применена компенсационная многоатрибутивная модель выбора оптимального состава мультиверсионного программного обеспечения [19].

Таким образом, состав мультиверсий программного модуля стал включать в себя 4 версии. Каждая из версий реализует свой алгоритм расчёта, а язык её

реализации уникален, то есть не совпадает с языками реализации других мультиверсий данного модуля.

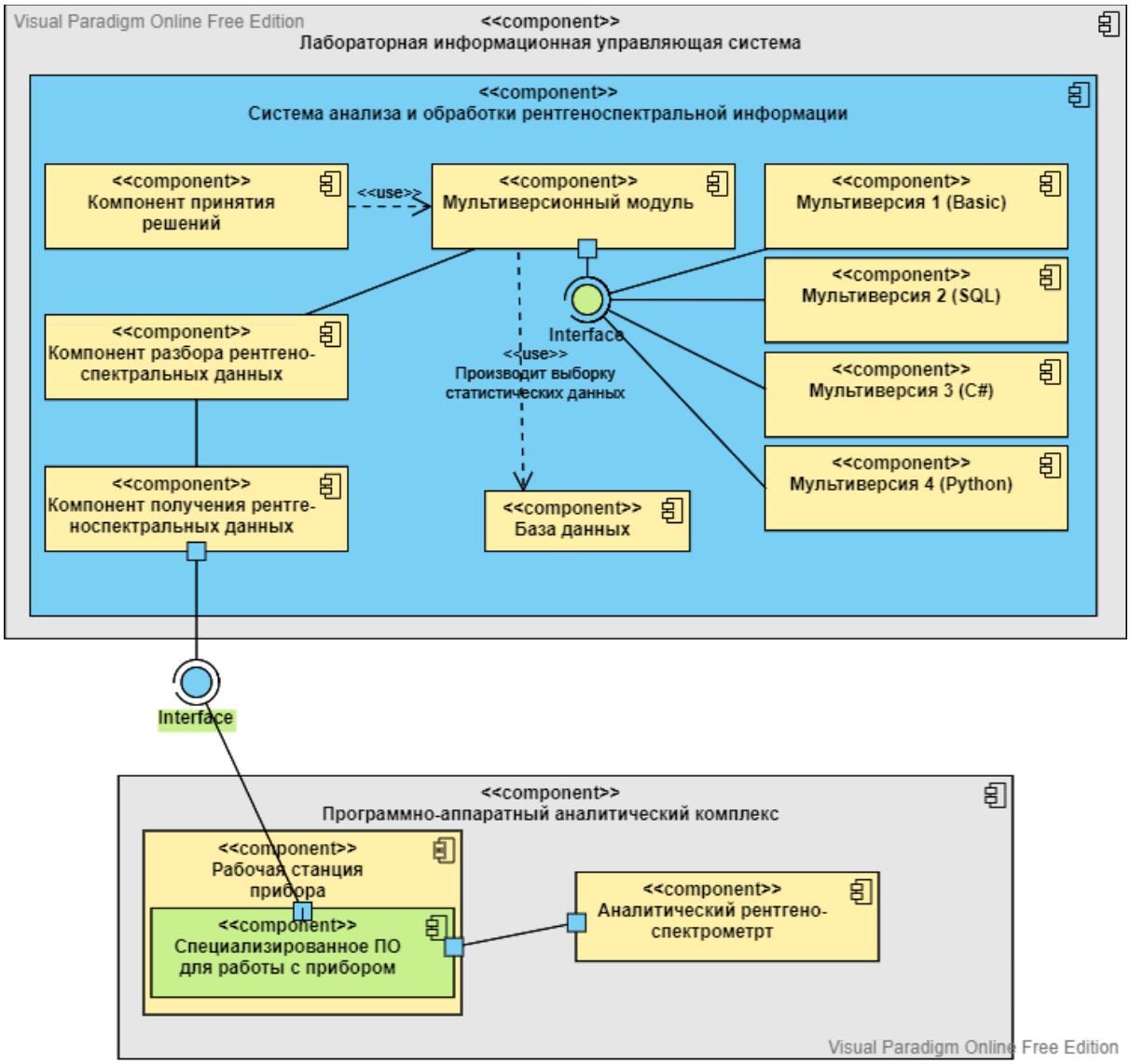

Рисунок 9 – Диаграмма компонентов системы анализа и обработки рентгеноспектральной информации

Мультиверсия, использующая метод Гаусса для решения СЛАУ, была реализована непосредственно внутри самой лабораторной информационной управляющей системы на внутреннем языке программирования – LIMS Basic. Оставшиеся три мультиверсии были интегрированы в LIMS. Метод Гаусса-Жордана был реализован на SQL, матричный метод – на языке C#. А мультиверсия, написанная на Python, использовала для решения СЛАУ уже готовый компонент linalg из пакета numpy [55].

Использование диверсифицированных мультиверсий, как на уровне алгоритмов, так и на уровне языков программирования, стало возможным, благодаря встроенным механизмам загрузки и использования сторонних программных модулей и библиотек в лабораторной информационной управляющей системе предприятия.

