Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ярославский государственный технический университет»

Кафедра «Информационные системы и технологии»

УДК 004.056

УДК 004.021 ДОПУСКАЕТСЯ К ЗАЩИТЕ

Заведующий кафедрой канд. тех. наук

С. Ю. Бойков

« » 2024

**РАЗРАБОТКА АЛГОРИТМА И ПРОГРАММЫ РАСЧЕТОВ ПО ПРОИЗВОЛЬНЫМ СХЕМАМ НАДЕЖНОСТИ С ИСПОЛЬЗОВАНИЕМ КОНТЕКСТНОЙ ЛОГИКИ**

Пояснительная записка к выпускной квалификационной работе   
по направлению подготовки «Информационные системы и технологии»

ЯГТУ 09.03.02 – 020 ВКР

СОГЛАСОВАНО

|  |  |
| --- | --- |
| Руководитель  канд. физ.-мат. наук, доцент  А.Б. Раухваргер  « » 2024 | Нормоконтролер  канд. физ.-мат. наук, доцент  А.Б. Раухваргер  « » 2024 |
| Консультант по экономике  и организации производства  М.И. Маркин  « » 2024 | Проект выполнил  студент группы ЭИСБ-44  В.А. Фокин  « » 2024 |

2024

РЕФЕРАТ

59 с., 30 рис., 11 табл., 18 источников, 1 прил.

СХЕМЫ НАДЕЖНОСТИ, ВЕРОЯТНОСТЬ, КОНТЕКСТНАЯ ЛОГИКА, ТАБЛИЦА ИСТИННОСТИ, МОДЕЛИРОВАНИЕ

Объектом исследования являются схемы надёжности, а именно, наиболее распространённые из них, мостиковые, а также алгоритмы по вычислению отказоустойчивости этих схем.

Цель работы - разработка автоматизируемого алгоритма для расчета отказоустойчивости сколь угодно комплексных мостиковых схем надёжности

В рамках исследования планируется использование следующих методов:

Анализ текущих методов в области вычисления мостиковых схем надёжности

Обзор существующих программных комплексов в области вычисления мостиковых схем надёжности

Определение характеристик разрабатываемого приложения

Проектирование пользовательского интерфейса приложения

Тестирование и отладка приложения

В результате выполнения работы будет разработано пользовательское приложение, которое позволит вычислять вероятность и частоту безотказной работы сколь угодно сложные схемы надёжности без их дополнительных преобразований.

Полученные результаты работы позволят автоматизировать вычисление сколь угодно сложных мостиковых схем надёжности без их дополнительного форматирования или изменения.

Практическая значимость исследования связана с повышением уровня цифровизации в сфере разработки и анализа надёжности вычислительных систем.

СОДЕРЖАНИЕ

[Введение 5](#_Toc168858261)

[1 Аналитическая часть 6](#_Toc168858262)

[1.1 Управление рисками 6](#_Toc168858263)

[1.2 Характеристика объекта исследования схемы надёжности 6](#_Toc168858264)

[1.3 Анализ существующих решений 15](#_Toc168858265)

[1.4 Постановка цели и задачи 21](#_Toc168858266)

[1.5 Вывод по аналитической части 22](#_Toc168858267)

[2 Проектная часть 23](#_Toc168858268)

[2.1 Разработка алгоритма для подсчёта мостиковых схем надёжности с возможностью кодирования 23](#_Toc168858269)

[2.2 Описание предметной области программы 30](#_Toc168858270)

[2.3 Функциональное проектирование 30](#_Toc168858271)

[2.4 Контекстная диаграмма 31](#_Toc168858272)

[2.5 Декомпозиция процесса 32](#_Toc168858273)

[2.6 Концептуальное проектирование 33](#_Toc168858274)

[2.7 Вывод по проектной части 35](#_Toc168858275)

[3 Технологическая часть 36](#_Toc168858276)

[3.1 Обоснование и выбор программных средств 36](#_Toc168858277)

[3.2 Разработка программных сущностей 37](#_Toc168858278)

[3.3 Логика работы приложения 40](#_Toc168858279)

[3.4 Вывод по технологической части 42](#_Toc168858280)

[4 Экономическая часть 43](#_Toc168858281)

[4.1 Обоснование целесообразности разработки проекта 43](#_Toc168858282)

[4.2 Расчётная часть 50](#_Toc168858283)

[4.3 Вывод по экономической части 56](#_Toc168858284)

[Заключение 57](#_Toc168858285)

[Список используемых источников 58](#_Toc168858286)

[Приложение А 60](#_Toc168858287)

# Введение

В настоящее время во всем мире и в особенности в России происходит процесс цифровизации экономики. Этот процесс затрагивает все сферы жизни, включая промышленность, торговлю, финансы и многие другие отрасли. Цифровизация позволяет автоматизировать многие процессы, сократить затраты и повысить эффективность работы предприятий. Кроме того, цифровизация способствует созданию новых рабочих мест и развитию инновационных технологий [1].

В нашей стране она реализуется в рамках Указов Президента Российской Федерации от 7 мая 2018 г № 204 «О национальных целях и стратегических задачах развития Российской Федерации на период до 2024 года» иот 21.07.2020 г. № 474 «О национальных целях развития Российской Федерации на период до 2030 года»**,** в том числе с целью решения задачи по обеспечению ускоренного внедрения цифровых технологий в экономике и социальной сфере, Правительством Российской Федерации сформирована национальная программа «Цифровая экономика Российской Федерации» утвержденная протоколом заседания президиума Совета при Президенте Российской Федерации по стратегическому развитию и национальным проектам от 4 июня 2019 г. № 7.

Цифровизация оценки рисков является неотъемлемой частью цифровизации бизнеса. Она позволяет автоматизировать процесс оценки рисков, анализировать большие объемы данных и принимать обоснованные решения. Цифровизация оценки рисков также позволяет снизить затраты на проведение оценки, повысить ее точность и эффективность [2].

# Аналитическая часть

## Управление рисками

Эффективное управление рисками при цифровой трансформации промышленного предприятия является его конкурентным преимуществом. Оценка рисков при планировании цифровой трансформации на промышленном предприятии тесно связана с процессом прогнозирования бюджета предприятия. Сопоставление имеющихся финансовых возможностей предприятия с его бюджетом цифровой трансформации дает возможность осуществить грамотное планирование, найти источники средств, а также возможности экономических интеграций в целях достижения конкурентных преимуществ в процессе цифровизации.

Для решения проблем, вызванных вышеуказанными рисками, необходимо проведение стратегического планирования по двум направлениям:

1. Оценка рисков в процессе проведения цифровизации, разработка системы управления ими и нахождение резервов для риск-менеджмента. Сначала определяются факторы, которые препятствуют выделению стратегических целей в сфере цифровизации деятельности предприятия. Далее эти факторы ранжируются по уровню разрушения последствия и вероятности реализации риска, разрабатываются мероприятия по ограничению влияния выявленных факторов. Заключительным этапом является анализ составленного бизнес-плана цифровизации и предоставление его руководителю предприятия.

2. Оценка воздействия резервов для нейтрализации рисков на бюджет цифровой трансформации предприятия представляет собой формирование сложной системы ликвидности резервов. Главным инструментом выступает метод анализа потоков денежных средств, который выделяет цифровую трансформацию в отдельные подсистемы в виде управляемой и управляющей. Менеджер управляет предприятием при помощи метода бюджетирования [3].

## Характеристика объекта исследования схемы надёжности

### Основные понятия теории надёжности, применяемые к программному обеспечению

Теория надёжности — это научная дисциплина, изучающая закономерности распределения отказов технических устройств и конструкций, причины и модели их возникновения. Она исследует методы обеспечения стабильности работы объектов в процессе проектирования, производства, приёмки, транспортировки, эксплуатации и хранения.

Теория надёжности устанавливает количественные показатели надёжности и изучает связь между показателями эффективности и надёжности. Она базируется на комплексном подходе к изучению надёжности и охватывает различные аспекты, такие как анализ отказов, прогнозирование надёжности, разработка методов повышения надёжности и поддержание надёжности в процессе эксплуатации.

Основные термины и определения теории надёжности включают понятия системы, элемента системы, надёжности, работоспособности, отказов и предельных состояний.

Техническая система представляет собой совокупность элементов, которые взаимодействуют друг с другом в процессе выполнения определённых функций. Эта система характеризуется упорядоченностью взаимодействия элементов и наличием свойств, не сводящихся к свойствам отдельных элементов.

Элемент системы — это составная часть, которая рассматривается как единое целое без дальнейшего разделения. Внутренняя структура элемента не является предметом исследования, так как она не влияет на его основные свойства и функции в рамках данной системы.

Надёжность — это комплексное свойство объекта или системы, которое определяет его способность сохранять свои функциональные возможности и параметры в установленных пределах на протяжении определённого периода времени. Это свойство связано с недопустимостью отказов в работе и сохранением работоспособного состояния объекта в течение некоторого времени или определённой наработки.

Работоспособность — это состояние изделия, при котором оно удовлетворяет всем требованиям, предъявляемым к его основным параметрам, таким как быстродействие, нагрузочная характеристика, устойчивость и точность выполнения операций.

Отказ — это событие, которое приводит к нарушению работоспособности системы и переходу её из работоспособного состояния в неработоспособное. Это явление характеризуется снижением или потерей способности системы выполнять свои функции и задачи.

Предельное состояние возникает, когда дальнейшая эксплуатация технической системы или оборудования становится невозможной или нецелесообразной. Это состояние характеризуется потерей работоспособности, снижением безопасности или ухудшением качества функционирования системы [4].

Схемы надёжности — это графические представления, используемые для анализа и оценки надёжности сложных систем. Они позволяют определить слабые места и оптимизировать структуру системы для повышения общей надёжности. Существуют различные виды схем надёжности, включая:

Блок-схемы: представляют систему в виде блоков, соединённых линиями связи. Каждый блок соответствует функциональному элементу системы, а линии связи показывают потоки информации или материальных ресурсов между ними (рис 1).

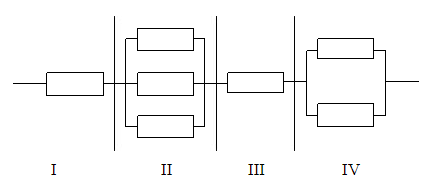


Рисунок – 1 Пример блок-схемы

Деревья отказов: изображают возможные пути развития отказов системы, начиная с основного события и заканчивая конечными отказами. Ветви дерева соответствуют возможным причинам отказов, а листья — конечным отказам (рис 2).

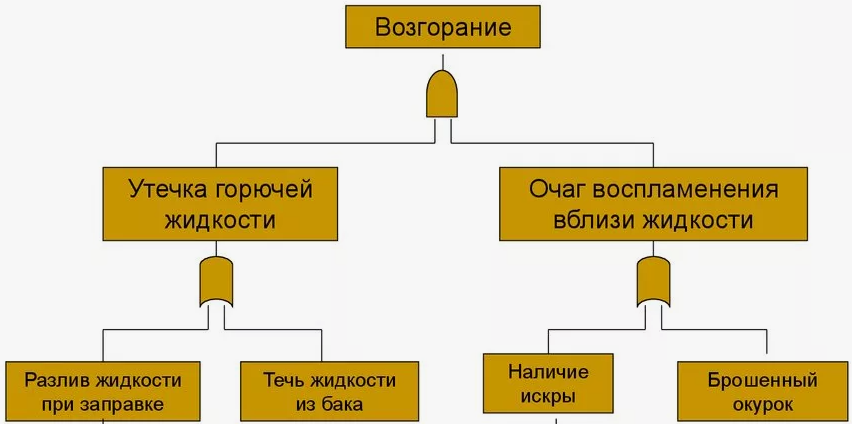


Рисунок – 2 Пример дерева отказов

Деревья событий: аналогичны деревьям отказов, но отражают возможные пути развития успешных событий в системе (рис 3) [5].

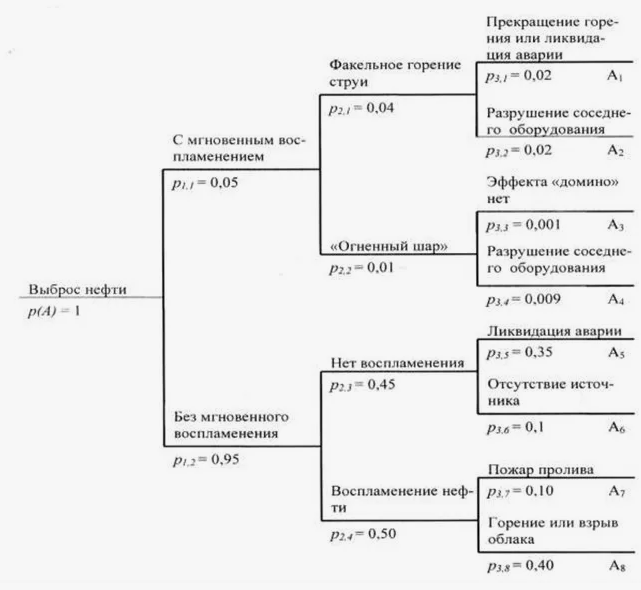


Рисунок – 3 Пример дерева событий

Схемы надёжности, также известные как структурные схемы надёжности (RBD), являются графическим представлением логических связей между функционирующими компонентами системы, необходимыми для её успешной работы. Они эквивалентны логическим уравнениям булевых переменных и используются для оценки показателей надёжности, таких как коэффициент готовности, вероятность безотказной работы и частота отказов.

Существует несколько методов анализа надёжности, один из которых — RBD. Цель каждого метода и их индивидуального или совместного применения — оценить указанные показатели надёжности. При анализе необходимо учитывать различные факторы, такие как зависимости между блоками, порядок отказов и временные зависимости.

RBD связан с анализом дерева отказов (FTA) и марковскими методами. Базовая математика одинакова для RBD и FTA, но RBD ориентирован на успех системы, в то время как FTA — на отказ системы. Преобразование RBD в FTA и наоборот возможно.

С математической точки зрения модели RBD и FTA представляют собой две стороны одного и того же логического выражения. Вычисления коэффициента готовности, частоты отказов и показателей безотказности могут быть проведены с использованием RBD, учитывая зависимости между блоками и временные зависимости.

