**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ЭКОНОМИЧЕСКИЙ ФАКУЛЬТЕТ**

**Кафедра информационных систем в экономике**

**ДОПУСТИТЬ К ЗАЩИТЕ**

**Заведующий кафедрой информационных систем в экономике**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**“\_\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2016 г.**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**По специальности 080500 «Бизнес-информатика»**

**На тему:** оценка трудозатрат на создание тестовой документации в процессе разработки программного обеспечения

**Бакалавриант 4 курса** Хральцова Татьяна Александровна

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**(Подпись студента)**

**Руководитель** доктор физико-математических наук, профессор кафедры информационных систем в экономике Юрков А.В.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**(Подпись руководителя)**

**Рецензент**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**(Подпись рецензента)**

**Санкт-Петербург**

**2016**

Оглавление

[ВВЕДЕНИЕ 4](#_Toc451690099)

[1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ. ПОСТАНОВКА ЗАДАЧИ 6](#_Toc451690100)

[1.1 Общая информация о тестовых документах и подходов в тестировании 6](#_Toc451690101)

[1.2 Описание исследования и сбор данных 7](#_Toc451690102)

[1.2.1 Тестирование без документации 8](#_Toc451690103)

[1.2.2 Тестирование по документации 8](#_Toc451690104)

[2 АНАЛИЗ ДАННЫХ 10](#_Toc451690105)

[2.1 Оценка трудозатрат 10](#_Toc451690106)

[2.2 Оценка окупаемости 10](#_Toc451690107)

[2.3 Оценка трудозатрат на тестирование по документации на уровне структурной декомпозиции 11](#_Toc451690108)

[2.4 Оценка тест кейсов и отчетов об ошибках. 12](#_Toc451690109)

[2.4.1 Покрытие требований тест кейсами. 12](#_Toc451690110)

[2.4.2 Общее устранение дефектов. 13](#_Toc451690111)

[3. ОБЗОР МОДЕЛЕЙ ОЦЕНКИ СТОИМОСТИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 15](#_Toc451690112)

[3.1 Основанная на стоимости разработка программного обеспечения 15](#_Toc451690113)

[3.1.1 Эффективность автоматизированного тестирования 15](#_Toc451690114)

[3.1.2 Преимущества разработки ПО основанной на стоимости 19](#_Toc451690115)

[3.2 Расширенная модель оценки стоимости разработки программного обеспечения с сертифицирующим уровнем надежности 21](#_Toc451690116)

[3.2.1 Методология «Чистого помещения» 22](#_Toc451690117)

[3.2.2 COCOMO 23](#_Toc451690118)

[3.2.3 Е-COCOMO (Extended Cost Constructive Model) 24](#_Toc451690119)

[3.2.4 COCOMO для Agile 26](#_Toc451690120)

[ЗАКЛЮЧЕНИЕ 28](#_Toc451690121)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 29](#_Toc451690122)

Список принятых обозначений

|  |  |
| --- | --- |
| ПО | Программное обеспечение |
| TDD – Test Driven Development | Разработка через тестирование |
| SW/IT – Software/Information Technology | Программное обеспечение/ Информационные технологии |
| Agile | Гибкий подход к разработке |
| VBSE – Value Based Software Engineering | Разработка программного обеспечения, основанная на стоимости |
| E-COCOMO  | Расширенная модель оценки стоимости разработки программного обеспечения |
| Ad Hoc | Исследовательское тестирование |
| Scrum | Методология управления проектами |
| ATG – Automated Test Generation | Автоматизированное тестирование |
| EAF – Effort Adjustment Factor | Поправочный коэффициент |

# ВВЕДЕНИЕ

В современном мире тестирование программного обеспечения – это неотъемлемая часть разработки информационного продукта. Но не во всех компаниях данный процесс контролируется. В значительной части организаций отдел по тестированию ПО отсутствует, поэтому промежуточное и итоговое тестирование осуществляется программистами, или же его не проводят вовсе. Данное отсутствие сказывается и на качестве производимого программного продукта, и на стоимости исправления его дефектов после официального выпуска. Данный этап разработки информационного продукта несомненно важен, но количество средств и ресурсов, которое необходимо выделять при планировании, зависит от целей компании, её масштаба и приоритетов в разработке.