Для формирования метрики диверсифицированности на уровне языков программирования в ходе реализации мультиверсионного модуля были приняты в качестве наиболее существенных следующие свойства языков программирования (критерии сравнения):

- Парадигма;
- Типизация;
- Механизмы управления памятью;
- Механизмы управления потоком вычислений.

Значения, доступные каждому из этих критериев, соотнесённые с языками программирования, на которых были реализованы мультиверсии для лабораторной ИУС, приведены в таблице 22.

Таблица 22. Значения свойств языков программирования, использовавшихся в написании мультиверсионного модуля для лабораторной ИУС

Критерий	**Значение**	**Python**	**Lims Basic**	**T-SQL**	**C#**
Парадигма	Императивная	+	+	+	+
Парадигма	Объектно-ориентированная	+	+	-	+
Парадигма	Функциональная	+	-	+	+
Парадигма	Рефлексивная	+	-	-	-
Парадигма	Обобщённое программирование	+	+	-	+
Парадигма	Логическая	-	-	-	-
Парадигма	Декларативная	+	-	+	-
Парадигма	Распределенная	-	-	-	-
Типизация	Статическая	-	+	+	+
Типизация	Динамическая	+	+	-	+
Типизация	Явная	+	+	+	+
Типизация	Неявная	+	+	-	-

Критерий	**Значение**	**Python**	**Lims Basic**	**T-SQL**	**C#**
Типизация	Неявное приведение типов без потери данных	+	+	-	+
Типизация	Неявное приведение типов с потерей данных	-	-	-	-
Типизация	Неявное приведение типов в неоднозначных ситуациях	-	-	-	+
Типизация	Алиасы типов	-	-	-	+
Типизация	Вывод типов переменных из инициализатора	-	+	-	+
Типизация	Вывод типов переменных из использования	-	+	-	-
Типизация	Вывод типов аргументов при вызове метода	-	-	-	+
Типизация	Вывод сигнатуры для локальных функций	-	-	-	-
Типизация	Параметрический полиморфизм	-	-	-	+
Типизация	Параметрический полиморфизм с ковариантностью	-	-	-	+
Типизация	Параметрический полиморфизм высших порядков	-	-	-	-
Типизация	Информация о типах в runtime	+	-	+	+
Типизация	Информация о типах-параметрах в runtime	+	-	+	+
Управление памятью	Создание объектов на стеке	-	+	-	+
Управление памятью	Неуправляемые указатели	-	-	-	+
Управление памятью	Ручное управление памятью	-	+	+	+
Управление памятью	Сборка мусора	+	-	-	+
Управление потоком вычислений	Инструкция goto	-	+	-	+
Управление потоком вычислений	Инструкции break без метки	+	+	-	+
Управление потоком вычислений	Инструкция break с меткой	-	-	-	-

Критерий	**Значение**	**Python**	**Lims Basic**	**T-SQL**	**C#**
Управление потоком вычислений	Поддержка try/catch	+	+	-	+
Управление потоком вычислений	Блок finally	+	+	-	+
Управление потоком вычислений	Блок else (исключения)	+	-	-	-
Управление потоком вычислений	Перезапуски	-	-	-	-
Управление потоком вычислений	Ленивые вычисления	+	-	-	-
Управление потоком вычислений	Continuations	-	-	+	+
Управление потоком вычислений	Легковесные процессы (Coroutines)	+	-	-	-

На основании данных, приведённых в таблице 22 по формулам (2.5), (2.4) и (2.6) были рассчитаны метрики диверсифицированности на уровне языков программирования для каждой из описанных мультиверсий, то есть их удалённость от начала координат, которым является «идеальный» язык программирования.

Результаты расчёта:

– $D_L(\text{Версия } 1) = 0{,}57831$;

– $D_L(\text{Версия } 2) = 0{,}75993$;

– $D_L(\text{Версия } 3) = 0{,}32353$;

– $D_L(\text{Версия } 4) = 0{,}51176$;

Поскольку метрика диверсифицированности на уровне алгоритмов пересчитывается после каждого запуска, её приведение в диссертации представляется нецелесообразным. Однако важно указать диапазон, в котором она изменялась: от 0,323529 до 0,999995.

Таким образом, у каждой мультиверсии появляются две координаты в метрическом пространстве после каждого запуска и исполнения мультирвесий модуля. По этим координатам определяется Евклидово расстояние между сравниваемым миультиверсиями и используется в качестве дополнительного критерия в случае неопределённости..

Для выбора верного ответа из набора значений, выданных всеми мультиверсиями, применяется алгоритм голосования согласованным большинством, модифицированный по описанной в предыдущей главе методике. В рамках применяемого алгоритма формируется модель каждой мультиверсии, то есть определяются её координаты в двумерном метрическом пространстве, как описано выше. Благодаря этому в случае выдачи двумя группами по две мультиверсии различающихся значений в качестве конечного варианта ответа принимается то значение, которое было получено группой наиболее диверсифицированных версий, то есть расположенных наиболее далеко друг от друга в метрическом пространстве [55].

На базе лабораторной информационной управляющей системы была реализована среда исполнения мультиверсионного ПО. Данная среда предоставляет возможности поочерёдного запуска мультиверсий, считывания промежуточных значений во время работы мультиверсий в контрольных точках работы алгоритмов, а также запуск алгоритма голосования. Разумеется, для изменения состава мультиверсий нет необходимости как-либо изменять или переразворачивать всю ИУС – достаточно просто подключить внешний модуль, представленный в виде exe- или dll-файла [55].