Соединение элементов в структурных схемах надёжности можно разделить на четыре типа:

1. последовательному,
2. параллельному,
3. cмешанному,
4. произвольному.

В качестве ключевых характеристик надёжности применяются вероятность безотказной работы и вероятность отказа [6].

### Структурные схемы надежности с последовательным соединением элементов

Последовательное соединение в структурной схеме надёжности — это тип соединения элементов, при котором отказ хотя бы одного элемента приводит к отказу всей системы в целом (рис. 4)

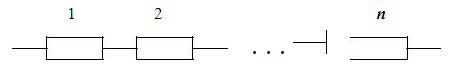


Рисунок 4 - Структурная схема надежности с последовательным соединением элементов

Этот тип соединения в теории надежности еще называет основным соединением.

Если предположить, что отказы элементов являются независимыми событиями, то на основе теоремы умножения вероятностей вероятность безотказной работы может быть выражена следующим образом:

Последовательная схема состоит из двух или более элементов, соединённых последовательно. Предполагая, что отказы элементов являются независимыми событиями, вероятность безотказной работы объекта определяется произведением вероятностей безотказной работы его элементов. Формула для определения вероятности безотказной работы объекта с последовательным соединением элементов выглядит следующим образом (1):

|  |  |
| --- | --- |
|  | (1) |

где – вероятность безотказной работы i -ого элемента;

– вероятность безотказной работы системы.

Приведённая выше формула показывает, что для достижения требуемой техническими условиями вероятности безотказной работы при увеличении количества последовательно соединённых элементов необходимо уменьшить интенсивность отказов каждого элемента или повысить среднюю наработку на отказ.

Нередко встречаются ситуации, когда система последовательно соединённых элементов состоит из k подсистем, и каждая j-я (j = 1, k) подсистема содержит nj равно надёжных элементов. В таких случаях вероятность безотказной работы системы определяется уравнением (2):

|  |  |
| --- | --- |
|  | (2) |

где P(t) — вероятность безотказной работы системы,

nj — количество элементов в j-й подсистеме,

Pj(t) — вероятность безотказной работы j-й подсистемы за время t.

Для увеличения вероятности безотказной работы необходимо уменьшать количество элементов, снижать интенсивность отказов элементов или увеличивать среднюю наработку до отказа.

Анализ полученных выражений показывает следующее:

1. Вероятность безотказной работы уменьшается с увеличением количества элементов. Это связано с тем, что при добавлении новых элементов увеличивается вероятность возникновения отказов из-за взаимодействия между ними.
2. Вероятность безотказной работы последовательного соединения ниже, чем у самого надёжного элемента системы. Это объясняется тем, что отказ одного элемента в последовательной системе приводит к отказу всей системы, в то время как отказ самого надёжного элемента не влияет на работу остальных элементов.

### Структурные схемы надежности с параллельным соединением элементов

Параллельное соединение элементов в структурной схеме надёжности представляет собой такой тип соединения, при котором система перестаёт функционировать только при отказе всех n элементов, составляющих схему (рис. 5).

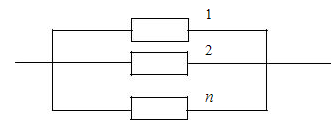


Рис. 5 - Структурная схема надежности с параллельным соединением элементов

Согласно определению (3),

|  |  |
| --- | --- |
|  | (3) |

Следовательно (4):

|  |  |
| --- | --- |
|  | (4) |

Вероятность безотказной работы для случая равнонадёжных элементов определяется по формуле (5):

|  |  |
| --- | --- |
|  | (5) |

где p(t) — вероятность безотказной работы элемента,

n — количество элементов.

Анализ формулы (5) показывает, что вероятность безотказной работы будет тем выше, чем больше элементов входит в систему, и тем выше вероятность безотказной работы параллельного соединения, чем выше вероятность безотказной работы самого надёжного элемента системы.

### Структурные схемы надежности со смешанным соединением элементов

В некоторых сложных системах структурные схемы надёжности включают как последовательные, так и параллельные структуры надёжности. Это указывает на наличие обоих типов соединений в схеме надёжности, что и показано на рисунке 6.

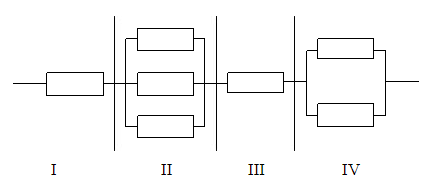


Рис. 6 - Пример структурной схемы надежности со смешанным соединением элементов

В этом случае для определения надёжности структурную схему разделяют на последовательные или параллельные сегменты таким образом, чтобы каждый сегмент имел либо только последовательную, либо только параллельную структурную схему. На каждом сегменте рассчитывается вероятность безотказной работы согласно формулам, соответствующим структурным схемам этого сегмента. В результате исходная структурная схема надёжности преобразуется в структуру с последовательным или параллельным соединением элементов. Эквивалентная последовательная структура представлена на рисунке 7. Здесь, на примере исходного рисунка 6, PI, PII, PIII, PIV — вероятности безотказной работы соответственно первого, второго, третьего и четвёртого последовательных сегментов, на которые структурная схема со смешанным соединением элементов была предварительно разделена.

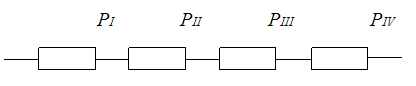


Рис. 7 - Преобразованная структура со смешанным соединением элементов

Тогда вероятность безотказной работы системы в представленном примере будет равна (6).

|  |  |
| --- | --- |
|  | (6) |

В общем случае, если система состоит из k последовательных участков, возникших в результате предварительных преобразований, функция вероятности безотказной работы будет описываться следующим уравнением (7):

|  |  |
| --- | --- |
|  | (7) |

где – вероятность безотказной работы j-о участка.

### Сложная произвольная структура схем надёжности

Если невозможно использовать последовательное, параллельное или комбинированное соединение при построении структурной схемы надёжности, возникает сложная произвольная структура. Для таких структур не существует универсальных методов расчёта надёжности. Одна из наиболее распространённых схем таких структур — мостиковая схема (рис. 8).

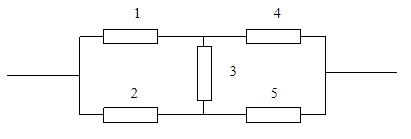
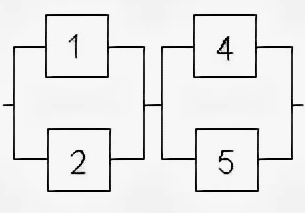


Рисунок 8 - Произвольная структурная «мостиковая» схема надежности

Мостиковые схемы надёжности представляют собой сложные структуры, состоящие из множества элементов, соединённых между собой различными способами. Для определения надёжности таких схем используется метод разложения по «мостам».

Расчёт начинается с разделения основной схемы на подсхемы до тех пор, пока не образуются схемы, сводящиеся к последовательным и параллельным соединениям. Затем проводится расчёт вероятностей безотказной работы по выведенным формулам, начиная с конечных схем и заканчивая исходной схемой - корнем дерева (рис 9-10).



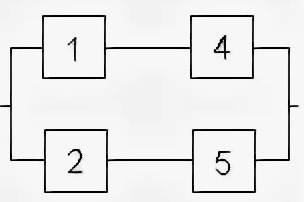


Рисунок 9-10 Конечные схемы надёжности для рисунка 8

Определение «мостов» — это ключевой этап расчёта, так как именно они определяют надёжность системы. «Мосты» — это элементы, отказ которых приводит к отказу всей системы. Эмпирический подход к определению мостов затрудняет автоматизацию процесса разложения и увеличивает время расчёта.

Таким образом, метод разложения по «мостам» позволяет определить надёжность мостиковых схем, однако его громоздкость и низкая производительность ограничивают применение в практических задачах [7].

## Анализ существующих решений

На рынке программных комплексов представлен ряд зарубежных и отечественных программных продуктов, предназначенных для автоматизации расчётов надёжности сложных технических систем, включая радиоэлектронную аппаратуру и электрорадиоизделия.

Наиболее распространенными среди зарубежных ПК являются: RELEX (Relex software Corporation, США); A.L.D.Group (Израиль); Risk Spectrum (Relcon AB, Швеция); среди отечественных ПК, которые применяются на ряде предприятий: ПК АСОНИКА-К (МИЭМ-ASKsoft);

ПК Relex и Risk Spectrum предоставляют возможность проведения логико-вероятностного анализа надёжности и безопасности технических систем, включая расчёт надёжности современных автоматизированных систем управления технологическими процессами (АСУТП), оптимизацию техногенного риска и определение оптимальных параметров системы технического обслуживания потенциально опасных объектов.

В список программных продуктов, которые прошли аттестацию в Государственном агентстве по атомному надзору России в 2003 году был включён Комплекс Spectrum. ПК Relex и Risk Spectrum могут использоваться для оценки надёжности не только производственных или управляющих систем, но и приборов, транспортных средств, вычислительной техники и оборудования в оборонной сфере.

Моделирование и расчёт показателей надёжности и безопасности технических систем, используемых в Европе и США, основаны на логико-вероятностных методах, использующих деревья событий (ДС) и деревья отказов (ДО) в качестве инструментов для создания графических моделей безопасности. Эти методы позволяют формализовать условия работоспособности сложных технических систем и рассчитывать их надёжность. (рис.11)

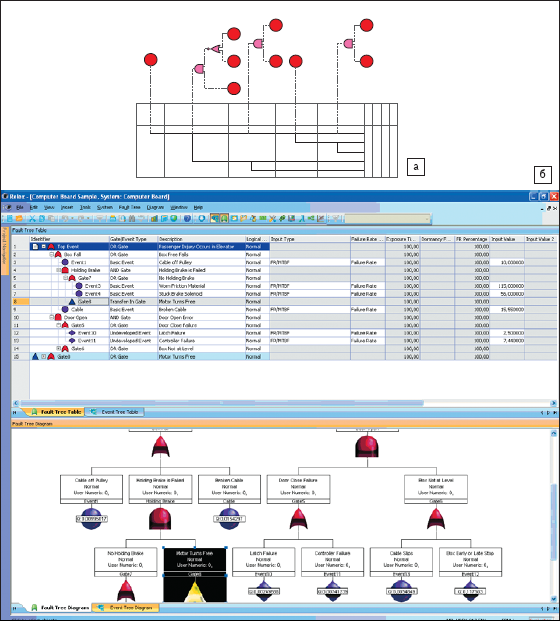


Рисунок 11 - а) Модель надежности (безопасности), представленная с помощью дерева отказов и событий; б) дерево отказов в ПК Relex

Дерево событий и отказов — это графические модели, используемые для анализа и оценки рисков аварийных ситуаций. Дерево событий отражает последовательность развития событий, ведущих к аварии, а дерево отказов показывает возможные причины отказа системы. Оба дерева состоят из вершин (событий) и связей между ними (логических операторов). Вершины представляют собой элементарные случайные события, а логические операторы соответствуют булевым операциям над ними.

Применение деревьев событий и отказов позволяет проводить комплексный анализ устойчивости функционирования промышленных и экологических систем, выявлять слабые места и разрабатывать меры по повышению надёжности и безопасности. Каждый логический оператор Булевой алгебры соответствует определённому графическому элементу, что позволяет декомпозировать сложные события на более простые (базисные или элементарные) (рис. 12).

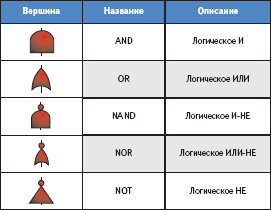


Рисунок 12 - Примерный список вершин и событий в ПК Relex

В модуле дерева отказов (ДО) программного комплекса Relex применяются логико-динамические операторы (вершины), которые принимают во внимание взаимосвязь событий, временные аспекты и приоритеты. (рис. 11б). Он предоставляет возможность расчёта таких показателей, как вероятность отказа, неготовность, параметр потока отказов и среднее количество отказов. Расчёт производится как для основного события, так и для каждого промежуточного. Для каждого отдельного события можно просматривать и анализировать соответствующие минимальные сечения.

В состав программного обеспечения Relex Reliability Studio 2007 входят разнообразные аналитические инструменты для решения разнообразных задач: система оценки человеческого фактора и анализа рисков (Human Factors, Risk Analysis), система оповещения об отказах, анализе и корректирующих действиях (FRACAS-система), деревья отказов/событий (Fault Tree/Event Tree), блок-схемы надёжности (Reliability Block Diagram), оценка стоимости срока службы оборудования (Life Cycle Cost), статистический анализ (Weibull Analysis), марковский анализ (Markov Analysis), прогнозирование надёжности (Reliability Prediction).

Компания A.L.D. Group (Израиль-США) состоит из двух фирм, занимающихся логистикой (информационная система логистики) и оценкой надёжности: SoHaR и FavoWeb.

Программный продукт FavoWeb представляет собой динамическую FRACAS-систему (Failure Reporting Analysis and Corrective Action System), функционирующую в интернете. Эта система используется многими зарубежными компаниями, такими как Lockheed Martin, для анализа отказов и принятия корректирующих мер.

FavoWeb использует современные возможности интернет-технологий и реализует полный замкнутый цикл методологии FRACAS, которая применима к различным продуктам, услугам и процессам. Он может быть использован на всех этапах жизненного цикла: разработка, макетирование, производство, эксплуатация, техническое обслуживание, контроль и испытания. Продукт подходит для применения в разных отраслях, таких как авиация, оборона, связь, электроника, фармацевтика, автомобилестроение и бытовая техника.

Система FRACAS предназначена для создания баз данных и преобразования неоднородных данных в структурированные сведения о качестве. Она обладает мощным механизмом корректирующих действий, включающим поддержку работы групп по анализу отказов, дефектов и материалов, изучение глубинных причин неисправностей и решение проблем. Также имеется модуль Workflow для автоматизации уведомлений об серийных номерах и отказах.

Программа обладает широким набором функций и возможностей для оценки и повышения надёжности оборудования благодаря интеграции с системой анализа RAM Commander (рис. 13). FavoWeb также предоставляет специализированные инструменты для внесения изменений, управления логистикой и отслеживания серийных номеров изделий (CALS-технологии).

