

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 ОБУЧАЮЩЕЙ СИСТЕМЫ ПО МАТЕМАТИЧЕСКИМ ДИСЦИПЛИНАМ**

КУРСОВАЯ РАБОТА

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  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Борщевский И.О.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Корогод Д.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Майоров С.В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Пяткин А.В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Федоров А.М.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Широкова С.С.

Руководитель:  
старший преподаватель  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Иваненко Ю.С.

г. Владивосток  
2023

**Оглавление**

[Введение 3](#_Toc156166785)

[1. Разработка плана проекта 4](#_Toc156166786)

[2. Разработка регламента проведения инспекции 6](#_Toc156166787)

[4. Разработка презентации проекта 10](#_Toc156166788)

[5. Разработка требований к проекту 12](#_Toc156166789)

[7. Разработка программы измерений проекта 17](#_Toc156166790)

[8. Разработка перечня задач проекта 20](#_Toc156166791)

[9. Разработка рекомендаций по кодированию 21](#_Toc156166792)

[10. Разработка плана тестирования проекта 23](#_Toc156166793)

[11. Тестирование проекта 23](#_Toc156166794)

[Заключение 24](#_Toc156166795)

[Список литературы 25](#_Toc156166796)

**Введение**

Промышленная разработка информационных систем – это сложный и продолжительный комплекс работ, включающий в себя множество этапов. Для выполнения данного комплекса работ необходимо иметь в распоряжении команду специалистов разных направленностей, налаженную коммуникацию между ними, а также правильно разделить обязанности членов команды в зависимости от их профиля.

Из написанного выше можно сделать вывод, что для успешного выполнения работы, которое включает в себя соответствие конечного продукта исходным требованиям, необходимо использовать различные технологии коллективной разработки.

В данной курсовой работе рассматривается задача коллективной разработки программного средства «Обучающая система по математическим дисциплинам» и составление технической документации к данному продукту.

Тогда целью курсовой работы является разработка программного средства «Обучающая система по математическим дисциплинам» с использованием методов коллективной промышленной разработки.

Для достижения поставленной цели необходимо решить 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 программу измерений проекта.
8. Разработать перечь задач проекта.
9. Разработать рекомендации по кодированию.
10. Разработать план тестирования проекта.
11. Провести тестирование проекта.
12. **Разработка плана проекта**

План проекта содержит в себе информацию об участниках команды, их ролях, задачах проекта и сроках их выполнения. Документ регламентируется в конце этапа планирования до начала любых работ.

Ниже перечислены участники команды и их роли:

1. Team Leader – Пяткин Алексей
2. Main Coder – Корогод Дмитрий
3. Coder 1 – Борщевский Иван
4. Coder 2 – Федоров Артур
5. Technical Writer – Широкова Софья
6. Build Engineer – Майоров Сергей

На рисунке 1 представлен план задач проекта с примерными сроками их реализации.

****

Рисунок 1 – План проекта

1. **Разработка регламента 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 (peer reviews).

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 формальности инспекции**

Критерии отнесения к формальной инспекции представлен в таблице 1.

Таблица 1 – Критерии отнесения к формальной инспекции

|  |  |
| --- | --- |
| **Рабочий продукт** | **Размер продукта или изменение в нём, не менее** |
| Документы по дизайну | 3 страницы или 6% рабочего продукта |
| Не комментируемый код | 15 NCLOC |
| Тесты (Test Cases) | 3 страницы или 6% рабочего продукта |

Неформальная инспекция проводится в случаях: не удовлетворяющих условиям таблицы 1.

**Перечень ролей и их обязанности**

Участники могут иметь следующие роли:

1. Автор — участник, внёсший изменения в рабочий продукт. Загружает изменения в систему контроля версий и инициирует инспекцию;
2. Председатель — тимлид (Team Leader), контролирующий процесс инспекции. Назначает инспектора и одобряет внесение ключевых изменений в основную версию продукта;
3. Инспектор — участник, проводящий проверку внесённых изменений. Оставляет замечания и выносит вердикт о внесении изменений в продукт.

В инспекции в обязательном порядке присутствуют два участника, имеющие роли автора и инспектора соответственно. При внесении изменений в дизайн рабочего продукта, необходимо присутствие председателя, который может также выполнять роль инспектора.

**Этапы инспекции**

1. Инспекция – запрос автора на внесение изменений в продукт и назначение инспектора.
2. Подготовка и проведение – анализ изменений и вынесение замечаний, если необходимо.
3. Завершение – вынесение вердикта о внесении изменений в продукт.