4.5 Результаты внедрения

В итоге модифицированный алгоритм голосования согласованным большинством показал свою эффективность, что подтверждается актом об использовании результатов диссертации. Число проб ювелирных сплавов на основе

золота, переданного на рентгеноспектральный анализ, увеличилось с 68,2% до 89,3% от общего числа проб. Сравнение производственных показателей, изменившихся после внедрения модифицированного алгоритма, приведено в таблице 23.

Таблица 23 – Производственные показатели, изменившиеся после внедрения модифицированного алгоритма голосования

Производственный показатель	**Вид улучшения**	**Значение**
Стоимость выполнения одного анализа	Снижена на	430 ₽
Срок анализа вторичного сырья	Сократился в сравнении с пробирным анализом в	5 раз
Количество ручных операций	Сокращено на	~80%
Время расчёта с поставщиками	Сокращено до	1 сутки

Как видно из результатов в таблице 22, мультиверсионный подход в совокупности с представленными результатами диссертационного исследования позволил достичь значимых экономических результатов для организации. Внедрение описанного мультиверсионного программного модуля в лабораторную информационную управляющую систему подтверждено актом об использовании результатов кандидатской диссертационной работы (Приложение А).

Таким образом, был применён модифицированный алгоритм голосования в реальной информационной управляющей системе для подтверждения на практике полученных теоретических результатов.

4.6 Выводы

В рамках применения полученных в ходе диссертационного исследования результатов был разработан мультиверсионный программный модуль, в последствии внедрённый в существующую лабораторную информационную управляющую систему аналитического центра ОАО «Красцветмет». Мультиверсионный модуль реализован как надстройка над лабораторной ИУС.

Разработанный мультиверсионный модуль включает четыре мультиверсии, каждая из которых реализовывала свой собственный алгоритм вычисления содержаний анализируемых компонентов в пробе по интенсивностям их излучения, полученным с рентгеновского спектрометра. Кроме того, все мультиверсии были реализованы на различных языках программирования:. первая версия реализована на внутреннем языке LIMS Basic, вторая – на SQL, третья – на C#, четвёртая – на Python.

Разработанная среда исполнения мультиверсионного ПО на базе лабораторной ИУС обеспечивает возможность последовательного запуска версий, считывания промежуточных значений и запуска модифицированного алгоритма голосования согласованным большинством. В случае возникновения ситуации неопределённости окончательное решение по выбору результата всего мультиверсионного модуля принималось в пользу той группы мультиверсий, чей уровень диверсифицированности выше. Результаты внедрения показали значительное производственных показателей.

На основании вышеописанного можно сделать следующие выводы:

1. Внедрение мультиверсионного подхода в существующую лабораторную информационную систему позволило повысить надёжность процесса анализа без необходимости полной замены базового программного обеспечения.

2. Модифицированный алгоритм голосования, учитывающий диверсифицированность версий, продемонстрировал свою эффективность в реальных условиях эксплуатации, что подтверждается значительными улучшениями производственных показателей.

Таким образом, в лабораторной информационной управляющей системе предприятия реализован мультиверсионный модуль с модифицированным (с применением комбинированной модели мультиверсии) алгоритмом голосования согласованным большинством, используемый для расчёта результатов рентгеноспектрального анализа золота. Результатом применения алгоритма стала возможность перевода части анализируемых номенклатур с

пробирного на рентгеноспектральный анализ, что в свою очередь позволило ощутить значительный экономический эффект.

ЗАКЛЮЧЕНИЕ

В рамках диссертационного исследования был выполнен обзор предметной области, в рамках которого также был проведён анализ моделей представления мультиверсий и мультиверсионных программных модулей, в ходе которого был сделан вывод об отсутствии формального аппарата представления, как мультиверсий, так и мультиверсионных модулей, позволяющего определить их меру различия.

Также был выполнен обзор подходов и методов принятия решений, сделаны выводы об их применимости в контексте выбора правильного ответа от множества мультиверсий в мультиверсионном программном обеспечении, конкретно о том, что наилучшим образом для применения в мультиверсионных ИУС подходят алгоритмы теории голосования, однако и они могут работать ненадёжно в условиях неопределённости, что обуславливает необходимость их развития и модификации с целью нивелирования этих недостатков.

С этой целью были разработаны новые модели описания мультиверсий и мультиверсионных программных модулей на основе метрики диверсифицированности, которые позволили формализовать процесс обеспечения разнообразия мультиверсионного программного обеспечения ИУС, а также управлять процессом выбора наиболее разнообразных групп мультиверсий с целью повышения надёжности мультиверсионной системы в целом.

На базе предложенных моделей разработан алгоритм оценки диверсифицированности для блоков принятия решений в мультиверсионных информационных управляющих системах, который позволяет определить группу наиболее различных между собой по набору заданных критериев мультиверсий.

Приведённый выше алгоритм лёг в основу новой разработанной методики модификации алгоритмов голосования с применением новой модели описания мультиверсий, которая позволяет повысить вероятность выбора корректного результата в условиях неопределённости.