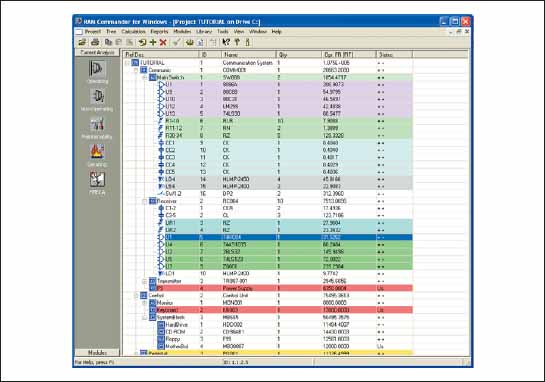


Рисунок 13 - Окно системы анализа надежности RAM Commander

Термин CALS-технологии (Continuous Acquisition and Lifecycle Support) обозначает совокупность принципов и технологий информационной поддержки жизненного цикла продукта на всех его этапах. Русский эквивалент CALS — информационная поддержка жизненного цикла изделий (ИПИ). В последнее время за границей наряду с CALS также используется понятие Product Lifecycle Management (PLM). Важный аспект CALS-технологий — информационная поддержка самого процесса оценки надёжности технической системы. RAM Commander позволяет рассчитывать среднюю наработку на отказ или критический отказ (MTBF/ MTBCF), среднее время ремонта (MTTR), среднее время между устранением отказов (MTBMA) и другие показатели.

Окно системы RAM Commander, предназначенной для анализа надёжности представлено на рисунке 13. Эта система охватывает все инженерные задачи, связанные с надёжностью электронных, электромеханических, механических и других систем. С помощью RAM Commander можно прогнозировать надёжность, готовность и ремонтопригодность различных видов оборудования, а также управлять данными о надёжности, готовности и ремонтопригодности. Система также позволяет проводить анализ надёжности проектируемого оборудования методом Монте-Карло и оптимизировать учёт складских запасов запчастей.

Возможности RBD-модуля RAM Commander для расчёта структурной надёжности во многом схожи с RBD-модулем ПК Relex. Однако возможности последнего значительно шире, поскольку он позволяет учитывать следующие факторы: вид резервирования, вероятность и время успешного подключения резерва, загруженность резерва, механизм проявления отказа, различные стратегии восстановления, наличие ЗИП, профилактического обслуживания и технических осмотров [8].

На российском рынке представлена успешно развивающаяся подсистема АСОНИКА-К, которая, по мнению разработчиков, в будущем станет программным комплексом (ПК АСОНИКА-К Это программное обеспечение предназначено для анализа и обеспечения надёжности в рамках автоматизированного проектирования РЭА (рис. 14). Подсистема АСОНИКА-К обладает возможностями, эквивалентными RBD-модулям зарубежных программных комплексов A.L.D. Group (например, RAM Commander), Relex, Isograph и других. Её использование является более предпочтительным, поскольку АСОНИКА-К позволяет проводить расчёт надёжности РЭА, производимой в России, на основе данных из отечественных справочников «Надёжность электрорадиоизделий» и «Надёжность электрорадиоизделий зарубежных аналогов». АСОНИКА-К соответствует требованиям комплекса военных стандартов «Мороз-6» для РЭА ответственного применения, а также стандартам США MIL-HDBK-217 и КНР GJB/z 299B.

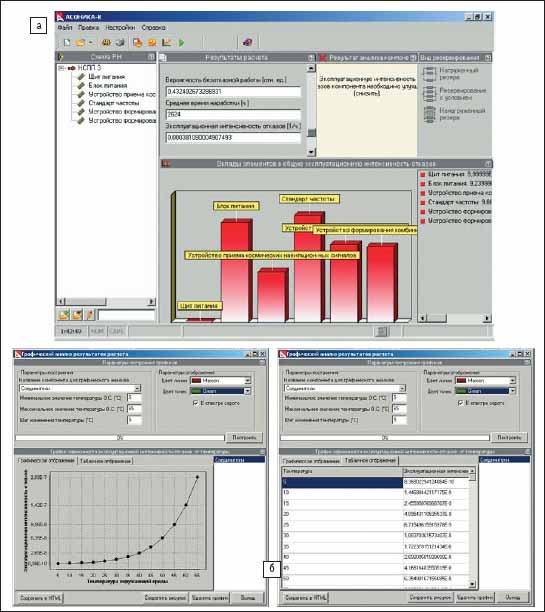


Рисунок - 14 ПК АСОНИКА-К. Система расчета надежности

Применение ПК АСОНИКА-К способствует реализации современных методов конструирования РЭА, основанных на CALS-технологиях. АСОНИКА-К представляет собой программное обеспечение, разработанное с использованием технологии «клиент-сервер». База данных серверной части ПК содержит постоянно обновляемую информацию о надёжности как российских, так и иностранных электронных компонентов, основанную на уникальных принципах, упрощающих процесс администрирования, включая редактирование информации о надёжности ЭРИ, математических моделей ЭРИ и добавление новых классов ЭРИ [9].

Теоретическим фундаментом представленных выше продуктов служит общий логико-вероятностный метод системного анализа, который использует весь потенциал базового аппарата моделирования алгебры логики на основе операций «И», «ИЛИ» и «НЕ». Однако в контексте мостиковых схем такой подход неприменим по следующим причинам.

В общем случае событие безотказной работы всей системы представляет собой объединение пересечений событий безотказной работы отдельных элементов по всем непересекающимся путям от входа к выходу. Однако в этом случае простая формула для вычисления вероятностей не может быть применена, так как объединяемые пересечения могут быть как совместными, так и зависимыми, поскольку в разных пересечениях могут присутствовать события безотказной работы идентичных элементов.

Это указывает на то, что для работы с этими программными комплексами требуется предварительно преобразовывать наиболее часто встречающиеся мостиковые схемы надёжности в смешанные схемы надёжности. Этот процесс является сложным и не поддаётся автоматизации из-за уникальности этих схем.

## Постановка цели и задачи

Объектом данного проекта являются процессы вычисления безотказной работы по мостиковым схемам надёжности. Разработка в данной сфере обеспечит автоматизацию вычислений безотказной работы для самых распространенных (мостиковых) схем надёжности.

Цель: разработать демонстрационную программу, реализующую расчеты по произвольным мостиковым схемам надёжности.

Задачи:

1. Разработка алгоритма на основе контекстной логики для расчётов по произвольным мостиковым схемам надёжности
2. Создание объекта класса, отражающего специфику реализованного алгоритма
3. Разработка пользовательского интерфейса для удобного доступа и применения результатов работы класса
4. Тестирование разработанной программы

## Вывод по аналитической части

На данном этапе был проведен анализ программных комплексов в области расчётов произвольных мостиковых схем надёжности. Были установлены определённые для выполнения цели и задачи, следование которым должно осуществлять при разработке. На основе этого были сформированы общие требования к приложению и его функционалу. Дополнительные задачи и цели, необходимость которых не заявлялась в данном пункте, будут опционально рассмотрены и добавлены в ход разработки в следующих пунктах.

# Проектная часть

## Разработка алгоритма для подсчёта мостиковых схем надёжности с возможностью кодирования

### Расчёт мостиковых схем в булевой алгебре

Когда схема не сводится к последовательному и параллельному соединению элементов, т.е. является мостиковой, прямой подход расчета вероятности в соответствии с булевой алгеброй становится невозможным по следующей причине.

Событие безотказной работы всей системы является объединением пересечений событий безотказной работы отдельных компонентов по всем непересекающимся путям от входа к выходу. Поэтому в этом случае простая формула для вычисления вероятности объединения не применима, так как объединяемые пересечения могут быть как совместными, так и зависимыми, поскольку в разных пересечениях могут присутствовать события безотказной работы одних и тех же элементов.

Как было описано выше, существуют специализированные эмпирические методы. Они позволяют преобразовать мостиковую схему в схемы с только последовательным и параллельным соединениями.

Предлагаемый алгоритм основан на контекстной логике, что позволит обойти данное ограничение ввиду следующих факторов.

### Контекстная логика – основа булевой теории вероятности

В контекстной логике рассматриваются предложения, которые в отличие от высказываний, не обладают априорным значением истинности, но могу становиться истинными или ложными при проверках контекста. Над такими предложениями определены логические операции (дизъюнкция, конъюнкция, отрицание). Определения этих операций отличаются от определений операций алгебры высказываний указанием на любую проверку данного контекста. При этом в контекстной логике предложения, конъюнкция которых тождественно ложна, могут не быть противоположными, просто не может быть такой проверки, в которой оба предложения оказались бы одновременно истинными или одновременно ложными. Такие предложения названы несовместными. Важную роль в контекстной логике имеет понятие базиса контекста: совокупность несовместных предложений (базисные предложения), такая, что любое предложение контекста может быть представлено дизъюнкцией некоторых базисных предложений, а полная дизъюнкция их тождественно истинна (оказывается истинной при любой проверке контекста). Постулирование наличия базиса у любого контекста делает контекст замкнутой булевой алгеброй.

Алгебра контекстных предложений может быть булевой платформой теории вероятностей, если в качестве предложений контекста рассматривать предложения, определяющие события (например, “элемент 1 работает безотказно и элемент 2 отказал”). Базис контекста, при этом, соответствует полной системе несовместных событий. Могут быть определены свойства вероятностей контекстных предложений в соответствии с аксиоматикой Колмогорова, только теперь объединению событий соответствует дизъюнкция контекстных предложений, пересечению – конъюнкция, а дополнению – отрицание [10].

Для выражений алгебры контекстных предложений можно построить апостериорные таблицы истинности, перебором всех возможных значений истинности входящих в выражение предложений при проверках контекста и вычислением результатов. Это определяет формальное соответствие контекстных предложений переменным алгебры высказываний, в частности возможность построения совершенных нормальных форм выражений алгебры контекстных предложений по тем же правилам, что и для выражений из переменных алгебры высказываний.

если заданы несколько контекстных предложений, можно построить минимальный контекст для этой совокупности, перебирая все совершенные конъюнктивные одночлены (СКО), тогда все исходные предложения, и любые выражения из них, можно представить в виде совершенной нормальной дизъюнктивной формы (СДНФ), то есть в виде дизъюнкции базисных предложений [11].

### Предлагаемый алгоритм на основе контекстной логики

На основе всех вышеперечисленных высказываний был разработан и сформулирован следующий алгоритм (рис. 15).

Из предложений типа “элемент i работает безотказно” формируется дизъюнкция конъюнкций таких предложений по всем самонепересекающимся путям от входа к выходу (что есть предложение “вся схема работает безотказно”). Далее строится таблица апостериорной истинности данного предложения и по ней, стандартным способом (как и в алгебре высказываний) строится СДНФ. Вычисление вероятностей СКО, входящих в данную СДНФ, вычисляется произведением вероятностей исходных предложений или их отрицаний, поскольку исходные предложения независимы. Вероятности безотказной работы всей системы тогда вычисляется суммой вероятностей всех СКО, входящих в СДНФ, поскольку они несовместны [12].

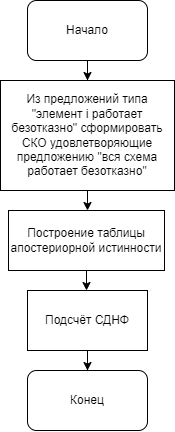


Рисунок 15 – Схема предложенного алгоритма

### Практическое подтверждение работоспособности алгоритма

Возьмём простейшую схему надёжности, в состав которой входят пять пронумерованных элементов. Элемент 3 в приведённой схеме является «мостом», поскольку он соединяет не параллельно и не последовательно две пары элементов, один – четыре и два – пять соответственно (рис. 16).

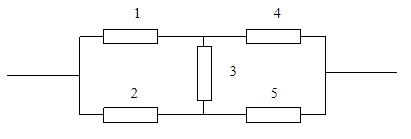


Рисунок 16 – Простейшая мостиковая схема надёжности

Для каждого из элементов можно составить предложение вида «элемент i работает безотказно». Теперь определим все самонепересекающиеся пути:

Это будут пути с элементами 1-4, 1-3-5, 2-5, 2-3-4. Если все элементы, входящие хотя бы в один путь сработали безотказно, то и вся схема сработала безотказно.

Теперь на основе имеющихся путей можно составить таблицу апостериорной истинности. Единица в пронумерованных столбцах соответствует высказыванию «элемент i работает безотказно», ноль – его отрицанию. Единица в столбце «AT» соответствует высказыванию «Вся схема сработала безотказно, ноль – его отрицанию (таблица 1).

Таблица 1 – Апостериорная таблица истинности для мостиковой схемы надёжности.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | AT |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

Следующим шагом будет построение на основе апостериорной таблицы истинности СДНФ. Для этого необходимо выбрать только те строки апостериорной таблицы истинности, которые соответствуют высказыванию «Вся схема сработала безотказно», т.е. единицы в столбце «AT» (таблица 2).

Таблица 2 – Таблица со значениями СДНФ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | AT |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

Для вычисления СДНФ для данной схемы, основываясь на приведённой выше таблице, будет справедлива формула (1).

|  |  |
| --- | --- |
|  | (1) |

Для вычисления вероятности безотказной работы всей схемы в полученную формулу необходимо занести вероятности безотказной работы каждого элемента, необходимо учесть, что отрицанию элемента соответствует обратная вероятность.

Тогда если для элементов схемы будут определены следующие вероятности (таблица 3) то будет эквивалентно следующее преобразование (2-3).

Таблица 3 – Вероятности элементов.

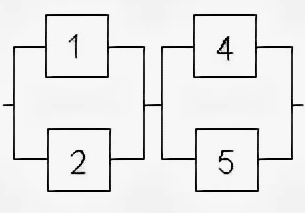
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | 1 | 2 | 3 | 4 | 5 |
| Вероятность | 0,2 | 0,3 | 0,5 | 0,4 | 0,7 |

|  |  |
| --- | --- |
|  | (2) |

|  |  |
| --- | --- |
| 0,0504+0,0336+0,0504+0,0144+0,0336+0,0084+0,0196+0,0294+0,0084+0,0196+0,0126+0,0036+0,0084+0,0126+0,0036+0,0084 = 0,317 | (3) |

Дальнейшие вычисления приведут к числу 0,317, что будет соответствовать вероятности безотказной работы представленной мостиковой схемы надёжности.

