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**

**РЕФЕРАТ**

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

Выполнили студенты гр. Б9121-09.03.04

Булах А.А.

Ермак В.М.

Лазарев А.Е.

Лебединский И.В.

Пасько У.Н.  
Руководитель:

ассистент кафедры ПИиИИ Логачев Е.М.

« » 2025 г.

(подпись)

Защищён с оценкой\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (И.О. Фамилия)

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г.

**Оглавление**

[Декомпозиция задач. План проекта. 3](#_Toc188888071)

[Разработка регламента инспекции рабочих продуктов 6](#_Toc188888072)

[Разработка модели состояний задач 9](#_Toc188888073)

[Разработка требований к проекту 12](#_Toc188888074)

[Общая характеристика 12](#_Toc188888075)

[Разработка архитектуры проекта 16](#_Toc188888076)

[Архитектурно-контекстная диаграмма системы 16](#_Toc188888077)

[Архитектурно контекстная диаграмма подсистемы «Пользовательский интерфейс» 16](#_Toc188888078)

[Диаграмма потока данных проекта 17](#_Toc188888079)

[Система метрик для оценки эффективности процесса разработки 17](#_Toc188888080)

[Время цикла 17](#_Toc188888081)

[Эффективность потока 17](#_Toc188888082)

[Список литературы 19](#_Toc188888083)

# Декомпозиция задач. План проекта.

1. Разработка плана проекта

1.1. Составление календарного плана:

- Определение основных этапов и сроков.

- Учет зависимостей между задачами.

1.2. Организация работы команды:

- Распределение ролей (тимлид, разработчик, тестировщик и др.).

- Разработка документа с ролями и обязанностями.

2. Разработка регламента инспекций

2.1. Определение критериев инспекций:

- Критерии проверки кода, требований, документации.

2.2. Создание ролей и этапов инспекций:

- Формализация ролей (модератор, автор, рецензент).

- Определение этапов (подготовка, рецензирование, обсуждение).

3. Разработка модели состояний задач

3.1. Определение возможных состояний задач:

- Backlog, To Do, In Progress, Review, Done, Blocked.

3.2. Разработка регламента переходов:

- Правила перехода задач между состояниями.

4. Разработка требований к проекту

4.1. Сбор требований:

- Определение функциональных и нефункциональных требований.

4.2. Разделение требований по подсистемам:

- Пользовательский интерфейс, модуль расчётов, визуализация.

5. Разработка архитектуры проекта

5.1. Создание архитектурных диаграмм:

- Диаграммы потоков данных, классов, вызовов.

5.2. Определение взаимодействий подсистем:

- Взаимодействие UI, расчётов и визуализации.

6. Разработка программы измерений проекта

6.1. Определение метрик эффективности разработки:

- Время цикла (Cycle Time), эффективность потока (Flow Efficiency).

6.2. Разработка метрик качества:

- Надёжность (Reliability) и точность расчётов.

6.3. Публикация программы измерений:

- Размещение документа с метриками в GitHub.

7. Реализация проекта

7.1. Разработка пользовательского интерфейса:

- Реализация функций редактирования конструкций и визуализации.

7.2. Разработка модуля расчётов:

- Расчёт реакций опор и проверка данных.

7.3. Реализация подсистемы визуализации:

- Визуализация расчётов, экспорт изображений.

7.4. Интеграция подсистем:

- Обеспечение взаимодействия UI, расчётов и визуализации.

8. Выпуск альфа-версии

8.1. Сборка альфа-версии:

- Интеграция основных функций в рабочую версию.

8.2. Размещение альфа-версии в репозитории:

- Публикация версии для начального тестирования.

9. Составление расписания тестирования

9.1. Разработка расписания тестирования:

- Планирование модульного, интеграционного и пользовательского тестирования.

10. Тестирование

10.1. Модульное тестирование:

- Проверка подсистем на соответствие требованиям.

10.2. Интеграционное тестирование:

- Тестирование взаимодействия подсистем.

10.3. Пользовательское тестирование:

- Проверка удобства работы системы студентами и преподавателями.

10.4. Исправление ошибок и повторное тестирование:

- Устранение замечаний и повторная проверка.

Проектная команда:

1. Тимлид – Ермак В.М.
2. Программист – Лебединский И.В.
3. Тестировщик – Лазарев А.Е.
4. Проектировщик программного продукта – Булах А.А.
5. Проектный менеджер – Пасько У.Н.