По этой методике был модифицирован алгоритм голосования относительным большинством путём применения в ходе голосования алгоритма оценки диверсифицированности с целью обеспечения его более надёжной работы в условиях неопределённости, то есть, когда классический алгоритм голосования согласованным большинством мог бы выбрать лишь случайным образом верный ответ из нескольких равночисленных групп мультиверсий.

С целью реального применения и практического подтверждения полученных результатов в лабораторной информационной управляющей системе промышленного предприятия реализован мультиверсионный модуль с модифицированным (с применением комбинированной модели мультиверсии) алгоритмом голосования согласованным большинством, используемый для расчёта результатов рентгеноспектрального анализа золота. Результатом применения алгоритма стала возможность перевода части анализируемых номенклатур с пробирного на рентгеноспектральный анализ, что в свою очередь позволило ощутить значительный экономический эффект.

В результате работы над диссертацией все поставленные задачи были выполнены, а цель исследования достигнута.

Проведен анализ существующих подходов к выбору корректного результата в ИУС, который показал отсутствие формального аппарата для оценки меры различия версий. В результате выполнения этой и следующей задач была разработана новая комбинированная модель формального описания мультиверсий на основе интегральной метрики диверсифицированности. Данная модель позволяет перевести понятие «различия версий» из качественного в количественное, что обеспечивает основу для объективного принятия решений в мультиверсионных информационных систем.

Предложена новая графовая модель представления мультиверсионного программного модуля, в которой вершинами графа являются версии, а ребрами – связи диверсифицированности. Особенностью модели является возможность формального сравнения групп версий через анализ изоморфизма графов, что

позволяет верифицировать наличие «клонов» в модуле и оценивать структуру его разнообразия как единого объекта.

Разработан и описан алгоритм оценки диверсифицированности для блоков принятия решений. Он обеспечивает автоматизированный выбор групп наиболее различных версий из общего набора, используя Евклидовы расстояния в многомерном пространстве метрик. Это позволяет блоку голосования с большей вероятностью выбрать действительно верный ответ в случае неопределённости.

Предложена методика модификации алгоритмов голосования на основе комбинированной модели представления мультиверсий для решения задач голосования в условиях неопределенности.

На основе предложенной методики модифицирован алгоритм голосования согласованным большинством, который в случаях равенства голосов использует уровень различия версий как дополнительный критерий для принятия решения, предотвращая случайный выбор и повышая вероятность получения верного ответа.

Модифицированный алгоритм реализован в реальной ИУС ОАО «Красцветмет» для рентгеноспектрального анализа золота. Внедрение показало рост доли корректно обработанных образцов, что подтверждено экономической эффективностью и сокращением трудозатрат в производственном цикле.

Результаты исследования рекомендуются к применению в задачах оценки качества мультиверсионного программного обеспечения, формирования его оптимального состава с учётом меры разнообразия, а также в задачах поддержки принятия решений.

Дальнейшая разработка темы будет направлена на дальнейшую модификацию алгоритмов голосования и их практическую реализацию. Дальнейшее развитие данного направления связано с введением в модель мультиверсии новых частных метрик диверсифицированности и применением более широкого круга методов из теории графов, направленных на более глубокое и многостороннее определение степени разнообразия групп мультиверсий.

СПИСОК ЛИТЕРАТУРЫ

1. Анич, И. Метод ЭЛЕКТРА и проблема ацикличности отношений и альтернатив / И. Анич, О.И. Ларичев // Автоматика и телемеханика. - 1996. - №8. - С.108-118

2. Багов М.А., Кудаев В.Ч. Математическое моделирование и оптимизация трубопроводной сети Штейнера //Известия Кабардино-Балкарского научного центра РАН. – 2017. – №. 1 (75). – С. 5-11.

3. Бождай А. С., Горшенин Л. Н. Обзор и анализ подходов к классификации объектов с гетерогенным набором информационных признаков //Известия высших учебных заведений. Поволжский регион. Технические науки. – 2024. – №. 2 (70). – С. 47-57.

4. Васильев Н. Метрические пространства //Квант. – 1990. – №. 1. – С. 16–23.

5. Верютина В. В., Дзюба А. Г., Король М. А. Классификация данных методом K-ближайших соседей (KNN) //Мировая наука в эпоху социально-политических трансформаций: новые возможности, пути развития. – 2022. – С. 37-40.

6. Войтицкий В. И. Средние функциональные величины набора действительных чисел, их сравнение и свойства //Математическое образование. – 2017. – №. 4 (84. – С. 12-19.

7. Гантмахер Ф.Р. Теория матриц. – 5-е изд. //М.: Физматлит. – 2010. – 560 с.

8. Грузенкин Д. В. Алгоритм формирования методик регрессионного анализа концентрации основного компонента в минеральном сырье рентгенофлуоресцентным методом //Информатика. Экономика. Управление/Informatics. Economics. Management. – 2023. – Т. 2. – №. 4. – С. 0209-0217.

9. Грузенкин Д. В. и др. Определение метрики диверсифицированности мультиверсионного программного обеспечения на уровне алгоритмов //Фундаментальные исследования. – 2017. – №. 6. – С. 36-40.

