ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА И ГОСУДАРСТВЕННОЙ СЛУЖБЫ ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»

КОЛЛЕДЖ МНОГОУРОВНЕВОГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

Специальность 09.02.07 «Информационные системы и программирование»

КУРСОВАЯ РАБОТА

на тему: «Расчет эффективности разработки и внедрения программного продукта»

Выполнил студент группы 412ИС-22
_____ Уткин Д.С.

Руководитель
_____ Хашина О.В.

МОСКВА 2026 г.

СОДЕРЖАНИЕ

ЗАЯВЛЕНИЕ

на утверждение темы курсовой работы

Кому: Руководителю курсовой работы
преподавателю дисциплины ОП.07 «Экономика отрасли»
Хашиной Оксане Владиславовне

От: студента 4 курса, группы 412ИС-22
специальности 09.02.07 «Информационные системы и программирование»
Уткина Даниила Сергеевича

Тема курсовой работы:
«Расчет экономических показателей разработки программного продукта с открытым исходным кодом»

Объект исследования:
Процесс разработки консольного приложения для архивирования и сжатия файлов с реализацией алгоритмов сжатия данных.

Прошу утвердить тему курсовой работы и её план:

План курсовой работы:

1. Введение
2. Материально-техническая база предприятий, оказывающих ИТ-услуги
3. Расчёт экономических показателей разработки программного продукта
4. Заключение
5. Список использованных источников

Подпись студента: _____ **Дата:** _____

Уткин Д.С.

Подпись руководителя: _____ **Дата утверждения:** _____

Хашина О.В.

Москва, 2025

ВВЕДЕНИЕ

Тема свободных программ актуальна как никогда. Программное обеспечение с открытым исходным кодом (Open-Source Software, OSS) признано стратегическим активом и драйвером инноваций на уровне экономик целых регионов, что подтверждается специализированными исследованиями, проводимыми для Европейского Союза [1]. Однако по мере того как OSS становится мейнстримом, возникает парадокс: согласно исследованию Linux Foundation (Census III, 2024), свободное и открытое программное обеспечение (FOSS) демонстрирует растущую зависимость мировой экономики, при этом анализ более 12 миллионов точек данных выявил, что 40% наиболее популярных проектов поддерживаются всего одним или двумя разработчиками [2]. Это создаёт ситуацию, когда критически важные компоненты цифровой инфраструктуры имеют минимальную ресурсную базу для долгосрочной поддержки и экономической оценки. Данная работа фокусируется на одном из таких классов проектов — консольных утилитах для обработки данных, типичным представителем которых является компрессор. Разработка подобных инструментов часто ведётся малыми командами с использованием открытого стека технологий, что делает задачу корректного расчёта их себестоимости и эффективности одновременно актуальной и методически сложной.

Степень научной разработанности темы. Социокультурные и организационные аспекты разработки open-source программного обеспечения глубоко исследованы в классической работе Эрика Реймонда «Собор и базар» [3]. Автор, анализируя успех Linux и собственного проекта fetchmail, противопоставляет закрытую «соборную» модель разработки (характерную для традиционной коммерческой разработки) открытой «базарной», где ключевую роль играет распределённое сообщество разработчиков, ранние и частые релизы, а также принцип «при достаточном количестве наблюдателей все ошибки становятся мелкими» (Закон Линуса). Однако, как отмечает сам Реймонд, мотивацией участников в такой модели служат репутация и личный интерес («egoboo»), а не прямое финансовое вознаграждение. Это создаёт фундаментальное противоречие: экономические модели оценки (такие как COCOMO) созданы для «соборной» модели с оплачиваемым трудом, в то время как значительная часть open-source экосистемы живёт по законам «базара». Данная работа направлена на частичное устранение этого противоречия путём создания методики расчёта, учитывающей специфику малых проектов, разрабатываемых в «базарной» парадигме.

Исследовательские проблемы, цель и задачи заключаются в отсутствии адаптированной и апробированной методики расчёта экономических показателей (себестоимости, эффективности) для open-source проектов, разрабатываемых индивидуально или малыми командами, на примере класса консольных утилит.