**Порядок организации инспекции**

Работа над продуктом ведётся в системе контроля версий GIT. Автор изменений оформляет Pull Request (Merge Request) и назначает инспектора, отправляет в рабочий чат в Telegram сообщение с упоминанием инспектора.

Столкнувшись с изменением дизайна проекта, инспектор обращается к председателю инспекции в рабочем чате в Telegram. По окончании своей работы инспектор одобряет Pull Request или отправляет на доработку автору, уведомляя его в рабочем чате в Telegram.

**Порядок подготовки и проведения инспекции**

Инспекция должна быть проведена в течение 10 рабочих дней с момента её объявления.

После анализа инспектор в GIT оставляет комментарии или замечания. При необходимости (наличие серьёзных замечаний) автору необходимо их исправить. Если нет серьёзных замечаний, то инспекция считается завершённой и изменения вступают в силу.

**Порядок статусов и степени важности замечаний**

* Комментарий – советы по оптимизации продукта, не требующая обязательных изменений;
* Ошибка – замечание, которое необходимо исправить.

**Порядок верификации учёта замечаний**

После вынесения повторных изменений инспектор сравнивает изменения и вынесенные замечания. Если нет дополнительных замечаний, то выносится вердикт верификации и изменения вступают в силу, иначе назначаются новые замечания.

**Метрики, характеризующие эффективность инспекции**

Inspection efficiency(IE): IE = Размер продукта / (Общее время инспекции (в часах) \* Количество исправлений).

1. **Разработка модели 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.

**Состояния задач и их интерпретация**

* Backlog – новая или отложенная задача;
* InProgress – задача находится в стадии реализации;
* Inspected – реализованная задача находится в стадии инспекции;
* Tested – задача в стадии тестирования;
* Done – задача реализована.

**Правила создания новой задачи**

Каждый участник команды может создавать задачи в рамках назначенной ему части разработки.

**Правила перехода задачи из состояния в состояние**

Системой отслеживания задач будет выступать GitHub Projects.

Новая задача имеет статус Backlog. Тимлид может назначать на задачу исполнителя и инспектора. Участник может взять роль исполнителя на себя.

Если участник приступает к задаче, на которую он назначен, то задача получает следующий статус InProgress.

Как только задача выполнена, она переходит в состояние Tested. Исполнитель на данном этапе проверяет работоспособность, вылавливает и исправляет ошибки.

После завершения тестирования задаче переходит в состояние Inspected. В данном случае происходит инспекция выполненной задачи. Если инспекция прошла без существенных замечаний, то задача переходит в статус завершённой − Done, иначе она получает статус InProgress, чтобы доработать задачу.

1. **Разработка презентации проекта**

На рисунке 2 представлена титульная страница презентации.



Рисунок 2 – Титульная страница

Что содержит в себе обучающая система представлено на рисунке 3



Рисунок 3 – Из чего состоит обучающая система

Описание особенностей программы представлено на рисунке 4.

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 4 – Описание особенностей программы

1. **Разработка требований к проекту**

**О продукте**

**Программным продуктом** выступает автоматизированная система обучения дисциплине «Теория игр» обучения работы со структурой данных «AVL Tree» в виде клиент-сервера.

Данный продукт предназначен для обучения дисциплине «Теория игр» и обучения работы со структурой данных «AVL Tree».

Программный продукт состоит из следующих подсистем:

1. Пользовательский интерфейс;
2. Модуль обучения деревьям;
3. Модуль обучения «Теории игр»;
4. Модуль работы с базами данных.

**Общие требования**

**Функциональные требования**

**Требование FU\_001** Продукт должен иметь возможность создания и редактирования курсов: преподаватель должен иметь возможность создавать новые курсы или редактировать существующие, добавляя или удаляя материалы, задания и тесты.

**Требование FU\_002** Продукт должен обеспечивать возможность управления пользователями и группами: система должна позволять управлять пользователями и группами, добавлять новых пользователей, назначать им роли и права доступа, а также управлять группами пользователей.

**Требование FU\_003** Продукт должен обеспечивать создание и редактирование материалов: преподаватели должны иметь возможность создавать и редактировать различные виды материалов, такие как тексты, видео, аудио, презентации и т.д.

**Требование FU\_004** Продукт должен позволять создавать и обеспечивать настройку заданий и тестов: система должна поддерживать создание и настройку различных типов заданий и тестов, таких как вопросы с одиночным и множественным выбором.

**Требование FU\_005** Продукт должен позволять отслеживать прогресс учащихся: cистема должна отслеживать прогресс учащихся, показывать им их результаты, давать рекомендации по улучшению своих навыков и знаний.