Для сравнения алгоритмов расчёта безотказной работы схемы надёжности посчитаем эту же схему методом разложения по мостам. Для этого эмпирическим методом определим, что «мостом» является элемент под номером 3 в корневой схеме надёжности (рис 16). Далее дополним исходную схему такими схемами, что выбранный мостовой элемент будет либо всегда безотказно работать, либо всегда отказывать (рис 17-18).



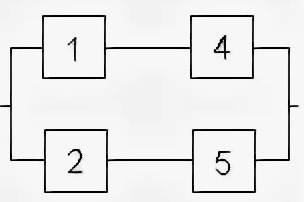


Рисунок – 17-18 Конечные схемы надёжности.

На данном этапе на конечных схемах присутствуют только последовательные или параллельные соединения, и теперь можно приступить к подсчётам вероятности безотказной работы на основе значений из таблицы 3. Вероятность исходной схемы рассчитывается как сумма произведений вероятностей корневых схем и вероятности моста. Опираясь на формулы булевой теории вероятности, будет справедлива следующая формула.

|  |  |
| --- | --- |
|  | (4) |

Данный пример показывает основное достоинство выведенного алгоритма на основе контекстной логики в виде отсутствия эмпирических методов расчета. Вычисления по выведенному алгоритму не зависят от числа и размеров мостовых элементов, в то время как эмпирическая часть существующего алгоритма может не справиться с преобразованиями исходной схемы без потери точности. Новый алгоритм решает ключевую проблему существующего: грамотное определение и выделение мостов. Соответственно скорость решения поставленной задачи с новым алгоритмом будет выше, а также будет решена основная цель, цифровизация аналитики рисков.

## Описание предметной области программы

Приложение разрабатывается для демонстрации вышеизложенного алгоритма. Поэтому оно должно соответствовать определённым системным требованиям, которые могут быть определены на этапе проектирования проекта. Эти требования включают:

1. Интерактивность: возможность ввода данных, на которых демонстрируется алгоритм, и задания наборов входных данных.
2. Простота использования: интуитивно понятный интерфейс
3. Платформо-независимость: работа на распространённых аппаратных конфигурациях и операционных системах.
4. Надежность и отказоустойчивость;
5. Автономность проектируемого приложения;

## Функциональное проектирование

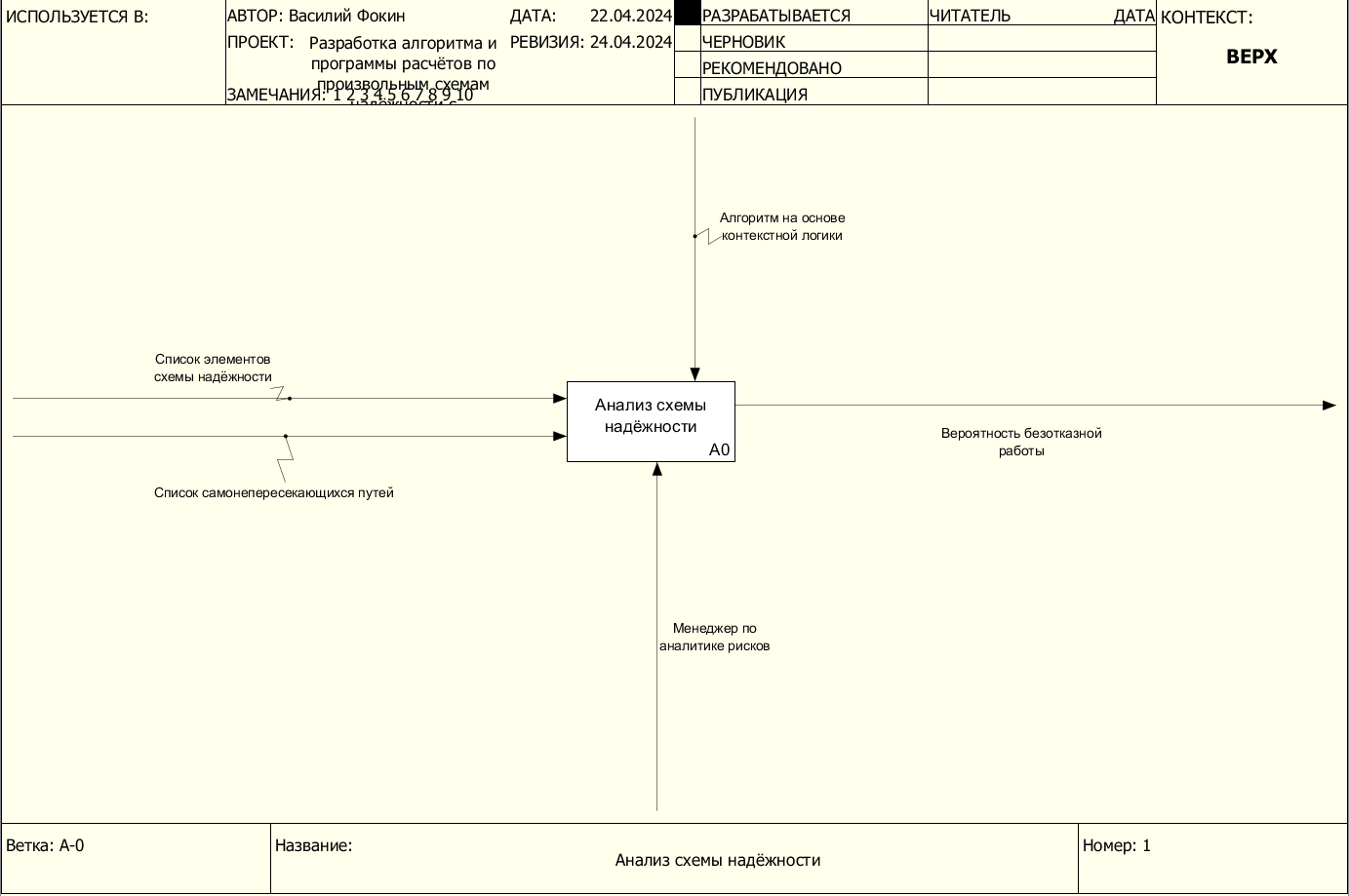
Бизнес-процессы представляют собой логическую структуру, которая определяет последовательность действий индивидов. Основная идея процесса заключается в анализе и стандартизации решения различных задач, возникающих в процессе создания продукта.

## Контекстная диаграмма

IDEF0 — это набор определённых блоков с собственными входными и выходными данными, а также процессами управления и механизмами. В совокупности этих аспектов происходит формализация и описание бизнес-процессов [13]. Диаграмма программы представлена на рисунке 19.

Точка зрения: Менеджер по аналитике рисков.

Тип диаграммы: AS – IS

Рисунок 19 – Диаграмма IDEF0

Вход:

1. Список элементов схемы надёжности

2. Список самонепересекающихся путей

Выход:

1. Вероятность безотказной работы

Управление:

1. Алгоритм на основе контекстной логики

Механизмы:

1. Менеджер по аналитике рисков

## Декомпозиция процесса

Декомпозиция — это метод разделения задачи на составные части для более детального изучения и оптимизации бизнес-процессов. Данный процесс будет показан на рисунке 20, а описание диаграммы осуществляется в таблице 4.

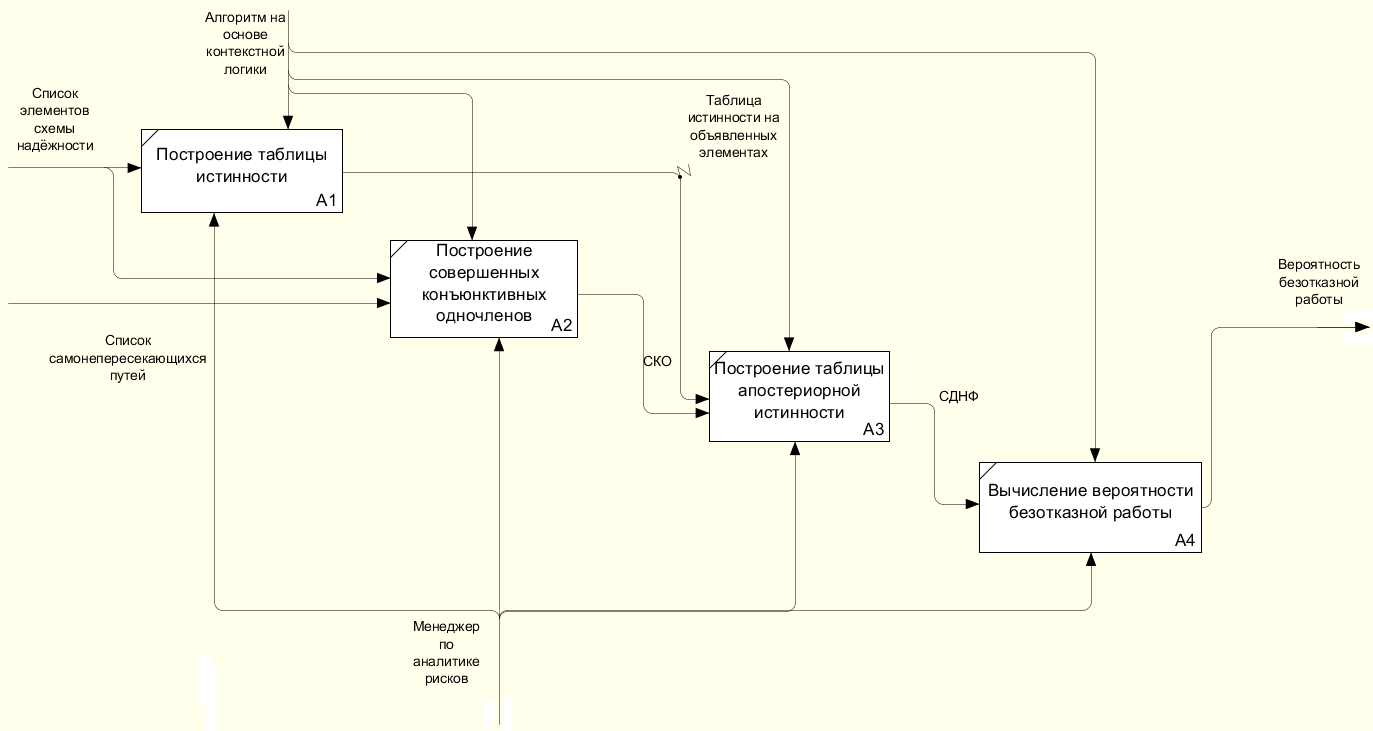


Рисунок 20 – Диаграмма декомпозиции

Таблица 4 – Описание диаграммы декомпозиции

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название блока | Вход | Выход | Управление | Механизм |
| Построение таблицы истинности | Список элементов схемы надёжности | Таблица истинности на объявленных элементах | Алгоритм на основе контекстной логики | Менеджер по аналитике рисков |
| Построение совершенных конъюнктивных одночленов | Список элементов схемы надёжности, список самонепересекающихся путей | Список совершенных конъюнктивных одночленов | Алгоритм на основе контекстной логики | Менеджер по аналитике рисков |

Окончание таблицы 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Построение таблицы апостериорной истинности | Таблица истинности на объявленных элементах, список совершенных конъюнктивных одночленов | Совершенная дизъюнктивная нормальная форма апостериорной таблицы истинности | Алгоритм на основе контекстной логики | Менеджер по аналитике рисков |
| Вычисление вероятности безотказной работы | Совершенная дизъюнктивная нормальная форма апостериорной таблицы истинности | Вероятность безотказной работы | Алгоритм на основе контекстной логики | Менеджер по аналитике рисков |

## Концептуальное проектирование

### Варианты использования

Диаграмма вариантов использования — это графическое представление, отражающее взаимосвязь между участниками (актёрами) и возможными сценариями использования системы. Она служит отправной точкой для проектирования и разработки системы, определяя её границы и основные функции.

Цели создания диаграммы вариантов использования включают:

1. Определение начальных границ и аспектов моделируемой предметной области,
2. Формирование требований к функциональным возможностям системы,
3. Создание базовой концептуальной модели для дальнейшей детализации в логическом и физическом аспектах.

Суть этой диаграммы заключается в том, что система представляет собой набор возможных сценариев использования, в которых участвуют внешние сущности (актёры), такие как технические устройства, программные системы или люди. Вариант использования определяет набор действий, выполняемых системой при взаимодействии с актёром [14].

Для данного демонстрационного приложения была создана диаграмма вариантов использования, представленная ниже (рис.21)

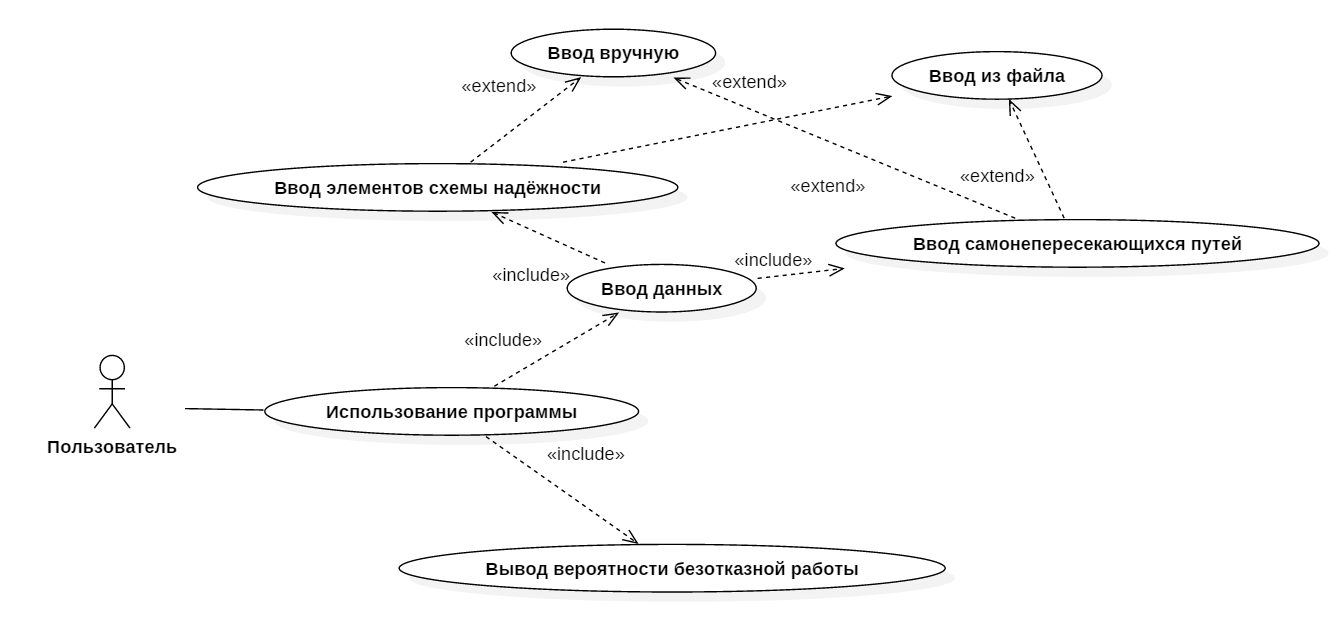


Рисунок 21 – Диаграмма вариантов использования

Использование программы пользователем понимается в виде ввода данных в программу и получения на основе введённых данных вероятности безотказной работы схемы надёжности. Ввод данных включает в себя ввод элементов схемы надёжности и ввод самонепересекающихся путей для той же схемы надёжности. Ввод элементов может выполняться как вручную, так и из файла.