Конкретный исследовательский вопрос: Как определить полную себестоимость и оценить экономическую эффективность разработки консольного компрессора, реализованного с использованием исключительно открытых инструментов и распространяемого по свободной лицензии?

Цель работы — разработать и апробировать методику экономического обоснования разработки консольного архиватора с открытым исходным кодом, рассчитав ключевые показатели затрат и эффективности.

Для достижения цели поставлены следующие задачи:

1. Изучить теоретические основы: материально-техническую базу ИТ-предприятий и классические методики оценки стоимости ПО (COCOMO, функциональные точки).
2. Определить основные характеристики разрабатываемого программного продукта — консольного компрессора: его функции (MVP), целевую аудиторию и дать оценку трудозатрат на разработку.
3. Рассчитать капитальные (CAPEX) и операционные (OPEX) затраты на проект, спрогнозировать потенциальную выручку и определить себестоимость разработки.
4. Провести анализ эффективности проекта путём расчёта точки безубыточности, срока окупаемости и возврата на инвестиции (ROI).
5. Сформулировать практические рекомендации по адаптации методики для реалий индивидуальной open-source разработки.

Объект исследования — процесс разработки консольного приложения с открытым исходным кодом для архивирования и/или компрессии потоков данных (файлов).

Предмет исследования — экономические показатели данного процесса: состав и структура затрат, се-

бестоимость, показатели экономической эффективности (точка безубыточности, ROI, срок окупаемости).

Теоретическая значимость исследования заключается в преодолении выявленного методологического разрыва между классическими экономическими моделями оценки ПО, созданными для «соборной» разработки, и реалиями «базарной» open-source модели. Работа вносит вклад в экономику программной инженерии, предлагая подход к адаптации таких методик, как COCOMO, для условий малых, некоммерческих или слабо финансируемых проектов, чья роль в цифровой инфраструктуре, однако, остаётся критической.

Практическая значимость состоит в том, что разработанная методика и конкретные расчёты для кейса консольного компрессора предоставляют готовый инструмент для:

- Индивидуальных разработчиков и малых команд, позволяя реалистично оценить полную стоимость создания и поддержки open-source продукта, что необходимо для планирования ресурсов, поиска финансирования или обоснования перехода на модели гибридного монетизирования.
- Менеджеров и аналитиков IT-проектов в компаниях, которые используют или вносят вклад в open-source, помогая количественно оценить вклад в экосистему и затраты на внутреннюю поддержку внешних зависимостей.
- Образовательных учреждений, предлагая конкретный, структурированный кейс для обучения основам технико-экономического обоснования программных продуктов в условиях современной open-source парадигмы.

Таким образом, работа не только отвечает на конкретный исследовательский вопрос, но и обеспечивает переносимый методический каркас для экономического анализа широкого класса малых open-source проектов.

1 Теоретические основы разработки программного продукта с открытым исходным кодом

1.1 Программное обеспечение: определение, классификация, особенности open-source

1.1.1 Базовые определения и классификация программного обеспечения

В современной цифровой экономике программное обеспечение (ПО) является ключевым активом и средством производства. Согласно межгосударственному стандарту ГОСТ 19781-90, программное обеспечение определяется как «совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ» [4, п. 3]. Это определение подчёркивает комплексный характер ПО, включающий не только исполняемый код, но и сопровождающую документацию, что важно для оценки полного объёма работ при его создании.

С функциональной точки зрения тот же стандарт устанавливает классификацию ПО по назначению, выделяя три основные категории [4, пп. 4, 7, 8]:

- Системные программы, предназначенные для поддержания работоспособности системы обработки информации (операционные системы, драйверы, утилиты сопровождения).
- Прикладные программы, решающие задачи в определённой области применения (текстовые и графические редакторы, системы управления базами данных, инженерные пакеты).
- Программы обслуживания (утилиты), оказывающие услуги общего характера пользователям и обслуживающему персоналу. К этому классу относятся средства архивации, диагностики, управления файлами и другие инструменты для обслуживания системы.

Внутри этих категорий, согласно исследованиям, доминирующую роль в инфраструктуре начинают играть решения с открытым исходным кодом [1].