**Требование FU\_006** Продукт должен быть способен интегрироваться с другими системами, такими как системы управления обучением (LMS), системы контроля версий (Git) и т.д.

**Нефункциональные требования**

**Требование UF\_001** Продукт должен быть безопасным: Система должна обеспечивать защиту персональных данных пользователей и преподавателей, а также обеспечивать безопасность передачи данных.

**Требование UF\_002** Продукт должен быть надежным: Система должна работать без сбоев и обеспечивать стабильную работу пользователей.

**Требование UF\_003** Продукт должен быть масштабируемым: Система должна иметь возможность масштабирования для увеличения количества пользователей и объема обрабатываемых данных.

**Требование UF\_004** Продукт должен быть производительным: Система должна обеспечивать высокую скорость работы и выполнения задач.

**Требование UF\_005** Продукт должен быть удобным для пользователей: Система должна быть удобной и интуитивно понятной для пользователей, обеспечивать простой и быстрый доступ к необходимым функциям и информации.

**Требование UF\_006** Продукт должен соответствовать стандартам качества: Система должна соответствовать международным стандартам качества и требованиям законодательства.

**Системные требования**

**Требование SM\_001** Продукт должен запускаться на таких браузерах, как Google Chrome, Opera, Safari, Firefox и т.д.

**Требования к подсистеме «Пользовательский интерфейс»**

**Требование UI\_001** На сайте должны быть следующие элементы интерфейса:

1. Страница личного аккаунта пользователя;
2. Страница выбора изучаемого курса;
3. Страница работы с курсом «Теория игр»;
4. Страница работы с курсом «Деревья».

**Требование UI\_002** Если пользователь ввёл некорректные данные, то по нажатии кнопки отправки данных *(см. UI\_001)* пользователь должен получать системное сообщение об ошибке.

**Требование UI\_003** Сообщение об ошибке *(см. UI\_002)* должно содержать в себе информацию о типе ошибки (лексическая, синтаксическая, неверный ввод) и способе исправления ошибки.

**Требования к подсистеме «Модуль обучения деревьям»**

**Требование LX\_001** В модуль должно входить 1 или несколько уроков по теме «Работа с алгоритмами деревьев».

**Требование LX\_002** Каждый урок должен содержать модуль теории и модуль практики для проверки знаний испытуемого**.**

**Требование LX\_003** Для отслеживания уже выполненных верно уроков, система должна иметь различную цветовую окраску для пройденных уроков и для еще не начатых.

**Требование LX\_004** Подсистема должна позволять начать задание заново.

**Требование LX\_005** Подсистема отрисовки деревьев должна корректно выводить дерево в соответствии с поданными данными.

**Требования к подсистеме «Модуль обучения «Теории игр»»**

**Требование SY\_001** В модуль должны входить один или несколько занятий по дисциплине «Теория игр».

**Требование SY\_002** Каждое занятие должно содержать модуль теории и модуль практики для проверки знаний испытуемого**.**

**Требование SY\_003** Для отслеживания уже выполненных верно занятий, система должна иметь различную цветовую окраску для пройденных занятий и для еще не начатых.

**Требование SY\_004** Подсистема должна позволять начать задание заново.

1. **Разработка архитектуры проекта**

Программный продукт обучающая система по математическим дисциплинам состоит из следующих подсистем:

1. Подсистема прохождения курса;
2. Подсистема отрисовки деревьев;
3. Подсистема редактирования данных пользователя;
4. Подсистема редактирования учебного материала.

На рисунке 5 приведена архитектурно-контекстная диаграмма обучающей системы.

