**ОТЧЕТ**

по производственной практике

обучающегося \_\_\_Данкова Дмитрия Алексеевича \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(фамилия, имя, отчество)*

3 курса группы \_22ВП2\_\_\_\_\_\_\_\_ факультета вычислительной техники

направления подготовки 03.09.04 - «Программная инженерия»\_\_\_\_\_\_\_\_\_\_\_\_\_\_,

проходившего практику с 25\_\_июня\_\_\_\_\_\_ 2025 по 8 июля\_\_\_\_\_\_\_\_\_2025

в АО “ИнфоТеКС”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

*(наименование профильной организации)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

*(подпись обучающегося) (расшифровка подписи)*

«Отчет УТВЕРЖДАЮ»

Руководитель практики \_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись) (расшифровка подписи) (должность)*

кафедры \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ПГУ.

*( название кафедры)*

Руководитель практики от \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*( наименование организации, предприятия, учреждения)*

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

*(занимаемая должность) (подпись) (расшифровка подписи)*

М.П.

# Реферат

Пояснительная записка содержит 18 листов, 4 рисунков, 1 таблиц,   
 3 использованных источника.

РЕЗЕРВНОЕ КОПИРОВАНИЕ, C#, UML 2.0, JSON, ЛОГИРОВАНИЕ, SERILOG, КОНФИГУРАЦИЯ, VISUAL STUDIO, ОБРАБОТКА ОШИБОК, МОДУЛЬНОСТЬ, ДИАГРАММА КОМПОНЕНТОВ, ДИАГРАММА ДЕЯТЕЛЬНОСТИ

Объектом исследования является методология UML 2.0, а также реализация программы резервного копирования файлов и каталогов на языке C#.

Целью является разработка консольного приложения, реализующего резервное копирование файлов из нескольких исходных директорий в целевую с временной меткой, с поддержкой конфигурации в формате JSON и журналирования, а также использование методологии UML 2.0 для демонстрации структуры программы, алгоритма работы и особенностей реализации.

Разработка проводилась на языке программирования C# в среде программирования Visual Studio. Для проектирования были созданы UML-диаграммы: диаграмма компонентов, описывающая структуру программы, и диаграмма деятельности, иллюстрирующая алгоритм копирования.

Осуществлено функциональное тестирование разработанного программного обеспечения, которое подтвердило корректность его работы, включая успешное копирование файлов и обработку всех предусмотренных ошибок.

**Содержание**

[Введение 4](#_Toc202306582)

[1 Общая характеристика АО “ИнфоТеКС” 5](#_Toc202306583)

[2 Обзор задач, решаемых в АО “ИнфоТеКС” 6](#_Toc202306584)

[3 Задание, постановку которой осуществил руководитель от организации 7](#_Toc202306585)

[3.1 Постановка задачи и анализ требований 7](#_Toc202306586)

[3.2 Разработка структуры программы и спецификация на программные модули 8](#_Toc202306587)

[3.3 Реализация 9](#_Toc202306588)

[3.4 Тестирование 13](#_Toc202306589)

[Заключение 17](#_Toc202306590)

[Список используемых источников 18](#_Toc202306591)

Введение

Данная практика проводится с целью приобретения практических навыков в разработке программного обеспечения, включая проектирование структуры программы, реализацию алгоритмов, тестирование и документирование. Разработанное решение должно быть совместимо с операционной системой Windows 10, компилироваться в среде Visual Studio или Visual Studio Code и поставляться в виде zip-архива, содержащего исходный код и бинарную версию приложения.

Основной задачей является создание консольного приложения на языке C#, которое обеспечивает копирование файлов из указанных исходных директорий в целевую директорию с учетом временной метки, а также обработку ошибок, связанных с доступом к файлам и директориям. Дополнительно требуется реализовать поддержку конфигурации в формате JSON, возможность указания нескольких исходных папок и журналирование процесса копирования с настраиваемым уровнем логирования.

