  
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
**«Дальневосточный федеральный университет»**

**ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ  
Департамент программной инженерии и искусственного интеллекта**

**РАЗРАБОТКА СИСТЕМЫ ДЛЯ ИССЛЕДОВАНИЯ ДИНАМИКИ ВЗАИМОДЕЙСТВУЮЩИХ ВИДОВ**

КУРСОВОЙ ПРОЕКТ  
по дисциплине «Технологии коллективной промышленной разработки информационных систем» по образовательной программе подготовки бакалавров по направлению 09.03.04 «Программная инженерия»

Выполнили:  
студенты гр. Б9120-09.03.04прогин  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Аникин Д. В.   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Мельникова Е. А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Разумов С. И.  
  
Руководитель:  
ассистент департамента ПИиИИ  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Иваненко Ю. С.

г. Владивосток  
2024

**Оглавление**

[Введение 3](#_Toc124728865)

[1 План проекта 4](#_Toc124728866)

[2 Регламент проведения инспекции 5](#_Toc124728867)

[3 Модель состояний задач 8](#_Toc124728868)

[4 Презентация проекта 10](#_Toc124728869)

[5 Требования к проекту 13](#_Toc124728870)

[6 Разработка архитектуры проекта 15](#_Toc124728871)

[7 Измерения проекта 19](#_Toc124728872)

[8 Перечень задач проекта 21](#_Toc124728873)

[9 Правила по кодированию 24](#_Toc124728874)

[10 Разработка плана тестирования проекта 29](#_Toc124728875)

[11 Тестирование проекта 36](#_Toc124728876)

[Заключение 45](#_Toc124728877)

[Список литературы 46](#_Toc124728878)

# Введение

Промышленная разработка информационных систем включает в себя множество этапов, начиная от разработки плана проекта, заканчивая его тестированием, для чего необходимо множество специалистов различных профилей, а также унифицированные методы коммуникации между ними, с помощью которых можно разделить обязанности членов команды по их специализации.

Исходя из описанного выше, необходимо использовать определенные технологии коллективной разработки для повышения эффективности работы в группе и соответствия конечного продукта заявленным требованиям.

В данной курсовой работе рассматривается задача коллективной разработки программного средства «Программная система для автоматического создания выкроек» и составление технической документации к нему.

Таким образом, целью курсовой работы является разработка программного средства с использованием подходов коллективной промышленной разработки.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Разработать план проекта.
2. Разработать регламент проведения инспекции.
3. Разработать модель состояний задач.
4. Разработать презентацию проекта.
5. Разработать требования к проекту.
6. Разработать архитектуру проекта.
7. Разработать измерения проекта.
8. Разработать перечень задач проекта.
9. Разработать рекомендации по кодированию.
10. Разработать план тестирования проекта.
11. Протестировать проект.

# 1 План проекта

План проекта – это документ, содержащий подробную информацию о проекте: исполнителях, задачах и сроках. Документ является конечным результатом этапа планирования, утверждается до начала любых работ и становится самым главным и достоверным источником информации о грядущем проекте.

В нашем случае исполнителями являются следующие лица:

* Team Leader – Мельникова Екатерина;
* Coder 1 – Разумов Сергей;
* Coder 2 – Аникин Данил;
* Build Engineer – Разумов Сергей;
* Technical Writer – Мельникова Екатерина.

На рисунке 1 представлен перечень задач для выполнения и примерные сроки их реализации.

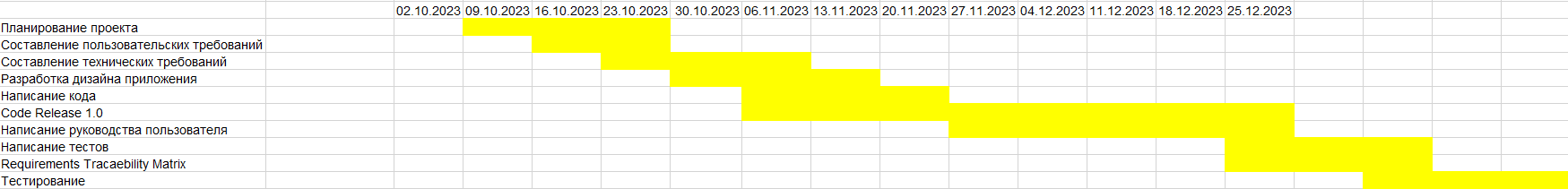


Рисунок 1 – План проекта

# 2 Регламент проведения инспекции

Верификация рабочих продуктов является неотъемлемой частью процесса по обеспечению их качества. Современной технологией программирования выработаны специальные стандарты, подходы и механизмы проведения верификаций рабочих продуктов, в формате так называемых инспекций.

