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. Теоретические основы разработки программного продукта с открытым исходным кодом
3. Расчёт экономических показателей разработки программного продукта с открытым исходным кодом
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 архиватор/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: Применимы ли классические методики экономической оценки (CAPEX/OPEX, ROI, точка безубыточности) для малых open-source проектов, и если нет, то как их адаптировать или чем дополнить или заменить?

Цель работы — проверить применимость классической методики экономического обоснования к проекту разработки open-source архиватора, выявить её ограничения и предложить адаптированный подход для оценки подобных проектов.

Для достижения цели поставлены следующие задачи:

1. Изучить теоретические основы экономики ПО и специфику open-source разработки.
2. Сформулировать кейс проекта, провести оценку трудозатрат и собрать исходные данные для расчёта по классической методике.
3. Выполнить формальный расчёт по классической методике (CAPEX, OPEX, выручка, точка безубыточности, ROI).
4. Проанализировать полученные результаты, выявив противоречия между расчётными показателями и реальной ценностью open-source проектов.
5. На основе анализа предложить адаптированный подход или комплекс рекомендаций для экономической оценки малых open-source проектов.

Объект исследования — процесс экономического обоснования разработки программного обеспечения с открытым исходным кодом. Предмет исследования — классические методики экономической оценки ПО и их применимость к условиям open-source разработки на примере проекта консольного архиватора.

Теоретическая значимость работы заключается в критическом анализе границ применимости классических экономических моделей (COCOMO, расчёт ROI) к open-source парадигме и в формулировке направлений для их развития.

Практическая значимость состоит в том, что работа:

- Предоставляет реалистичный кейс с полным расчётом, наглядно демонстрирующий, почему стандартные модели дают негативную оценку open-source проекту.
- Систематизирует актуальные модели монетизации open-source, предоставляя разработчикам и менеджерам структурированный обзор возможностей.
- Формулирует практические рекомендации о том, какие метрики и подходы (помимо прямых финансовых) следует учитывать при оценке целесообразности участия в open-source проектах.

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 содержательный 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

Таблица 2 (продолжение)

Компонент МТБ	Традиционная коммерческая разработка	Типичный open-source проект
Рабочее место	Стандартизированная офисная рабочая станция	Личное устройство разработчика, удалённый доступ
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, к ак 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.

1.4 Модели монетизации программного обеспечения с открытым исходным кодом

Классическая коммерческая модель, основанная на продаже лицензий на использование ПО, противоречит философии open-source. В результате сформировались специфические бизнес-модели, позволяющие извлекать экономическую выгоду, сохраняя код открытым [15].

1. Модель Open Core (Открытое ядро): Базовый функционал распространяется свободно, а расширенные функции (безопасность, управление, интеграции) — в проприетарной «корпоративной» версии. Приме-

ры: GitLab, Redis, Elasticsearch. Это позволяет финансировать развитие ядра за счёт премиум-продаж.

2. Модель SaaS / Хостинг (ПО как услуга): Пользователям предлагается не дистрибутив, а готовый облачный сервис на основе open-source кода. Примеры: WordPress.com, GitLab.com. Поток выручки становится регулярным (подписка).
3. Модель профессиональных услуг: Заработок идёт от сопутствующих услуг: технической поддержки, консалтинга, кастомизации, обучения. Пример: Red Hat (до перехода к модели подписки).
4. Краудфандинг и спонсорство: Финансирование от сообщества пользователей и компаний-бенефициаров через платформы (Open Collective, GitHub Sponsors). Пример: Vue.js, Laravel.
5. Двойное лицензирование: Один и тот же код доступен под двумя лицензиями: открытой (GPL) и коммерческой. Компании, не желающие соблюдать строгие условия GPL, покупают коммерческую лицензию. Пример: MySQL (ранее).

Данные модели решают ключевое противоречие: они отделяют источник финансирования (деньги) от объекта финансирования (открытый код). Для целей экономического обоснования это означает, что графа «Выручка» в расчёте должна отражать не продажу *исходного кода архиватора/компрессора*, а поступления от одной или нескольких перечисленных выше моделей.