Сроки выполнения:

1. Составление плана разработки – 02.10.24 – 23.10.24
2. Разработка регламента инспекций – 23.10.24 - 06.11.24
3. Разработка модели состояний задач– 30.10.24 - 13.11.24
4. Разработка требований к проекту – 06.11.24 - 20.11.24
5. Разработка архитектуры проекта – 20.11.24 - 11.12.24
6. Разработка программы измерений проекта – 27.11.24 - 25.12.24
7. Реализация проекта– 11.12.24 - 15.01.25
8. Выпуск альфа-версии – 25.12.24 - 15.01.25
9. Составление расписания тестирования – 15.01.25 - 22.01.25
10. Тестирование – 22.01.25 - 28.02.25

Изображение выглядит как текст, линия, Шрифт, График

Автоматически созданное описание

Рисунок 1 – сроки выполнения задач проекта

# Разработка регламента инспекции рабочих продуктов

Регламент проведения инспекции рабочих продуктов для студенческих проектов, выполняемых по дисциплине «Технологии коллективной промышленной разработки информационных систем».

Критерии отнесения к формальной / неформальной инспекции

1. Формальная инспекция проводится для:
   1. Документов требований с объемом более 20 страниц.
   2. Документов дизайна, если они влияют на критические функции системы.
   3. Кода с высоким уровнем риска (например, связанного с вычислительными модулями, оптимизацией или безопасностью).
   4. Тестов, предназначенных для проверки критических путей.
2. Неформальная инспекция допускается для:
3. Промежуточных версий требований, дизайна и тестов.
4. Кода, который не содержит критических функций или является экспериментальным.

Роли участников инспекции и их обязанности

1. Автор (создатель рабочего продукта):
   1. Подготавливает материалы к инспекции.
   2. Объясняет сложные моменты участникам.
2. Модератор:
   1. Организует инспекцию, контролирует соблюдение процесса и сроков.
   2. Составляет итоговый отчет.
3. Рецензент:
   1. Анализирует рабочий продукт, выявляет недостатки.
4. Записывающий (секретарь):
   1. Фиксирует замечания и комментарии во время обсуждения.
5. Эксперт:
   1. Оценивает узкоспециализированные аспекты продукта.
6. Рекомендации по числу участников:
7. Для продуктов объемом до 10 страниц – 3-4 участника.
8. Для продуктов от 10 до 50 страниц – 4-6 участников.
9. Для более крупных продуктов – от 6 участников.

Этапы инспекции

1. Подготовка: сбор материалов, назначение участников, определение сроков.
2. Рецензирование: участники анализируют продукт индивидуально.
3. Обсуждение: проводится собрание для обсуждения замечаний.
4. Заключение: фиксация замечаний, определение статусов.
5. Верификация: тестирование исправлений выявленных ошибок.

Порядок организации

1. Автор выгружает материалы в корпоративное хранилище.
2. Модератор организует встречи, отправляет ссылки на материалы, и приглашения участникам.
3. Для критических продуктов используется инструмент отслеживания задач.

Порядок подготовки к инспекции

1. Материалы предоставляются за 3 рабочих дня до инспекции.
2. Время обсуждения согласовывается с учётом доступности участников.
3. Участники получают четкие инструкции по анализу продукта.

Порядок проведения инспекции

1. Автор: представляет продукт, отвечает на вопросы.
2. Рецензенты и эксперты: озвучивают замечания.
3. Модератор: фиксирует итоги и управляет дискуссией.
4. Время проведения: до 1 часа на каждые 10 страниц продукта.

Перечень статусов и степени важности замечаний

1. Статусы:
   1. "К исправлению" (несоответствие или неполнота выполнения задачи).
   2. "Информационное" (уточнение по реализации).
   3. "Отклонено" (критическое несоответствие).
2. Степени важности:
   1. Высокая (критическое влияние на проект).
      1. Компоненты, участвующие в основном функционале
      2. Вызывающие критические ошибки
   2. Средняя (может повлиять на сроки или качество).
      1. Дополнительный функционал
      2. Удобство интерфейса и широта его функционирования
      3. Советы и подсказки для пользователя
   3. Низкая (косметические замечания).
      1. Оптимизация отдельных частей программы
      2. Интуитивная понятность программы
      3. Интерактивное обучение использования программы

Порядок верификации учёта замечаний

1. Исправления выполняются автором в течение 5 рабочих дней.
2. Модератор проверяет устранение замечаний и сообщает участникам.

Метрики эффективности инспекций

1. Процент устраненных замечаний (≥ 90%).
2. Среднее время исправления замечаний.
3. Соотношение замечаний по важности (высокие ≤ 10% от общего числа).
4. Количество выявленных дефектов на единицу объёма продукта.