В зависимости от количественной классификации предприятий определяется подход к общему процессу разработки. Малые предприятия, с численностью сотрудников до 100 человек, владеют ограниченным капиталом. Отсутствие отдела по тестированию или назначение данных обязанностей сотруднику смежной профессии объясняется недостатком ресурсов и специалистов. Поэтому рассматривать такие предприятия в разрезе организации процесса разработки программного обеспечения не следует. Так же не стоит ориентироваться на крупные предприятия, с численностью сотрудников свыше 500, так как в большинстве случаев в таких компаниях актуальны другие вопросы, например, инновационный подход к разработке ПО. Средние предприятия, с численностью сотрудников до 500 человек, наиболее подвержены изменениям, вопросы оптимизации процесса разработки и его эффективности у них стоят остро. Именно в таких предприятиях отдел тестирования программного обеспечения подвергается постоянными интуитивными изменениями.

**Актуальность**

В тестировании, как и в любом другом этапе разработки, существует много формальных моментов. Одним из них является тестовая документация – документация, которая помогает инженерам по тестированию осуществлять проверки продукта. К тестовой документации относятся тест планы, тест сценарии, тест кейсы, отчёты об ошибках, матрица трассируемости и чек-листы. Необходимость вести и поддерживать такое количество документации – важный вопрос, который затрагивает проблемы эффективности и стоимости тестовой документации.

Средние компании-разработчики в большинстве случаев придерживаются такого процесса разработки ПО, как Аджайл (Agile), то есть гибкой методологии разработки, которая характеризуется итеративными подходом [19]. Каждая итерация по своей сути является отдельным проектом, который состоит из этапов: планирование, анализ требований, проектирование, кодирование, тестирование, документирование. Данный подход позволяет минимизировать риски.

Гибкая методология основывается на нескольких принципах, один из которых – работающий продукт важнее исчерпывающей документации. Поэтому в настоящее время встает вопрос, а нужна ли тестовая документация. В средних компаниях, которые существуют на рынке уже не первый год, имеется свой массив тестовых документов, которые они накапливали с начала своего основания. Актуальность информации в данных документах находится под сомнением. Кто должен нести ответственность за её поддержание и сколько данный процесс занимает времени и ресурсов - вопросы, на которые ответить сложно.

**Цель работы**

Целью данной выпускной квалификационной работы является оценка трудозатрат на создание тестовой документации в процессе разработки программного обеспечения.

**Для достижения поставленной цели были решены следующие задачи:**

* Знакомство с теорией тестирования и организацией работ, связанных с тестированием в компании Скаут
* Составление тестовой документации в рамках PBI (Product Backlog Item) и проведение проверок
* Сбор статистики о времени прохождения тестов и написания документации, выгрузка данных из системы Redmine
* Экономический анализ эффективности написания документации и оценка трудозатрат
* Обзор моделей оценки стоимости разработки программного обеспечения

# 1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ. ПОСТАНОВКА ЗАДАЧИ

 Группа компаний Скаут является разработчиком и производителем программно-аппаратного комплекса Скаут – решения для спутникового мониторинга и контроля всех видов транспорта и спецтехники. Руководство компании приняло решение ввести в эксплуатацию новый плагин «Отчёт Х». Соответственно на основе данного решения будет произведена оценка написания тестовой документации и непосредственно анализ эффективности решения.