1.4.1 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 Расчёт экономических показателей разработки программного продукта с открытым исходным кодом

2.1 Основные характеристики разрабатываемого программного продукта

2.1.1 Формулировка проблемы и постановка целей разработки

Несмотря на обилие существующих решений для сжатия данных, анализ современного open-source ландшафта, проведённый в отчёте Linux Foundation (Census III), выявляет системную проблему: многие критически важные проекты, включая базовые утилиты, десятилетиями поддерживаются силами 1-2 разработчиков, что создаёт риски для безопасности и устойчивости цифровой инфраструктуры [2]. Парадокс заключается в том, что при высокой общественной ценности такие проекты не имеют прозрачных экономических моделей, позволяющих оценить реальную стоимость их создания и поддержки.

Таким образом, проблема заключается не в отсутствии функционального аналога, а в недостатке методик для экономического обоснования разработки и поддержки малых, но инфраструктурно значимых open-source проектов. Данная работа рассматривает создание нового консольного архиватора не как цель саму по себе, а как практический кейс для апробации такой методики.

Целью практической части работы является разработка и апробация методики экономического обоснования для малого open-source проекта на примере создания консольного архиватора. Проект следует критериям SMART:

- Specific (Конкретная): Разработать консольную утилиту, реализующую алгоритм DEFLATE для сжатия/распаковки отдельных файлов.
- Measurable (Измеримая): Достичь степени сжатия, сопоставимой с ‘gzip -6‘, для эталонного набора данных.
- Achievable (Достижимая): Реализовать силами одного разработчика средней квалификации за 3 месяца.
- Relevant (Значимая): Создать открытый кейс для отработки методики расчёта, актуальной в условиях, описанных Census III.
- Time-bound (Ограниченная по времени): Срок полной разработки MVP — 3 календарных месяца.

2.1.2 Анализ целевой аудитории и сценариев использования

Разрабатываемый продукт позиционируется как учебно-демонстрационный, но его целевая аудитория отражает реальные сегменты потребителей подобных утилит:

1. Студенты и начинающие разработчики: Для изучения основ алгоритмов сжатия, работы с файловыми системами и практики разработки на Си/C++.
2. Системные администраторы и DevOps-инженеры: Для автоматизации резервного копирования и логменеджмента в средах, где предъявляются требования к лицензионной чистоте и минимализму зависимостей.
3. Сообщество open-source: Как потенциальная основа для форка или экспериментальной площадки для тестирования новых модификаций алгоритмов сжатия.

2.1.3 Формирование требований к минимально жизнеспособному продукту (MVP)

Минимально жизнеспособная версия продукта (MVP) включает следующий функционал, структурированный по модулям:

- Модуль ввода/вывода: Чтение и запись данных из файлов, передача данных через стандартные потоки (stdin/stdout).
- Модуль алгоритма сжатия: Реализация алгоритма DEFLATE (LZ77 + кодирование Хаффмана) с одним предустановленным уровнем сжатия.

- Модуль интерфейса командной строки (CLI): Обработка аргументов для выбора режима работы (сжатие/распаковка), указания входного и выходного файлов.
- Модуль тестирования: Набор unit-тестов для проверки корректности работы ядра алгоритма.

Архитектура проекта сознательно упрощена по сравнению с промышленными аналогами (такими как ‘bzip2‘ или ‘xz‘) и ориентирована на наглядность реализации и лёгкость оценки трудозатрат.

2.1.4 Верификация оценки трудозатрат на основе анализа исторических данных аналогов

Для обоснования реалистичности планируемых трудозатрат был проведён самостоятельный количественный анализ истории разработки трёх классических open-source архиваторов. Методология заключалась в сборе и обработке данных из публичных git-репозиториев с использованием скриптов на основе команд ‘git log‘ и ‘git diff‘ [14].

Таблица 3: Результаты сравнительного анализа истории разработки open-source архиваторов (по данным git-репозиториев)

Критерий	`bzip2`	`xz utils`	`zstd`
Период анализа (гг.)	1997–2023	2009–2023	2015–2023
Общее число коммитов	180	> 1000	> 3000
Ориентировочный чистый прирост строк кода*	≈ 16 600	Значительный	Наибольший
Выявленная модель разработки	Индивидуальная, с переходом к поддержке сообществом	Распределённая с выделенным мейнтейнером	Промышленная, командная
Оценка релевантности как аналога	Высокая (базовый кейс)	Умеренная (верхняя граница сложности)	Низкая (промышленный масштаб)

*Расчёт выполнен автором на основе агрегированной статистики git-репозиториев.

