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

1.4.1 Экономический парадокс открытого кода

Традиционная модель продажи лицензий, основанная на ограничении доступа к интеллектуальной собственности, вступает в прямое противоречие с принципами Open Source, закрепленными в лицензиях семейства MIT, Apache или GNU GPL [6; 14]. Эти лицензии гарантируют пользователям «четыре свободы»:

запускать, изучать, распространять и изменять программу [5; 15]. В таких условиях классическая монетизация через продажу «копий» становится невозможной. Тем не менее, рынок Open Source продолжает расти, привлекая миллиардные инвестиции и порождая компании с многомиллиардной капитализацией [16; 17].

Современные методы монетизации сместились от продажи самого кода к продаже сопутствующей ценности, которую невозможно легко скопировать: экспертизы, операционной надежности, удобства эксплуатации и гарантий безопасности [18; 19]. В 2024–2025 годах наметился четкий тренд на профессионализацию открытых проектов, где разделение на «хобби-проекты» и «профессиональные решения» становится все более выраженным [17; 20]. Ключевым фактором выживания становится не просто качество кода, а наличие жизнеспособной бизнес-стратегии, которая позволяет поддерживать критическую массу разработчиков и обеспечивать безопасность цепочки поставок [16; 18].

1.4.2 Теоретические основы экономики открытого ПО

Для понимания экономических механизмов, лежащих в основе open source, необходимо обратиться к фундаментальным работам в этой области. Йохай Бенклер, профессор Гарвардской школы права, в своей работе «Freedom in the Commons» вводит концепцию «commons-based peer production» — производства, основанного на сотрудничестве равноправных участников в цифровой среде [21]. Бенклер показывает, что открытое программное обеспечение является наиболее ярким примером новой формы экономической организации, где мотивация участников не сводится к прямому денежному вознаграждению, а включает репутационные, социальные и идеалистические стимулы. Эта теоретическая рамка объясняет, почему тысячи разработчиков добровольно вносят вклад в открытые проекты, создавая колоссальную экономическую стоимость без традиционных рыночных механизмов.

Урсула Хольтгрюве в статье «Articulating the Speed(s) the Internet» дополняет этот анализ социологическим измерением [22]. Она исследует, как самоорганизованные open source проекты создают особую временную и социальную динамику. Добровольный труд в таких проектах, по ее мнению, основывается на сочетании креативности, обучения и обмена знаниями, что формирует устойчивую экосистему, способную конкурировать с коммерческими разработками. Эти теоретические положения служат базой для понимания того, как open source проекты могут генерировать экономическую ценность даже при отсутствии прямых продаж.

1.4.3 Классификация бизнес-моделей открытого ПО

Карл Михаэль Попп в книге «Best Practices for commercial use of open source software» предлагает систематизацию бизнес-моделей, используемых компаниями, которые строят свой бизнес на открытом коде [23]. Он выделяет несколько основных подходов, которые получили развитие в последние десятилетия. В более ранней работе «Profit from Software Ecosystems» Попп и Мейер рассматривают open source в контексте более широких программных экосистем, анализируя, как компании могут извлекать прибыль, участвуя в них [24]. Дирк Рихле в статье «The Single-Vendor Commercial Open Source Business Model» детально исследует модель, при которой одна компания контролирует разработку открытого продукта и монетизирует его через дополнительные проприетарные компоненты или услуги [25].

Основываясь на этих работах, а также на современных обзорах рынка, можно выделить следующие ключевые модели монетизации.

Модель профессиональных услуг и поддержки. Одной из наиболее исторически сложившихся и надежных моделей является коммерциализация экспертизы. Компания-разработчик или сервисный интегратор не взимает плату за программный продукт, а продает свое время и знания, необходимые для его успешного внедрения и эксплуатации в корпоративной среде [18; 19]. Для крупных организаций переход на Open Source связан с высокими операционными рисками. Отсутствие вендорской поддержки в классическом понимании означает, что в случае критического сбоя ответственность ложится на внутреннюю ИТ-команду. Модель платной поддержки решает эту проблему, предлагая бизнесу привычный уровень защиты.

Как отмечает Майк Олсон, основатель Cloudera, в своей лекции в Стэнфорде, построение бизнеса на open source технологиях требует фокуса на услугах для корпоративных клиентов, которые ценят гарантии и операционную надежность выше, чем сам код [26]. Основным продуктом в сервисной модели является не код, а гарантия времени реакции и восстановления системы, закрепленная в соглашении об уровне обслуживания (SLA). Параметры SLA, такие как время реакции на критический инцидент (обычно от 1 часа для премиум-поддержки), доступность системы на уровне 99,9% или 99,99%, превращают открытое ПО в надежный корпоративный актив. Дополнительным источником дохода выступает профессиональное обучение и сертификация, что подтверждается практикой Red Hat, IBM и других вендоров.

Модель «Открытого ядра» (Open Core). Модель Open Core на сегодняшний день является доминирующей для коммерческих компаний, создающих программные продукты на базе открытого кода. Ее суть заключается в разделении продукта на две части: базовое «ядро» (Core), распространяемое под свободной лицензией, и проприетарные дополнения (Extensions), доступные только платным клиентам [23; 25]. Ключевым искусством в этой модели является проведение границы между бесплатным и платным функционалом. Если ядро будет слишком слабым, проект не наберет популярности и не сформирует сообщество. Если же ядро будет слишком мощным, у пользователей не будет стимула переходить на платную версию.

Как правило, корпоративные (платные) функции включают в себя инструменты управления и мониторинга (графические панели для администрирования сотен узлов), безопасность корпоративного уровня (интеграция с Single Sign-On, LDAP, Active Directory и расширенный аудит), масштабируемость и отказоустойчивость (модули для автоматического шардирования, гео-репликации и расширенного резервного копирования), а также оптимизацию производительности (специфические алгоритмы сжатия данных или ускорения запросов, не включенные в общую ветку).

Примером успешной реализации этой модели является Nginx. Свободная версия проекта решила проблему C10k и завоевала более 36% рынка веб-серверов [27]. Это позволило компании Nginx Inc. успешно продавать Nginx Plus — коммерческую версию с расширенными функциями балансировки и мониторинга [28]. Данные IPO компаний, использующих модель Open Core (Elastic, MongoDB, Mulesoft), демонстрируют впечатляющую финансовую эффективность. Так, Elastic при выходе на IPO в 2018 году имел капитализацию $2,5 млрд при годовой выручке $227 млн и среднем чеке $38 тыс. [29].