# Разработка модели состояний задач

Множество состояний задач, принятых на итоговом проекте. Разработка регламента создания задач и перевода их из состояния в состояние.

1. Перечень возможных состояний задачи и их интерпретация
   1. Backlog
      1. Задача добавлена в проект, но ещё не утверждена к работе.
      2. Интерпретация: задача требует уточнения требований, приоритезации или проверки.
   2. To Do
      1. Задача утверждена для выполнения и ожидает начала работы.
      2. Интерпретация: приоритет задачи определён, она включена в ближайший спринт.
   3. In Progress
      1. Над задачей ведётся активная работа.
      2. Интерпретация: задача выполняется исполнителем согласно требованиям.
   4. Review
      1. Работа по задаче завершена и передана на проверку.
      2. Интерпретация: ожидается подтверждение выполнения требований или исправление замечаний.
   5. Done
      1. Задача полностью выполнена, проверена и утверждена.
      2. Интерпретация: результат интегрирован в проект.
   6. Blocked
      1. Задача приостановлена из-за внешних факторов (зависимость от другой задачи, отсутствие ресурсов и т.д.).
      2. Интерпретация: задача требует разрешения проблемных вопросов.
   7. Closed
      1. Задача закрыта (выполнена или отклонена).
      2. Интерпретация: дальнейшие действия по задаче не требуются.
2. Правила создания новой задачи
   1. Кто создает:

2.1.1. Руководитель проекта – для постановки целей и ключевых задач.

2.1.2. Разработчики – при выявлении технических, уточняющих или вспомогательных задач.

2.1.3. Участники инспекций – при обнаружении замечаний в ходе проверки.

* 1. Когда создается:
     1. При выявлении новой функциональной потребности.
     2. В случае обнаружения дефекта или проблемы.
     3. При необходимости уточнить существующую задачу.
  2. Требования к задаче:
     1. Чёткое название, отражающее суть задачи.
     2. Описание с критериями выполнения (Acceptance Criteria).
     3. Метки (Labels), указывающие тип задачи (bug, feature, improvement, documentation).
     4. Назначение исполнителя (Assignee).
     5. Привязка к вехе (Milestone).

1. Правила перехода задачи из состояния в состояние
   1. Backlog ↔ To Do:
      1. Требования уточнены, задача согласована руководителем проекта.
      2. Задача приоритезированы.
   2. To Do → In Progress:
      1. Исполнитель готов приступить к задаче.
      2. Все блокирующие зависимости сняты.
   3. In Progress → Review:
      1. Работа завершена, выполнены все Acceptance Criteria.
      2. Результат доступен для проверки.
   4. Review → Done:
      1. Задача проверена, все замечания устранены.
      2. Финальная версия подтверждена руководителем или рецензентом.
   5. Review ↔ In Progress:
      1. Замечания не устранены, требуется доработка.
   6. Blocked → To Do:
      1. Проблемы, блокирующие выполнение задачи, решены.

Критерии перевода в состояние:

Каждое состояние требует выполнения описанных выше условий.

Переходы допускаются только между логически связанными состояниями.

Возможные переходы:

1. Backlog ↔ To Do
2. To Do → In Progress
3. In Progress → Review
4. Review ↔ In Progress
5. Review → Done
6. Any → Blocked
7. Blocked → To Do

Результаты переходов:

1. При каждом переходе обновляется статус задачи.
2. Исполнители уведомляются об изменениях через комментарии в GitHub Issues.

# Разработка требований к проекту

Продукт позволяет рассчитывать реакции опор конструкций (задача C1) через визуальный интерфейс и модуль автоматических расчётов.

Программный продукт предназначен для:

1. Студентов, изучающих теоретическую механику.
2. Преподавателей, проверяющих задания студентов.
3. Инженеров, которым необходимы базовые расчеты опор.

Приложение работает на настольных системах с операционной системой Windows. Основной язык интерфейса: русский.

Определения и сокращения

C1: Задача на определение реакций опор твердого тела.

## Общая характеристика

**Функциональность системы**

1. Пользовательский интерфейс позволяет пользователю:
   1. Задавать конструкцию (опоры, силы, стержни) визуально.
   2. Запускать расчёты.
2. Модуль расчетов:
   1. Рассчитывает реакции опор на основе введенных данных.
   2. Выводит числовые результаты для задачи C1.
3. Подсистема визуализации:
   1. Отображает схему конструкции и направления сил.
   2. Позволяет взаимодействовать с элементами схемы (перемещать, редактировать).