Инспекция – это мероприятие по обеспечению качества рабочих продуктов проектов по разработке ПО и иной деятельности, которая проводится разработчиками, возможно – с участием представителей заказчика.

Концептуально инспекция имеет следующие цели:

* обнаружить ошибки в функциях, логике, содержании или реализации рабочих продуктов на ранних этапах их разработки и предотвратить их наследование;
* рационально донести замысел или реализацию продукта до всех заинтересованных лиц (через их участие);
* оптимизировать, оценить или улучшить рабочий продукт.

## Критерии отнесения к формальной/неформальной инспекции для различных типов рабочих продуктов

|  |  |  |
| --- | --- | --- |
|  | **Формальная инспекция в случае CRUD-операций над** | **Неформальная инспекция в случае CRUD-операций над** |
| **Требования** | >1 требованием | 1 требованием |
| **Документы дизайна** | >1 объектом дизайна внешнего вида готового продукта | 1 объектом дизайна внешнего вида готового продукта (изображение) |
| **Код** | >7 строчками | <8 строчками |
| **Тесты** | >3 тестами | < 4тестами |

## Перечень ролей участников инспекции и их обязанности

Автор (Author) – сотрудник, разработавший инспектируемый рабочий продукт, либо сделавший инспектируемые изменения в существующем рабочем продукте.

Инспектор (Inspector) – сотрудник, ответственный за эффективную проверку инспектируемого рабочего продукта.

## Этапы инспекции

## Планирование инспекции.

## При планировании инспекции коллективно выбирается дата, время, формат (очный или заочный) и платформа (при заочной инспекции) проведения инспекции.

## Подготовка к инспекции.

## Инспектор самостоятельно изучает предоставленный для инспекции рабочий продукт, используя накопленный опыт и стандарты.

## Собрание по инспекции.

## На собрании происходит обсуждение замечаний и рекомендаций инспектора по рабочему продукту. На собрании по инспекции обязательно присутствует инспектор и автор рабочего продукта, требующего инспекции. Присутствие остальных участников команды разработки по желанию.

## Завершение инспекции.

## Если рабочий продукт требует доработки, то автор фиксирует все замечания и рекомендации инспектора, разрабатывает план предстоящих работ и далее согласовывает его с инспектором. Если рабочий продукт не требует доработки, то инспектор подтверждает слияние рабочей ветки в «master» ветку.

## Перечень статусов и степени важности замечаний

1. Ошибка – проблема, которая найдена на той же фазе, на которой внесена. Допустимые значения степени серьёзности ошибки:

* Критическая (Critical)
* Средняя (Moderate)
* Мелкая, незначительная (Minor)
* Другие (Other)

1. Комментарий – это наблюдение, предложение, рекомендация или улучшение, предложенное для будущего выпуска рабочего продукта или вопрос, требующий разъяснения.

## Метрики, характеризующие эффективность инспекций

Стратегическая цель метрики – повысить качество разрабатываемого ПО.

В качестве метрики, характеризующей эффективность инспекции, была выбрана Inspection Preparation Rate (IPR):

IPR = (Количество инспекторов \* Размер продукта) / Общее время подготовки

Изучаемый объект метрики – подготовка к инспекции, измеряемый атрибут – производительность подготовки к инспекции.

Единица измерения – <страница, требование, LOC, тест> / час

# 3 Модель состояний задач

Каждая задача, являясь отражением делового процесса, проходит определенные состояния. Сначала идет создание задачи, потом идет выполнение работ по задаче, после выполнения задача завершается.

## Перечень состояний задач:

## New – новая подзадача.

## Analysis – в процессе анализа. В это состояние подзадачу переводит сотрудник после того, как начнёт её анализ.

## Forward – в данном случае имеет значение «переданный на дальнейшую разработку». В это состояние задача переводится CCB после анализа при назначении задачи на разработку конкретному сотруднику.

## Coding – кодирование. В это состояние задача переводится сотрудникомразработчиком, при начале работы по кодированию, связанному с задачей.

## Inspected – проинспектировано. В это состояние задача переводится сотрудником-разработчиком после завершения кодирования и инспектирования изменений рабочего продукта.

## Integrated – заинтегрировано. Переводится сотрудником, осуществляющим интеграцию изменений в основную ветку рабочего продукта после успешной интеграции этих изменений.