Несмотря на финансовый успех, модель Open Core создает внутреннее напряжение в сообществе. Рихле отмечает, что независимые участники часто ощущают, что их волонтерский труд используется для обогащения корпоративного владельца [25]. Это приводит к тому, что независимые участники реже вкладываются в разработку новых функций, ограничиваясь исправлением ошибок в ядре. Более того, агрессивное стремление монетизировать проприетарные части может подтолкнуть сообщество к созданию форка (ответвления) проекта, как это произошло с Nginx после приобретения компанией F5 [30].

Программное обеспечение как сервис (SaaS) и облачный хостинг. В эпоху облачных вычислений модель SaaS стала наиболее динамично развивающимся методом монетизации Open Source. По данным исследований, до 67% организаций, платящих за открытое ПО, делают это ради удобства облачного развертывания [31]. Многие современные системы (ClickHouse, Kafka, Kubernetes) обладают чрезвычайно высоким порогом входа в плане администрирования. Для бизнеса зачастую дешевле платить ежемесячную подписку облачному провайдеру, чем нанимать штат квалифицированных инженеров для поддержания инфраструктуры в режиме 24/7.

Преимущества SaaS для корпоративного заказчика включают снижение капитальных затрат (отсутствие необходимости инвестировать в собственное серверное оборудование), гарантированную безопасность и соответствие (провайдер берет на себя прохождение аудитов SOC 2, ISO 27001, GDPR, что критически важно для финтеха и медицины), мгновенную масштабируемость (возможность расширения ресурсов по клику мышки в ответ на рост нагрузки) и фокус на продукте (команда разработчиков клиента может заниматься бизнес-логикой, а не администрированием инфраструктуры).

Главным вызовом для создателей Open Source в модели SaaS стала деятельность гиперскейлеров (AWS, Google Cloud, Azure). Эти компании могут взять любой популярный проект, имеющий пермиссивную лицен-

зию (например, Apache 2.0), и предложить его как управляемый сервис в своем облаке, не возвращая прибыли и не всегда возвращая код оригинальным авторам. Это привело к появлению защитных лицензий, таких как SSPL (Server Side Public License) или BSL (Business Source License). Хотя они не являются «чистым» Open Source согласно критериям OSI, они позволяют авторам защитить свою бизнес-модель, запрещая облачным провайдерам перепродавать софт как услугу без коммерческих договоренностей. Примеры Elasticsearch и MongoDB показывают, что переход на такие лицензии часто является вынужденной мерой для выживания компании-разработчика.

Платформы коллективного финансирования и микроспонсорство. Для проектов, которые не имеют амбиций стать крупными корпорациями, но являются критическими узлами программной экосистемы (например, библиотеки cURL или Babel), на первый план выходят альтернативные методы финансирования, основанные на сообществе. Бенклер предвидел появление таких механизмов ещё в 2003 году, описывая, как цифровая среда позволяет создавать новые формы финансирования общественных благ [21].

Запуск программы GitHub Sponsors в 2019 году стал важным этапом, позволив пользователям напрямую поддерживать индивидуальных разработчиков. Однако более устойчивой формой для командных проектов оказался Open Collective. Эта платформа обеспечивает финансовую прозрачность, позволяя любому участнику видеть, откуда приходят деньги и на что они тратятся. Преимущества Open Collective для проектов включают фискальное спонсорство (возможность принимать деньги от корпораций без необходимости регистрации собственного юридического лица), децентрализацию власти (поддержка сообщества, а не конкретного человека, что снижает риски при уходе основателя) и прозрачность для доноров (компании-спонсоры видят реальный вклад своих денег в проект). Проекты, такие как Webpack (годовой бюджет $83 тыс., наем первого штатного разработчика), Babel ($100 тыс.) и cURL ($15 тыс.), успешно используют эту модель для обеспечения своей устойчивости.

Государственное финансирование и концепция цифрового суверенитета. В 2024–2025 годах отчетливо проявился новый вектор монетизации — государственные инвестиции в Open Source как в общественное благо. Это связано с осознанием того, что уязвимость в одной открытой библиотеке может парализовать цифровую экономику целой страны. Германия стала первопроходцем, создав Sovereign Tech Fund (STF). Его миссия — укрепление цифрового суверенитета путем финансирования поддержки «невидимой» инфраструктуры: библиотек, протоколов и инструментов разработки [32; 33]. С 2022 года фонд инвестировал более 24,6 миллионов евро в 60+ проектов по всему миру.

Ключевые аспекты этой модели: фокус на обслуживании, а не на инновациях (деньги выделяются не на создание новых «фич», а на исправление багов и аудит безопасности), борьба с «проблемой безбилетника» (государство берет на себя расходы, которые частный бизнес склонен игнорировать, считая их общими) и экономическая эффективность (по оценкам Еврокомиссии, каждый вложенный в Open Source евро приносит 4 евро экономического эффекта) [34]. Европейские лидеры призывают увеличить финансирование до 350 миллионов евро на период до 2034 года, чтобы создать надежный технологический фундамент, независимый от иностранных корпораций.

1.4.4 Российский опыт: специфика и успешные кейсы

Россия обладает уникальной школой разработки системного ПО, что позволило ряду проектов занять лидирующие позиции на мировом рынке, используя различные стратегии монетизации.

- Postgres Professional: модель Open Core в сегменте СУБД. Компания Postgres Professional, основанная в 2015 году ведущими разработчиками PostgreSQL, является ярким примером трансформации экспертизы в продукт [35; 36]. Их стратегия сочетает активный вклад в международный проект (upstream) с продажей собственной коммерческой версии Postgres Pro Enterprise. Коммерческая версия включает нативные технологии сжатия данных CFS, шифрование TDE, маскирование данных и инструменты горизонтального масштабирования Shardman. Выручка компании в 2023 году выросла на 50%, достигнув 5 миллиардов

рублей. Это доказывает, что в условиях импортозамещения и высоких требований к надежности бизнеса модель «Open Core + Сервис» является максимально эффективной.