Анализ показал, что проект `bzip2`, являющийся полнофункциональным архиватором, был создан и длительно поддерживался силами, эквивалентными работе одного разработчика. Его итоговый объём кода (≈ 16.6 тыс. строк) и история коммитов позволяют сделать вывод о реалистичности разработки аналогичного по масштабу, но более простого (использующего стандартный DEFLATE) продукта в сжатые сроки.

2.1.5 Оценка трудозатрат и ресурсов для разработки

На основе проведённого сравнительного анализа, а также с учётом принципов декомпозиции работ по методологии, изложенной в [13, с. 86–90], составлена детальная оценка трудозатрат. Для расчёта фонда оплаты труда (ФОТ) принята рыночная ставка разработчика средней квалификации (С-программист/инженер ПО) в размере 1 200 руб./час, что соответствует данным по региональному рынку труда на 2024–2025 гг.

Таблица 4: Оценка трудозатрат на разработку программного продукта

Этап разработки	Трудозатраты, час	Ставка, руб./час	Стоимость, руб.
Анализ требований и проектирование архитектуры	40	1 200	48 000
Программирование (реализация модулей)	100	1 200	120 000
Тестирование и отладка	40	1 200	48 000
Написание документации и подготовка релиза	20	1 200	24 000
Итого по ФОТ	200		240 000

Таким образом, общая оценка трудозатрат на создание MVP консольного архиватора составляет 200 человеко-часов, а фонд оплаты труда (ФОТ) — 240 000 рублей. Данная оценка является консервативной и опирается на анализ наиболее релевантного аналога (`bzip2`), что обеспечивает запас при планировании и соответствует принципам реалистичного прогнозирования в условиях малого open-source проекта.

2.2 Расчёт экономических показателей разработки программного продукта

2.2.1 Расчёт капитальных затрат (CAPEX)

Капитальные затраты (CAPEX) включают все единовременные инвестиции, необходимые для создания программного продукта. Для проекта, использующего исключительно открытые инструменты, структура CAPEX существенно отличается от типичной коммерческой разработки.

Таблица 5: Структура капитальных затрат (CAPEX)

Статья расходов	Основание для расчёта	Сумма, руб.
Фонд оплаты труда (ФОТ)	Данные из Таблицы 4	240 000
Лицензионное ПО и инструменты	Используется только open-source стек (NeoVim, GCC, Git, Forgejo)	0
Оборудование (амортизация)	Ноутбук разработчика (80 000 руб.), срок службы 3 года, период разработки 3 мес. (0.25 года). Амортизация: $\frac{80\,000}{3} \times 0.25 = 6\,667$	6 667
Накладные расходы	15% от ФОТ (аренда рабочего места дома, интернет, электричество) $240\,000 \times 0.15 = 36\,000$	36 000
Резервный фонд	10% от суммы прямых затрат (ФОТ + Оборудование) на непредвиденные расходы $(240\,000 + 6\,667) \times 0.10 = 24\,667$	24 667
ИТОГО CAPEX		307 334

Таким образом, общий объём капитальных вложений, необходимых для запуска проекта, составляет 307 334 рубля. Критически важным отличием от коммерческого сценария является нулевая стоимость лицензионного ПО, что напрямую вытекает из философии открытого кода и является ключевым фактором снижения первоначального барьера для входа [3].

2.2.2 Расчёт операционных затрат (OPEX)

Операционные затраты (OPEX) — это ежегодные расходы на поддержку и эксплуатацию работающего программного продукта после его релиза.

Таблица 6: Структура годовых операционных затрат (OPEX)

Статья расходов	Метод расчёта	Сумма, руб./год
Хостинг и инфраструктура	Использование бесплатного тарифа публичной платформы (GitHub, GitLab). Для сценария с полным контролем: аренда VPS 500 руб./мес. [15]	0 (6 000)*
Техническая поддержка	0.25 ставки инженера (реакция на issues, правка документации). З/п: 120 000 руб./мес. $120\,000 \times 0.25 \times 12 \times 1.3$ (с учётом страховых взносов 30%)	468 000
Маркетинг и продвижение	Минимальный бюджет (публикация на форумах, базовое SEO)	5 000
Обновление лицензий	Все инструменты и зависимости — open-source, обновления бесплатны	0
ИТОГО OPEX		473 000 (479 000)*

*В расчётах используется вариант с бесплатным хостингом как наиболее типичный для малых проектов [2]. Стоимость VPS указана для справки.