Оценка трудозатрат на написание тестовой документации и общая оценка трудозатрат на тестирование проводится на основе нескольких экономических моделях: линейной модели, VBSE, COCOMO. В современном мире существуют и другие модели оценки трудозатрат на разработку программного обеспечения. Модель COSMIC Алана Абрана, основанная на автоматизированном тестировании. [2] Данная модель не была выбрана, так как оценивание по ней производится на больших масштабах, чем в данном исследовании. Модель SLIM Путнэма, в которой применяется распределение Рейлиха, несет в себе тоже несколько недостатков: не применима для небольших проектов и на стадиях планирования и кодирования. [18]

## 1.1 Общая информация о тестовых документах и подходов в тестировании

В группе компаний Скаут используются следующие тестовые документы:

Тест кейс – набор входных значений, предварительных условий выполнения, ожидаемых результатов, выходных условий выполнения, разработанных для конкретной цели или тестового случая, например, осуществить конкретный путь программы или проверить соответствие определённому требованию.[3]

Чек-лист – набор идей проверок.

Представленные выше документы хранятся в электронном виде в базе данных в формате .docx.

Отчёт об ошибке (баг-репорт) – документ, описывающий и приоритизирующий обнаруженный дефект, а также содействующий его устранению.[1] Данный документ является документом-карточкой, который создаётся и хранится в баг-трекинговой системе Visual Studio Team Foundation Server.

В своём исследовании я проведу анализ двух подходов к тестированию программного обеспечения – это Эд Хок (Ad Hoc) и «Разработка через тестирование» (TDD).[4]

Разработка через тестирование – это сценарный подход. Его суть состоит в том, что разработка ПО состоит из повторяющихся коротких циклов: сначала пишется тестовая документация, покрывающая желаемое изменение, затем пишется код, который поможет пройти тест, далее осуществляется регрессионное тестирование.

Эд Хок тестирование – это исследовательское тестирование. Его суть заключается в параллельном проектировании и выполнении тестов. То есть тестовая документация не пишется, а выполнение тестов происходит без точного соответсвия какому-либо плану. Тестировщик может писать идеи проверок для себя и проходить тестирование по ним.

## 1.2 Описание исследования и сбор данных

Для того, чтобы рассчитать эффективность создания и поддержания тестовой документации, необходимо выгрузить статистику времени из системы Redmine и рассчитать стоимость каждого затраченного часа для каждого сотрудника. Для последующего качественного и количественного анализа необходимо провести исследование двух методов: тестирование по документации и без документации.

В группе компаний Скаут существует следующее ранжирование должностей: специализация Миддл, Джуниор и Стажёр. Заработная плата рассчитана в Таблице 1.

*Таблица 1 «Информация о сотрудниках»*

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Сотрудник 1 | Сотрудник 2 | Сотрудник 3 |
| Специализация | Миддл | Джуниор | Стажёр |
| Заработная плата, ₽ | 52 800 | 37 000 | 26 400 |
| Стоимость часа работы, ₽ | 300 | 210 | 150 |

Распределение обязанностей следующее:

Сотрудник с уровнем специализации «миддл» тестирует плагин «Отчёт Х» без документации. В его работу входит следующее: чтение технического задания и выявление требований, написание кратких направлений и идей проверок, дополнительное уточнение требований и проведение проверок, фиксация результата.

Сотрудник с уровнем специализации «джуниор» пишет тестовую документацию. Подзадачи, которые он выполняет: анализ технического задания «Отчёт Х», составление требований к функциональности, составление списка проверок на требования, форматирование и агрегирование данных для плана тестирования. Сотрудник с уровнем специализации «стажёр» будет проводить тестирование по документации, то есть осуществлять проверки и фиксировать дефекты.

Перед началом выполнения задач каждый сотрудник экспертным путем оценивает время на выполнение своего задания. Так же в системе Redmine будет рассчитана статистика на каждую задачу сотрудников, в которой будет указано фактически затраченное время.

### 1.2.1 Тестирование без документации

Считаем, что рабочее время – 8 часов 5 дней в неделю.

Экспертная оценка затраченного времени и фактический результат выполнения задач первым сотрудником представлено в таблице 2.