- Nginx: от волонтерского кода до сделки на $670 млн. История Игоря Сысоева и Nginx иллюстрирует классический путь Open Source стартапа. Начав как решение внутренних проблем портала Rambler в 2002 году, Nginx был открыт под пермиссивной лицензией BSD в 2004-м [27]. Монетизация началась лишь через 9 лет с созданием Nginx Inc. и запуском продукта Nginx Plus [28]. Успех был обусловлен тем, что команда сохранила верность открытому коду, добавляя в коммерческую версию лишь те функции, которые действительно необходимы только крупным предприятиям. Итогом стала покупка компании американской корпорацией F5 в 2019 году за 670 миллионов долларов — одна из крупнейших сделок в истории российского ПО.
- ClickHouse: аналитическая СУБД и облачная стратегия. ClickHouse, созданная в Яндексе для задач Метрики, стала глобальным стандартом в области аналитических СУБД (OLAP) [37]. После выделения в отдельную компанию ClickHouse Inc. основным методом монетизации стал ClickHouse Cloud (SaaS). Хотя саму систему можно запустить самостоятельно, сложность настройки шардирования, бэкапов и обеспечения отказоустойчивости заставляет клиентов выбирать облачную версию, где эти задачи автоматизированы.

1.4.5 Тренды и прогнозы на 2025–2030 годы

Будущее монетизации Open Source будет определяться тремя ключевыми факторами: безопасностью, искусственным интеллектом и новыми нормами регулирования.

- Искусственный интеллект как драйвер и вызов. ИИ радикально меняет процесс разработки. С одной стороны, AI-ассистенты (GitHub Copilot) ускоряют написание кода, но с другой — они могут наводнить проекты низкокачественными патчами, увеличивая нагрузку на мейнтейнеров [20]. В плане монетизации 2025 год обещает рост моделей, где AI-функции (например, интеллектуальная оптимизация запросов в СУБД) становятся частью премиальных Open Core пакетов.
- Безопасность цепочки поставок. После инцидентов с закладками (как в случае с xz или Log4Shell) доверие к Open Source стало прагматичным [17]. Компании все чаще готовы платить за «проверенные» и «подписанные» сборки открытого ПО. Это открывает новые возможности для бизнеса, который специализируется на непрерывном аудите и поставке безопасных репозиториев.
- Регуляторное давление. Введение в Европе Cyber Resilience Act (CRA) заставляет разработчиков Open Source серьезнее относиться к ответственности за свой код. Это создает рыночную нишу для организаций, которые берут на себя юридическую ответственность и помощь в соблюдении комплаенса для открытых проектов, используемых в критической инфраструктуре.

1.4.6 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) 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). При этом выбор моде-

ли монетизации (например, Open Core или SaaS) будет определять структуру потенциальных доходов проекта.

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

1.4.7 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). При этом выбор модели монетизации (например, Open Core или SaaS) будет определять структуру потенциальных доходов проекта.

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‘ [38].

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.

2.1.6 Оценка трудоёмкости по модели COCOMO I

Для дополнительной верификации полученной оценки воспользуемся классической параметрической моделью COCOMO I (Constructive Cost Model). Базовая формула для органического типа проектов, к которому может быть отнесён разрабатываемый архиватор, имеет вид [13, p. 87]:

$$PM = 2,4 \times (KSLOC)^{1,05} \times \prod_{i=1}^{n} EMF_i, \tag{3}$$

где PM — трудозатраты в человеко-месяцах, $KSLOC$ — размер кода в тысячах строк, EMF_i — поправочные коэффициенты.

Принимая ожидаемый размер кода равным 5 KSLOC (5000 строк) и используя стандартные значения поправочных коэффициентов для средних условий разработки (надёжность ПО RELY = 1,00; сложность модулей CPLX = 1,00; опыт команды AEXP = 1,00; использование современных инструментов TOOL = 0,90; ограничения времени разработки SCED = 1,00), получаем произведение коэффициентов, равное 0,90. Подставляя значения в формулу:

$$PM = 2,4 \times 5^{1,05} \times 0,90 \approx 2,4 \times 5,38 \times 0,90 = 11,6 \text{ человеко-месяцев.}$$

При стандартной продолжительности рабочего месяца 160 часов это соответствует $11,6 \times 160 = 1856$ часов, что значительно превышает предварительную оценку в 200 часов. Данное расхождение объясняется тем, что COCOMO ориентирована на крупные проекты с полным жизненным циклом и не учитывает специфику малых утилит, возможность повторного использования готовых наработок, а также открытый характер разработки, где часть трудозатрат может быть компенсирована вкладом сообщества [25]. Тем не менее, полученное значение служит верхней границей трудозатрат и подтверждает, что даже по пессимистичным оценкам проект остаётся в рамках реализуемости.

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.

*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.

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. [18]	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)*

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.

Согласно исследованию Д. Рихле, для компаний, работающих по модели single-vendor commercial open source, типичная конверсия пользователей в платящих клиентов составляет от 0,5

Исходные данные для прогнозирования представлены в таблице 7. Конверсия в установку $C_{install} = 2\%$ оценивает долю компаний, которые заинтересуются продуктом и начнут его использовать. На её основе определяется число активных пользователей U. Затем с помощью конверсии C_{active} (пессимистичной, реалистичной и оптимистичной) рассчитывается число клиентов платной поддержки и годовая выручка.

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 клиенты платной поддержки	C_{active}	0,5% (пессимистичный), 1,5% (реалистичный), 5% (оптимистичный)
Среднее количество контрактов в год на пользователя	Q	1
Средний годовой контракт на поддержку	P_{avg}	25 000 руб.

На основе этих параметров рассчитаны три варианта денежного потока (таблица 8).

Таблица 8: Расчётный денежный поток от эксплуатации по сценариям

Показатель	Формула расчёта	Значение (песс. / реал. / оптим.)
Активные пользователи в месяц (компании)	$U = N_{total} \cdot C_{install}$	100
Клиенты платной поддержки в год	$U_{.} = U \cdot C_{active}$	0,5 / 1,5 / 5
Годовая выручка	$V = U_{.} \cdot P_{avg}$	12 500 / 37 500 / 125 000 руб.