Основной статьёй OPEX является техническая поддержка. В модели open-source проекта [3] часть этой работы может выполняться сообществом, однако для гарантированного поддержания проекта в работоспособном состоянии в расчёт заложены минимальные затраты на частичную занятость разработчика.

2.2.3 Прогнозирование выручки от внедрения продукта

Для open-source проекта, распространяемого под лицензией GPL, традиционная модель монетизации через продажу лицензий неприменима. В качестве основной модели принята модель платной поддержки и консалтинга для корпоративных пользователей, желающих гарантировать работоспособность и безопасность утилиты в своей инфраструктуре.

Таблица 7: Данные для прогнозирования годовой выручки

Параметр	Обозначение	Значение
Общий размер целевой аудитории (ИТ-компании малого и среднего размера в РФ)	N_{total}	5 000
Конверсия в установку/пользование продуктом	$C_{install}$	2%
Конверсия пользователей в клиентов платной поддержки	C_{active}	5%
Среднее количество контрактов в год на пользователя	Q	1
Средний годовой контракт на поддержку	P_{avg}	25 000 руб.

На основе этих параметров рассчитывается прогнозный денежный поток (Таблица 8).

Таблица 8: Расчётный денежный поток от эксплуатации

Показатель	Формула расчёта	Значение
Активные пользователи в месяц (компании)	$U = N_{total} \cdot C_{install}$	$5\,000 \cdot 0.02 = 100$
Клиенты платной поддержки в год	$U_{.} = U \cdot C_{active}$	$100 \cdot 0.05 = 5$
Месячная выручка	$V = U_{.} \cdot Q \cdot \frac{P_{avg}}{12}$	$5 \cdot 1 \cdot \frac{25\,000}{12} \approx 10\,417$
Годовая выручка	$V = V \cdot 12$	125 000 руб.

Прогнозируемая годовая выручка составляет 125 000 рублей. Данная цифра консервативна и отражает реалии нишевого open-source продукта, где монетизация является сложной задачей и часто не является основной целью разработки [2].

2.2.4 Расчёт прибыли и рентабельности

На основе прогноза выручки и затрат формируется прогнозный отчёт о прибылях и убытках (Таблица 9).

Таблица 9: Прогнозный отчёт о прибылях и убытках (годовой)

Показатель	Формула расчёта	Значение, руб.
Выручка (Revenue)	V (из Табл. 8)	125 000
Переменные расходы (комиссии платёжных систем 5%)	$C_{var} = V \cdot 0.05$	6 250
Валовая прибыль (Gross Profit)	$GP = V - C_{var}$	118 750
Операционные расходы (OPEX)	C_{opex} (из Табл. 6)	473 000
Прибыль до налогообложения (EBIT)	$EBIT = GP - C_{opex}$	-354 250
Налог на прибыль (20%)*	$Tax = 0$ (при убытке)	0
Чистая прибыль (Net Profit)	$NP = EBIT - Tax$	-354 250

*При расчёте налога на прибыль учитывается, что налогооблагаемая база не может быть отрицательной.

Как видно из расчётов, проект является убыточным при рассмотрении исключительно прямых финансовых потоков. Годовой убыток составляет 354 250 рублей. Рентабельность продаж (ROS) отрицательна. Данный результат является типичным для многих open-source проектов, ценность которых заключается не в прямой монетизации, а в создании общественного блага, построении репутации, портфолио разработчика и косвенных экономических эффектах [1].

2.2.5 Адаптация методики для индивидуальной open-source разработки

Приведённый выше расчёт соответствует формальному «коммерческому» сценарию. Однако для индивидуального разработчика или малой команды энтузиастов экономическая модель кардинально меняется.

Для индивидуального разработчика основными «доходами» являются:

- Накопление человеческого капитала: Приобретённые навыки (оптимизация C, работа с алгоритмами) эквивалентны прохождению углублённого курса стоимостью 50 000–100 000 руб.
- Укрепление репутации в сообществе: Наличие успешного open-source проекта повышает стоимость часа будущей работы разработчика.