10. Грузенкин Д. В. Повышение надёжности систем управления БПЛА экологического мониторинга путём применения общей метрики диверсифицированности для модификации алгоритма голосования согласованным большинством. – 2022.

11. Грузенкин Д. В. Поиск клонов среди автоматически сгенерированных учебных планов с применением метрики диверсифицированности программного обеспечения на уровне алгоритмов //Информатизация образования и методика электронного обучения: цифровые технологии в образовании. – 2022. – С. 31-35.

12. Грузенкин Д.В., Едреев В.В., Пантелеев Д.А. t / (n – 1)-вариантное программирование //Вестник Воронежского государственного технического университета. – 2022. – Т. 18. – №. 6. – С. 46-56. (К2)

13. Грузенкин Д.В., Ковалев И.В., Ковалев Д.И., Черных Н.С. Исследование эффективности различных подходов к выбору альтернатив для реализации мультиверсионного программного обеспечения // Авиакосмическое приборостроение. – 2025. – №6. – С. 39-47. (К2)

14. Грузенкин Д. В., Кузнецов А. С., Селезнев И. В. Оценка меры различия алгоритмов в многовариантной системе составления производственных планов //Системы анализа и обработки данных. – 2020. – №. 4 (80). – С. 65-80.

15. Грузенкин Д. В., Михалев А. С. Общая метрика диверсифицированности мультиверсий //ПРИБОРЫ Учредители: Союз общественных объединений Международное научно-техническое общество приборостроителей и метрологов (МНТО ПМ). – 2023. – №. 4. – С. 30-40.

16. Грузенкин Д. В., Михалев А. С. Определение метрики диверсифицированности мультиверсионного программного обеспечения на уровне языков программирования //Программная инженерия. – 2019. – Т. 10. – №. 9-10. – С. 384-390.

17. Грузенкин Д. В., Новиков О. С., Суханова А. В. Мультиверсонное ПО и блоки восстановления–два способа защиты от ошибок //Новая наука: От идеи к результату. – 2016. – №. 11-2. – С. 72-75.

18. Грузенкин Д.В., Царев Р.Ю., Кузнецов А.С. МОДЕЛЬ ДВУХФАЗНОЙ ТРАНСЛЯЦИИ КОДА МУЛЬТИВЕРСИЙ ПРОГРАММНЫХ МОДУЛЕЙ // Фундаментальные исследования. – 2015. – № 12-5. – С. 886-890;

19. Грузенкин Д.В., Черниговский А.С., Царёв Р.Ю. Requirements for N-version Software Modules Design and Development // Сборник докладов конференции Международная конференция «Математические и информационные технологии, MIT-2016» – 2016.

20. Грузенкин Д. В., Шаварин Д. О. Метод блоков восстановления для повышения надежности программного обеспечения: сравнение с мультиверсионным программированием //Современные инновации, системы и технологии. – 2022. – Т. 2. – №. 3. – С. 0127-0138.

21. Дворяк Д. А. Методика повышения качества разработки программного обеспечения с использованием тестирования на ранних стадиях //Актуальные исследования. – 2024. – С. 67.

22. Доросинский Л. Г., Виноградова Н. С. Основы теории принятия решений. – 2022.

23. Ежеманская С. Ступина А. Технология надежностного программирования задач автоматизации управления в технических системах. – ЛитРес, 2019.

24. Ефремова С. В. Метод мультиверсионного программирования для обработки телеметрической информации малых космических аппаратов //Сибирский аэрокосмический журнал. – 2023. – Т. 24. – №. 3. – С. 436-449.

25. Ильюшин Ю. В. ТЕОРИЯ ПРИНЯТИЯ РЕШЕНИЙ (ДОПОЛНИТЕЛЬНЫЕ ГЛАВЫ).

26. Казанцев А. А., Прохоров М. В., Худякова П. С. Обзор подходов к классификации текстов актуальными методами //Экономика и качество систем связи. – 2021. – №. 1 (19). – С. 57-67.

27. Ковалев И. В. и др. К вопросу реализации мультиверсионной среды исполнения бортового программного обеспечения автономных беспилотных объектов средствами операционной системы реального времени //Сибирский аэрокосмический журнал. – 2017. – Т. 18. – №. 1. – С. 58-61.

28. Ковалев И. В. Система мультиверсионного формирования программного обеспечения управления космическими аппаратами: дис. д-ра техн. наук //Красноярск: КГТУ. – 1997. – С. 228.

29. Корчагин В. Д. Анализ современных SOTA-архитектур искусственных нейронных сетей для решения задач классификации изображений и детекции объектов //Программные системы и вычислительные методы. – 2023. – №. 4. – С. 73-87.

30. Котенок А. В. Реализация алгоритмов мультиверсионного голосования //Современные наукоемкие технологии. – 2007. – №. 8. – С. 34-35.

31. Кузнецов А. С. Автоматизация разработки трансляторов мультисинтаксических языков програмирования мультиверсионных программных систем : дис. – Сибирский федеральный университет, 2009.

32. Кулямин В. В. Обзор методов динамического анализа программного обеспечения //Труды Института системного программирования РАН. – 2023. – Т. 35. – №. 4. – С. 8-45.