Прогнозируемая годовая выручка в реалистичном сценарии составляет 37 500 рублей, что заметно ниже оптимистичной оценки в 125 000 руб. Данная цифра отражает реальную сложность монетизации нишевого open-source продукта, где лишь небольшая доля пользователей готова оплачивать поддержку [2]. В дальнейших расчётах будем опираться на реалистичный сценарий.

2.2.4 Расчёт прибыли и рентабельности

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

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

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

Показатель	Формула расчёта	Значение, руб.
Выручка (Revenue)	V (реалистичный сценарий)	37 500
Переменные расходы (комиссии платёжных систем 5%)	$C_{var} = V \cdot 0.05$	1 875
Валовая прибыль (Gross Profit)	$GP = V - C_{var}$	35 625
Операционные расходы (OPEX)	C_{opex} (из Табл. 6)	473 000
Прибыль до налогообложения (EBIT)	$EBIT = GP - C_{opex}$	–437 375
Налог на прибыль (20%)*	$Tax = 0$ (при убытке)	0
Чистая прибыль (Net Profit)	$NP = EBIT - Tax$	–437 375

Как видно из расчётов, проект является убыточным при рассмотрении исключительно прямых финансовых потоков. Годовой убыток составляет 437 375 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. Как отмечает Рихле в своём исследовании single-vendor open source, важным нематериальным активом становится так называемая «коммиттер-премия» — повышение рыночной стоимости разработчика благодаря статусу коммиттера в значимом проекте [25]. Ссылаясь на эмпирические данные Ханна и др., Рихле указывает, что разработчики, достигшие статуса коммиттера, могут рассчитывать на прирост заработной платы в размере 20–40%.

Количественная оценка нематериальных выгод может быть выполнена с использованием следующего подхода:

$$V_{reputation} = \Delta S \times T_{car} \times P_{comm},$$

где ΔS — ожидаемый прирост годовой зарплаты после получения статуса коммиттера, T_{car} — оставшийся период карьеры (в годах), P_{comm} — вероятность достижения этого статуса в ходе работы над проектом. Дополнительно следует учитывать эквивалентную стоимость обучения (например, специализированные курсы по алгоритмам сжатия и системному программированию, которые могут стоить 50–100 тыс. руб.) и вклад в экосистему, измеряемый через предотвращённые затраты других разработчиков, использующих данный код.

В таблице 10 сопоставлены формальный коммерческий подход и реальный сценарий индивидуальной разработки.

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	12 500 – 125 000 руб./год (прогноз)	0 – 50 000 руб./год (нерегулярные донаты)
Прибыль	–437 375 руб./год (убыток)	Нематериальная выгода (портфолио, знания, репутация)

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 (эффект коммиттер-премии).
- 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 437 тыс. руб. в год в реалистичном сценарии), 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). Эмпирические данные подтверждают существование «коммиттер-премии», что позволяет частично квантифицировать эти выгоды [25].

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:

$$Q_{\text{ТБУ}} = \frac{C_{\text{пост}}}{P_{\text{ед}} - C_{\text{пер.ед}}}, \tag{4}$$

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 1,5 контракта (реалистичный сценарий) 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).

Таблица 12: Данные для расчёта срока окупаемости

Параметр	Обозначение	Значение
Капитальные затраты	CAPEX	307 334 руб. (из Табл. 5)
Чистая прибыль в год	$P_{\text{чист.год}}$	–437 375 руб./год (из Табл. 9)

При отрицательной прибыли срок окупаемости формально стремится к бесконечности: $PP = \frac{307\,334}{-437\,375} \approx -0,7$ (не окупается).

Рентабельность инвестиций (ROI) также отрицательна:

$$ROI = \frac{P_{\text{чист.год}} - \text{CAPEX}}{\text{CAPEX}} \times 100\% = \frac{-437\,375 - 307\,334}{307\,334} \times 100\% \approx -242\%. \tag{5}$$

Вывод 2: С точки зрения классического финансового анализа, проект абсолютно неэффективен: CAPEX не окупается, на каждый вложенный рубль приходится 2,42 рубля убытка.

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

Полученные в предыдущих расчётах отрицательные показатели — убыток 437 тыс. руб. в год, отрицательная рентабельность инвестиций (–242 %), недостижимая точка безубыточности — на первый взгляд свидетельствуют о полной экономической несостоятельности проекта. Однако такой вывод был бы поспешным и методологически неверным. Как справедливо отмечается в отраслевых дискуссиях, попытки оценить open-source исключительно через прямые финансовые потоки заведомо ведут к отрицательным выводам, поскольку игнорируют принципиально иную природу ценности, создаваемой открытыми проектами [39]. Этот феномен получил название «парадокс open source»: высокая общественная полезность сочетается с низкой или отрицательной коммерческой рентабельностью, если измерять её традиционными показателями.

Для понимания корней этого парадокса обратимся к работе Д. Рихле, посвящённой экономической мотивации участников open-source экосистемы [40]. Рихле подчёркивает, что разные стейкхолдеры имеют принципиально различные интересы и, соответственно, по-разному извлекают выгоду из существования открытого ПО:

- Системные интеграторы (компании, которые строят решения на базе готовых компонентов) получают возможность увеличить свою прибыль за счёт снижения затрат на лицензионное ПО. Если раньше они вынуждены были делиться выручкой с проприетарными вендорами, то с переходом на open source эти средства остаются у интегратора или могут быть направлены на расширение клиентской базы благодаря более гибкому ценообразованию. Для них open source — это инструмент повышения операционной эффективности.
- Разработчики (как индивидуальные, так и наёмные) видят в участии в open-source проектах возможность накопления человеческого капитала. Работа над реальным, востребованным проектом позволяет приобрести уникальные навыки, получить признание в сообществе и в конечном счёте повысить свою рыночную стоимость. Рихле, ссылаясь на эмпирические исследования Ханна и др., указывает на существование так называемой «коммиттер-премии» — устойчивой прибавки к зарплате у разработчиков, достигших статуса коммиттера в значимых проектах.
- Вендоры (компании, разрабатывающие ПО) используют open source как стратегический инструмент для захвата рынка. Открывая базовую функциональность, они снижают барьеры входа для пользователей, формируют сообщество и на этой основе строят монетизацию через дополнительные услуги или расширенные версии (модели Open Core, SaaS). Это позволяет быстрее обойти традиционных конкурентов, даже если прямая прибыль от самого открытого ядра отсутствует.