## Tested – протестировано. Переводится сотрудником, осуществляющим тестирование изменений в рабочий продукт (tester).

## Closed – закрыто. В это состояние задача переводится CCB по результатам отчёта о тестировании сделанных изменений.

## Правила создания новой задачи

Созданием новых задач могут заниматься все участники команды. Происходит это на начальном этапе, когда необходимо организовать структурированную работу над проектом и в процессе работы над проектом в случае присутствия в плане слишком объемных задач, требующих упрощения. Также, когда необходимо организовать баг-фиксинг или разработать новый функционал.

## Правила перехода задачи из состояния в состояние

Состояния задач всегда идут последовательно друг за другом, в некоторых случаях пункты могут опускаться или повторяться. Схема перехода из состояния в состояние показана на рисунке 2.

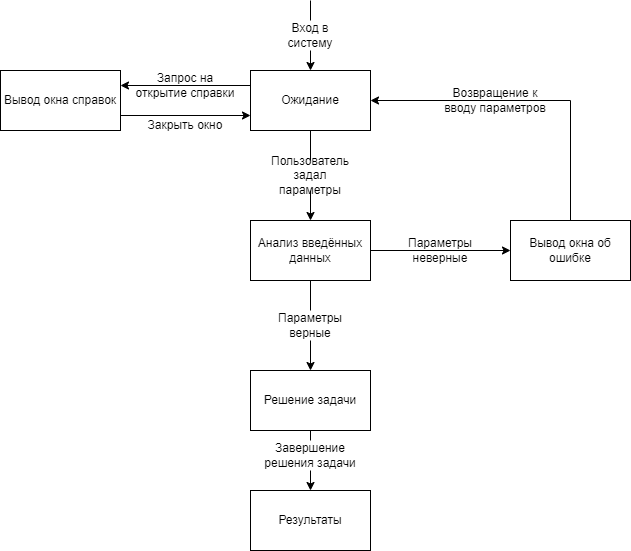


Рисунок 2 – Состояния задачи

# 4 Презентация проекта

На рисунке 3 представлена титульная страница презентации.

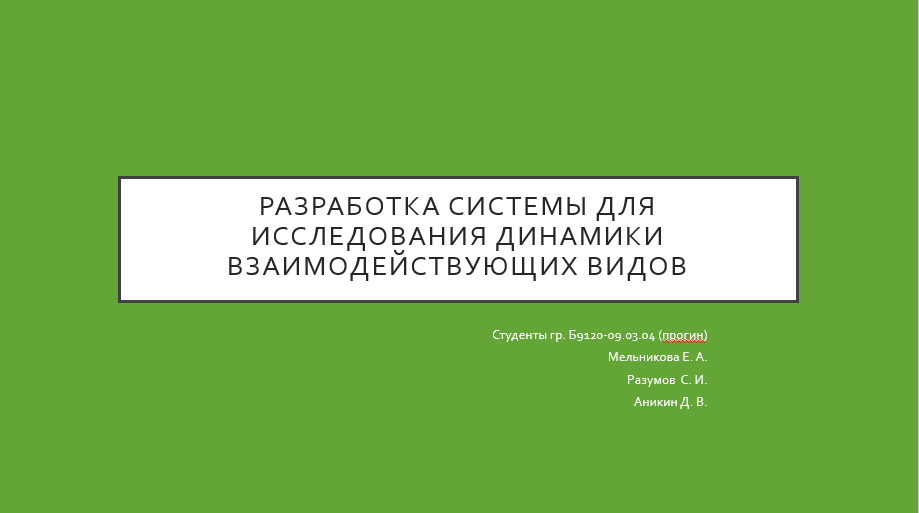


Рисунок 3 – Титульная страница

Проблемы, возникающие в предметной области разрабатываемого средства, представлены на рисунке 4.