Данный отчет должен включать следующие разделы: постановка задачи и анализ требований, разработка структуры программы и спецификация программных модулей, описание реализации, а также результаты тестирования. Каждый раздел подробно описывает этапы выполнения задания, включая проектирование, программирование и проверку работоспособности решения.

1 Общая характеристика АО “ИнфоТеКС”

АО «ИнфоТеКС» (Информационные Технологии и Коммуникационные Системы) — российская компания, специализирующаяся на разработке программно-аппаратных VPN-решений и средств криптографической защиты информации. Основанная 6 сентября 1991 года группой экспертов по информационной безопасности во главе с Андреем Чапчаевым, компания входит в пятерку крупнейших российских организаций в области защиты информации [1].

В России у ИнфоТеКС есть три дочерние компании:

• ОАО «ИнфоТеКС Интернет Траст», основанное в 2001 году, предоставляет услуги по защите информации с использованием продуктов и технологий ViPNet, включая электронный документооборот. Компания является удостоверяющим центром и оператором сдачи отчетности в электронном виде для ПФР, ФНС и других государственных учреждений, входя в ТОП-10 крупнейших удостоверяющих центров России.

• ЗАО «Перспективный мониторинг», созданное в 2007 году, занимается исследованием состояния безопасности информационных систем организаций, выявлением уязвимостей и недокументированных сервисов. Компания оказывает экспертную поддержку в разработке политик, требований и инструкций по информационной безопасности, а также в актуализации существующих регламентов под изменяющиеся бизнес-требования.

• Учебный центр ИнфоТеКС, функционирующий с 2000 года, сотрудничает с ведущими вузами страны, предоставляя консультации и обучение специалистам в сфере информационной безопасности. Центр обеспечивает теоретическую и практическую подготовку клиентов, сотрудников и партнеров компании в вопросах построения комплексных систем защиты информации и использования средств защиты в автоматизированных системах.

Кроме того, в состав ГК «ИнфоТеКС» входят компании «Системы практической безопасности» и «СФБ лаборатория», предоставляющая услуги по сертификации средств защиты информации и проведению контрольных исследований программных и программно-аппаратных СЗИ по требованиям ФСБ России.

2 Обзор задач, решаемых в АО “ИнфоТеКС”

Компания «ИнфоТеКС» — ведущий отечественный разработчик и производитель высокотехнологичных программных и программно-аппаратных средств защиты информации. Основная продукция компании, известная благодаря решениям для защиты каналов связи корпоративных сетей, была преобразована в современный комплекс, ориентированный на защиту от актуальных угроз информационной безопасности. В портфолио компании также входят решения по обнаружению и предотвращению угроз и вторжений, защите серверов и рабочих станций от несанкционированного доступа, а также решения для защиты мобильных устройств и коммуникаций, ИБ АСУ ТП, IIoT, криптографические сервисы, библиотеки и модули для встраивания [2].

Некоторые из наиболее популярных продуктов компании включают:

• ViPNet CSS Connect HW: стационарный телефон с сенсорным экраном, предназначенный для общения пользователей сети ViPNet по защищенному каналу.

• ViPNet SafeBoot 3: новое поколение программного модуля доверенной загрузки (ПМДЗ), сертифицированного ФСБ и ФСТЭК России. Предназначен для создания точки доверия к платформе и её компонентам, а также загружаемой операционной системе, обеспечивая защиту UEFI BIOS и организацию доверенной загрузки ОС.

• ViPNet Client: программный комплекс для защиты рабочих мест корпоративных пользователей, надежно защищающий от сетевых атак за счет фильтрации трафика и обеспечивающий защищенную работу с корпоративными данными через зашифрованный канал, включая удаленных пользователей.

• ViPNet OSSL: программное обеспечение на базе библиотеки OpenSSL, позволяющее использовать российские криптографические алгоритмы ГОСТ через интерфейс OpenSSL.

• ViPNet Hardware Security Module (ViPNet HSM): универсальный криптографический модуль для реализации криптографических операций по запросу различных прикладных сервисов.