*Таблица 2 «Время на выполнение тестирования без документации»*

|  |  |  |
| --- | --- | --- |
|  | Оценка времени | Фактический результат |
| Анализ ТЗ | 2 | 4 |
| Выявление требований | 7 | 9 |
| Проведение проверок | 20 | 18 |
| Фиксация результата | 4 | 4,5 |
| **Итого:** | **33 часа** | **35,5 часа**  |

### 1.2.2 Тестирование по документации

Экспертная оценка затраченного времени и фактический результат выполнения задач вторым сотрудником представлено в таблице 3.

*Таблица 3 «Время на написание документации»*

|  |  |  |
| --- | --- | --- |
|  | Оценка времени | Фактический результат |
| Анализ ТЗ | 5 | 4 |
| Составление требований | 8 | 6 |
| Составление списка проверок | 16 | 20 |
| Форматирование и агрегирование данных  | 8 | 20 |
| **Итого:** | **37 часов** | **50 часов** |

Экспертная оценка затраченного времени и фактический результат выполнения задач третьим сотрудником представлено в таблице 4.

*Таблица 4 «Время на проведение проверок по документации»*

|  |  |  |
| --- | --- | --- |
|  | Оценка времени | Фактический результат |
| Анализ документации | 1 | 0,3 |
| Проведение проверок | 4 | 3,5 |
| Фиксация результата | 2 | 2 |
| **Итого:** | **7 часов** | **6 часов** |

Фактический результат затраченного времени на проведение проверок по двум методам тестировании намного превысил оценочное время, Таблица 5.

*Таблица 5 «Сравнение оценочного и фактического затраченного времени»*

|  |  |  |  |
| --- | --- | --- | --- |
|  | Оценка времени | Фактический результат | Итоговый результат\* |
| Тестирование без документации | 33 часа | 35,5 часов | 30,2 часа |
| Тестирование по документации | 44 часа | 56 часов | 47,6 часа |

\*Итоговый результат – это чистое затраченное время на выполнение работы, из которого исключили поправки на регрессионное тестирование, которое заняло 10% от всего времени, и 5% коррекция на независимую деятельность.

#

# 2 АНАЛИЗ ДАННЫХ

## 2.1 Оценка трудозатрат

В простейшем случае определить стоимость написания тестовой документации можно исходя из количественной оценки трудозатрат, в человеко-часах. В такой линейной модели используется формула 1:

T = С \* В \* 0,85 , (1)

где Т – трудозатраты, С – стоимость часа сотрудника, В – время, которое понадобилось на выполнение задания. Коэффициент 0, 85 выделяет чистое затраченное время.

В таблице 6 видно, что затраты на чистое время проведения обоих методов тестирования не сильно различаются.

*Таблица 6 «Оценка трудозатрат на тестирование»*

|  |  |  |
| --- | --- | --- |
|  | Тестирование без документации | Тестирование по документации |
| Трудозатраты, ₽ | 9 060  | 9 690 |

## 2.2 Оценка окупаемости

ГК Скаут придерживается смешанного типа процесса разработки программного обеспечения, в настоящее время в отделе тестирования руководство приняло решение постепенно вводить методологию управления проектами Скрам (scrum).[25] Скрам – наиболее известный Agile фреймворк, который помогает командам-разработчикам реализовать продукт за короткие циклы (называемые спринтами), предоставляя быструю обратную связь, непрерывные улучшения и быструю адаптацию к изменению.

Для того, чтобы ответить на вопрос, через какое время окупится написание полной тестовой документации, необходимо рассчитать количество часов на каждый этап спринта. Спринт длится 3 недели – это 120 часов, из которых 2 дня (16 часов) отводится на планирование.

В таблице 7 представлены расчеты затраченного времени и трудозатраты на тестирование по документации для второго и третьего сотрудников.