### Диаграмма классов

Диаграмма классов — это структурная диаграмма языка моделирования UML, которая демонстрирует общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов и интерфейсов, а также взаимосвязи между ними. Она используется для документирования, визуализации и конструирования систем через прямое или обратное проектирование. На рисунке 22 приведена диаграмма классов, используемых для реализации разрабатываемого алгоритма. Атрибуты и операции изображённых классов представлены в таблице 5.

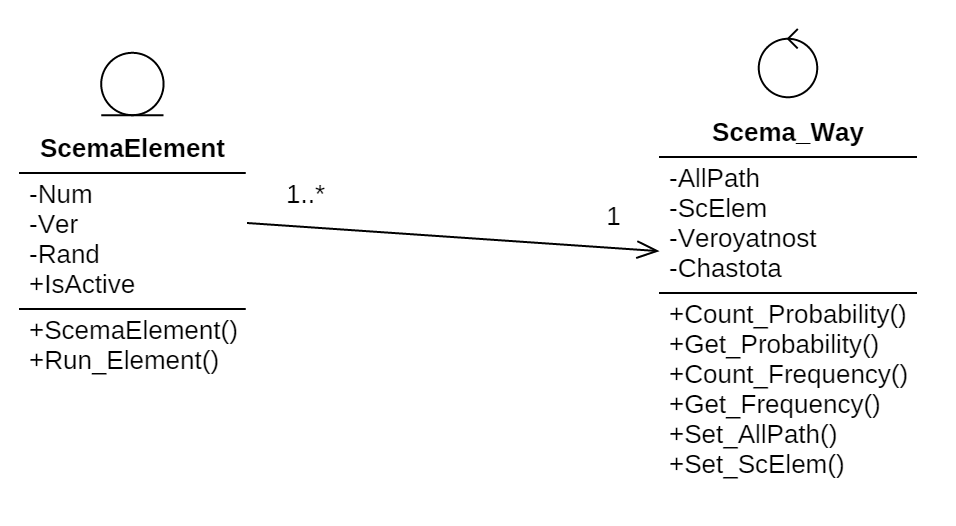


Рисунок 22 – Диаграмма классов

Таблица 5 – Атрибуты и операции диаграммы

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя класса | Стереотип | Атрибуты | | | Операции | |
| Имя | Тип | Квантор видимости | Имя | Тип |
| ScemaElement | <<entity>> | Num | Int | Private (закрытый) | ScemaElement | Num,Ver,  Rand |
| Ver | Double | Private (закрытый) | Run\_Element |  |
| Rand | Random | Private (закрытый) |  |  |
| IsActive | Bool | Public (открытый) |  |  |
| Scema\_Way | <<control>> | AllPath | List  <List  <int>> | Private (закрытый) | Count\_Probability |  |
| ScElem | List  <ScemaElement> | Private (закрытый) | Get\_Probability |  |
| Veroyatnost | Double | Private (закрытый) | Count\_Frequency | Count |
| Chastota | Double | Private (закрытый) | Get\_Frequency |  |
|  |  |  | Set\_AllPath | AllPath |
|  |  |  | Set\_ScElem | ScElem |

## Вывод по проектной части

На данном этапе был проведен системный анализ предметной области. Были предоставлены научные основания достоверности алгоритма. Так же представлены диаграммы и модели процессов разрабатываемого программного обеспечения.

# Технологическая часть

## Обоснование и выбор программных средств

Разработка может вестись на одном из нескольких актуальных языков программирования. Ниже предложена их сравнительная характеристика и в связи с этим обоснован выбор одного языка.

Java — это строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems. Он был создан для написания приложений для Android, промышленных программ, десктопных приложений, банковских программ, программ для работы с Big Data, научных программ, корпоративного софта, веб-серверов и веб-приложений. Java обладает мульти функциональностью, простотой синтаксиса, надёжностью, возможностью параллельной разработки и хорошей организацией кода [15].

Плюсы Java:

1. Автоматический сборщик мусора, который удаляет ненужные данные.
2. Возможность работы одного и того же кода на разных устройствах благодаря JVM.

Минусы Java:

1. Высокие требования к структурированию кода
2. Наличие множества фреймворков и библиотек, что может затруднить выбор подходящих инструментов для конкретной задачи

Python — это высокоуровневый язык программирования, эффективный, простой и универсальный. Он применяется в веб-разработке, машинном обучении, обработке больших данных и науке о данных [16].

1. Многофункциональность и обширная стандартная библиотека, охватывающая разные области программирования
2. Интеграция с различными инструментами и технологиями, такими как Django, Flask, NumPy, Pandas, и другими.

Минусы Python:

1. Медленная скорость выполнения по сравнению с компилируемыми языками, такими как C++.
2. Динамическая типизация и интерпретируемая природа могут замедлять выполнение кода.
3. Управление памятью осуществляется автоматически, что иногда приводит к менее эффективному использованию памяти.

C# — это объектно-ориентированный язык программирования, разработанный Microsoft. Он используется для создания различных типов приложений, включая веб-приложения, настольные приложения, мобильные приложения и игры. C# является частью платформы .NET и использует управляемую модель памяти, что обеспечивает безопасность и стабильность работы приложений [17].

Плюсы C#:

1. Высокая производительность и низкие требования к ресурсам.
2. Широкое использование в различных областях, таких как системное программирование, разработка операционных систем, баз данных и встраиваемых систем.
3. Множество фреймворков и библиотек для упрощения разработки.

Минусы C#:

1. Сложная архитектура и высокие требования к качеству кода.
2. Более медленное клиент-серверное программирование по сравнению с современными языками, такими как Java или Python.

Исходя из всех вышеперечисленных достоинств и недостатков считаю оптимальным разработку приложения на языке C#. Данный язык позволит обеспечить как высокую надёжность кода, так и позволит удобно встроить класс с алгоритмом в уже существующие программные решения.

## Разработка программных сущностей

### Пространство имён

После выбора необходимых программных средств для реализации можно приступить к созданию сущностей в контексте используемого языка программирования.

Для выполнения поставленных задач реализовано пространство имен RelSc. В нём будут храниться классы, с помощью объектов которых станет реализован алгоритм по расчету смешанных мостиковых схем надёжности.

### Класс ScemaElement

Это класс, описывающий определённый элемент схемы надёжности (рис 23).

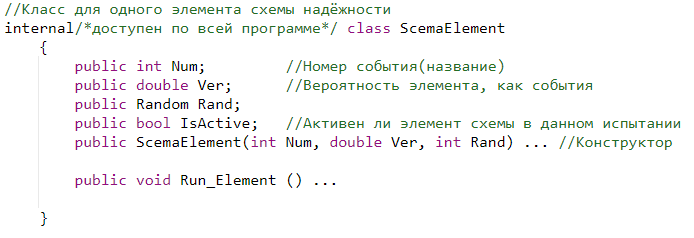


Рисунок 23 – Класс ScemaElement

Внутри этого класса определены такие переменные как:

Num – числовая переменная, в ней содержится уникальный номер события из схемы надёжности,

Ver – числовая переменная с плавающей точкой, в ней хранится вероятность описываемого события. Подразумевается, что вероятность находится в промежутке от 0 до 1.

Rand – переменная класса random, уникальный генератор псевдослучайных чисел используется для каждого элемента схемы надёжности, описывая его уникальность, как события.

IsActive – булева переменная, используется при определении частоты отказоустойчивости конкретного события.

Помимо переменных так же класс ScemaElement содержит функцию Run\_Element. Эта функция используется для определения частоты отказоустойчивости элемента схемы надёжности. Эта функция работает по следующему принципу: если число с плавающей точкой от 0 до 1, полученное с помощью псевдослучайного генератора чисел для конкретного элемента будет меньше, чем заявленная вероятность этого элемента, то данный элемент пройдёт проверку на отказоустойчивость.

### Класс Scema\_Way

Данный класс предназначен для определения частоты и вероятности событий в смешанных мостиковых схемах надёжности. Он принимает на вход список всех путей и элементов схемы и применяет функции, описывающие алгоритмы подсчёта этих величин.

Внутри этого класса определены такие переменные как:

AllPath – вложенный список, предназначен для хранения путей схемы надёжности.

ScElem – список состоящий из элементов описанного выше класса ScemaElement

Veroyatnost - числовая переменная с плавающей точкой, используется для подсчёта и хранения вероятности введённой схемы.

Chastota - числовая переменная с плавающей точкой, используется для подсчёта и хранения частоты введённой схемы.

Подсчёт частоты для мостиковой схемы надёжности выполняется по следующему алгоритму:

На вход функции подаётся число испытаний, количество раз, которое схема будет испытана на отказоустойчивость. Каждое из этих испытаний будет состоять из последующих шагов.

Для каждого элемента схемы из списка элементов будет выполнена проверка на отказоустойчивость, соответствующая функция из класса ScemaElement.

Далее выполняется проход по всем элементам из вложенного списка путей и если существует хотя бы один такой путь, все элементы которого прошли проверку на отказоустойчивость, то данная попытка работы схемы считается успешной.

Производится расчет количества успешных попыток работы схемы надёжности к общему числу попыток и сохраняется, и выводится соответствующая частота работы мостиковой схемы надёжности.

Вычисление вероятности для мостиковой схемы надёжности выполняется по следующему алгоритму (рис.24).

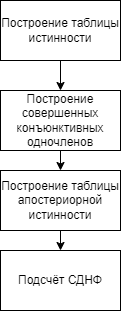


Рисунок 24 – Общая схема алгоритма для вычисления вероятности мостиковых схем надёжности

Происходит построение таблицы истинности на объявленных переменных. Число элементов таблицы соответствует числу уникальных элементов объявленной схемы надёжности.

Затем выполняется на основе объявленных путей. Составляется вложенный список, где если конкретный элемент содержится в конкретном пути, то ему сопоставляется единица, в противном случае ноль.

Далее с полученную таблицу совершенных конъюнктивных одночленов расширяем до таблицы апостериорной истинности, используя общую таблицу истинности и сохраняем только члены, входящие в совершенную дизъюнктивную нормальную форму.

Теперь по полученной таблице подсчитывается соответствующая сумма произведений, что является как СДНФ, так и вероятностью объявленной мостиковой схемы надёжности.

Код для описанных выше классов и их функций представлен в приложении А.

## Логика работы приложения

После описания класса для реализации алгоритма по подсчёту мостиковых схем надёжности следует описать общую логику работы с приложением, а также описать исключения, которые могут возникнуть в работе класса.

При запуске приложения пользователя встречает окно (рис.25)

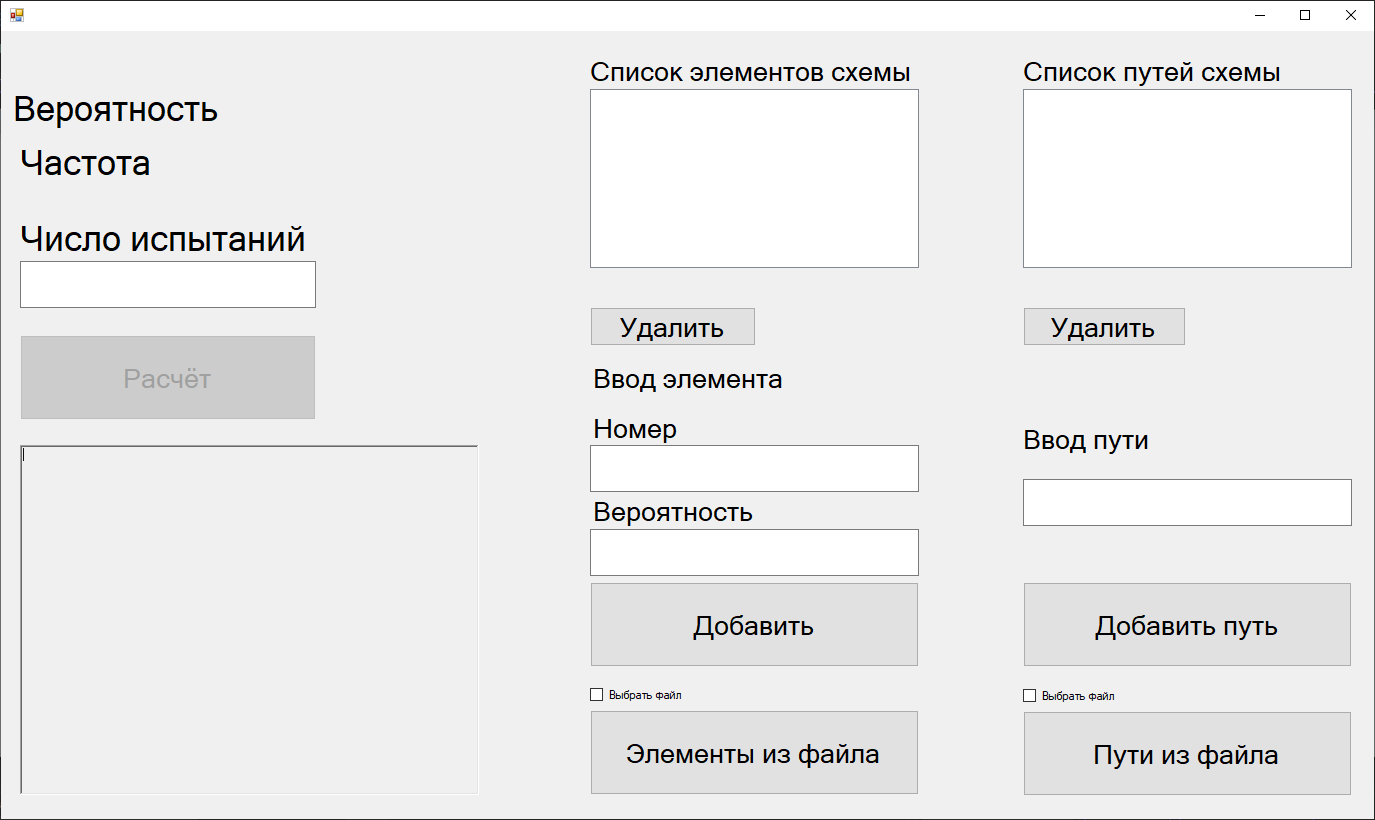


Рисунок 25 – Окно приложения

Первое, что должен сделать пользователь, это ввод данных в программу. Этот процесс может быть произведён либо вручную, с помощью специальных ячеек, либо из файла с заранее определённой структурой.