1.1.2 Философские и практические основы открытого кода

Феномен ПО с открытым исходным кодом (Open-Source Software, OSS) базируется на двух взаимодополняющих, но идеологически различных концепциях: «свободное программное обеспечение» (Free Software) и «открытое программное обеспечение» (Open Source).

Концепция свободного ПО, продвигаемая Фондом свободного программного обеспечения (Free Software Foundation, FSF), делает акцент на этических и социальных свободах пользователя. Она определяет свободное ПО через четыре неотъемлемых права: свободу запуска программы с любой целью, изучения и адаптации её исходного кода, распространения точных копий, а также распространения модифицированных версий [5]. Данный подход, сформулированный в 1980-х годах, рассматривает доступ к коду как необходимое условие для контроля пользователя над технологиями.

Более поздняя и прагматичная концепция открытого ПО, формализованная организацией Open Source Initiative (OSI), фокусируется на практических преимуществах открытой модели разработки для качества, безопасности и скорости инноваций. Её критерии, изложенные в The Open Source Definition (OSD), включают свободное распространение, обязательное наличие исходного кода, разрешение на создание производных работ и недискриминационный характер лицензии [6].

Несмотря на различия в философском обосновании, на практике набор лицензий, удовлетворяющих определению OSI, почти полностью совпадает с лицензиями, соответствующими критериям FSF. Для целей настоящего исследования, сфокусированного на экономических аспектах, используется термин «open-source software» (OSS) как более распространённый в деловой и академической среде, подразумевая соблюдение как критериев OSD, так и базовых свобод пользователя.

1.1.3 Основные типы лицензий открытого программного обеспечения

Юридической основой, регулирующей использование, модификацию и распространение OSS, являются лицензии открытого кода. Они определяют баланс между свободой сообщества и правами авторов. Наиболее

распространённые лицензии можно разделить на две основные категории, представленные в таблице 1.

Таблица 1: Сравнительная характеристика основных лицензий открытого программного обеспечения

Лицензия	Ключевой принцип (тип)	Основные условия и особенности
GNU GPL (General Public License)	Копилефт (Copyleft)	Обязательное лицензирование производных работ на условиях той же лицензии. Гарантирует сохранение открытости всех последующих модификаций. Является основой для многих проектов, включая ядро Linux.
MIT License	Разрешительная (Permissive)	Предоставляет максимальную свободу при минимальных ограничениях. Позволяет использование, изменение, распространение, в том числе в составе проприетарного ПО, с обязательным сохранением уведомления об авторских правах и лицензии.
Apache License 2.0	Разрешительная (Permissive)	Аналогична MIT, но дополнительно содержит явное предоставление патентных прав от contributors пользователям и защиту от патентного троллинга. Также требует сохранения уведомлений об изменениях.
BSD 2-Clause/3-Clause License	Разрешительная (Permissive)	Краткая лицензия, близкая по духу к MIT. Основное различие в 3-пунктной версии — наличие условия о запрете использования имени авторов для одобрения производных продуктов.

Выбор лицензии имеет прямые экономические последствия. Копилефт-лицензии (GPL) способствуют сохранению экосистемы полностью открытых проектов, но могут ограничивать коммерческое использование в проприетарных продуктах. Разрешительные лицензии (MIT, Apache) обеспечивают максимальную гибкость для бизнеса, позволяя интегрировать код в коммерческие решения, что способствует их широкому распространению в отраслевых стандартах и облачных сервисах [2].

1.1.4 Компрессоры как класс служебных программ (утилит)

В контексте классификации ГОСТ 19781-90, программы-архиваторы относятся к категории программ обслуживания (утилит). Их основное функциональное назначение — уменьшение объёма данных (сжатие) для экономии дискового пространства и ускорения передачи по сетям, а также упаковка множества файлов в единый архив для удобства хранения и переноса.

Исторически и технологически развитие компрессоров и архиваторов тесно связано с открытым исходным кодом. Классические консольные утилиты, такие как `gzip` (использующий алгоритм DEFLATE), `bzip2` и `xz`, являются стандартными компонентами любой UNIX-подобной системы и распространяются под свободными лицензиями (GPL). Кроссплатформенный архиватор `7-Zip`, поддерживающий современный формат 7z с высоким коэффициентом сжатия, также является open-source проектом (лицензия LGPL).