Таблица 10: Сравнение экономических моделей разработки

Показатель	Формальный коммерческий сценарий	Реальный индивидуальный сценарий
CAPEX	307 334 руб. (полный расчёт)	8 000 руб. (амортизация ПК + энергия)
ФОТ	240 000 руб. (рыночная ставка)	0 руб. (альтернативная стоимость времени, труд добровольца)
OPEX	473 000 руб./год (включая поддержку)	0 руб./год (поддержка по мере возможностей)
Выручка	125 000 руб./год (прогноз)	0 – 50 000 руб./год (нерегулярные донаты)
Прибыль	-354 250 руб./год (убыток)	Нематериальная выгода (портфолио, знания, репутация)

- Социальная польза: Вклад в экосистему свободного ПО, которой, согласно исследованиям, пользуются миллионы [2].

Таким образом, экономическая целесообразность индивидуальной open-source разработки оценивается не через призму прямых финансовых результатов, а через анализ долгосрочных нематериальных выгод и альтернативных издержек.

2.2.6 Выводы по подразделу 2.2

Расчёт экономических показателей разработки консольного архиватора с открытым исходным кодом показал:

1. Структура затрат для open-source проекта радикально смещена: CAPEX минимизирован за счёт бесплатных инструментов, основную долю в нём составляет ФОТ.
2. В рамках традиционной финансовой модели проект является убыточным (чистый убыток 354 тыс. руб. в год), что является нормальным для open-source продуктов, не ориентированных на прямую монетизацию.
3. Ключевой экономический парадокс open-source заключается в несоответствии высокой общественной полезности проекта и его низкой или отрицательной коммерческой рентабельности.
4. Для индивидуального разработчика методика расчёта должна быть адаптирована: вместо финансовых потоков на первый план выходит оценка альтернативной стоимости времени и нематериальных выгод (портфолио, репутация, человеческий капитал).

Полученные цифры служат основой для финального анализа эффективности проекта в следующем подразделе.

2.3 Анализ эффективности внедрения программного продукта

2.3.1 Расчёт точки безубыточности (ТБУ)

Точка безубыточности определяет минимальный объём продаж, необходимый для покрытия операционных затрат. Для проекта с выручкой от платной поддержки она рассчитывается по формуле:

$$Q_{\text{ТБУ}} = \frac{C_{\text{пост}}}{P_{\text{ед}} - C_{\text{пер.ед}}}, \tag{3}$$

где $C_{\text{пост}}$ — постоянные затраты (OPEX), $P_{\text{ед}}$ — цена контракта поддержки, $C_{\text{пер.ед}}$ — переменные затраты на единицу.

Таблица 11: Исходные данные для расчёта точки безубыточности

Параметр	Обозначение	Значение
Постоянные затраты (OPEX)	$C_{\text{пост}}$	473 000 руб./год (из Табл. 6)
Цена за единицу (средний контракт)	$P_{\text{ед}}$	25 000 руб. (из Табл. 7)
Переменные затраты на единицу (5% комиссия)	$C_{\text{пер.ед}}$	1 250 руб.

Подставив значения, получаем: $Q_{\text{ТБУ}} = \frac{473\,000}{25\,000-1\,250} \approx 20$ контрактов в год. В денежном выражении: $V_{\text{ТБУ}} = 20 \times 25\,000 = 500\,000$ руб./год.

Вывод 1: Для покрытия затрат проект должен заключать не менее 20 контрактов платной поддержки в год. При прогнозе в 5 контрактов (Таблица 8) проект заведомо убыточен в рамках данной модели.

2.3.2 Расчёт срока окупаемости и возврата на инвестиций (ROI)

Срок окупаемости (PP) и возврат на инвестиции (ROI) рассчитываются для капитальных затрат (CAPEX).

При отрицательной прибыли срок окупаемости формально стремится к бесконечности: $PP = \frac{307\,334}{-354\,250} \approx -0.87$ (не окупается).

Таблица 12: Данные для расчёта срока окупаемости

Параметр	Обозначение	Значение
Капитальные затраты	CAPEX	307 334 руб. (из Табл. 5)
Чистая прибыль в год	$P_{\text{чист.год}}$	-354 250 руб./год (из Табл. 9)

Рентабельность инвестиций (ROI) также отрицательна:

$$ROI = \frac{P_{\text{чист.год}} - \text{CAPEX}}{\text{CAPEX}} \times 100\% = \frac{-354\,250 - 307\,334}{307\,334} \times 100\% \approx -215\%. \quad (4)$$

Вывод 2: С точки зрения классического финансового анализа, проект абсолютно неэффективен: CAPEX не окупается, на каждый вложенный рубль приходится 2.15 рубля убытка.