Для ввода элемента в ячейку «Номер» могут вводиться только целые положительные числа, а в ячейку «Вероятность» только числа с плавающей точкой на промежутке от 0 до 1. Ввод элемента схемы надёжности в программу осуществляется нажатием на кнопку «Добавить». Если указанный номер уже существует в списке элементов, тогда будет обновлена его вероятность. В ячейке «Ввод пути» должны быть указаны через пробел номера введённых элементов, образующих один путь схемы надёжности. Ввод пути в программу осуществляется нажатием на кнопку «Добавить путь». При внесении некорректных данных в поля, в информационном окне слева будет выведена соответствующая ошибка.

После того, как схема надёжности будет внесена и будет указано корректное число испытаний, станет активна кнопка «Расчёт» (рис.26)

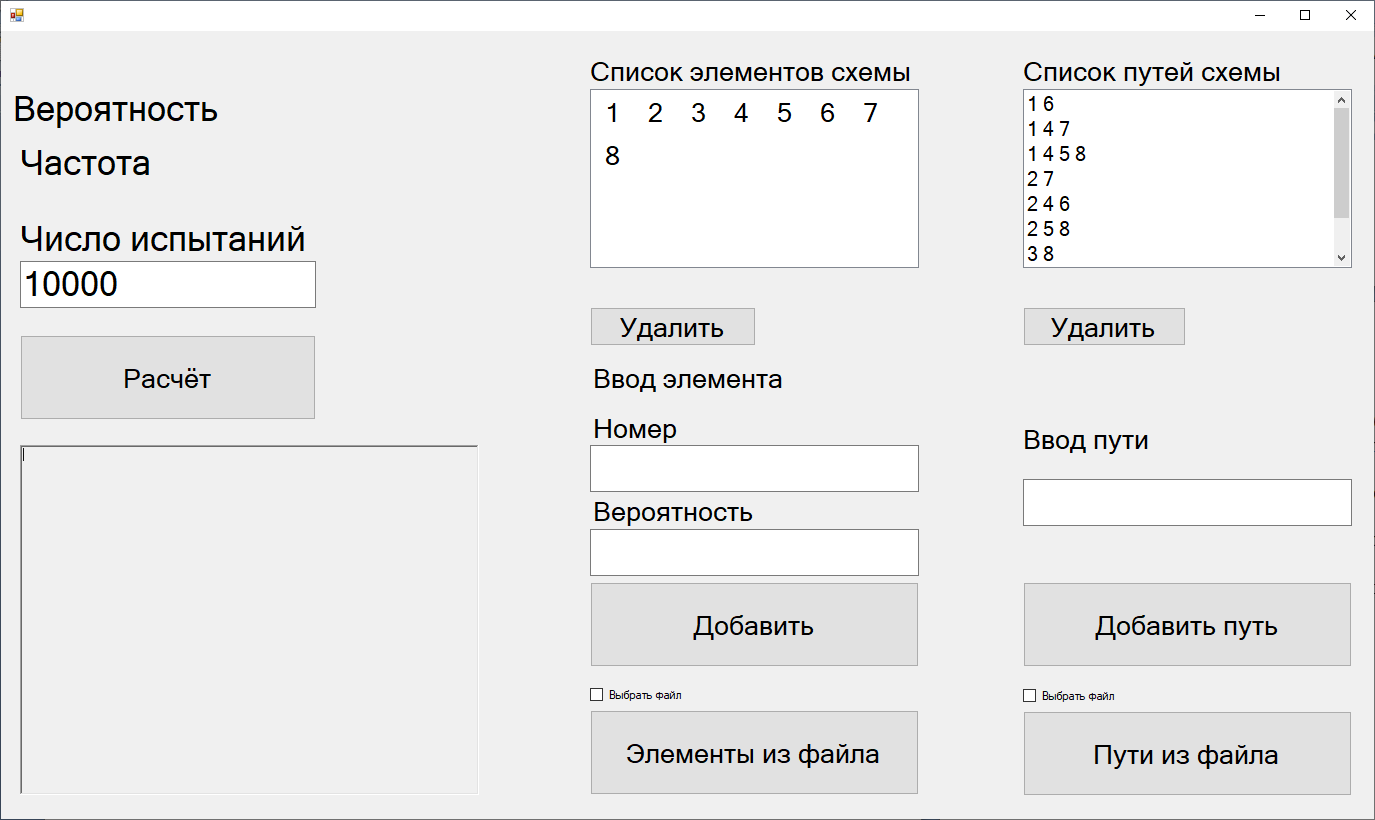


Рисунок 26 - Окно приложения после ввода основных данных

При нажатии на кнопку «Расчёт», выше, напротив соответствующих слов, будут выведены частота и вероятность для введённой схемы надёжности. (рис. 27)

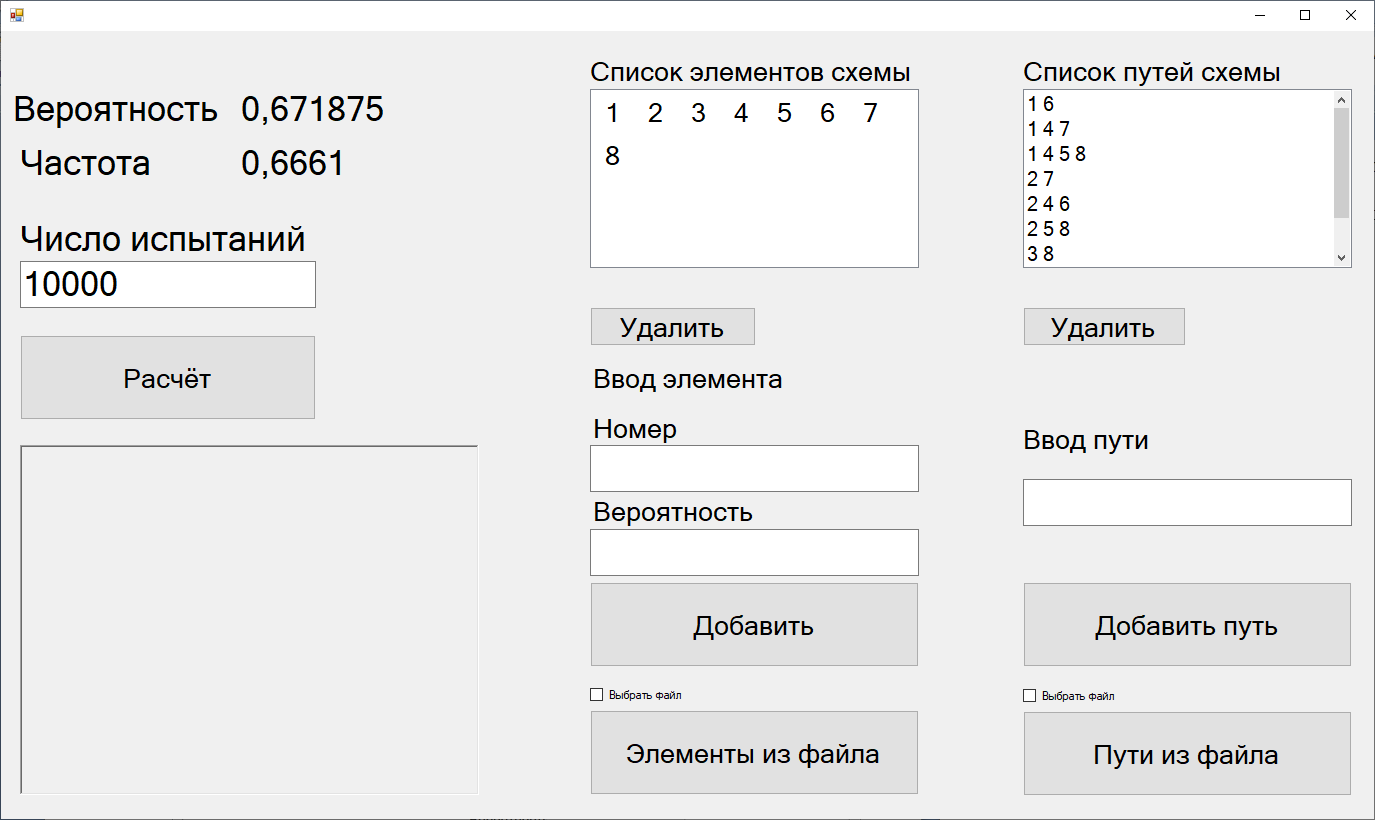


Рисунок 27 - Окно приложения поле выполнения расчётов

Если были внесены лишние элементы или пути схемы надёжности, то удалить их можно, сначала выбрав не нужный элемент, затем нажав соответствующую кнопку «Удалить» под блоком, где располагается элемент.

## Вывод по технологической части

На данном этапе был выбран язык программирования, на котором будет происходить реализация проекта, было разработано клиентское приложение. Также, были рассмотрены сценарии использования приложения и логика работы приложения.

# Экономическая часть

## Обоснование целесообразности разработки проекта

### Проблема, решаемая проектом

На сегодняшний день с текущими алгоритмами расчета надёжности мостиковых схем происходит не эффективно по времени, а самое важное, что существующий алгоритм невозможно цифровизовать ввиду наличия в нем шагов, использующих эвристические методы.

В связи с этим предлагается провести исследование в области контекстной логики и сформулировать новый алгоритм, который можно будет закодировать.

Причины выбора именно контекстной логики и математическое доказательство алгоритма были представлены выше.

Разработка предложенного алгоритма несёт ряд выгод:

1. Универсальность и гибкость решения, благодаря чему вероятность безотказной работы абсолютно любой схемы надёжности будет вычисляться автоматически.
2. Повышение эффективности работы аналитиков в области рисков, которым больше не придётся вручную форматировать полученные схемы надёжности для их анализа.

Цель проекта

Целью проекта обозначены формулировка алгоритма на основе контекстной логики для вычисления вероятности безотказной работы сколь угодно комплексных мостиковых схем надёжности. Написание класса на языке C#, реализующего вычисления на основе указанного выше алгоритма. Создание однопользовательского приложения, демонстрирующего работу класса.

### Жизненный цикл проекта

1. Разработка аванпроекта
   1. Исследования в области вычисления вероятности мостиковых схем надёжности и обоснование некомпетентности существующих алгоритмов.
      1. Исследования в области вычисления вероятности мостиковых схем надёжности
      2. Обоснование некомпетентности существующих алгоритмов.
   2. Исследования в области математики для формулирования алгоритма вычисления вероятности мостиковых схем надёжности без эвристических шагов.
      1. Математические исследования в области контекстной логики
2. Разработка технического задания
   1. Разработка ТЗ с учётом модульности решения
      1. Определение требований к системе: функциональные и нефункциональные требования, пользовательские сценарии, ограничения и предположения.
      2. Разработка структуры и содержания технического задания.
   2. Согласование ТЗ
      1. Подготовка и отправка технического задания заказчику для ознакомления и проверки соответствия требованиям
      2. Подписание акта согласования технического задания обеими сторонами.
3. Проектирование
   1. Программирование
      1. разработка прототипа продукта и его компонентов;
      2. создание структуры данных и программных кодов;
      3. написание технической документации.
   2. Отладка
      1. проверка работоспособности;
      2. исправление ошибок в коде.
   3. Тестирование
      1. проведение различных видов тестирования (функциональное, нефункциональное, интеграционное, системное);
      2. анализ результатов тестирования и устранение найденных ошибок;
      3. составление отчёта о качестве продукта.

Визуальная иллюстрация жизненного цикла продукта [18] представлена на рисунках 28-29.