Таким образом, отрицательный финансовый результат в нашем расчёте — это не свидетельство «бесполезности» проекта, а отражение того факта, что мы рассматривали его изолированно, в отрыве от более широкого контекста, где создаются основные выгоды.

Дополнительный ракурс анализа даёт классификация рисков, предложенная К. Поппом в его фундаментальном руководстве по коммерческому использованию open source [23]. Попп выделяет четыре категории рисков, которые необходимо учитывать при попытке построить бизнес на открытом коде. Применим эту классификацию к нашему проекту:

- Операционный риск — отсутствие формальных коммерческих услуг (поддержки, гарантированного времени реакции, соглашений об уровне обслуживания) снижает привлекательность продукта для корпоративных заказчиков. В нашем расчёте это проявилось в крайне низкой прогнозируемой конверсии пользователей в платящих клиентов (1,5% в реалистичном сценарии). Корпоративные заказчики, привыкшие к SLA и гарантиям, не готовы полагаться на «энтузиазм сообщества» в критических инфраструктурах.
- Коммерческий риск — лицензия GPL, под которой распространяется архиватор, не препятствует монетизации через поддержку, но блокирует возможность встраивания кода в проприетарные продукты. Это автоматически исключает из потенциального рынка всех независимых производителей ПО, которые могли бы использовать библиотеку сжатия в своих коммерческих приложениях, если бы она была доступна под более либеральной лицензией (например, MIT или Apache). Рынок ограничивается только теми компаниями, которые готовы либо открывать свои наработки, либо приобретать коммерческую лицензию (если бы таковая предлагалась).
- Риск лицензионных атрибутов - даже если текущая модель (поддержка) соответствует GPL, при попытке расширения бизнеса, например, создания SaaS-версии (облачного сервиса сжатия), могут возникнуть сложности. GPL требует раскрытия исходного кода при распространении, но в SaaS-модели распространения как такового не происходит, и возникает так называемая «ASP-лазейка» (Application Service Provisioning loophole), которую закрывает только AGPL. Разработчику необходимо заранее продумать, какая лицензия будет использоваться, чтобы не столкнуться с неожиданными ограничениями в будущем.
- Патентный риск - для нашего архиватора, использующего алгоритм DEFLATE, который является общеизвестным (описан в RFC 1951) и сроки действия основных патентов на него истекли, этот риск минимален. Однако в общем случае использование open source компонентов может нести потенциальные патентные претензии, особенно в быстроразвивающихся областях (кодеки, беспроводные протоколы).

Осознание этих рисков подводит нас к ключевому вопросу: можно ли вообще получить деньги от open-source проекта, и если да, то как?

Ответ на этот вопрос необходимо формулировать в два этапа.

1. Можно ли получить деньги с open-source проекта?

Да, безусловно, можно, но не через прямую продажу лицензий на само ядро (в классическом понимании). Мировая практика выработала несколько устойчивых моделей монетизации, которые успешно применяются десятками компаний [18; 19]:

- Платная поддержка и консалтинг — именно эта модель была использована в наших расчётах. Как показал анализ, в чистом виде она недостаточна для достижения безубыточности: для покрытия годовых операционных затрат требуется не менее 20 контрактов поддержки, тогда как реалистичный прогноз даёт лишь 1,5 контракта. Это подтверждает тезис о том, что модель поддержки эффективна либо для проектов с очень большой пользовательской базой, либо в сочетании с другими подходами.
- Модель Open Core — применительно к архиватору это означало бы, что консольная утилита остаётся открытой (GPL), а графический интерфейс, планировщик задач, облачная синхронизация или интеграция с корпоративными системами становятся платными расширениями. Такой подход позволил бы привлечь корпоративных пользователей, не готовых работать из командной строки, и получать доход, не затрагивая открытое ядро.
- SaaS (ПО как услуга) — создание веб-сервиса для сжатия файлов с платными тарифами (по объёму, скорости, дополнительным функциям). Эта модель особенно привлекательна для пользователей, которые не хотят самостоятельно устанавливать и администрировать ПО. Для архиватора можно реализовать как простое веб-приложение с загрузкой файлов, так и API для разработчиков, встраивающих сжатие в свои приложения.
- Краудфандинг и спонсорство — если проект станет популярным в сообществе, он может получать финансирование через платформы Open Collective или GitHub Sponsors. Это не замена коммерческим моделям, но источник средств для покрытия инфраструктурных расходов или оплаты работы ключевых разработчиков.

Важно понимать, что эти модели не исключают друг друга. Многие успешные проекты комбинируют несколько подходов: например, предлагают бесплатную версию (Open Core), платную поддержку и облачный хостинг (SaaS). Это позволяет диверсифицировать доходы и охватить разные сегменты пользователей.

2. Как правильно рассчитать рентабельность open-source проекта?

Классический расчёт ROI, основанный на сопоставлении прямых денежных притоков и оттоков, в чистом виде неприменим, поскольку игнорирует ключевые нематериальные выгоды, ради которых, собственно, и затеваются многие открытые проекты. Необходима адаптированная методика, включающая следующие компоненты:

1. Расчёт «затратной» компоненты - он выполняется по классической схеме: определение капитальных затрат (CAPEX) и операционных расходов (OPEX). Именно это было проделано в разделе 2.2. Затратная часть остаётся объективной и обязательной для понимания минимальных ресурсов, необходимых для создания и поддержки продукта.
2. Оценка «доходной» компоненты - здесь требуется перейти от простого прогноза продаж к многокритериальной модели, учитывающей:
    - Денежные потоки от гибридных моделей - как показано выше, это могут быть поступления от поддержки, продажи проприетарных расширений, подписки на SaaS, донаты. В нашем расчёте мы ограничились только поддержкой, но реальная доходная часть могла бы быть шире. Необходимо прогнозировать несколько сценариев с разной комбинацией моделей и оценивать вероятные денежные потоки по каждой.
    - Экономию на рекламе и найме - успешный open-source проект сам по себе является мощным маркетинговым инструментом. Компания-разработчик может существенно экономить на привлечении клиентов и сотрудников, поскольку сообщество генерирует «сарафанное радио», а лучшие контрибьюторы со временем могут стать ценными кадрами с уже доказанной квалификацией. Эту экономию можно оценить через сравнение с затратами на традиционный маркетинг и рекрутинг.
    - Накопление человеческого капитала - участие в разработке повышает квалификацию команды. Для индивидуального разработчика это выражается в росте рыночной ставки («коммиттер-премия»). Согласно данным, приведённым у Рихле [25], разработчики, имеющие статус коммиттера в значимых проектах, могут рассчитывать на прирост зарплаты в размере 20–40%. Если разработчик планирует работать в индустрии ещё, скажем, 10 лет, то приведённая стоимость этого прироста может составить сотни тысяч рублей - сумма, сопоставимая с прямыми затратами на разработку.
    - Социальный и экосистемный вклад - этот компонент сложнее всего поддаётся количественной оценке, но именно он часто оказывается доминирующим в макроэкономических исследованиях. Например, отчёт Еврокомиссии показывает, что каждый евро, вложенный в развитие open source, приносит обществу 4 евро экономического эффекта за счёт ускорения инноваций, снижения зависимости от конкретных вендоров и повышения прозрачности [1]. Вклад конкретного проекта можно оценить через количество зависимых проектов, сэкономленные часы разработчиков, предотвращённые затраты на лицензии.
3. Расчёт «скорректированной рентабельности» - итоговый показатель должен представлять собой отношение суммы всех выгод (как денежных, так и оценённых нематериальных) к сумме всех затрат. Формально:

$$ROI_{\text{скорр}} = \frac{B_{\text{денежные}} + B_{\text{репутация}} + B_{\text{человеч.капитал}} + B_{\text{экосистема}}}{C_{\text{прямые}} + C_{\text{альтернативные}}}.$$

При таком подходе даже проект с отрицательным денежным потоком может оказаться вполне оправданным с точки зрения совокупной выгоды для разработчика или организации.

Таким образом, отрицательные цифры, полученные нами, — это не приговор, а сигнал о необходимости расширения модели оценки. Они показывают, что чисто сервисная модель поддержки в данном конкретном случае недостаточна, и проект требует либо комбинирования с другими подходами, либо должен рассматриваться как инвестиция в человеческий капитал и репутацию.

2.3.4 Сводные показатели и практические рекомендации

Для наглядности сведём основные количественные результаты и дадим их интерпретацию с учётом проведённого анализа.

На основе проведённого анализа и обобщения работ Рихле и Поппа можно сформулировать развёрнутые практические рекомендации для разработчика, планирующего создание open-source проекта (в частности, консольного архиватора или аналогичной утилиты):

1. Не рассчитывайте на прямую прибыль от открытого ядра. Воспринимайте его как бесплатную визитную карточку, инструмент для привлечения пользователей и формирования сообщества. Основная ценность создаётся не в коде, а в сопутствующих сервисах и дополнительных продуктах.
2. С первого дня планируйте гибридную бизнес-модель. Проанализируйте, какие функции могут быть интересны только корпоративным клиентам (управление, безопасность, интеграция, масштабирование), и вынесите их в платную «премиум» версию (Open Core). Как подчёркивает Рихле, продажа «whole product» (целостного продукта, решающего все задачи клиента) является одним из основных источников дохода single-vendor open source компаний [25]. Одновременно продумайте возможность предоставления облачного сервиса (SaaS) — многие пользователи готовы платить за готовое решение, избавляющее их от самостоятельной установки и администрирования.
3. При оценке эффективности используйте расширенную формулу рентабельности, учитывающую нематериальные выгоды. Включите в расчёт:
    - прогнозируемый рост вашей рыночной ставки после получения статуса коммиттера («коммиттер-премия»);
    - стоимость знаний и навыков, приобретённых в ходе разработки (эквивалентную стоимости специализированных курсов);
    - экономию на рекламе и найме, которую обеспечит популярность проекта;
    - вклад в экосистему, измеряемый через количество зависимых проектов и предотвращённые затраты других разработчиков.

    Эти величины могут быть оценены экспертно или на основе аналогий, но их включение делает анализ более реалистичным.

Таблица 13: Сводные экономические показатели проекта

Показатель	Значение (реалистичный сценарий)	Интерпретация и рекомендации
CAPEX (капитальные затраты)	307 334 руб.	Может быть снижен до 8–10 тыс. руб. при использовании личного оборудования и бесплатного хостинга (для индивидуального разработчика). В коммерческом сценарии эти затраты необходимо закладывать в бюджет.
OPEX (операционные расходы)	473 000 руб./год	Основная статья — техническая поддержка. При успешном формировании самообслуживаемого сообщества (self-supporting community) эти расходы могут быть существенно сокращены, как показывает практика успешных open-source проектов [25]. Рекомендуется активно развивать документацию, форумы и поощрять взаимопомощь среди пользователей.
Годовая выручка (только поддержка)	37 500 руб.	Очевидно недостаточна для окупаемости. Необходимо комбинировать модели: добавить Open Core (продажа премиум-функций) и/или SaaS (облачный сервис с подпиской). Например, можно оставить консольную утилиту бесплатной, а разработать веб-интерфейс с расширенными возможностями (планировщик, интеграция с облачными хранилищами) и продавать к нему доступ.
Точка безубыточности	20 контрактов/год	При текущей модели поддержки это целевой показатель, который достижим только при активных продажах и выходе на корпоративный сегмент. Для сравнения, в модели Open Core точка безубыточности могла бы быть ниже за счёт более высокой цены премиум-лицензий.
ROI (классический)	–242%	Отражает неэффективность изолированной модели поддержки, но не бесполезность проекта как такового. При учёте нематериальных выгод этот показатель может стать положительным.