33. Лебедянцев В. В., Озерова М. И. Влияние архитектуры нейронной сети и исходных данных на работу нейронной сети для задач классификации //Информационные технологии в науке и производстве. – 2020. – С. 145-152.

34. Мейера О. В. Сравнительный анализ сходимости итерационных методов Якоби и Гаусса-Зейделя решения систем линейных алгебраических уравнений //ТЕЗИСЫ ABSTRACTS. – 2025. – С. 39.

35. Морозов В. А. Алгоритмы голосования для мультиверсионных информационно-управляющих систем : дис. – Сибирская аэрокосмическая академия им. МФ Решетнева, 2007.

36. Морозов В. А. Голосование согласованным большинством в мультиверсионном ПО // Вестник российских университетов. Математика. 2007. №1.

37. Муродов П. С. Методы классификации текстов //Материалы конференции «Информационный обмен в междисциплинарных исследованиях. – 2022. – С. 33-39.

38. Никулин В. С. Методика сбора и обработки эксплуатационных данных для оценки надежности функционирования инфокоммуникационных систем при малом количестве отказов : дис. – Сибирский федеральный университет, 2024.

39. Орлов С. А. О-66 Теория и практика языков программирования: Учебник для вузов. Стандарт 3-го поколения.—СПб.: Питер, 2013.—688 с.: ил. – 2014.

40. Пальмов С. В. Использование метода k-ближайших соседей для классификации данных //WORLD OF SCIENCE. – 2023. – С. 43-46.

41. Петухов Д. Е., Ткаченко А. В., Белов Ю. С. Линейный дискриминантный анализ как контролируемый подход в задачах уменьшения размерности данных //Научное обозрение. Технические науки. – 2020. – №. 2. – С. 5-9.

42. Побединский В. В. и др. Разработка программы нейронной сети обратного распространения ошибки. – 2022.

43. Подиновский В. Многокритериальные задачи принятия решений: теория и методы анализа. Учебник для вузов. – ЛитРес, 2022.

44. Свищёв А. В., Гревцов М. А. Нейронные сети прямого распространения //Моя профессиональная карьера. – 2021. – Т. 1. – №. 24. – С. 206-214.

45. Севастьянов Л. А., Щетинин Е. Ю. О методах повышения точности многоклассовой классификации на несбалансированных данных //Информатика и её применения. – 2020. – Т. 14. – №. 1. – С. 63-70.

46. Седых В. В. Методы решения задач многокритериальной оптимизации с линейными функциями цели //Актуальные исследования. – 2024. – №. 16 (198). – С. 66-71.

47. Семенов, С.С. Обзор методов принятия решений при разработке сложных технических систем / С.С. Семенов [и др.] // Функциональная надежность. Теория и практика. - 2014. - №3. - С. 72-84

48. Серебряная Л. В. Методы построения искусственных нейронных сетей для классификации данных //Цифровая трансформация. – 2022. – Т. 28. – №. 1. – С. 20-26.

49. Соловьев Д. С. Метод объективизации значений весовых коэффициентов для принятия решений в многокритериальных задачах //Научно-технический вестник информационных технологий, механики и оптики. – 2023. – Т. 23. – №. 1. – С. 161-168.

50. Торгашин А. А., Ковалев Д. И. Анализ эффективности принятия решений в мультиверсионном программировании //Информатика. Экономика. Управление-Informatics. Economics. Management. – 2025. – Т. 4. – №. 3. – С. 4014-4027.

51. Турсунмуротов Д. Х. ВЫБОР ОПТИМАЛЬНОГО ЧИСЛА k БЛИЖАЙШИХ СОСЕДЕЙ //World Scientific Research Journal. – 2025. – Т. 46. – №. 3. – С. 93-96.

52. Тюнькин А. Б., Басов А. А., Пармузина М. С. Аппроксимация с помощью среднеквадратического приближения //E-Scio. – 2021. – №. 7 (58). – С. 167-177.

53. Увайсова А. С. и др. Анализ современных методов кластеризации и классификации //Журнал Вестник Международного университета природы, общества и человека «Дубна». Серия «Естественные и инженерные науки». – 2020. – №. 4 (49). – С. 19-22.

54. Федеральная служба по аккредитации : официальный сайт. URL: https://fsa.gov.ru/about/ (дата обращения: 18.05.2026)

55. Хабеев И.А., Царенко В.А., Хабеев С.И., Чехмарев В.С., Грузенкин Д.В. Разработка и внедрение методики рентгенофлуоресцентного определения золота в ювелирных сплавах в аналитическом центре ОАО «Красцветмет» //Заводская лаборатория. Диагностика материалов. – 2020. – Т. 86. – №. 6. – С. 14-23.

56. Харахинов В. А. Нейросетевые технологии решения задач кластеризации и классификации данных в технических системах : дис. – Иркутск : дис.... канд. техн. наук, 2023.

57. Чечнев В. Б. Анализ и классификация многокритериальных методов принятия решений //Онтология проектирования. – 2024. – Т. 14. – №. 4. – С. 607-624.

58. Штарик Е., Штарик А., Царев Р. Мультиверсионное программное обеспечение. Алгоритмы голосования и оценка надёжности. – ЛитРес, 2019.