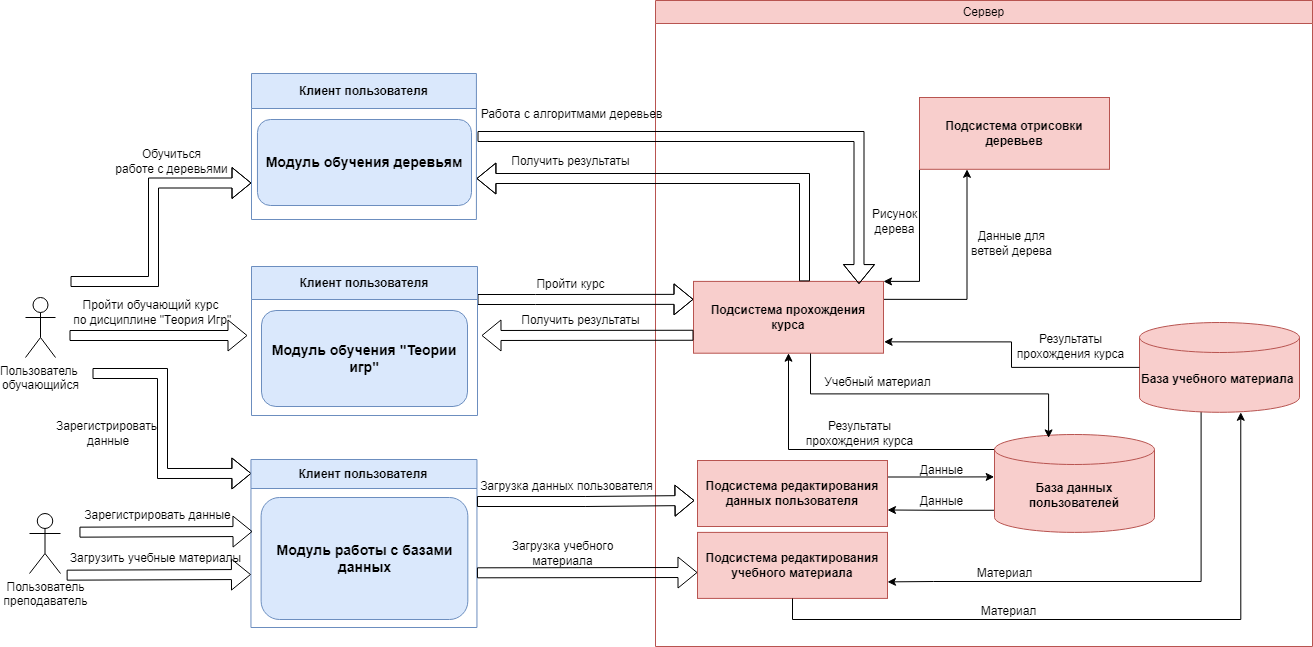


Рисунок 5 – Архитектурно-контекстная диаграмма обучающей системы

1. **Разработка программы измерений проекта**

Контроль за производственным процессом и его результатами является одним из определяющих видов деятельности на современном предприятии, производящем ПО. Для оценки эффективности используются специальные метрики.

Комплекс мероприятий, направленных на количественную оценку эффективности работы компании, называется программой измерений компании. Она может выполняться как в рамках одного проекта компании, так и в рамках более мелких задач.

**Метрика эффективности процесса производства**

1. Working Norm Day (Выполнение дневной рабочей нормы)

Working Norm *Day= (Количество строчек кода[документа] за рабочий день / минимальное количество строчек кода[документа] за рабочий день) \* 100%, где минимальное количество строчек кода[документа] = 50[от 40 до 80];*

*Стратегическая цель метрики – повысить производительность труда, путём соблюдения нормы.*

*Изучаемый объект метрики – проект, измеряемый атрибут – производительность труда на проекте.*

*Единица измерения – %.*

2. Time to market (скорость разработки и внедрения новых функций в проект)

*TTM = [Количество человек \* количество затраченных рабочих часов / ожидаемое количество человеко-часов] \* 100%*

*Стратегическая цель метрики – повысить качество разрабатываемого ПО, оценка эффективности работы команды над задачей, выявление наиболее затратных типов задач для команд.*

*Изучаемый объект метрики – задача, измеряемый атрибут – временная эффективность работы команды над задачей.*

*Единица измерения – %.*

**Метрика качества продукта**

1. Product Fault Density (плотность ошибок, внесённых на каком-либо этапе проекта)

*PFD-Req = Число ошибок, допущенных на этапе разработки требований / Количество требований проекта;*

*PFD-Coding = Число ошибок, допущенных на этапе кодирования / Количество строчек кода;*

*PFD-Test = Число ошибок, допущенных на этапе тестирования / Количество тестов;*

*Стратегическая цель метрики – повысить качество разрабатываемого ПО.*

*Изучаемый объект метрики – продукт, измеряемый атрибут – плотность неполадок (учитывает все ошибки).*

*Единица измерения – неполадка / единица размера.*

1. Test coverage (степень покрытия тестами кода)

Test coverage *= (количество покрытых тестами строк кода / количество строк кода) \* 100%;*

*Стратегическая цель метрики – увеличить процент покрытия кода тестами.*

*Изучаемый объект метрики – продукт, измеряемый атрибут – процент протестированного кода.*

*Единица измерения – %.*

**Метрика качества продукта**

1. Product Fault Density (плотность ошибок, внесённых на каком-либо этапе проекта)

*PFD-Req = Число ошибок, допущенных на этапе разработки требований / Количество требований проекта;*