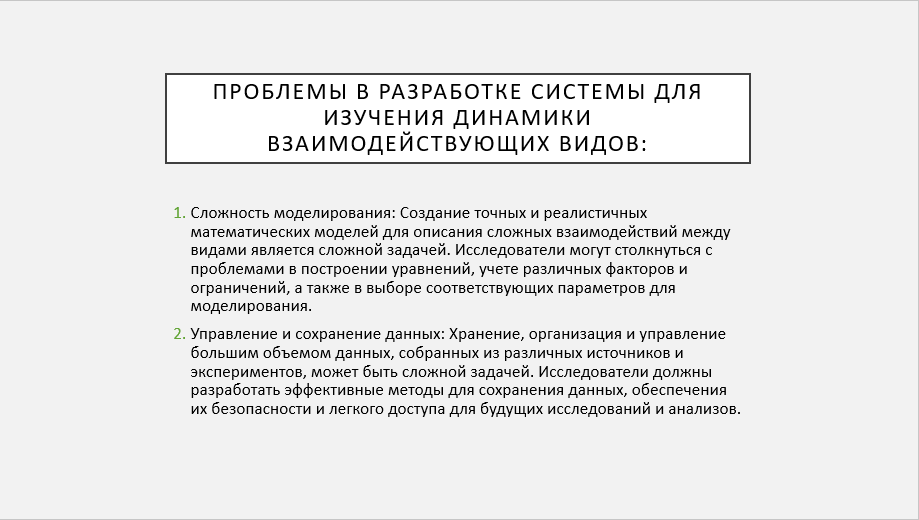


Рисунок 4 – Проблемы

На рисунке 5 демонстрируется, что можно получить и как, используя программное средство.

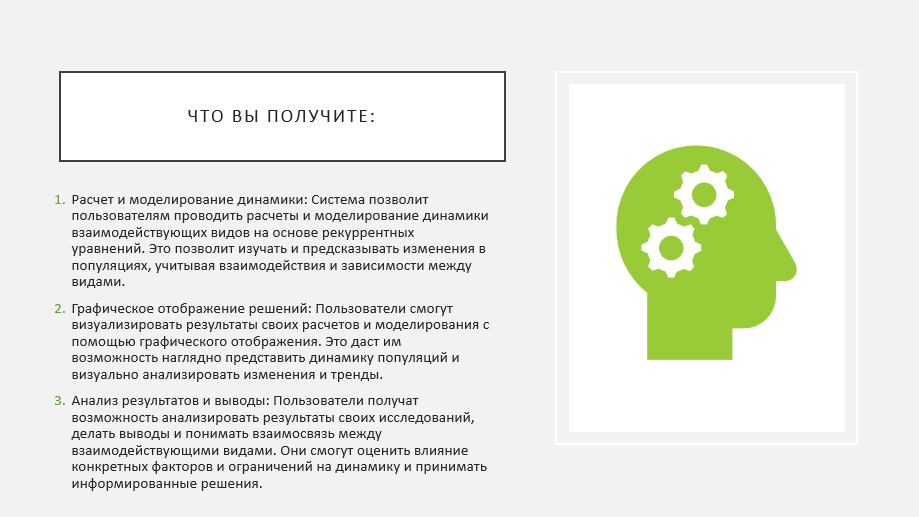


Рисунок 5 – Принцип работы

Внешний вид разрабатываемого средства представлен на рисунках 6, 7, 8.