59. Aho A. V., Lam M. S., Sethi R., Ullman J. D. Compilers: principles, techniques and tools, 2-nd edition, 2007, 1009 p.

60. Alawida M., Teh J. S., Alshoura W. H. A new image encryption algorithm based on DNA state machine for UAV data encryption //Drones. – 2023. – Т. 7. – №. 1. – С. 38.

61. Alves J. et al. Characterization and identification of programming languages //12th Symposium on Languages, Applications and Technologies (SLATE 2023). – Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2023. – С. 13: 1-13: 13.

62. Beiroumi M. Z., Iversen V. B. Recovery method based on communicating extended finite state machine (CEFSM) for mobile communications //10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'05). – IEEE, 2005. – С. 384-393.

63. Chen K. et al. Statistical Inference for Software Reliability Constrained by the Shape of the Mean Value Function //Journal of Systems Science and Complexity. – 2026. – Т. 39. – №. 1. – С. 334-362.

64. Cong S., Zhou Y. A review of convolutional neural network architectures and their optimizations //Artificial Intelligence Review. – 2023. – Т. 56. – №. 3. – С. 1905-1969.

65. David Callahan, Ken Kennedy Analysis of interprocedural side effects in a parallel programming environment // Journal of Parallel and Distributed Computing, Volume 5, Issue 5, October 1988, Pages 517-550.

66. Dugan J. B., Lyu M. R. System reliability analysis of an N-version programming application //IEEE Transactions on Reliability. – 2002. – Т. 43. – №. 4. – С. 513-519.

67. Eriş O. et al. N-version programming for railway interlocking systems: Synchronization and voting strategy //IFAC Proceedings Volumes. – 2012. – Т. 45. – №. 24. – С. 177-180.

68. Fiterau-Brostean P. et al. Automata-Based Automated Detection of State Machine Bugs in Protocol Implementations //NDSS. – 2023.

69. Gruzenkin D. V. et al. Algorithm diversity metric for N-version software //Journal of Physics: Conference Series. – IOP Publishing, 2019. – Т. 1333. – №. 3. – С. 032086. (Scopus)

70. Gruzenkin D. V. et al. Algorithm source codes generation for ensuring N-version software diversity //Journal of Physics: Conference Series. – IOP Publishing, 2019. – Т. 1333. – №. 3. – С. 032026. (Scopus)

71. Gruzenkin D. V., Chernigovskiy A. S., Tsarev R. Y. N-version software module requirements to grant the software execution fault-tolerance //Proceedings of the Computational Methods in Systems and Software. – Springer, Cham, 2017. – С. 293-303.

72. Gruzenkin, D.V. Neural networks to solve modern artificial intelligence tasks / D.V. Gruzenkin et. al. // Journal of Physics: Conference Series. - 2019. - Vol. 1399, issue 3. - pp.

73. Hancock J. T., Khoshgoftaar T. M. Survey on categorical data for neural networks //Journal of big data. – 2020. – Т. 7. – №. 1. – С. 28.

74. Huang Y. S. et al. A study on optimal release schedule for multiversion software //INFORMS Journal on Computing. – 2024. – Т. 36. – №. 1. – С. 121-140.

75. Hu T., Bertolott I. C., Navet N. Towards seamless integration of N-Version Programming in model-based design //2017 22nd IEEE International conference on emerging technologies and factory automation (ETFA). – IEEE, 2017. – С. 1-8.

76. Kernighan B. W., Ritchie D. M. The C Programming Language, 2-nd edition, 1988, 274 p.

77. KhoKhar F. A., Zoppi T., Shah J. H. Orchestrating Fail-Safe, Black-Box Models Within Federated Learning Scenarios //2025 IEEE 30th Pacific Rim International Symposium on Dependable Computing (PRDC). – IEEE, 2025. – С. 45-57.

78. Kim Y. S. et al. A software reliability model with dependent failure and optimal release time //Symmetry. – 2022. – Т. 14. – №. 2. – С. 343.

79. Kovalev D. I., Zaitsev P. K. Improving decision-making algorithms on the correctness of the states of multiversions of fault-tolerant software systems //Информатика. Экономика. Управление/Informatics. Economics. Management. – 2023. – Т. 2. – №. 3. – С. 0201-0209.

80. Krishna Mohan K. et al. Integration of black-box and white-box modeling approaches for software reliability estimation //International Journal of Reliability, Quality and Safety Engineering. – 2010. – Т. 17. – №. 03. – С. 261-273.

81. Levitin G., Xing L., Xiang Y. Optimization of time constrained N-version programming service components with competing task execution and version corruption processes //Reliability Engineering & System Safety. – 2020. – Т. 193. – С. 106666.

82. Lin Z., Xue H., Pan W. Robust multivariable Mendelian randomization based on constrained maximum likelihood //The American Journal of Human Genetics. – 2023. – Т. 110. – №. 4. – С. 592-605.

83. Maspupah A. Literature review: Advantages and disadvantages of black box and white box testing methods //Jurnal Techno Nusa Mandiri. – 2024. – Т. 21. – №. 2. – С. 151-162.