4. Используйте open-source разработку как стратегию снижения барьеров входа на рынок. Выход с проприетарным продуктом требует значительных вложений в маркетинг и «раскрутку». Открытый код позволяет привлечь первых пользователей практически бесплатно, получить обратную связь и постепенно сформировать базу для будущих продаж. Как отмечается в отраслевых дискуссиях, такой подход в долгосрочной перспективе может оказаться выгоднее традиционного [39].
5. Учитывайте множественность мотиваций стейкхолдеров. Успешная стратегия должна балансировать интересы различных участников экосистемы [40]:
    - Для *сообщества разработчиков* создайте понятный и прозрачный путь от пользователя до контрибьютора и, возможно, коммиттера. Люди должны видеть, что их вклад ценится и может привести к росту репутации.
    - Для *корпоративных пользователей* делайте упор на снижение совокупной стоимости владения, предоставляйте гарантии и SLA (за плату).
    - Для *потенциальных бизнес-партнёров* (интеграторов, вендоров) предлагайте гибкие условия лицензирования и возможность совместного развития продукта.
6. Внедрите элементы open source governance с самого начала. Следуя рекомендациям Поппа [23]:
    - На *стратегическом уровне* определите приемлемый уровень риска. Например, решите, какие лицензии для сторонних зависимостей вы готовы допустить (только пермиссивные? можно и копилефтные?). Это важно для будущей коммерциализации.
    - На *тактическом уровне* настройте автоматическую проверку лицензий всех используемых библиотек. Существуют инстру-

менты (FOSSology, ScanCode и др.), которые могут сканировать код и выявлять лицензионные условия. Это предотвратит случайное включение кода с несовместимой лицензией.

- На *операционном уровне* ведите реестр всех использованных open-source компонентов и их лицензий. Это не только требование многих лицензий, но и необходимая база для due diligence, если в будущем проект заинтересует инвесторов или покупателей.

7. Активно работайте с сообществом, но сохраняйте контроль над стратегическими решениями. Как показывает модель single-vendor open source, ключевым фактором успеха является создание активного и самообслуживаемого сообщества пользователей, которое помогает друг другу, пишет документацию, сообщает об ошибках. Однако, если вы планируете коммерческое использование, необходимо сохранять авторские права на код и требовать передачи прав на вносимые изменения (или как минимум права на перелицензирование) [25]. Это позволит избежать ситуации, когда проект «разветвляется» сообществом в нежелательном направлении.

Таким образом, курсовая работа не только провела расчёт по заданной методике, но и выявила её ограничения применительно к open-source, предложив пути адаптации. Полученные отрицательные финансовые показатели следует интерпретировать не как свидетельство неудачи, а как подтверждение того, что open-source проекты требуют особых подходов к оценке эффективности, учитывающих долгосрочные нематериальные выгоды и множественность интересов участников экосистемы. Это подтверждает достижение цели работы — разработки и апробации методики экономического обоснования для малых open-source проектов.

Заключение

В ходе выполнения курсовой работы была разработана и апробирована методика экономического обоснования для малого open-source проекта на примере создания консольного архиватора, реализующего алгоритм DEFLATE.

Основные результаты работы:

1. Проведён анализ теоретических основ разработки открытого программного обеспечения, включая классификацию ПО, философию open source, типы лицензий и современные модели монетизации. Особое внимание уделено работам Д. Рихле, раскрывающим экономическую мотивацию различных участников open-source экосистемы, и исследованиям К. Поппа, систематизирующим риски и практики коммерческого использования открытого кода.
2. Выполнен количественный анализ истории разработки трёх классических open-source архиваторов (`bzip2`, `xz utils`, `zstd`), который подтвердил реалистичность планируемых трудозатрат и позволил оценить масштаб проекта-аналога (`bzip2` с объёмом кода 16,6 тыс. строк).
3. Разработана детальная оценка трудозатрат на создание MVP (200 человеко-часов, ФОТ 240 тыс. руб.), которая была верифицирована с помощью параметрической модели COCOMO I. Расхождение между декомпозиционной оценкой и моделью COCOMO (1856 часов) объясняется спецификой малых утилит и открытым характером разработки, что позволило использовать результат COCOMO как верхнюю границу трудозатрат.
4. Произведён расчёт капитальных и операционных затрат. Общий объём CAPEX составил 307 тыс. руб., годовые операционные расходы (OPEX) — 473 тыс. руб. Показано, что структура затрат для open-source проекта радикально смещена в сторону ФОТ при нулевой стоимости лицензионного ПО.
5. Выполнено прогнозирование выручки по трём сценариям с учётом данных о типичной конверсии пользователей в клиенты (0,5–2%) для single-vendor open source компаний [25]. В реалистичном сценарии годовая выручка составила 37,5 тыс. руб., что привело к убытку 437 тыс. руб. и отрицательной рентабельности инвестиций (–242%).
6. Рассчитана точка безубыточности (20 контрактов поддержки в год), показана недостижимость этого показателя в рамках чисто сервисной модели для данного проекта.
7. Предложена адаптированная методика оценки эффективности open-source проектов, учитывающая не только денежные потоки, но и нематериальные выгоды: накопление человеческого капитала («коммиттер-премия»), репутационные эффекты, вклад в экосистему. Разработаны формулы для количественной оценки этих компонентов.
8. Сформулированы практические рекомендации для разработчика open-source проекта, включающие выбор гибридной бизнес-модели (Open Core, SaaS), использование расширенной формулы рентабельности, работу с сообществом и внедрение элементов open source governance.

Выводы:

Проведённое исследование подтвердило исходную гипотезу о том, что классические методы экономической оценки, ориентированные на коммерческую разработку, дают отрицательные результаты применительно к open-source проектам и не отражают их реальной ценности. Парадокс open source заключается в несоответствии высокой общественной полезности и низкой или отрицательной коммерческой рентабельности, измеряемой традиционными финансовыми показателями.

Предложенная в работе адаптированная методика, сочетающая классические подходы (COCOMO, расчёт CAPEX/OPEX) с оценкой нематериальных выгод (человеческий капитал, репутация, экосистемный вклад), позволяет более адекватно оценивать эффективность малых open-source проектов и обосновывать целесообразность их разработки, особенно в образовательном и академическом контексте.

Практическая значимость работы заключается в создании инструментария, который может быть использован студентами, начинающими разработчиками и исследователями для экономического обоснования собственных open-source инициатив, а также для выбора оптимальной стратегии монетизации.

Направления дальнейших исследований:

- Эмпирическая верификация предложенных коэффициентов для оценки «коммиттер-премии» на российском рынке труда.
- Разработка методики количественной оценки вклада проекта в экосистему через анализ графа зависимостей.
- Исследование применимости предложенного подхода для других типов open-source проектов (библиотеки, фреймворки, инфраструктурное ПО).

Таким образом, цель курсовой работы — разработка и апробация методики экономического обоснования для малого open-source проекта — достигнута, все поставленные задачи выполнены.