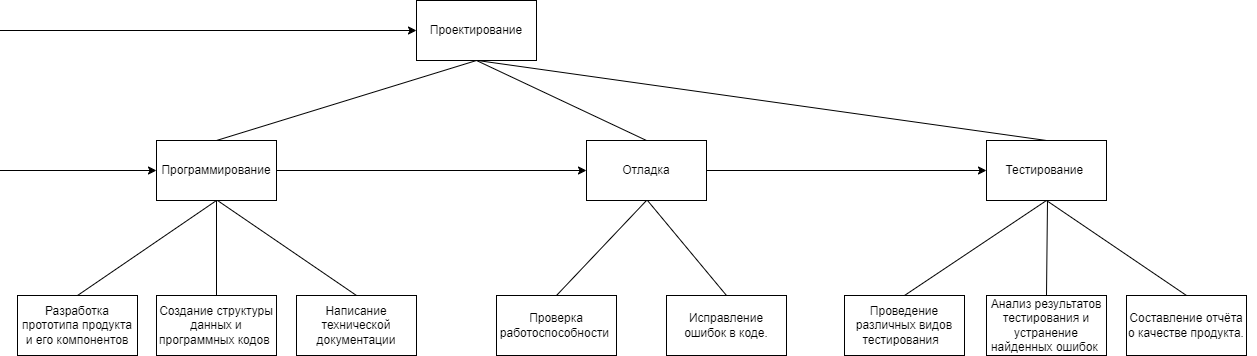
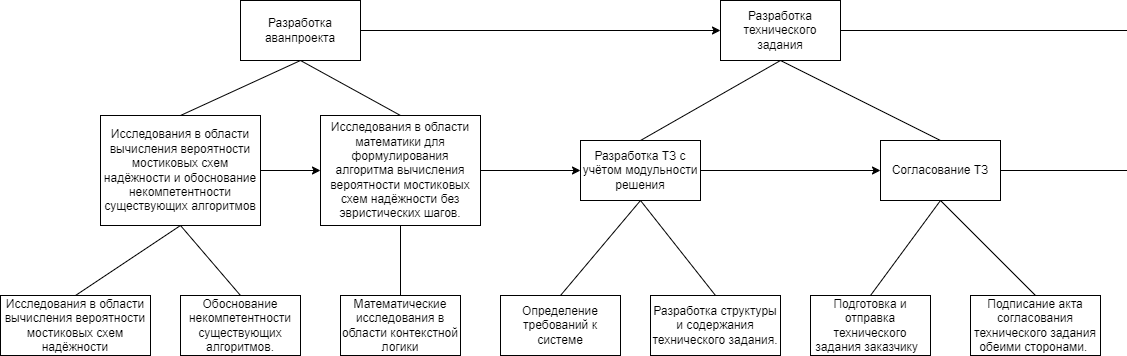


Рисунок 28-29 – Жизненный цикл продукта

### Реестр заинтересованных лиц

Заинтересованные стороны (стейкхолдеры) — это индивиды или группы, оказывающие влияние на проект или компанию в целом. У каждого стейкхолдера есть свои уникальные цели, ожидания и потребности, и он вносит вклад в проект или организацию. К заинтересованным сторонам могут относиться инвесторы, различные команды в организации, клиенты, поставщики, производители и регулирующие органы — все те, кого положительно или отрицательно затрагивает проект.

Таблица 6 – Заинтересованные лица проекта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование | Описание сторон | Цели и интересы сторон, степень их проявления | Степень влияния/вероятные риски от стороны |
| 1 | Компании разрабатывающие ПК для расчётов надёжности | Правообладатель ПК для расчётов надёжности | Повышение удобства пользователя  Повышение цифровизации продукта  Повышение престижа | Уровень власти – высокий  Уровень интереса – средний |
| 2 | Аналитики рисков | Конечный пользователь ПК для расчётов надёжности | Повышение качества и скорости работы | Уровень власти – низкий  Уровень интереса – высокий |
| 3 | Государство | Страна в лице Российской Федерации | Заинтересована в повышении цифровизации процессов, как в общем, так и в области аналитики рисков. | Уровень власти – низкий  Уровень интереса – средний |

### Структура команды проекта

Команда проекта — это совокупность людей, имеющих различные компетенции, должности, опыт, знания и ответственность, и объединённых общей целью — успешным завершением проекта. Команда необходимая для реализации данного проекта представлена в таблице 7.

Таблица 7 – Команда проекта

|  |  |
| --- | --- |
| Специалист | Обязанности |
| Data Scientist | Разработка инновационного алгоритма  Проверка функционирования построенного алгоритма |
| Программист С# | Создание программного обеспечения на языке программирования C#.  Отладка и исправление кода, интеграция программного обеспечения. |
| Руководитель | определение целей и задач проекта  составление списка работ и определение круга сотрудников для их выполнения  составление договоров с поставщиками и подрядчиками;  проведение деловых переговоров; |

### Бизнес-модель

Вместо того, чтобы создавать новый программный комплекс, способный конкурировать с текущими средствами анализа рисков предлагается распространять данное решение по модели Enterprise. Это значит, тесное сотрудничество с одной или несколькими компаниями, а программный компонент будет распространяться в виде модуля, который можно будет интегрировать в существующий программный комплекс, и осуществлять его дальнейшие поддержку и совершенствование.

### Планирование комплекса работ по разработке темы и оценка трудоемкости

Определение трудоёмкости работ основывается на учёте сроков выполнения задач, объёма выполняемых функций и выбранной среды программирования. Комплекс работ приведён в таблице 8.

Таблица 8 – Комплекс работ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Содержание работ | Исполнители | Длительность, дни | Загрузка | |
| дни | % |
| 1. Разработка аванпроекта | | | | |
| 1.1 Исследования в области вычисления вероятности мостиковых схем надёжности и обоснование некомпетентности существующих алгоритмов | | | | |

Продолжение таблицы 8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1.1.1 Исследования в области вычисления вероятности мостиковых схем надёжности | Data Scientist | 30 | 30 | 100 |
| 1.1.2 обоснование некомпетентности существующих алгоритмов | Data Scientist | 30 | 30 | 100 |
| 1.2 Исследования в области математики для формулирования алгоритма вычисления вероятности мостиковых схем надёжности без эвристических шагов. | | | | |
| 1.2.1 Математические исследования в области контекстной логики | Data Scientist | 30 | 30 | 100 |
| Итого по стадии 1 | Data Scientist | 90 | 90 | 100 |
| 2.Разработка технического задания | | | | |
| 2.1 Разработка ТЗ с учётом модульности решения | | | | |
| 2.1.1 Определение требований к системе: функциональные и нефункциональные требования, пользовательские сценарии, ограничения и предположения | Data Scientist | 2 | 1 | 50 |
| Руководитель | 2 | 100 |
| 2.1.2Разработка структуры и содержания технического задания | Data Scientist | 1 | 1 | 100 |
| Программист С# | 1 | 100 |
| 2.2Согласование ТЗ | | | | |

Продолжение таблицы 8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2.2.1Подготовка и отправка технического задания заказчику для ознакомления и проверки соответствия требованиям | Руководитель | 1 | 1 | 100 |
| 2.2.2 Подписание акта согласования технического задания обеими сторонами | Руководитель | 1 | 1 | 100 |
| Итого по стадии 2 | Data Scientist | 5 | 2 | 40 |
| Программист С# | 1 | 20 |
| Руководитель | 4 | 80 |
| 3. Проектирование | | | | |
| 3.1 Программирование | | | | |
| 3.1.1 разработка прототипа продукта и его компонентов; | Программист С# | 7 | 7 | 100 |
| 3.1.2создание структуры данных и программных кодов; | Программист С# | 7 | 7 | 100 |
| 3.1.3 написание технической документации | Программист С# | 10 | 5 | 50 |
| Руководитель | 10 | 100 |
| 3.2 Отладка | | | | |
| 3.2.1 проверка работоспособности | Программист С# | 2 | 2 | 100 |
| 3.2.2 исправление ошибок в коде | Программист С# | 1 | 1 | 100 |

Продолжение таблицы 8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3.3Тестирование | | | | |
| 3.3.1проведение различных видов тестирования (функциональное, нефункциональное, интеграционное, системное); | Программист С# | 3 | 3 | 100 |
| 3.3.2анализ результатов тестирования и устранение найденных ошибок; | Программист С# | 2 | 2 | 100 |
| 3.3.3 составление отчёта о качестве продукта. | Руководитель | 1 | 1 | 100 |
| Итого по стадии 3 | Программист С# | 33 | 27 | 81 |
| Руководитель | 11 | 33 |
| Итого по проекту | Data Scientist | 128 | 92 | 71 |
| Программист С# | 28 | 21 |
| Руководитель | 15 | 11 |

## Расчётная часть

### Анализ структуры затрат

Расходы на реализацию проекта включают затраты на оплату труда исполнителей, приобретение или аренду оборудования, обустройство рабочих мест и дополнительные издержки (1):

|  |  |
| --- | --- |
| С = Сзп + Сэл + Соб + Сорг + Снакл | (1) |

где: Сзп – заработная плата исполнителей;

Сэл – затраты на электроэнергию;

Соб – затраты на обеспечение требуемым оборудованием;

Сорг – затраты на обеспечение рационального трудового процесса;

Снакл – накладные расходы.

Определение затрат на выплату заработной платы исполнителям происходит согласно следующему соотношению (2):

|  |  |
| --- | --- |
| *Сзп = Сз.осн + Сз.доп + Сз.отч* | (2) |

где: Сз.осн – основная заработная плата;

Сз.доп – дополнительная заработная плата;

Сз.отч – отчисление с заработной платы.

Определение базового вознаграждения при оплате труда исполнителей на основе дневного тарифа производится с учётом их окладов и графика занятости (3):

|  |  |
| --- | --- |
| Сз.осн = *Одн* ×*Тзан* | (3) |

где: Одн – дневной оклад исполнителя;

Тзан – число дней, выработанных исполнителем проекта.

При 8-и часовом рабочем дне оклад рассчитывается (4):

|  |  |
| --- | --- |
|  | (4) |

где: Омес – месячный оклад;

Fм – месячный фонд рабочего времени (8).

Таблица 9 представляет расчёт заработной платы с указанием исполнителей, их месячных и дневных окладов, а также времени участия в проекте для каждого сотрудника.

Таблица 9 – Расчёт заработной платы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Должность | Оклад, руб. | Дневной оклад, руб | Трудовые затраты, дни. | Заработная плата, руб. |
| 1 | Data Scientist | 120 000 | 5818 | 92 | 535 256 |
| 2 | Программист С# | 60 000 | 2909 | 28 | 81 452 |
| 3 | Руководитель | 80 000 | 3878 | 15 | 58 170 |
| Итого | |  |  |  | 674 878 |

Расходы на дополнительную заработную плату включают все выплаты, производимые непосредственно исполнителям за время, не отработанное, но предусмотренное законодательством, включая оплату очередных отпусков, компенсацию за неиспользованный отпуск и другие платежи. Эти выплаты составляют 20 % от суммы основного заработка. (5-6):

|  |  |
| --- | --- |
| Сз.доп = 0,2 × Сз.осн | (5) |
| 134 975 = 0,2\*674 878 | (6) |

Общая дополнительная заработная плата будет равна 809 853 руб.

Отчисления с заработанной платы составят для ИТ-компаний — 7,6% с выплат в пределах 1917000₽ на сотрудника. Сверх этой суммы взносы не начисляют (7-8):

|  |  |
| --- | --- |
| Сз.отч = (Сз.осн + Сз.доп)×7,6% | (7) |
| 61 548 = 809 853 ×7,6% | (8) |

Таблица 10 – Расходы на заработную плату

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Должность | Заработная плата, руб. | Расходы на дополнительную заработную плату | Отчисления с заработной платы. | Итого затраты на оплату труда |
| 1 | Data Scientist | 535 256 | 107 051 | 48 815 | 691122 |
| 2 | Программист С# | 81 452 | 16290 | 7428 | 105170 |
| 3 | Руководитель | 58 170 | 11634 | 5305 | 75109 |
| Итого | |  |  |  | 871402 |

### Затраты на оборудование

Затраты, связанные с обеспечением рабочих процессов оборудованием и программным обеспечением, должны начинаться с определения состава оборудования и оценки необходимости его приобретения или аренды. Основным оборудованием, необходимым для работы, является ноутбук DEXP Aquilon, который был куплен.

В данном случае покупки подсчитывается величина годовых амортизационных отчислений по следующей формуле (9):

|  |  |
| --- | --- |
| *Аг* = *Сбал* × *Нам* | (9) |

где: Аг – сумма годовых амортизационных отчислений, руб;

Сбал – балансовая стоимость компьютера, руб./шт.;

Нам – норма амортизации, %.

|  |  |
| --- | --- |
| АП = Аг / 365×Тк | (10) |

где: АП – сумма амортизационных отчислений за период использования дней, руб.;

ТК – время эксплуатации устройства при разработке программы.

Согласно информации из проекта, для программной реализации требуется 128 дней. При этом время использования компьютера во время создания программы также составило 128 дней.

Амортизационные отчисления на устройство и программное обеспечение производятся линейным методом относительно срока эксплуатации.

Балансовая стоимость ПЭВМ определяется исходя из отпускной цены, затрат на транспортировку, монтаж оборудования и его наладку, и рассчитывается по формуле (11):

|  |  |
| --- | --- |
| Сбал = Срын × Зуст | (11) |

где: Сбал – балансовая стоимость ПЭВМ, руб.;

Срын – рыночная стоимость устройства, руб./шт.;

Зуст – затраты на доставку и установку, %.

В рассматриваемом случае компьютер, на котором выполнялась работа, был приобретён до разработки программного обеспечения по стоимости 18 000 рублей с учётом затрат на доставку.

Общая амортизация за время эксплуатации компьютера при создании программы вычисляется по формуле (12):

|  |  |
| --- | --- |
| АО=АЭВМ | (12) |

где: АЭВМ – амортизационные отчисления на компьютер за время его

эксплуатации;

Отсюда следует (Срок полезного использования установлен 2 года) (13-14):

|  |  |
| --- | --- |
| АЭВМ = ((18 000 × 0,5)/365) × 128 = 3156 руб.; | (13) |
| АП =3156 руб. | (14) |

### Расчет затрат на содержание и эксплуатацию оборудования

Расходы на обслуживание и использование оборудования определяются как 4 % от стоимости основных средств:

|  |  |
| --- | --- |
| Зтр = Сбал × Пр ×Тк / 365 | (15) |

где: Пр – процент затрат на содержание и эксплуатацию оборудования, %.

Отсюда (16):

|  |  |
| --- | --- |
| Зтр = 18 000 × 0,04 × 128 / 365 = 252 руб | (16) |

### Затраты на электроэнергию

В данном пункте рассмотрена стоимость электроэнергии, потребляемой компьютерами, серверами и другим оборудованием, используемым во время разработки программы.

Стоимость электроэнергии, потребляемой за год, определяется по формуле (17):

|  |  |
| --- | --- |
| ЗЭЛ=РЭВМ×ТЭВМ×СЭЛ | (17) |

где: РЭВМ – суммарная мощность ЭВМ, кВТ;

ТЭВМ – время работы компьютера, часов;

СЭЛ – стоимость 1 кВт/ч электроэнергии, руб.