С алгоритмической точки зрения, компрессоры реализуют фундаментальные методы сжатия данных:

- Алгоритмы словарного сжатия (LZ77, LZ78), которые заменяют повторяющиеся последовательности символов ссылками на их предыдущие вхождения.
- Энтропийное кодирование (Алгоритм Хаффмана, Арифметическое кодирование), которое присваивает более короткие коды более частым символам.

Наиболее распространённый алгоритм DEFLATE, используемый в форматах ZIP и gzip, комбинирует метод LZ77 с кодированием Хаффмана.

Таким образом, консольный архиватор представляет собой типичный и технологически содержатель-

ный пример open-source утилиты. Его разработка требует реализации нетривиальных алгоритмов, но при этом проект обладает чётко очерченным функционалом, что делает его идеальным объектом для детального экономического анализа в рамках данной работы.

1.2 Состав материально-технической базы и специфика организации труда в ИТ

1.2.1 Состав и амортизация основных фондов

Материально-техническая база (МТБ) современного ИТ-предприятия представляет собой комплекс материальных активов, необходимых для осуществления деятельности по разработке, тестированию и эксплуатации программного обеспечения. К основным фондам традиционно относятся [4]:

- Здания и сооружения: офисные помещения, дата-центры.
- Вычислительная техника: рабочие станции разработчиков и инженеров, серверное оборудование для сборки, тестирования и развёртывания.
- Сетевое и телекоммуникационное оборудование: маршрутизаторы, коммутаторы, системы бесперебойного питания.
- Мебель и оргтехника.

Срок полезного использования такого оборудования для целей налогового учёта определяется в соответствии с Классификацией основных средств, включаемых в амортизационные группы (утверждённой Постановлением Правительства РФ № 1). Согласно данному классификатору, вычислительная техника (код ОКОФ 330.28.23.23) относится ко Второй амортизационной группе со сроком полезного использования от 2 до 3 лет включительно [7], что коррелирует с периодом её морального устаревания в ИТ-отрасли.

Особенностью ИТ-отрасли является высокая доля активов с быстрым моральным устареванием (3-5 лет), таких как серверы и компьютеры. В бухгалтерском учёте, согласно ФСБУ 6/2020, организация вправе выбирать способ начисления амортизации, в том числе линейный или способ уменьшаемого остатка, исходя из характера будущих экономических выгод [8]. В налоговом же учёте, наряду с линейным и нелинейным методами, Налоговый кодекс РФ (ст. 259.3) предусматривает возможность применения повышающих коэффициентов к норме амортизации для основных средств, используемых в условиях повышенной сменности, агрессивной среды или имеющих высокую энергетическую эффективность [9].

Таким образом, для быстроустаревающего ИТ-оборудования экономически обосновано применение ускоренных методов амортизации, что важно для корректного вычисления себестоимости разработки и налогового планирования.

1.2.2 Программные инструменты и экосистема разработки

Наряду с материальными активами, критически важной частью МТБ является программная экосистема. Согласно глобальному опросу разработчиков Stack Overflow (2025), доминирующими инструментами являются [10]:

- Среды разработки (IDE): Visual Studio Code (используется 75.9% респондентов), Visual Studio (29%), IntelliJ IDEA (27.1%). Преимущество open-source и условно-бесплатных решений (VS Code) подтверждает общий тренд на снижение лицензионных затрат.
- Системы контроля версий и коллаборации: Git как стандарт де-факто, с GitHub как самой желаемой платформой для совместной работы (желаема для 59.3% против 25.6% для GitLab и 22% для Jira) [10].
- Языки программирования и инфраструктура: Широкое распространение Python (57.9%), JavaScript (66%), а также инструментов контейнеризации и оркестрации (Kubernetes, Docker), что отражает переход к микросервисным и облачным архитектурам.

Эта стандартизированная экосистема снижает порог входа в проекты и формирует единую технологическую основу как для коммерческих, так и для open-source команд.