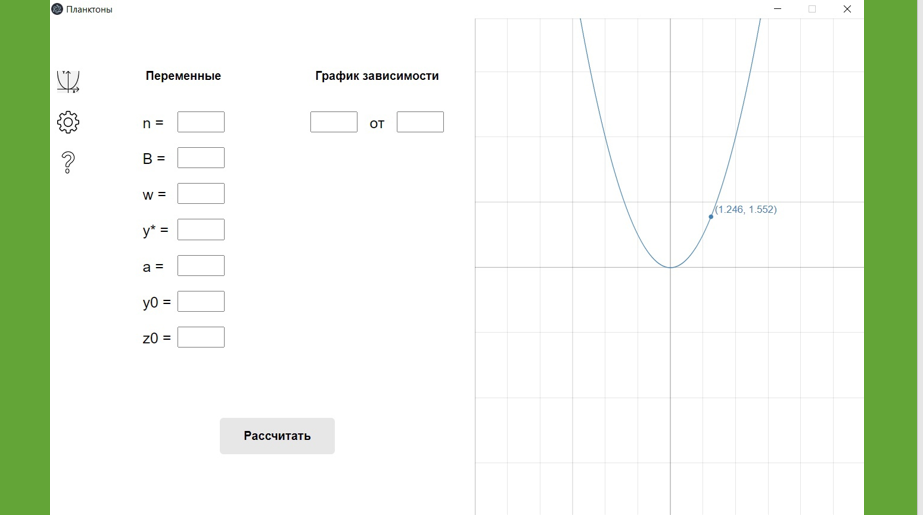


Рисунок 6 – Дизайн программного средства. Главное окно

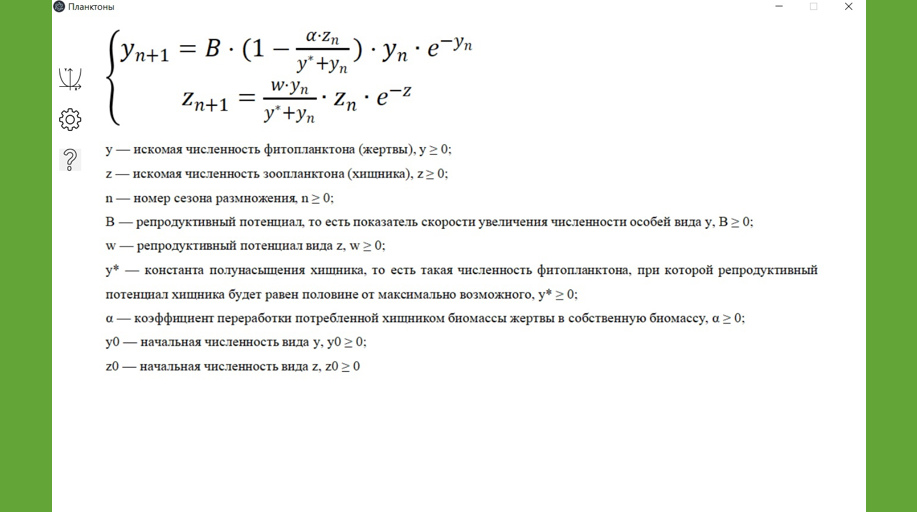


Рисунок 7 – Дизайн программного средства. Окно справки

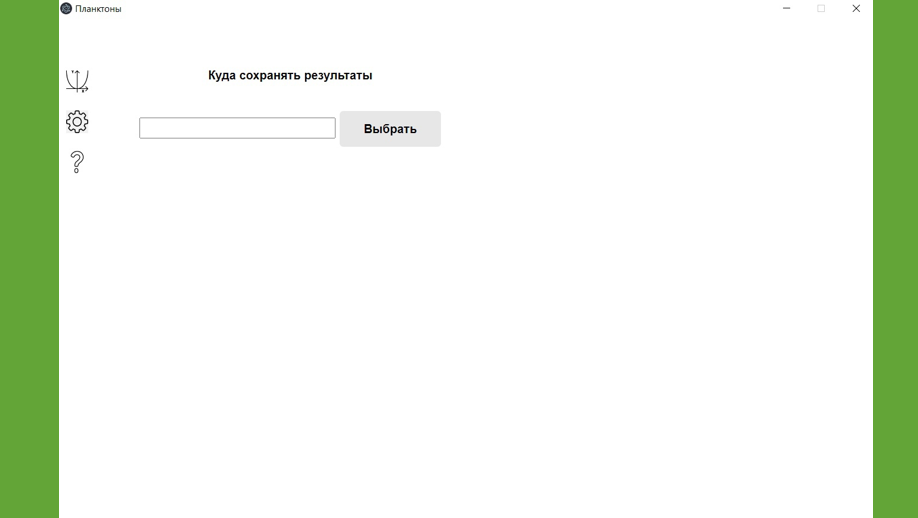


Рисунок 8 – Дизайн программного средства. Окно настроек

Актуальность разрабатываемого средства представлена на рисунке 9.

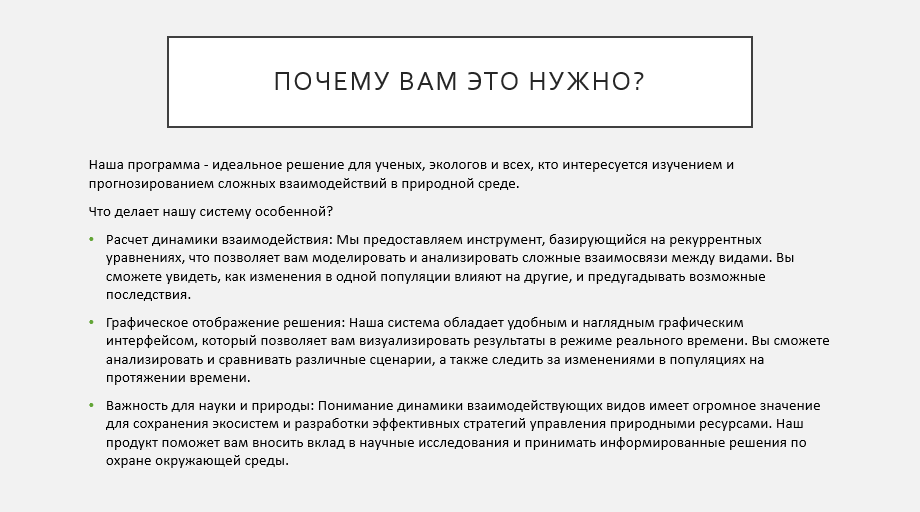


Рисунок 9 – Актуальность

# 5 Требований к проекту

Программный продукт «Компьютерная система для исследования динамики взаимодействующих видов» предназначен для расчета динамики взаимодействующих видов на основе рекуррентных уравнений и графического отображения решения.