Согласно техническому паспорту ЭВМ РЭВМ= 0,5 кВт/ч электроэнергии, а стоимость 1 кВт/ч электроэнергии в г. Ярославль компании на первое полугодие 2024 года СЭЛ = 5,6 руб. Тогда расчетное значение затрат на электроэнергию равна (18):

|  |  |
| --- | --- |
| Зэл = 0,5 × 128× 8× 5,6 = 2867 руб | (18) |

### Накладные расходы

Накладные расходы, связанные с реализацией проекта, определяются исходя из затрат на основную заработную плату. Обычно они варьируются от 60 % до 100 % расходов на основную заработную плату (19).

|  |  |
| --- | --- |
| Снакл = 0,6×Сз осн | (19) |

Накладные расходы составят (20):

|  |  |
| --- | --- |
| Снакл = 0,6× 674878= 404926 руб | (20) |

### Общие затраты на разработку

Таблица 11 – Расчет затрат на разработку

|  |  |  |
| --- | --- | --- |
| Статьи затрат | Затраты на проект, руб. | Удельный вес, % |
| Расходы по заработной плате | 871 402 | 67,72 |
| Амортизационные отчисления | 3 156 | 0,25 |
| Затраты на электроэнергию | 2 867 | 0,22 |
| Затраты на содержание и эксплуатацию оборудования | 315 | 0,02 |

Продолжение таблицы 11

|  |  |  |
| --- | --- | --- |
| Затраты на внедрение ИС | 4 091 | 0,32 |
| Накладные расходы | 404 926 | 31,47 |
| Прочие расходы | 0 | 0,00 |
| Итого | 1 286 759 | 100 |

### Оценка эффективности

Существующий алгоритм по нахождению вероятности безотказной работы мостиковых схем имеет циклический эмпирический шаг «Определение мостового элемента» в то время, как предлагаемый алгоритм линеен. Это означает что он не только покажет меньшее время работы на комплексных мостиковых схемах, но и увеличится точность подсчётов, поскольку будет убран человеческий фактор в решении задачи. Сравнение алгоритмов представлено на рисунке 30.

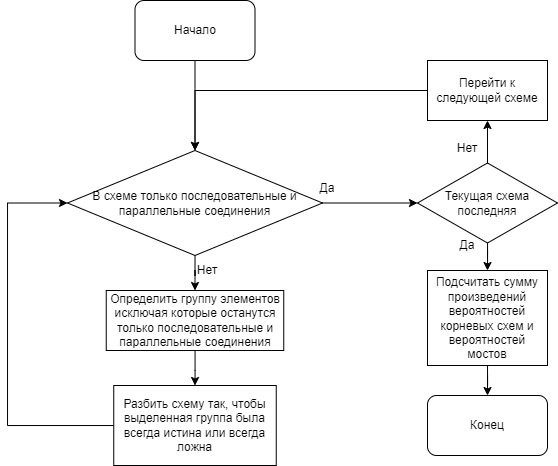
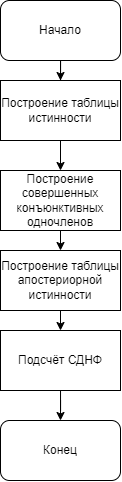


Рисунок 30 – Сравнение предложенного (слева) и текущего (справа) алгоритмов

Главное достоинство алгоритма это его возможность быть полностью закодированным. Его программное представление может быть на модульной основе добавлено в существующий программный комплекс, рассчитывающий вероятность безотказной работы схем надёжности.

Поскольку аналогичного полностью цифрового решения не существует, то наличие подобного модуля в программном комплексе может являться и элементом престижа.

## Вывод по экономической части

В этой главе были изучены экономические аспекты приобретения и внедрения разработанного продукта. Кроме того, были проанализированы факторы, определяющие экономическую обоснованность разработки.

# Заключение

Целью выпускной квалификационной работы была разработка алгоритма и программы расчетов по произвольным схемам надежности с использованием контекстной логики.

Для достижения этой цели было проведено исследование в области контекстной логики, и разработан алгоритм для вычисления безотказной работы сколь угодно сложных мостиковых схем надёжности.

В рамках разработки алгоритма была опубликована научная статья в сборнике студенческой конференции: Применение контекстной логики для разработки алгоритма расчета систем с произвольной схемой надежности. Семьдесят седьмая всероссийская научно-техническая конференция студентов, магистрантов и аспирантов с международным участием. 17-18 апреля 2024 г., Ярославль: сб. материалов конф. В 3 ч. Ч. 3. – Ярославль : Изд-во ЯГТУ, 2024, с. 82-85.

В ходе работы был создан класс на основе выведенного алгоритма и создана программа, демонстрирующая его работу.

Была проведена оценка экономической эффективности проекта. Подсчитаны расходы на исследования и разработку программного обеспечения. Оценена производительность труда. Итогом выполнения выпускной квалификационной работы стало разработанное программное обеспечение, реализующее выведенный алгоритм на основе контекстной логики для подсчёта безотказной работы сколь угодно сложных мостиковых схем надёжности.

# Список используемых источников

### ПРОГРАММА "Цифровая экономика Российской Федерации" УТВЕРЖДЕНА распоряжением Правительства Российской Федерации от 28 июля 2017 г. № 1632-р [Электронный ресурс]. Режим доступа: <http://static.government.ru/media/files/9gFM4FHj4PsB79I5v7yLVuPgu4bvR7M0.pdf> (дата обращения: 24.05.2024);

### Управление цифровыми рисками в современной экономике [Электронный ресурс]. Режим доступа: <https://esj.today/PDF/15ECVN124.pdf> (дата обращения: 24.05.2024);

### УПРАВЛЕНИЕ РИСКАМИ КОМПАНИИ В ПРОЦЕССЕ ЕЕ ЦИФРОВИЗАЦИИ [Электронный ресурс]. Режим доступа: <https://cyberleninka.ru/article/n/upravlenie-riskami-kompanii-v-protsesse-ee-tsifrovizatsii> (дата обращения: 24.05.2024);

### ОСНОВЫ ТЕОРИИ НАДЕЖНОСТИ [Электронный ресурс]. Режим доступа: [https://library.pguas.ru/xmlui/bitstream/handle/123456789/1112/ Лянден%20Основы%20теор%20надеж.pdf?sequence=1&isAllowed=y](https://library.pguas.ru/xmlui/bitstream/handle/123456789/1112/%20Лянден%20Основы%20теор%20надеж.pdf?sequence=1&isAllowed=y) (дата обращения: 24.05.2024);

### Надёжность систем управления [Электронный ресурс]. Режим доступа: [https://portal.tpu.ru/departments/otdel/publish/ izdaniya\_razrabotanye\_v\_ramkah\_IOP/Tab/ik\_shklyar\_nadezhnost\_system\_upravleniya.pdf](https://portal.tpu.ru/departments/otdel/publish/izdaniya_razrabotanye_v_ramkah_IOP/Tab/ik_shklyar_nadezhnost_system_upravleniya.pdf) (дата обращения: 24.05.2024);

### НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ. Надежность в технике СТРУКТУРНАЯ СХЕМА НАДЕЖНОСТИ [Электронный ресурс]. Режим доступа: https://docs.cntd.ru/document/1200180919 (дата обращения: 24.05.2024);

### Надежность технических систем и техногенный риск [Электронный ресурс]. Режим доступа: https://studfile.net/preview/16567654/ (дата обращения: 24.05.2024);

### Обзор программных комплексов по расчету надежности сложных технических систем [Электронный ресурс]. Режим доступа: https://cyberleninka.ru/article/n/obzor-programmnyh-kompleksov-po-raschetu-nadezhnosti-slozhnyh-tehnicheskih-sistem (дата обращения: 24.05.2024);

### Отечественные программные комплексы для расчета надежности технических систем [Электронный ресурс]. Режим доступа: https://scienceforum.ru/2024/article/2018035358 (дата обращения: 24.05.2024);

### Раухваргер А.Б., Калинин В.Б. О возможности построения контекстной логики и ее использования в качестве логической основы программирования и булевой платформы теории вероятностей. Актуальные проблемы математики и информатики. Сборник статей к 20–летию факультета ИВТ.-Ярославль, ЯрГУ, 2007, с. 81-86. (дата обращения: 24.05.2024);

### Раухваргер А.Б. О контекстной логике и возможных сферах ее применения. Математические методы в технике и технологиях. Сборник трудов XX Международной научной конференции.Том 8. - Ярославль, ЯГТУ, 2007, с. 198-202. (дата обращения: 24.05.2024);

### Раухваргер А.Б. Надконтекстные функции и контекст доказательств. Математические методы в технике и технологиях: сб. тр. междунар. науч. конф.: в 12 т. Т.4. - СПб.: Изд-во Политехн. ун-та, 2018. - с 95-98. (дата обращения: 24.05.2024);

### МЕТОДОЛОГИЯ ФУНКЦИОНАЛЬНОГО МОДЕЛИРОВАНИЯ IDEF0 [Электронный ресурс]. Режим доступа: https://advanced-quality-tools.ru/assets/idef0-rus.pdf (дата обращения: 24.05.2024);

### Использование диаграммы вариантов использования UML при проектировании программного обеспечения [Электронный ресурс]. Режим доступа: https://habr.com/ru/articles/566218/ (дата обращения: 24.05.2024);

### The Arrival of Java 22 [Электронный ресурс]. Режим доступа: https://blogs.oracle.com/java/post/the-arrival-of-java-22 (дата обращения: 24.05.2024);

### General Python FAQ [Электронный ресурс]. Режим доступа: https://docs.python.org/3/faq/general.html (дата обращения: 24.05.2024);

### Документация по языку C# [Электронный ресурс]. Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения: 24.05.2024);

### ГОСТ Р 53622-2009 Информационные технологии. Информационно-вычислительные системы [Электронный ресурс]. Режим доступа: https://ohranatruda.ru/upload/iblock/4a5/4293812344.pdf (дата обращения: 24.05.2024);

# Приложение А

Код для пространства имён RelSc

namespace RelSc

{

//Класс для одного элемента схемы надёжности

internal/\*доступен по всей программе\*/ class ScemaElement

{

public int Num; //Номер события(название)

public double Ver; //Вероятность элемента, как события

public Random Rand;

public bool IsActive; //Активен ли элемент схемы в данном испытании

public ScemaElement(int Num, double Ver, int Rand) //Конструктор

{

if (Ver <= 1 && Ver >= 0)

{

this.Num = Num;

this.Ver = Ver;

this.Rand = new Random(Rand);

}

else throw new Exception("Не корректная вероятность");

}

public void Run\_Element ()

{

if (Rand.NextDouble() <= Ver)

{

IsActive = true;

}

else { IsActive = false; }

}

}

internal class Scema\_Way

{

List<List<int>> AllPath = new List<List<int>>();

List<ScemaElement> ScElem = new List<ScemaElement>();

double Veroyatnost = 0;

double Chastota = 0;

public Scema\_Way(List<List<int>> AllPath, List<ScemaElement> ScElem)

{

bool a;

bool may = true;

//Каждый из элементов хотя бы в одной строке

foreach (var item in ScElem)

{

foreach (var item2 in AllPath)

{

a = item2.Contains(item.Num);

if (a)

{ may = false; break; }

may = true;

}

if (may)

{

//Элемента нет ни в одной строке

throw new Exception("Элемента нет ни в одной строке");

}

}

//В строке есть только элементы из списка

foreach (var stroka in AllPath)

{

foreach (var elstroka in stroka)

{

ScemaElement tmp = ScElem.Find(it => it.Num == elstroka);

if (tmp == null)

{

may = true;

throw new Exception("Один из путей содержит необъявленные элементы " + elstroka.ToString());

};

}

}

this.AllPath = AllPath;

this.ScElem = ScElem;

}

public double Count\_Probability()

{

//Составление таблицы истинности (на объявленных элементах схемы)

List<List<int>> truthTable = new List<List<int>>();

int n = ScElem.Count;// Количество переменных

for (int i = 0; i < (1 << n); i++)

{

List<int> row = new List<int>();

for (int j = 0; j < n; j++)

{

if ((i & (1 << j)) != 0)

{

row.Add(1);

}

else

{

row.Add(0);

}

}

truthTable.Add(row);

}

//Построение совершенных конъюнктивных одночленов на основе объявленных путей

List<List<int>> WaySKO = new List<List<int>>();

foreach (var way in AllPath)

{

List<int> row = new List<int>();

foreach (var item in ScElem)

{

if (way.Exists(i => i == item.Num))

{

row.Add(1);

}

else

{

row.Add(0);

}

}

WaySKO.Add(row);

}

//Таблица апостериорной истинности (только СДНФ)

List<List<int>> ApostTT = new List<List<int>>();

foreach (var itemS in WaySKO)

{

for (int j = 0; j < truthTable.Count; j++)

{

var itemT = truthTable[j];

bool fail = false;

for (int i = 0; i < ScElem.Count; i++)

{

if (itemS[i] == 1 && itemT[i] == 0)

{

fail = true;

break;

}

}

if (!fail)

{

ApostTT.Add(itemT);

truthTable.Remove(itemT);

j--;

}

}

}

//Подсчёт СДНФ

double chway = 1;

Veroyatnost = 0;

foreach (var item in ApostTT)

{

for (int i = 0; i < ScElem.Count; i++)

{

if (item[i] == 1)

{

chway \*= ScElem[i].Ver;

}

else

{

chway \*= (1 - ScElem[i].Ver);

}

}

Veroyatnost += chway;

chway = 1;

}

return Veroyatnost;//Вывод

}

public double Get\_Probability()

{

return Veroyatnost;

}

public double Count\_Frequency(int Count)

{

int Count\_SC = 0;

bool Way\_open = false;

for (int i = 0; i < Count; i++)

{

ScElem.ForEach(item => item.Run\_Element()); //Проверка на отказоустойчивость

foreach (var way in AllPath)

{

foreach (var elem in way)

{

ScemaElement tmp = ScElem.Find(it => it.Num == elem);

if (tmp.IsActive)

{

Way\_open = true;

}

else

{

Way\_open = false;

break;

}

}

if (Way\_open)

{

Count\_SC++;

Way\_open = false;

break;

}

}

}

Chastota = (double)Count\_SC / (double)Count;

return Chastota;

}

public double Get\_Frequency()

{

return Chastota;

}

}

}