1.2.3 Типовое рабочее место и операционные затраты

Типовая рабочая станция для разработки программного обеспечения представляет собой высокопроизводительный компьютер, характеристики которого (производительность процессора, объём оперативной памяти, скорость накопителя) определяются необходимостью одновременной работы со средами разработки (IDE), виртуальными машинами, контейнерами и инструментами сборки проектов. Такое рабочее место требует значительных единовременных капитальных вложений в МТБ. Это соответствует общей тенденции роста инвестиций в основной капитал ИТ-отрасли, которая, согласно данным НИУ ВШЭ, в 2023 году составляла десятки миллиардов рублей и демонстрировала высокие темпы прироста [11].

К операционным затратам (OPEX), связанным с МТБ, помимо амортизации, относятся: аренда помещений и облачных мощностей (IaaS, PaaS), оплата лицензий на проприетарное ПО (где оно применяется), расходы на электроэнергию и высокоскоростной доступ в интернет, что является критическим ресурсом для распределённых команд. Данный состав затрат характерен для управленческого учёта в ИТ-проектах и соответствует принципам классификации и калькулирования, описанным в литературе по предмету [12].

1.2.4 Специфика материально-технической базы для open-source проектов

МТБ проектов, разрабатываемых в парадигме открытого кода, имеет существенные отличия, вытекающие из их философских основ [3] и экономических моделей.

- Смещение затрат с CAPEX на OPEX: Максимальное использование бесплатных open-source инструментов (Linux, Git, VS Code, GCC) и публичной облачной инфраструктуры (GitHub Actions, GitLab CI) минимизирует первоначальные капитальные вложения (CAPEX) в лицензионное ПО, но может увеличивать операционные расходы на облачные сервисы.
- Распределённая и удалённая инфраструктура: Модель «базара» по Реймонду изначально предполагает географическую распределённость команды. Это подтверждается данными Stack Overflow: 32.4% разработчиков работают полностью удалённо, а ещё 29.5% — в гибридном формате [10]. МТБ такой команды — это совокупность личных или корпоративных устройств разработчиков, связанных через интернет, а не централизованный офис.
- Сообщество как расширенная МТБ: В open-source проектах тестирование на различных аппаратных и программных конфигурациях часто осуществляется добровольцами из сообщества, что, как отмечалось в Census III, особенно важно для обеспечения совместимости и безопасности [2]. Таким образом, сообщество выступает как «виртуальный» и масштабируемый тестовый полигон.
- Приоритет экосистемы над отдельным инструментом: Успех проекта зависит от интеграции в существующие экосистемы пакетов (npm, PyPI, Crates.io). Census III отмечает растущую зависимость от облачно-специфичных пакетов и компонентов, написанных на memory-safe языках, таких как Rust [2].

На основании проведённого анализа структуры МТБ, данных об инструментах разработки [10] и специфики open-source проектов [2; 3] можно сформулировать следующие сравнительные характеристики:

Таблица 2: Сравнительная характеристика МТБ коммерческой и open-source разработки

Компонент МТБ	Традиционная коммерческая разработка	Типичный open-source проект
Лицензии на инструменты	Крупные статьи CAPEX (проприетарные IDE, ОС)	Минимальные (доминируют бесплатные OSS-инструменты)
Инфраструктура разработки	Корпоративные серверы, выделенные линии	Публичные облачные платформы (GitHub, GitLab), интернет
Рабочее место	Стандартизированная офисная рабочая станция	Личное устройство разработчика, удалённый доступ

Таблица 2 (продолжение)

Компонент МТБ	Традиционная коммерческая разработка	Типичный open-source проект
Тестовые среды	Выделенный парк устройств, стенды	Добровольное сообщество, эмуляторы, limited CI/CD
Ключевые затраты	CAPEX (оборудование, лицензии), OPEX (аренда, зарплата)	OPEX (облачные сервисы, хостинг), альтернативная стоимость времени

Таким образом, материально-техническая база open-source проектов характеризуется высокой степенью виртуализации, зависимостью от публичных экосистем и смещением финансовой модели с прямых капитальных вложений на операционные расходы и нематериальные ресурсы сообщества. Это необходимо учитывать при построении методики экономического обоснования подобных проектов.

1.3 Методики экономической оценки разработки программного обеспечения

Данный подраздел посвящён анализу классических и современных методик оценки трудоёмкости и стоимости создания программного обеспечения. Понимание этих методик является теоретическим фундаментом для последующего практического расчёта экономических показателей разработки open-source архиватора.