Программный продукт «Компьютерная система для исследования динамики взаимодействующих видов» состоит из следующих подсистем:

1. FE-1 – Решение задачи.
2. FE-2 – Настройки.
3. FE-3 – Справка.

## Матрица требований к подсистемам

|  |  |
| --- | --- |
| **Матрица требований** | **Идентификаторы** |
| *Операции с решателем уравнений* | *FE-1 SE* |
| Изменение параметров | REQ-SE-1 |
| Решение | REQ-SE-2 |
| Сохранение результатов | REQ-SE-3 |
| *Операции с построителем графиков* | *FE-1 GF* |
| Изменение параметров | REQ-GF-1 |
| Построение | REQ-GF-2 |
| Сохранение результатов | REQ-GF-3 |
| *Операции с настройками* | *FE-2 S* |
| Просмотр | REQ-S-1 |
| Изменение | REQ-S-2 |
| *Операции со справкой* | *FE-3 IS* |
| Просмотр | REQ-IS-1 |

## Расшифровка идентификаторов

SE - Solving Equations (решение уравнений).

GF - Graph of a Function (график функции).

S – Settings (настройки).

IS - Information Sheet (информационная справка).

\*на данный момент покрыто ~77% требований

\*на данный сделано ~71% документации

# 6 Разработка архитектуры проекта

Архитектура программного обеспечения относится к фундаментальным структурам программной системы и дисциплине создания таких структур и систем. Каждая структура включает элементы программного обеспечения, отношения между ними, а также свойства как элементов, так и отношений.

Архитектура программной системы – это метафора, аналогичная архитектуре здания. Он функционирует как план для системы и проекта разработки, в котором излагаются задачи, которые должны быть выполнены командами разработчиков.

Одним из способов представления архитектуры является диаграмма потоков данных.

Диаграммы потоков данных (Data Flow Diagrams — DFD) представляют собой иерархию функциональных процессов, связанных потоками данных.

Цель такого представления — продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами.

На рисунке 9 представлена диаграмма потоков данных между подсистемами компьютерной системы для исследования взаимодействующих видов.

На вход интерфейс принимает числовые параметры, вводимые пользователем, необходимые для расчета динамики и последующего построения графика. Далее, полученные параметры передаются в подсистему для решения уравнений, вычисляется множество значений. Вычисленные значения передаются в подсистему визуализации, где на их основе строится график.

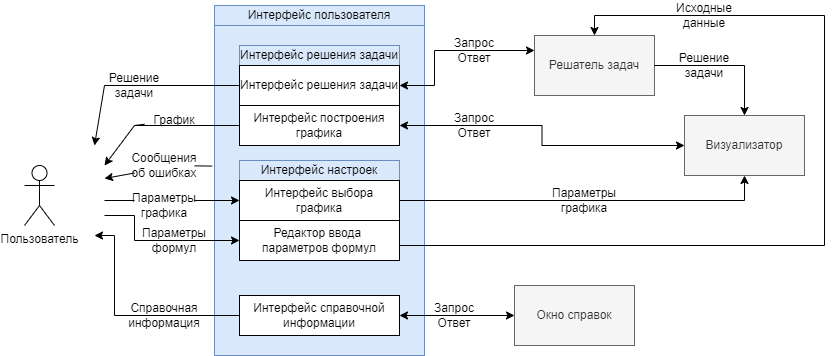


Рисунок 9 — Диаграмма потоков данных

# 7 Измерения проекта

Контроль над производственным процессом и его результатами является ключевым видом деятельности на современном предприятии, производящем программное обеспечение на заказ. В силу специфики такого продукта, как программное обеспечение, для оценки эффективности процесса и качества конечного продукта применяются особые методы.

Комплекс мероприятий, направленных на количественную оценку эффективности работы, называется программой измерений.