**Ограничения**

1. Программа предназначена для работы на настольной системе Windows 11.
2. Ввод данных ограничивается двумя-тремя телами и фиксированным количеством сил (максимум 3).
3. Визуализация ограничена плоскими и пространственными схемами.

**Предположения и зависимости**

1. Пользователь должен обладать базовыми знаниями в теоретической механике.
2. Для работы требуется установленный Python (для версий без компиляции).

### Требования к подсистеме «Пользовательский интерфейс»

Функциональные требования:

1. REQ\_UI\_001

Приложение должно позволять пользователю создавать и редактировать конструкцию (добавлять балки, узлы, силы, опоры).

1. REQ\_UI\_002

Интерфейс должен поддерживать ввод характеристик для каждого элемента конструкции (длина балки, величина и направление сил, тип опор и т.д.).

1. REQ\_UI\_003

Необходимо реализовать простую и понятную навигацию, чтобы пользователь мог легко переключаться между режимами работы (создание конструкции и расчёт).

1. REQ\_UI\_004

Пользователь может редактировать элементы конструкции (перемещать узлы, изменять величины сил).

Нефункциональные требования:

1. REQ\_UI\_005

Интерфейс должен быть интуитивно понятным и не требовать длительного обучения.

1. REQ\_UI\_006

Программа должна реагировать на действия пользователя без значительных задержек (менее 1 секунды для большинства операций).

1. REQ\_UI\_007

Интерфейс должен быть адаптивным для использования на экранах разного размера (от ноутбуков до больших мониторов).

### Требования к подсистеме «Модуль расчетов»

Функциональные требования:

1. REQ\_CALC\_001

Модуль должен рассчитывать реакции опор, используя уравнения равновесия.

1. REQ\_CALC\_002

Модуль должен поддерживать расчёт сил.

1. REQ\_CALC\_003

Должен быть реализован механизм обработки различных типов нагрузок (точечные силы, распределённые нагрузки).

1. REQ\_CALC\_004

Должна быть возможность расчёта моментов инерции и других характеристик балки.

1. REQ\_CALC\_005

Модуль должен проверять введённые данные на корректность.

1. REQ\_CALC\_006

При ошибках расчета программа должна выдавать пользователю сообщение с указанием причины.

Нефункциональные требования:

1. REQ\_CALC\_007

Алгоритм расчёта должен быть оптимизирован для конструкций с большим количеством узлов и сил (время выполнения не должно превышать 1 секунды для систем до 100 элементов).

1. REQ\_CALC\_008

Программа должна рассчитывать реакции опор с точностью до двух знаков после запятой.

1. REQ\_CALC\_009

Модуль должен быть гибким и легко масштабируемым для добавления новых функций.

### Требования к подсистеме «Подсистема визуализации»

Функциональные требования:

1. REQ\_VIS\_001

Программа должна визуализировать конструкцию, включая узлы, балки, силы и опоры.

1. REQ\_VIS\_002

Должна быть возможность масштабировать и перемещать изображение конструкции.

1. REQ\_VIS\_003

Сила и её направление должны быть визуализированы.

1. REQ\_VIS\_004

Результаты расчётов (реакции опор, распределение нагрузок) должны отображаться в диалоговом окне в текстовом виде.

1. REQ\_VIS\_005

Подсистема должна поддерживать экспорт изображения конструкции в виде файла (PNG, JPEG, SVG).

Нефункциональные требования:

1. REQ\_VIS\_006

Отображение должно быть плавным даже для сложных конструкций (до 100 элементов).

1. REQ\_VIS\_007

Визуализация должна быть чёткой и разборчивой, с использованием цветовой схемы для различения типов элементов (опоры, силы, реакции).

1. REQ\_VIS\_008

Визуализация должна автоматически обновляться при изменении параметров пользователем.

# Разработка архитектуры проекта

## Архитектурно-контекстная диаграмма системы

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Автоматически созданное описание

Рисунок 2 – Архитектурно контекстная диаграмма подсистемы  
«Пользовательский интерфейс»

Программный продукт состоит из следующих подсистем:

1. Пользовательский интерфейс
2. Модуль расчетов
3. Подсистема визуализации

## Архитектурно контекстная диаграмма подсистемы «Пользовательский интерфейс»

Подсистема "Пользовательский интерфейс" взаимодействует с пользователем, модулем расчётов и подсистемой визуализации. Основной поток данных:

1. Пользователь вводит данные через интерфейс.
2. Данные передаются в модуль расчетов для обработки.
3. Результаты расчетов передаются в подсистему визуализации.
4. Визуализация отображает результаты пользователю.