1.3.1 Классические модели оценки трудоёмкости: COCOMO и функциональные точки

Проблема прогнозирования затрат на разработку ПО является одной из центральных в экономике программной инженерии. Исторически сложилось два основных подхода: параметрическое моделирование, основанное на метриках исходного кода, и функционально-ориентированное оценивание.

Наиболее известной параметрической моделью является COCOMO (Constructive Cost Model), разработанная Барри Боэмом. Её базовая форма (COCOMO I) устанавливает зависимость между объёмом кода в тысячах строк (KSLOC) и требуемыми для разработки трудозатратами (в человеко-месяцах) [13, с. 87]. Формула модели имеет вид:

$$\text{PM} = a \times (\text{KSLOC})^b \times \prod_{i=1}^{n} EMF_i, \tag{1}$$

где:

PM (Person-Month) – оценка трудозатрат;

a, b – эмпирические коэффициенты, зависящие от типа проекта (органический, полунезависимый, встроенный);

EMF (Effort Multiplier Factor) – поправочные коэффициенты, учитывающие атрибуты проекта (надёжность, опыт команды, современность инструментов и др.).

Несмотря на свою структурированность, прямое применение COCOMO к небольшим open-source проектам затруднено из-за неявного учёта таких факторов, как волонтёрский труд и отсутствие формальных процессов.

Альтернативой является метод функциональных точек (Function Point Analysis, FPA IFPUG). Вместо строк кода он оценивает объём функциональности, предоставляемой пользователю, через пять типов компонентов: входы, выходы, запросы, файлы и интерфейсы [13, с. 86]. Метод лучше подходит для проектов с высокоуровневыми требованиями и менее зависит от реализации, однако требует высокой квалификации оценщика и также ориентирован на коммерческую разработку с чётко определёнными границами проекта.

1.3.2 Структура затрат и калькуляция себестоимости разработки ПО

Экономическая оценка проекта не ограничивается трудозатратами; она требует учёта полной структуры затрат. В общем виде себестоимость разработки программного продукта (C) можно представить как сумму

следующих статей [13, с. 36-41]:

$$C = C_{\text{ФОТ}} + C_{\text{соц}} + C_{\text{аморт}} + C_{\text{наклад}} + C_{\text{проч}}, \tag{2}$$

где:

$C_{\text{ФОТ}}$ – фонд оплаты труда ключевого персонала (аналитиков, разработчиков, тестировщиков);

$C_{\text{соц}}$ – страховые взносы и иные обязательные отчисления от ФОТ;

$C_{\text{аморт}}$ – амортизация оборудования и нематериальных активов (лицензий);

$C_{\text{наклад}}$ – накладные расходы (аренда, коммунальные услуги, административный персонал);

$C_{\text{проч}}$ – прочие прямые затраты (лицензии на инструменты, облачные услуги).

Для коммерческого проекта расчёт каждой статьи является обязательным. Однако, как показано в подразделе 1.2, для open-source проектов характерно смещение структуры: затраты на лицензионное ПО ($C_{\text{проч}}$) стремятся к нулю, амортизация ($C_{\text{аморт}}$) часто не учитывается при использовании личного оборудования, а накладные расходы ($C_{\text{наклад}}$) минимизируются за счёт удалённой работы [10]. Это приводит к тому, что основной статьёй затрат в открытой разработке становится альтернативная стоимость времени разработчиков ($C_{\text{ФОТ}}$), которая в случае волонтёрского участия не имеет прямого денежного выражения.

1.3.3 Специфика экономической оценки open-source проектов и существующие пробелы

Экономика свободного и открытого ПО строится на принципиально иных, по сравнению с проприетарной моделью, основаниях [13, с. 99]. Если классическая модель ориентирована на максимизацию прибыли от продажи лицензий, то open-source проекты часто следуют моделям, основанным на предоставлении сопутствующих платных услуг (поддержка, кастомизация, хостинг), продаже branded-версий или получении грантов.