84. Mohan K. K., Verma A. K., Srividya A. Software reliability estimation through black box and white box testing at prototype level //2010 2nd International Conference on Reliability, Safety and Hazard-Risk-Based Technologies and Physics-of-Failure Methods (ICRESH). – IEEE, 2010. – С. 517-522.

85. Nikolaeva D., Petrova D. A survey of the programming paradigms used in programming languages //2024 5th International Conference on Communications, Information, Electronic and Energy Systems (CIEES). – IEEE, 2024. – С. 1-5.

86. Parhami B. New Designs for Voting Networks Based on the Boyer-Moore Majority-Finding Algorithm //2025 IEEE 16th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). – IEEE, 2025. – С. 0058-0063.

87. Patel U., Hati A., Bhilare S. Black-box adversarial defense for enhancing robustness in speaker recognition systems with multimodel consensus //Seventeenth International Conference on Machine Vision (ICMV 2024). – SPIE, 2025. – Т. 13517. – С. 449-456.

88. Pierce B. C. (ed.). Advanced topics in types and programming languages. – MIT press, 2024.

89. Procaccia A. D. Computational voting theory: Of the agents, by the agents, for the agents. – Hebrew University, 2008.

90. Pu J. Voting Theory and Machine Learning : дис. – 2025.

91. Rasmus Fonseca ESMT-Smith: Smiths algorithm for the Euclidean Steiner Minimal Tree problem [Электронный ресурс]. – URL: https://github.com/RasmusFonseca/ESMT-Smith (дата обращения: 20.11.2020).

92. Rosales R., Paulitsch M. Composable finite state machine-based modeling for quality-of-information-aware cyber-physical systems //ACM Transactions on Cyber-Physical Systems. – 2021. – Т. 5. – №. 2. – С. 1-27.

93. Sabah A. S. et al. Comparative analysis of the performance of popular sorting algorithms on datasets of different sizes and characteristics. – 2023.

94. Syaikhuddin M. M. et al. Conventional software testing using white box method //Kinetik: game technology, information system, computer network, computing, electronics, and control. – 2018. – С. 65-72.

95. Taherdoost H., Madanchian M. Multi-criteria decision making (MCDM) methods and concepts //Encyclopedia. – 2023. – Т. 3. – №. 1. – С. 77-87.

96. Topping C. J., Høye T. T., Olesen C. R. Opening the black box—Development, testing and documentation of a mechanistically rich agent-based model //Ecological Modelling. – 2010. – Т. 221. – №. 2. – С. 245-255.

97. Trivedi K. S., Bobbio A. Reliability and availability engineering: modeling, analysis, and applications. – Cambridge University Press, 2017.

98. Tsarev R. Y. et al. Application of majority voting and consensus voting algorithms in N-version software //Journal of Physics: Conference Series. – IOP Publishing, 2018. – Т. 1015. – №. 4. – С. 042059.

99. Tsarev R. Y. et al. Classification of voting algorithms for N-version software //Journal of Physics: Conference Series. – IOP Publishing, 2018. – Т. 1015. – №. 4. – С. 042060.

100. Tsarev R. Y. et al. Fuzzy voting algorithms for N-version software //Journal of Physics: Conference Series. – IOP Publishing, 2019. – Т. 1333. – №. 3. – С. 032087.

101. Verma A., Khatana A., Chaudhary S. A comparative study of black box testing and white box testing //International Journal of Computer Sciences and Engineering. – 2017. – Т. 5. – №. 12. – С. 301-304.

102. Yang L. et al. A robust cyclegan-l2 defense method for speaker recognition system //IEEE Access. – 2023. – Т. 11. – С. 82771-82783.

103. Zhao S. et al. Linear discriminant analysis //Nature Reviews Methods Primers. – 2024. – Т. 4. – №. 1. – С. 70.

ПРИЛОЖЕНИЕ А

КРАСЦВЕТМЕТ

АКТ

23.09.2021 № б/н

Об использовании результатов кандидатской диссертационной работы Грузенкина Дениса Владимировича

Составлен:

Председатель: И.о. руководителя аналитического центра — В.А. Царенко

Члены комиссии: 1. Руководитель практики МЕС — О.В. Давыденко

2. Ведущий специалист направления ЛИМС — И.С. Чернов

Настоящий акт составлен о том, что результаты диссертационной работы Грузенкина Дениса Владимировича, представленной на соискание ученой степени кандидата технических наук, были апробированы в условиях производства ОАО «Красцветмет» (г. Красноярск).

Результаты использования предложенных программных решений показали их корректную работу на реальных входных данных, а также их полезность при проведении анализа золотосодержащих образцов рентгенофлуоренесцентным методом.

Результаты диссертационной работы были внедрены в работу лаборатории рентгеноспектральных методов анализа аналитического центра ОАО «Красцветмет» с целью повышения информационной надёжности расчётных методов анализа содержания золота во вторичном сырье.

В.А. Царенко
И.о. руководителя аналитического центра

О.В. Давыденко
Руководитель практики МЕС

И.С. Чернов
Ведущий специалист направления ЛИМС

Царенко Виталий Анатольевич
VCsarenko@krastsvetmet.ru, +7 391 259 3333 (4767)

Открытое акционерное общество «Красноярский завод цветных металлов имени В.Н. Гулидова» (ОАО «Красцветмет»)