## Метрика эффективности процесса производства

1. **Faults Screening (FS)**

FS = (Общее количество ошибок – Число не найденных ошибок)/ Общее количество ошибок \* 100%;

Стратегическая цель метрики – повысить качество разрабатываемого ПО.

Изучаемый объект метрики – проект, измеряемый атрибут –эффективность обнаружения дефектов.

Единица измерения – %.

В прошлом семестре измерения следующие: (5-1)/5\*100 = 80%

На данный момент измерения следующие: (7-2)/7\*100 = 71%

1. **Inspection Fault Density (IFD)**

IFD = (Количество найденных ошибок / Размер рабочего продукта)

Стратегическая цель метрики – повысить качество разрабатываемого ПО.

Изучаемый объект метрики – инспекция, измеряемый атрибут – плотность найденных в ходе инспекции ошибок.

Количество найденных ошибок – ошибки, обнаруженные на этапе инспектирования.

Единица измерения – ошибка / <страница, требование**,** LOC, тест>.

В прошлом семестре измерения следующие: 0/9 = 0

На данный момент измерения следующие: 0/9 = 0

**Метрика качества продукта**

1. **In Process Faults (IPF)**

IPF = (Число обнаруженных ошибок до выпуска его релиза) / LOC

Стратегическая цель метрики – повысить качество разрабатываемого ПО.

Изучаемый объект метрики – продукт, измеряемый атрибут – плотность неполадок.

Число обнаруженных ошибок – количество ошибок, обнаруженных на этапе тестирования

Единица измерения – неполадка / LOC.

В прошлом семестре измерения следующие: (2+3)/200 = 0,025

На данный момент измерения следующие: (5+2)/212 = 0,033

# 8 Перечень задач проекта

# 9 Правила по кодированию

Для создания качественного кода на любом языке программирования, обладающего таким свойствами, как удобочитаемость (readability) и понятность (understandability), необходимо следовать хорошо определённым стандартам и руководствам. Особенно это актуально при коллективной разработке программ.

Любой стандарт кодирования призван определить набор правил, которые способствуют разработке более единообразного кода и минимизации числа общераспространенных ошибок в нем, не ущемляя при этом права разработчика на творчество.

## Требования

1. Отступы: использовать 2 пробела для отступа.

2. Именование: использовать camelCase для имен переменных и функций.

3. Объявление переменных: объявлять переменные, используя `const` или `let`, избегать использование `var`.

4. Длина строки: Длина строки не должна превышать 80 символов.

5. Точка с запятой: использовать точку с запятой в конце выражения.

6. Кавычки: Используйте одинарные кавычки для строк.

7. Фигурные скобки: Открывающая скобка должна находиться на одной строке с выражением или объявлением функции.

8. Пробелы: ставить операторы между пробелами, после запятой ставить пробел, но не перед запятой.

9. Комментарии: использовать `//` для однострочных комментариев и `/\* \*/` для многострочных комментариев.

10. Модули: использовать модули ES6 (`import` and `export`) для организации и управления зависимостями кода.

# 10 Разработка плана тестирования проекта

В данном разделе описаны тесты для различных ситуаций, покрывающие все требования.

**Тест TEST\_SE\_01**

Тестируемые требования: REQ-SE-1, REQ-SE-2

В этом тесте проверяется способность программы обрабатывать верно введенные параметры для уравнения. На вход подаются следующие значения для параметров:

n = 15;

B = 10.3;

w = 10.0;

yg = 0.3;

a = 0.5;

y0 = 1.2;

z0 = 1.3;

Далее пользователь нажимает кнопку «Рассчитать», после чего программа решает задачу с заданными параметрами.

**Тест TEST\_SE\_02**

Тестируемые требования: REQ-SE-1

В этом тесте проверяется способность программы обрабатывать неверно введенные параметры для уравнения. На вход подаются следующие значения для параметров:

n = -15;

B = 10.3;

w = 10.0;

yg = 0.3;

a = 0.5;

y0 = 1.2;

z0 = 1.3;

Далее пользователь нажимает кнопку «Рассчитать», после чего программа выдает сообщение об ошибке.

**Тест TEST\_SE\_03**

Тестируемые требования: REQ-SE-2

В этом тесте проверяется способность программы решать задачу. На вход подаются следующие значения для параметров:

n = 15;

B = 5.4;

w = 0.2;

yg = 0.5;

a = 0.5;

y0 = 0.1;

z0 = 0.1;

Далее пользователь нажимает кнопку «Рассчитать», после чего программа решает задачу и выводит результаты расчётов на экран.