• ViPNet Quantum Trusted System Lite (ViPNet QTS Lite): квантовая криптографическая система для выработки и распределения ключей, обеспечивающая квантовозащищенными ключами средства криптографической защиты информации (СКЗИ) в автоматическом режиме.

• ViPNet CryptoSmart: криптопровайдер для работы с распределенными реестрами, предназначенный для защиты распределенных реестров и смарт-контрактов с использованием криптографических алгоритмов ГОСТ, встраиваемый в блокчейн-платформы на базе Hyperledger Fabric.

Эти продукты демонстрируют широкий спектр решений компании «ИнфоТеКС» для защиты информации в различных областях и обеспечивают надежную защиту данных от актуальных угроз.

3 Задание, постановку которой осуществил руководитель от организации

3.1 Постановка задачи и анализ требований

В файле настроек хранятся пути для исходной и целевой папки.

При запуске программы происходит создание папки с временным штампом в целевой папке и копирование в неё всех доступных файлов из исходной. Требуется обрабатывать ситуации с невозможностью доступа к файлам в исходной папке.

Пункты со звездочкой являются дополнительными и не обязательны для выполнения.

\* Файл настроек имеет формат JSON.

\* Есть возможность указать несколько исходных папок.

\* Ведется журналирование процесса копирования. Каждый запуск создает свой файл журнала. Уровень журналирования можно указать в файле настроек.

Примеры распределения событий:

• Error - Ошибки приложения. Например, те, которые вызвали неожиданное падение.

• Info - Основные события приложения: старт приложения, обработка одной исходной папки или обработанные ошибки.

• Debug - Отладочная информация. Например, скопирован отдельный файл

Требования к присылаемым решениям.

• Готовые задания должны быть переданы в zip архиве.

• Каждый из проектов должен находиться в своей папке и должен в обязательном порядке содержать отдельно исходный код приложения и отдельно собранную бинарную версию, работающую в среде MS Windows 10.

• Собранная программа не должна требовать настроек системы или нахождения определенных файлов в специфичном месте.

• Исходный код должен компилироваться средствами Visual Studio или Visual Studio Code.

• В архиве не должно быть неиспользуемых исходных кодов, ресурсов или промежуточных файлов сборки.

• Максимальное время на выполнение задания – 2 недели.

3.2 Разработка структуры программы и спецификация на программные модули

Программа разрабатывается на языке C# и представляет собой консольное приложение для выполнения резервного копирования файлов и каталогов. Структура программы организована в соответствии с принципами модульности и включает следующие ключевые компоненты:

Program – основной класс, содержащий точку входа в приложение (Main), который инициирует процесс резервного копирования через вызов метода StartBackup класса BackupService.

BackupService – сервисный класс, реализующий логику резервного копирования. Отвечает за загрузку конфигурации, настройку логгера и выполнение операций копирования.

Logger – класс для ведения журнала операций, использующий библиотеку Serilog. Поддерживает уровни логирования (Information, Debug, Error) и сохраняет лог в файл.

AppConfig – класс конфигурации, содержащий параметры приложения, такие как пути исходных и целевой папок, а также уровень логирования. Загружается из JSON-файла AppConfig.json.

Для демонстрации структуры программы была разработана диаграмма компонентов[3], изображённая на рисунке 1.

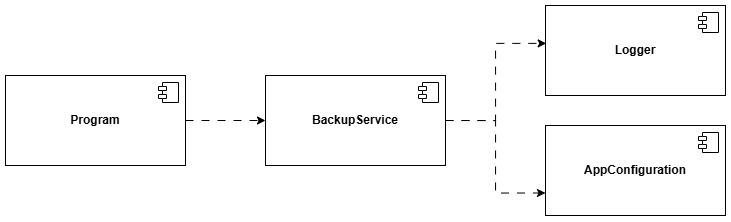


Рисунок 1 – Диаграмма компонентов

3.3 Реализация

В ходе разработки были реализованы следующие программные модули:

**1. Класс Program**

* **Назначение**: Точка входа в приложение.
* **Функциональность**:
  + Инициирует создание экземпляра BackupService.
  + Вызывает метод StartBackup для выполнения резервного копирования.
  + Обрабатывает критические ошибки с выводом сообщений в консоль.
* **Входные параметры**: Аргументы командной строки (args), которые в текущей версии не используются.
* **Выходные данные**: Сообщения об ошибках в случае их возникновения.