Классические методики, такие как COCOMO и FPA, создавались для «соборной» модели разработки и не учитывают ключевые особенности «базарной» модели [3]:

1. Асинхронный и распределённый вклад: Трудозатраты складываются из неравномерных усилий множества независимых контрибьюторов, что делает оценку в «человеко-месяцах» некорректной.
2. Нематериальная мотивация: Весомую часть стоимости составляет мотивация, основанная на репутации, обучении или идеализме («egoboo»), которая не конвертируется напрямую в денежный эквивалент.
3. Экосистемная зависимость: Стоимость проекта резко снижается за счёт повторного использования существующих открытых компонентов, что отражено в данных Census III о повсеместном использовании языковых пакетов [2]. Однако это создаёт скрытые затраты на поддержание совместимости и безопасность зависимостей.
4. Парадокс «критического ресурса»: Как показано в Census III, многие жизненно важные для инфраструктуры проекты поддерживаются 1-2 разработчиками [2]. С точки зрения классической оценки это «низкозатратный» проект, но с макроэкономической позиции — актив с высокой ценностью и высокими рисками.

Таким образом, в научной и методической литературе существует пробел: отсутствует адаптированная методика, которая бы, с одной стороны, использовала формальный аппарат классических моделей (например, для оценки внутренней сложности алгоритмов), а с другой — адекватно учитывала специфику открытой разработки: распределённость, нематериальную мотивацию, экосистемную интеграцию и парадоксальное соотношение затрат и общественной ценности.

1.3.4 Выводы по теоретической части и обоснование методики для практического раздела

Проведённый анализ позволяет сформулировать следующие выводы, являющиеся основой для практического расчёта в Главе 2:

1. Для оценки алгоритмической сложности разрабатываемого архиватора возможно использование аппарата параметрических моделей (адаптированной COCOMO) на этапе проектирования.

2. Структура затрат для open-source проекта будет кардинально отличаться от коммерческого: необходимо рассчитать два сценария — «коммерческий» (полная калькуляция по формуле 2) и «реальный open-source» (учёт только прямых материальных затрат и альтернативной стоимости времени).
3. Ключевым экономическим показателем для подобного проекта является не прямая прибыль, а соотношение общественной полезности (ценности) к понесённым затратам. Это требует введения в анализ качественных и косвенных количественных метрик.
4. Предлагаемая в работе методика должна быть гибридной: сочетать формальный расчёт для частей проекта, поддающихся оценке (трудозатраты на реализацию ядра), и экспертную оценку для специфических аспектов open-source (стоимость поддержки сообщества, ценность портфолио).

Данный теоретический базис позволяет перейти к практической части работы — экономическому обоснованию разработки конкретного программного продукта с открытым исходным кодом.

2 Практическая часть

... PLACEHOLDER ...

3 Заключение

В ходе выполнения курсовой работы была настроена рабочая среда для подготовки документов в системе LaTeX.

Был создан структурированный шаблон курсовой работы, включающий основные разделы, что позволяет в дальнейшем сосредоточиться непосредственно на написании содержательной части работы.