*PFD-Coding = Число ошибок, допущенных на этапе кодирования / Количество строчек кода;*

*PFD-Test = Число ошибок, допущенных на этапе тестирования / Количество тестов;*

*Стратегическая цель метрики – повысить качество разрабатываемого ПО.*

*Изучаемый объект метрики – продукт, измеряемый атрибут – плотность неполадок (учитывает все ошибки).*

*Единица измерения – неполадка / единица размера.*

1. **Разработка перечня задач проекта**

**Общие задачи**

**TASK\_GEN\_1. Реализация обучающую систему по математическим дисциплинам.**

До 27.02.2024 реализовать обучающую систему по математическим дисциплинам. Система должен соответствовать требованиям, указанным в REQ\_CX\_001/002 (см. Software Requirement Specification).

**TASK\_GEN\_2. Реализация подсистемы «Модуль теории игр».**

До 13.02.2024 реализовать модуль теории игр. Данная подсистема должна иметь 1 реализованный урок, а также теорию к нему и соответствовать требованиям (см. Software Requirement Specification, REQ\_САN\_001).

**TASK\_GEN\_3. Реализация подсистемы «Модуль алгоритмов с деревьями».**

До 17.02.2024 реализовать модуль алгоритмов с деревьями. Должен быть реализован 1 урок, а также теория к нему и соответствовать требованиям (см. Software Requirement Specification, REQ\_САN\_001).

**TASK\_GEN\_4. Реализация интерфейса.**

До 21.02.2024 реализовать интерфейс разрабатываемого программного средства. Интерфейс должен соответствовать требованиям (см. Software Requirement Specification, REQ\_UI\_001, REQ\_UI\_002).

1. **Разработка рекомендаций по кодированию**

**Рекомендации по кодированию**

В основе рекомендаций — руководство по оформлению кода на языке python PEP 8. Все правила PEP 8 должны соблюдаться в проекте. Ниже перечислены самые важные рекомендации и требования по оформлению кода.

**Рекомендации и требования к оформлению кода**

1. Запрет на неосмысленные названия для классов, переменных, и т. д. Название переменной должно четко отражать ее смысл. Не допускается создание переменных из одной буквы (за исключением итераторов и общепринятых обозначений), бессмысленного набора символов, на транслите, и т.д.

Хорошо:

username = …

class Lesson…

def add\_user():…

Плохо:

g = ‘’

class fjdklsajkl…

def func(a, c, d):…

1. Требование не использовать wildcard import’ы:

Хорошо:

import os

from math import sqrt

Плохо:

from os import \*

from math import sqrt

1. Рекомендация разделять пробелами арифметические и логические операторы для улучшения читаемости кода

Хорошо:

result = first + second

Плохо:

result=first+second

1. Запрет на «широкую» обработку исключений с помощью except.

Хорошо:

try:

    # код...

except ValueError:

    # обработка исключения ValueError

except FileNotFoundError as e:

    # обработка исключения FileNotFoundError

Плохо:

try:

    # код...

except:

    # обработка всех исключений

1. Требование при итерации по коллекции использовать функцию enumerate вместо стандартного цикла for in range().

Хорошо:

for i, item in enumerate(my\_list):

    # код...

Плохо:

for i in range(len(my\_list)):

    item = my\_list[i]

    # код...

1. Рекомендация избегать написания файлов с более 250 строками.

**Чек-лист для проверки корректности кода:**

1. проверка соответствия наименования сущностей в коде согласно стандарту PEP 8;
2. нет файлов, где больше 250 строк;
3. соблюдено нужное количество разделяющих пустых строк между классами, функциями, блоками кода, секцией импортов, и т. д.;
4. использованы оптимальные конструкции для достижения требуемого результата;
5. корректная обработка исключений, отсутствие «широкой» обработки исключений;
6. порядок импортов соблюден согласно стандарту PEP 8, нет wildcard импортов;
7. в коде нет лишних пробелов, между арифметическими и логическими операторами пробелы стоят (кроме исключительных случаев, по типу передачи именованных параметров в функцию/метод).
8. **Разработка плана тестирования проекта**
9. **Тестирование проекта**

**Заключение**

В рамках курсовой работы было разработано программное средство «Обучающая система по математическим дисциплинам» с использованием подходов коллективной промышленной разработки, для чего были решены следующие поставленные 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 программу измерений проекта;
8. разработать перечь задач проекта;
9. разработать рекомендации по кодированию;
10. разработать план тестирования проекта;
11. провести тестирование проекта.

Таким образом, цель данной курсовой работы была достигнута.

**Список литературы**

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