**2. Класс BackupService**

* **Назначение**: Управление процессом резервного копирования.
* **Функциональность**:
  + Загружает конфигурацию из файла AppConfig.json с использованием библиотеки System.Text.Json.
  + Проверяет существование целевой папки и создает подпапку с временной меткой.
  + Инициирует логгер с заданным уровнем логирования.
  + Выполняет копирование файлов и папок из исходных директорий в целевую, сохраняя структуру каталогов.
  + Обрабатывает ошибки доступа, отсутствия файлов или директорий, а также ошибки десериализации JSON.
* **Методы**:
  + BackupService(): Конструктор, выполняющий инициализацию конфигурации и логгера.
  + StartBackup(): Запускает процесс копирования для всех исходных директорий.
  + CopyFromDirectory(string sourceDirectory): Выполняет копирование файлов и папок из указанной исходной директории.
* **Входные параметры**: Конфигурация из файла AppConfig.json.
* **Выходные данные**: Скопированные файлы и папки в целевой директории, а также лог-файл.

**3. Класс Logger**

* **Назначение**: Ведение журнала операций резервного копирования.
* **Функциональность**:
  + Настраивает библиотеку Serilog для записи логов в файл с временной меткой.
  + Поддерживает три уровня логирования: Information, Debug, Error.
  + Обеспечивает журналирование событий, таких как начало и завершение копирования, ошибки доступа и другие исключения.
* **Методы**:
  + Logger(string logDirectory, string logLevel): Конструктор, настраивающий логгер.
  + LogInformation(string message): Записывает информационные сообщения.
  + LogDebug(string message): Записывает отладочные сообщения.
  + LogError(string message): Записывает сообщения об ошибках.
* **Входные параметры**: Путь для логов и уровень логирования из конфигурации.
* **Выходные данные**: Текстовый файл лога в целевой директории.

**4. Класс AppConfig**

* **Назначение**: Хранение конфигурационных параметров приложения.
* **Функциональность**:
  + Содержит свойства для списка исходных директорий (SourceDirectories), целевой директории (TargetDirectory) и уровня логирования (LogLevel).
  + Загружается из файла AppConfig.json с использованием десериализации JSON.
* **Входные параметры**: JSON-файл с настройками.
* **Выходные данные**: Объект с заполненными свойствами для использования в BackupService.

**5. Файл конфигурации AppConfig.json**

* **Назначение**: Хранение настроек приложения в формате JSON.
* **Структура**:
  + SourceDirectories: Список путей к исходным директориям.
  + TargetDirectory: Путь к целевой директории для резервных копий.
  + LogLevel: Уровень логирования (Debug, Information, Error).
* **Пример**:

{

"SourceDirectories": ["D:\\Down\\Data"],

"TargetDirectory": "D:\\Down\\Backups",

"LogLevel": "Debug"

}

Программа использует обработку исключений для повышения надежности, включая ошибки отсутствия файлов, проблем с доступом и некорректной десериализации JSON.

Логирование реализовано с использованием библиотеки Serilog, что обеспечивает гибкость в настройке уровней журналирования.

Структура программы позволяет легко расширять функциональность, например, добавлять новые параметры в конфигурацию или дополнительные методы обработки данных.

Для описания реализованного алгоритма бекапирования была реализована диаграмма деятельности, изображённая на рисунке 2.