2.3.3 Критический анализ результатов и адаптация методики оценки для open-source

Полученные отрицательные показатели являются не ошибкой расчёта, а отражением фундаментального противоречия, описанного в исследованиях: попытки оценить open-source исключительно через прямые финансовые потоки заведомо ведут к отрицательным выводам, так как не учитывают природу его ценности [16].

Ответ на вопрос курсовой работы формулируется в два этапа:

1. Можно ли получить деньги с open-source проекта? Да, но не через прямую продажу лицензий на ядро проекта. Анализ практик монетизации показывает, что доход генерируется через сопутствующие услуги и модели [15]:

- Платная поддержка и консалтинг (реализовано в нашей модели).
- Модель Open Core: бесплатное ядро (GPL) + платные проприетарные дополнения (например, графический интерфейс, облачная синхронизация).
- SaaS (ПО как услуга): хостинг управляемой версии архиватора в облаке.
- Краудфандинг и спонсорство (через Open Collective, GitHub Sponsors).

2. Как рассчитать рентабельность open-source проекта? Классический расчёт ROI неприменим. Необходима адаптированная методика, включающая:

1. Расчёт «затратной» компоненты: По классической схеме (CAPEX, OPEX), как было проделано в п. 2.2.
2. Оценка «доходной» компоненты: Не как прогноз продаж, а как многокритериальная модель, учитывающая:
   - Денежные потоки от гибридных моделей (Open Core, SaaS).
   - Экономию на рекламе и найме за счёт репутации.
   - Накопление человеческого капитала (рост стоимости часа разработчика).
   - Социальный и экосистемный вклад, который, как показывают макроэкономические исследования, может в десятки раз превышать прямые затраты на разработку [16].
3. Расчёт «скорректированной рентабельности»: Сопоставление совокупности выгод (в т.ч. нематериальных) с совокупными затратами.

2.3.4 Сводные показатели и практические рекомендации

Таблица 13: Сводные экономические показатели проекта

Показатель	Значение (формальный расчёт)	Интерпретация и рекомендации
CAPEX	307 334 руб.	Может быть снижен до 8-10 тыс. руб. при использовании личного оборудования и бесплатного хостинга.
OPEX	473 000 руб./год	Основная статья — поддержка. Может быть снижена за счет вовлечения сообщества.
Годовая выручка	125 000 руб.	Недостаточна для окупаемости. Необходимо комбинировать модели: добавить Open Core (премиум-функции) и SaaS.
Точка безубыточности	20 контрактов/год	Достижима только при расширении продукта до уровня B2B-решения и активных продажах.
ROI	-215%	Отражает провал данной конкретной финансовой модели, а не бесполезность проекта.

Итоговые практические рекомендации для разработчика:

1. Не рассчитывайте на прямую прибыль от open-source ядра. Рассматривайте его как публичное портфолио и основу для бизнеса.
2. Для достижения финансовой устойчивости с первого дня планируйте гибридную модель (напр., Open Core), где премиум-функции закрыты и платны.
3. Рассчитывайте рентабельность не как ‘(Выручка - Затраты) / Затраты‘, а как ‘(Выгоды(денежные + нематериальные) - Затраты) / Затраты‘. В эту формулу включайте рост вашей рыночной ставки как разработчика.
4. Используйте open-source разработку как стратегию снижения барьеров входа на рынок и проверки гипотез, что в долгосрочной перспективе может оказаться выгоднее традиционной коммерческой разработки [15; 16].

Таким образом, курсовая работа не только провела расчёт по заданной методике, но и выявила её ограничения применительно к open-source, предложив пути адаптации. Это подтверждает достижение цели работы — разработки методики экономического обоснования для малых open-source проектов.

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».
14. *The Git Development Community*. Git - git-log Documentation. — 2024. — URL: https://git-scm.com/docs/git-log (visited on 03/24/2025) ; Официальная документация по команде `git log`, использованной для сбора исходных данных.
15. *GitVerse*. Как заработать на Open Source. — 09.2024. — URL: https://gitverse.ru/blog/articles/open-source/47-kak-zarabotat-na-open-source (дата обр. 25.03.2025).
16. *Beeline Cloud*. Как «взвесить» open source: разбираем противоречивые мнения об исследованиях ценности открытого программного обеспечения. — 03.2024. — URL: https://habr.com/ru/companies/beeline_cloud/articles/799121/ (дата обр. 25.03.2025).

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 14: Соответствие параметров оформления

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