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).
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).
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. *Snyk*. Open Source Licenses: Types and Comparison. — URL: https://snyk.io/articles/open-source-licenses/ (дата обр. 24.02.2026).

15. *Free Software Foundation*. Four Principles of Free Software. — 2024. — URL: https://www.gnu.org/philosophy ; 4 принципа свободного программного обеспечения.

16. *The New Stack*. Open Source: Inside 2025's 4 Biggest Trends. — 2025. — URL: https://thenewstack.io/open-source-inside-2025s-4-biggest-trends/ (дата обр. 24.02.2026).

17. *InfoWorld*. Open source trends for 2025 and beyond. — 2025. — URL: https://www.infoworld.com/article/3800992/open-source-trends-for-2025-and-beyond.html (дата обр. 24.02.2026).

18. *Palark*. How companies make millions on Open Source. — URL: https://palark.com/blog/open-source-business-models/ (дата обр. 24.02.2026).

19. *Wikipedia*. Business models for open-source software. — 2026. — URL: https://en.wikipedia.org/wiki/Business_models_for_open-source_software (дата обр. 24.02.2026).

20. *Academy D. A.* Open source trends 2025. — 2025. — URL: https://duckalignment.academy/open-source-trends-2025/ (дата обр. 24.02.2026).

21. *Benkler Y.* Freedom in the Commons: Towards a Political Economy of Information // Duke Law Journal. — 2003. — Vol. 52, no. 6. — P. 1245–1276. — URL: https://scholarship.law.duke.edu/dlj/vol52/iss6/3/ ; Фундаментальная работа по экономике commons-based peer production.

22. *Holtgrewe U.* Articulating the Speed(s) of the Internet: The Case of Open Source/Free Software // Time & Society. — 2004. — Vol. 13, no. 1. — P. 129–146. — DOI: 10.1177/0961463X04040750. — Социологическое исследование самоорганизованных open source проектов.

23. *Popp K. M.* Best Practices for commercial use of open source software: Business Models, Processes and Tools for Managing Open Source Software. — Norderstedt, Germany : Books on Demand, 2015. — 124 p. — ISBN 978-3-7386-1909-6. — Описание современных бизнес-моделей на основе открытого ПО и управления OSS в коммерческой разработке.

24. *Popp K. M.*, *Meyer R.* Profit from Software Ecosystems: Business Models, Ecosystems and Partnerships in the Software Industry. — Professional ed. — Norderstedt, Germany : Books on Demand, 2010. — 240 p. — ISBN 978-3-8391-6983-4. — Предисловие Karl-Heinz Streibich (CEO Software AG). Включает анализ экосистем Google, Microsoft, SAP и open source.

25. *Riehle D.* The Single-Vendor Commercial Open Source Business Model // Information Systems and e-Business Management. — 2012. — Vol. 10, no. 1. — P. 5–17. — DOI: 10.1007/s10257-010-0149-x. — Анализ бизнес-моделей коммерческих open source компаний.

26. *Olson M.* Opportunities Abound in the Big Data Space. — Stanford University, 11/2013. — URL: https://ecorner.stanford.edu/videos/opportunities-abound-in-the-big-data-space/ ; Лекция основателя Cloudera о построении бизнеса на open source технологиях. Stanford eCorner.

27. *Wikipedia*. Nginx. — 2026. — URL: https://en.wikipedia.org/wiki/Nginx (дата обр. 24.02.2026).

28. *Whiteley R.*, *The NGINX Team*. Do Svidaniya, Igor, and Thank You for NGINX. — 2022. — URL: https://blog.nginx.org/blog/do-svidaniya-igor-thank-you-for-nginx (дата обр. 24.02.2026).

29. *Ventures O.* Building venture-scale open core. — URL: https://www.opencoreventures.com/blog/building-venture-scale-open-core/ (дата обр. 24.02.2026).

30. *Slashdot*. Nginx Core Developer Quits Project, Says He No Longer Sees Nginx as 'Free and Open Source Project For the Public Good'. — 2024. — URL: https://developers.slashdot.org/story/24/02/16/164217/nginx-core-developer-quits-project-says-he-no-longer-sees-nginx-as-free-and-open-source-project-for-the-public-good (дата обр. 24.02.2026).

31. *GetMonetizely*. How Can SaaS Companies Balance Open Source Strategy with Commercial Pricing? — URL: https://www.getmonetizely.com/articles/how-can-saas-companies-balance-open-source-strategy-with-commercial-pricing (дата обр. 24.02.2026).

32. *SPRIND*. Sovereign Tech Fund. — URL: https://www.sprind.org/en/actions/projects/sovereign-tech-fund (дата обр. 24.02.2026).

33. *Interoperable Europe Portal*. Funding open source: case study on the Sovereign Tech Fund. — URL: https://interoperable-europe.ec.europa.eu/collection/open-source-observatory-osor/document/funding-open-source-case-study-sovereign-tech-fund (дата обр. 24.02.2026).

34. *Open Source Initiative*. Investing in Open Source sustainability: OSI supports OpenForum Europe's EU Sovereign Tech Fund proposal. — 2025. — URL: https://opensource.org/blog/investing-in-open-source-sustainability-osi-supports-open-forum-europes-eu-sovereign-tech-fund-proposal (дата обр. 24.02.2026).

35. *TAdviser*. Postgres Pro Enterprise. — URL: https://tadviser.com/index.php/Product:Postgres_Pro_Enterprise (дата обр. 24.02.2026).

36. *Postgres Professional*. Postgres Pro Enterprise. — URL: https://postgrespro.com/products/postgrespro/enterprise (дата обр. 24.02.2026).

37. *Lindesvärd J.* ClickHouse: The Good, The Bad, and The Ugly. — 2023. — URL: https://dev.to/lindesvard/clickhouse-the-good-the-bad-and-the-ugly-2pi7 (дата обр. 24.02.2026).

38. *The Git Development Community*. Git - git-log Documentation. — 2024. — URL: https://git-scm.com/docs/git-log (дата обр. 24.03.2025).

39. *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) ; Отраслевой блог.

40. *Riehle D.* The Economic Motivation of Open Source Software: Stakeholder Perspectives // Computer. — 2007. — Май. — Т. 40. — С. 25—32. — DOI: 10.1109/MC.2007.147.

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.