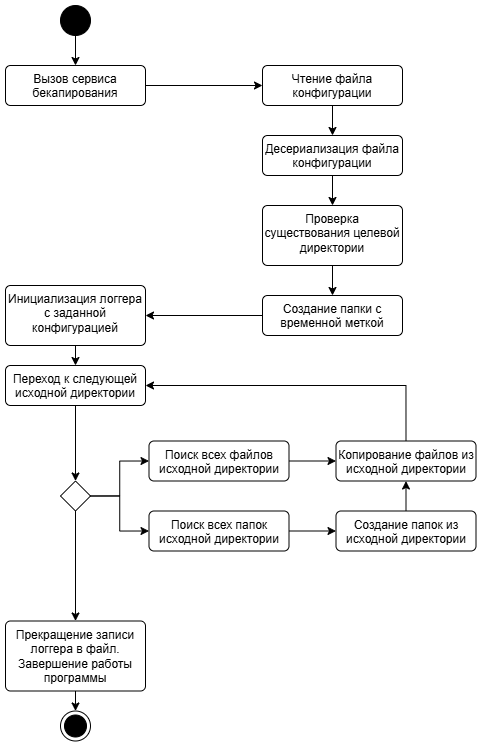


Рисунок 2 – Диаграмма деятельности

3.4 Тестирование

Было проведено функциональное тестирование разработанного программного обеспечения. Сценарии тестирования отображены в таблице 1.

Таблица 1 – Сценарии тестирования

|  |  |  |
| --- | --- | --- |
| Описание | Статус | Результат |
| Успешный сценарий: Программа запускается с корректным файлом AppConfig.json, содержащим существующие исходные и целевые директории. Все файлы и папки копируются без ошибок, логгер записывает события на уровне Debug. | Успешно пройдено | Все файлы и папки скопированы в целевую директорию с временной меткой, создан лог-файл с записями о начале, процессе и завершении копирования. (Рис 3) |
| Файл конфигурации отсутствует: Программа запускается, но файл AppConfig.json не существует в директории приложения. | Успешно пройдено | Выброшено исключение FileNotFoundException с сообщением: "Файл настроек не найден." |
| Некорректные данные конфигурации: Файл AppConfig.json корректен, но десериализация возвращает null (например, из-за несоответствия структуры). | Успешно пройдено | Выброшено исключение Exception с сообщением: "Не удалось десериализовать настройки." |
| Целевая директория не указана или не существует: Файл AppConfig.json содержит пустое или несуществующее значение для TargetDirectory. | Успешно пройдено | Выброшено исключение DirectoryNotFoundException с сообщением: "Ошибка пути для целевой папки!" |
| Исходная директория не существует: В AppConfig.json указана несуществующая исходная директория в SourceDirectories. | Успешно пройдено | Логгер записывает ошибку: "Исходная директория не найдена: {sourceDirectory}". Копирование для этой директории пропускается, программа продолжает работу с другими директориями. (Рис 4) |