## Диаграмма потока данных проекта

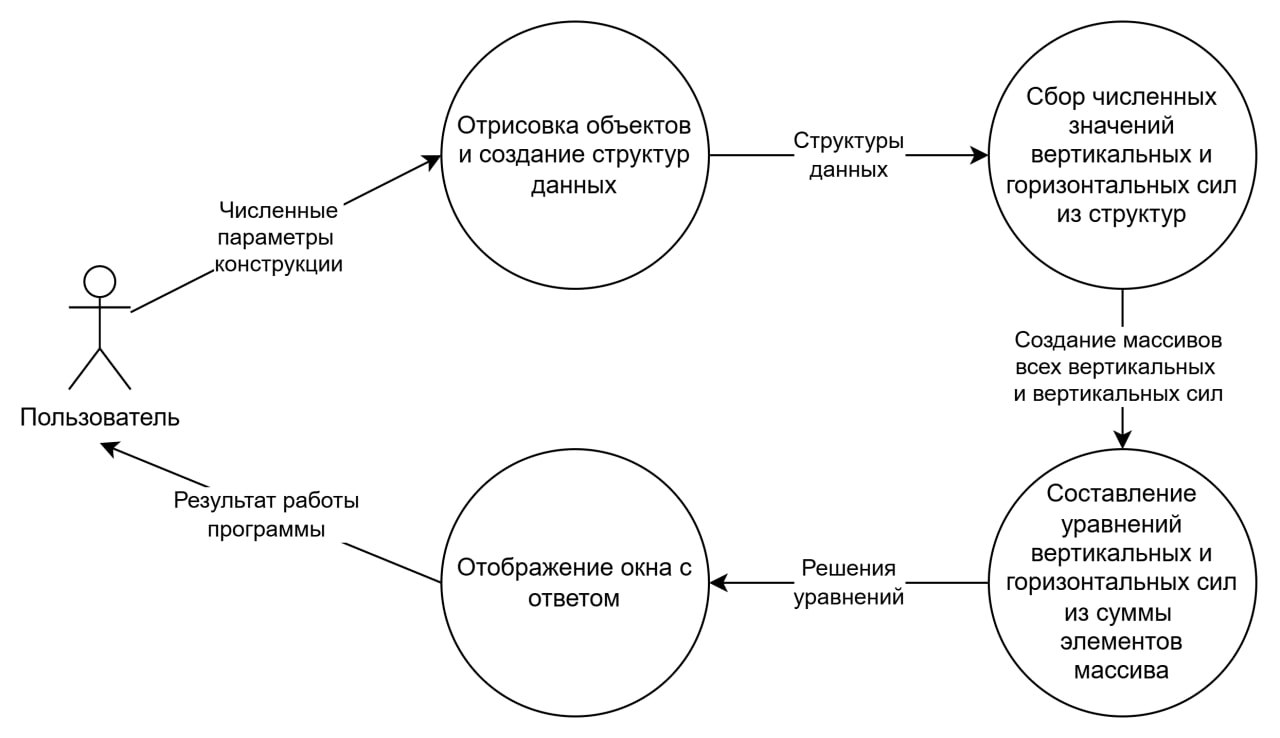


Рисунок 3 – Диаграмма потока данных

# Система метрик для оценки эффективности процесса разработки

## Время цикла

Описание: эта метрика измеряет время, которое задача проводит в активной разработке, от момента начала работы до завершения. Она не учитывает время ожидания до начала работы над задачей.

Формула:

Дата перехода задачи в статус «Завершено»‎‎ - дата перехода в статус «В работе»

Цель: позволяет оценить скорость работы команды и эффективность процесса разработки. Уменьшение времени цикла обычно указывает на повышение производительности команды

## Эффективность потока

Описание: это метрика, которая показывает, какая часть времени задачи действительно затрачивается на активную работу, в сравнении с общим временем, проведенным в системе. Измеряется как процент активной работы от общего времени Lead Time.

Формула:

Flow Efficiency = (Время активной работы / Общее Lead Time)\*100%

Цель: выявление "мёртвого времени" и узких мест в процессе разработки. Понимание этой метрики помогает оптимизировать процессы и улучшить планирование качества программного продукта.

Эта метрика качества оценивает надежность (Reliability) ПС.

Описание: надежность измеряет способность системы функционировать без сбоев в течение определенного времени или при выполнении определенных задач.

# Список литературы

1. Учебно-методический комплекс дисциплины «Технология коллектив­ной разработки информационных систем» [Электронный ресурс]. URL: <https://bb.dvfu.ru/webapps/blackboard/content/listContent.jsp?course_id=_5025_1&content_id=_172365_1> (дата обращения: 21.01.2025)