**Тест TEST\_SE\_04**

Тестируемые требования: REQ-SE-3

В этом тесте проверяется способность программы сохранять результаты решения уравнения в соответствии с заданными параметрами сохранения. На вход подаются значение параметра пути для сохранения результатов и нажатие пользователем кнопки сохранения результата. Программа должна сохранить решение уравнения в .txt файл по указанному пути.

**Тест TEST\_GF\_01**

Тестируемые требования: REQ-GF-1, REQ-GF-2

В этом тесте проверяется способность программы работать с разными параметрами. Пользователь выбирает параметр X для построения графика, после чего нажимает кнопку «Рассчитать», и на экране отображается график по выбранному пользователем параметру.

**Тест TEST\_GF\_02**

Тестируемые требования: REQ-GF-1, REQ-GF-2

В этом тесте проверяется способность программы работать с разными параметрами. Пользователь выбирает параметр Y для построения графика, после чего нажимает кнопку «Рассчитать», и на экране отображается график по выбранному пользователем параметру.

**Тест TEST\_GF\_03**

Тестируемые требования: REQ-GF-3

В этом тесте проверяется способность программы сохранять результаты построения графика соответствии с заданными параметрами сохранения. На вход подаются значение параметра пути для сохранения результатов и нажатие пользователем кнопки сохранения результата. Программа должна сохранить построенный график в .png файл по указанному пути.

**Тест TEST\_S\_01**

Тестируемые требования: REQ-S-1

В этом тесте проверяется способность программы показывать пользователю окно настроек. На вход подаются нажатие пользователем иконки настроек. Программа должна перейти в окно настроек.

**Тест TEST\_S\_02**

Тестируемые требования: REQ-S-2

В этом тесте проверяется способность программы предоставлять пользователю возможность изменения настроек. На вход подается значение пути для сохранения результатов. При изменении настройки пути для сохранения результатов последующее сохранение результатов производится в указанный путь.

**Тест TEST\_IS\_01**

Тестируемые требования: REQ-IS-1

В этом тесте проверяется способность программы предоставлять пользователю возможность просмотра окна справок. На вход подается нажатие пользователем иконки окна справок. Программа должна перейти в окно справок.

## Матрица покрытия тестами требований

Матрица соответствия требований используется QA-инженерами для валидации покрытия требований по продукту тестами.

Цель «Traceability Matrix» состоит в том, чтобы выяснить:

* какие требования «покрыты» тестами, а какие нет;
* избыточность тестов (одно функциональное требование покрыто большим количеством тестов).

В соответствии с написанными требованиями и тестами на рисунке 12 представлена матрица покрытия тестами требований.

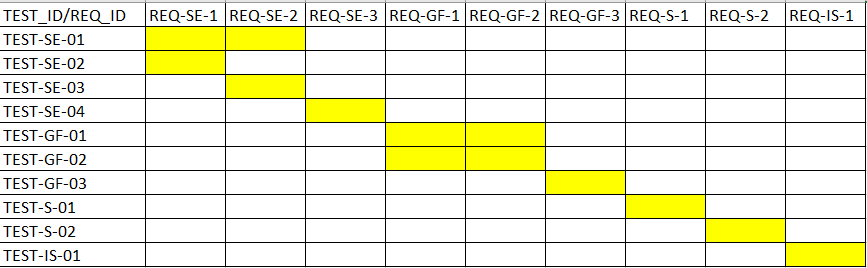


Рисунок 12 – Матрица покрытия тестами требований

# 11 Тестирование проекта

# 

# Заключение

В рамках курсовой работы было разработано программное средство «Программная система для автоматического создания выкроек» с использованием подходов коллективной промышленной разработки, для чего были решены следующие поставленные задачи:

1. Разработан план проекта.
2. Разработан регламент проведения инспекции.
3. Разработана модель состояний задач.
4. Разработана презентация проекта.
5. Разработаны требования к проекту.
6. Разработана архитектура проекта.
7. Разработаны измерения проекта.
8. Разработан перечень задач проекта.
9. Разработаны рекомендации по кодированию.
10. Разработан план тестирования проекта.
11. Протестирован проект.

Таким образом, цель данной курсовой работы была достигнута.

По окончанию всех работ посчитаем метрики, описанные в 7 главе. Эффективность процесса разработки программного средства – 7 дней работы. Качество продукта выразилось в 28% плотности неполадок.

# Список литературы

1. Гриняк В.М. Лекции по дисциплине «Технологии коллективной промышленной разработки информационных систем». Электронный вариант.