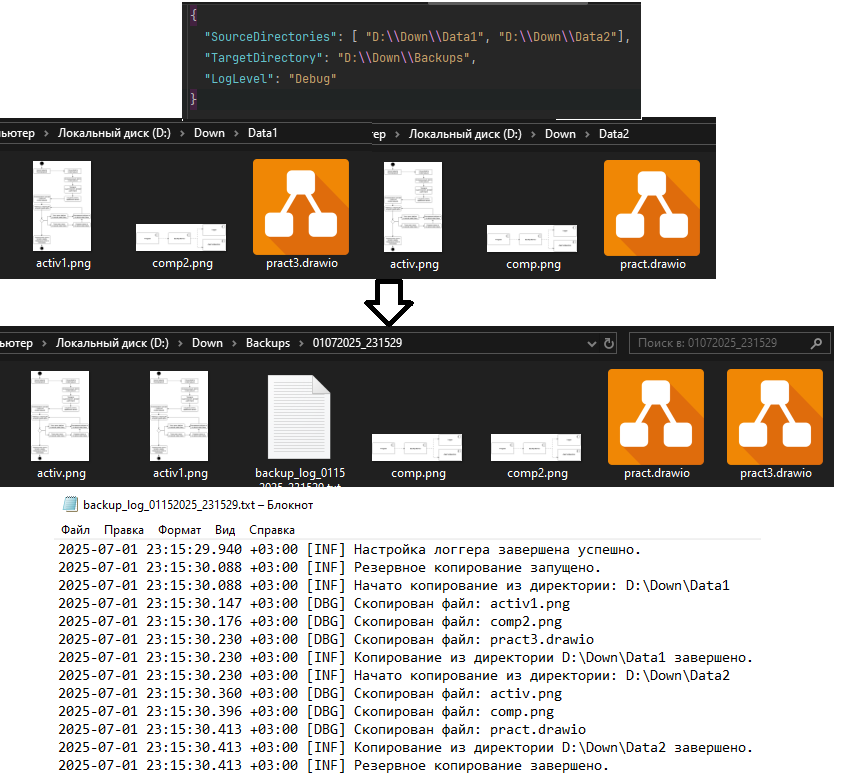


Рисунок 3 – Успешный сценарий

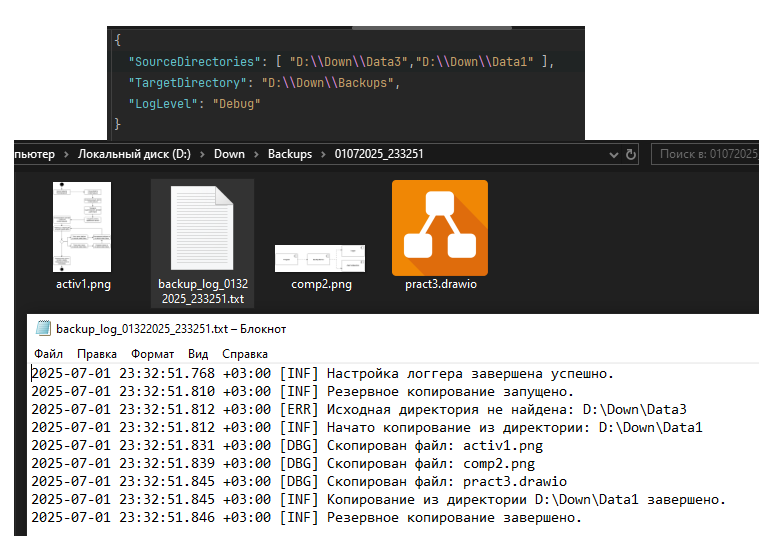


Рисунок 4 – Исходная директория не существует

В ходе тестирования расхождения ожидаемого и фактического результата не выявлено.

Заключение

В ходе практики было успешно разработано консольное приложение на языке C# для резервного копирования файлов и каталогов, полностью соответствующее поставленным требованиям. Программа реализует функциональность копирования файлов из нескольких исходных директорий в целевую папку с временной меткой, поддерживает конфигурацию в формате JSON и обеспечивает журналирование процесса с использованием библиотеки Serilog. Обработка исключений, таких как отсутствие файлов, некорректная конфигурация или ошибки доступа, гарантирует надежность работы приложения.

Тестирование подтвердило корректность работы программы: успешный сценарий копирования и обработка всех предусмотренных ошибок прошли без расхождений между ожидаемыми и фактическими результатами. Структура программы, построенная на принципах модульности, позволяет легко расширять функциональность, например, добавлять новые параметры конфигурации или интегрировать дополнительные возможности обработки данных. В ходе практики были закреплены навыки программирования на C#, работы с JSON, логирования и тестирования программного обеспечения.

Список используемых источников

1. Компания ИнфоТеКС (электронный ресурс) - https://ru.wikipedia.org/wiki/ИнфоТеКС#cite\_note-1.
2. ИнфоТеКС (Infotecs) (электронный ресурс) - https://www.tadviser.ru/index.php/Компания:ИнфоТеКС\_(Infotecs)#.D0.A1.D1.82.D1.80.D1.83.D0.BA.D1.82.D1.83.D1.80.D0.B0\_.D0.BA.D0.BE.D0.BC.D0.BF.D0.B0.D0.BD.D0.B8.D0.B8
3. Джим Арлоу. UML2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование/Джим Арлоу, Айла Нейштадт. – Санкт-Петербург, Издательство Символ-Плюс, 2007. – 624с.