Использование LaTeX в процессе подготовки учебных и научных работ способствует повышению качества оформления документов и упрощает работу с большими объёмами текста.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *European Commission, Directorate-General for Communications Networks, Content and Technology*. Study about the impact of open source software and hardware on technological independence, competitiveness and innovation in the EU economy. — 2021. — URL: https://digital-strategy.ec.europa.eu/en/library/study-about-impact-open-source-software-and-hardware-technological-independence-competitiveness-and (visited on 02/27/2025) ; Final report. Luxembourg. DOI: 10.2759/430161.
2. Census III of Free and Open Source Software — Application Libraries / F. Nagle [et al.] ; The Linux Foundation. — 12/2024. — URL: https://www.linuxfoundation.org/hubfs/LF%20Research/lfr_censusiii_120424a.pdf (visited on 02/25/2025) ; Отчёт подготовлен совместно с Harvard Business School.
3. *Raymond E. S.* The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. — O'Reilly & Associates, 2001. — ISBN 978-0-596-00108-7.
4. Обеспечение систем обработки информации программное. Термины и определения. — Государственный стандарт СССР, 1990. — Взамен ГОСТ 19781-83 и ГОСТ 19.004-80. Дата введения 01.01.92.
5. *Free Software Foundation*. The Free Software Definition. — 2024. — URL: https://www.gnu.org/philosophy/free-sw.html (visited on 02/28/2025) ; Определение свободного программного обеспечения от Free Software Foundation.
6. *Open Source Initiative*. The Open Source Definition. — 2024. — URL: https://opensource.org/osd (visited on 02/27/2025) ; Последнее изменение: 16 февраля 2024 г.
7. *Правительство Российской Федерации*. Постановление Правительства Российской Федерации от 1 января 2002 г. № 1 «О Классификации основных средств, включаемых в амортизационные группы». — 2002. — URL: https://www.consultant.ru/document/cons_doc_LAW_34710/f8649e13eaeec3b1984402f1bee3 (дата обр. 22.03.2025) ; В редакции от 18.11.2022.
8. *Министерство финансов Российской Федерации*. Федеральный стандарт бухгалтерского учёта ФСБУ 6/2020 «Основные средства». — 2020. — URL: https://minfin.gov.ru/ru/document?id_4=133537 (дата обр. 22.03.2025) ; Утверждён Приказом Минфина России от 17.09.2020 № 204н.
9. *Российская Федерация*. Налоговый кодекс Российской Федерации (часть вторая). Глава 25. Статья 259.3. Применение повышающих (понижающих) коэффициентов к норме амортизации. — 2000. — URL: https://www.consultant.ru/document/cons_doc_LAW_28165/8f5e360a53d29554be20fe46a6b79f85e5bbbd0d/ (дата обр. 22.03.2025) ; Принят Государственной Думой 19 июля 2000 года.
10. *Stack Overflow*. Stack Overflow Developer Survey 2025. — 2025. — URL: https://survey.stackoverflow.co/2025/ (visited on 02/28/2025) ; Ежегодный опрос разработчиков о технологиях, рабочих практиках и предпочтениях.
11. *НИУ ВШЭ, Институт статистических исследований и экономики знаний*. Российский сектор ИКТ: ключевые показатели. Январь-сентябрь 2023. Квартальный дайджест на основе официальной статистической информации / Национальный исследовательский университет «Высшая школа экономики». — 2024. — URL: https://issek.hse.ru/mirror/pubs/share/898604421.pdf (дата обр. 22.03.2025) ; DOI: 10.17323/ICT_Sector_2023_I-IIIQ.
12. *Исаев Д. В.*, *Кравченко Т. К.* Информационные технологии управленческого учета. — Москва : Государственный университет - Высшая школа экономики, 2006. — 291 с. — URL: https://www.hse.ru/data/929/290/1238/%D0%98%D1%81%D0%B0%D0%B5%D0%B2%D0%94%D0%92%20-%20%D0%98%D0%A1%20%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B3%D0%BE%20%D1%83%D1%87%D0%B5%D1%82%D0%B0.pdf.

13. Экономика программной инженерии: учебное пособие / под ред. А. М. Полянский, Д. В. Кочкин. — Вологда : Вологодский государственный университет, 2017. — URL: https://dokumen.pub/0c5f8b2c644f8cf8b027b75html (дата обр. 01.03.2025) ; Для подготовки бакалавров по направлению 09.03.04 «Программная инженерия».

ПРИЛОЖЕНИЕ А

Соответствие оформления работы требованиям методических рекомендаций

Требование методических рекомендаций	**Параметр в данной работе**
Шрифт: Times New Roman, 14 пт	Шрифт: Times New Roman, 14 pt
Межстрочный интервал: 1.5	Интервал: 1.5
Поля: левое — 30 мм, остальные — 15 мм	Поля: left=30mm, right=15mm, top=20mm, bottom=20mm
Абзацный отступ: 1.25 см	`\parindent = 1.25cm`
Нумерация страниц: арабские цифры, внизу по центру	`\pagestyle{fancy}`, `\fancyfoot[C]{\thepage}`
Заголовки разделов: прописные, полужирные, по центру	`\section{...}` с соответствующим оформлением

Таблица 3: Соответствие параметров оформления

Данная работа подготовлена в системе компьютерной вёрстки LaTeX, которая гарантирует точное и неизменное соблюдение заданных параметров оформления на протяжении всего документа.