*Таблица 7 «Трудозатраты на тестирование по документации»*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|   | 1 спринт | 2 спринт | 3 спринт | 4 спринт | Всего |
|   | для джуниора | для стажёра |   |
| Продолжительность спринта, ч | 120 | 20 | 120 | 120 | 120 |   |
| Планирование, ч | 16 | 0 | 16 | 16 | 16 |   |
| Проведение проверок плагина «Отчёт Х», ч | 0 | 6 | 8 | 8 | 8 |   |
| Написание документации для плагина «Отчёт х», ч | 50 | 0 | 0 | 0 | 0 |   |
| Трудозатраты на тестирование плагина «Отчёт Х», ₽ | 10500 | 900 | 1200 | 1200 | 1200 | 15000 |

*Таблица 8 «Трудозатраты на тестирование без документации»*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|   | 1 спринт | 2 спринт | 3 спринт | 4 спринт | Всего |
| Продолжительность спринта, ч | 120 | 120 | 120 | 120 |   |
| Планирование, ч | 16 | 16 | 16 | 16 |   |
| Проведение проверок плагина «Отчёт Х», ч | 35,5 | 35,5 | 35,5 | 35,5 |   |
| Написание документации, ч | 0 | 0 | 0 | 0 |   |
| Трудозатраты на тестирование плагина «Отчёт Х», ₽ | 10650 | 10650 | 10650 | 10650 | 42600 |

В долгосрочной перспективе, четыре спринта, ГК Скаут следует придерживаться метода TDD, который предусматривает написание тестовой документации. Разделение обязанностей между сотрудниками низшей и средней квалификации является экономически выгодным решением. Выгода составляет около 65 процентов, что является эквивалентом 27 600 рублей.

##  2.3 Оценка трудозатрат на тестирование по документации на уровне структурной декомпозиции

Сотрудником специализации Джуниор создано 22 новых тест кейса по исследуемому плагину, 4 тест кейса были созданы ранее. Сведём данные в таблицу 9, где также отразим количество проходов. Этот показатель появляется из соображения, что некоторые тест-кейсы будут находить дефекты, которые потребует повторного выполнения тест-кейса для верификации исправления дефекта. В некоторых случаях дефекты будут открыты заново, поэтому потребуется повторноая верификация. Это относится лишь к части тест-кейсов, поэтому количество проходов может быть дробным числом. Данные условия необходимы для того, чтобы оценка была более точной. Количество проходов для тестирования новой функциональности в общем случае можно грубо оценивать так:

* Простая функциональность: 1–1.5 (не все тесты повторяются).
* Функциональность средней сложности: 2.
* Сложная функциональность: 3–5. [1]

Время на создание и выполнение тест-кейсов оценено экспертным путем. Выполнение тест кейсов и написание отчетов об ошибках осуществляется сотрудником со специализацией Стажер.

*Таблица 9 «Структурная декомпозиция»*

|  |  |  |  |
| --- | --- | --- | --- |
|  | Создание | Выполнение | Отчёты о дефектах |
| Количество | 22 | 26 | 13 |
| Повторения (проходы) | 1 | 1,2 | 1 |
| Общее количество | 22 | 31,2 | 13 |
| Время на 1 тест кейс/отчёт, ч | 0,7 | 0,4 | 0,2 |
| Итоговое время, ч  | 15,4 | 12,48 | 2,6 |

Суммируя итоговое время, затраты на тестирование составили 30, 5 часов (что эквивалентно практически четырем рабочим дням).

## 2.4 Оценка тест кейсов и отчетов об ошибках.

### 2.4.1 Покрытие требований тест кейсами.

По формуле на рисунке 1 посчитаем процентный показатель покрытия требования тест кейсами, которые разработал сотрудник 2. Количество требований в разработанном тестовом документе составило 49.

*Рис. 1 «Процентный показатель покрытия требованиями»*



RC$ =\frac{22}{49}×100 \% =45\% $ (2)

В формуле 2 процентный показатель покрытия равен 45%. Так как фаза проекта – начальная, полученное значение выше минимальной границы данного этапа, что является нормой.

### 2.4.2 Общее устранение дефектов.

За время существование проекта было зарегистрировано 22 дефекта. Критический приоритет имел 1 дефект, высокий приоритет – 3 дефекта, средний – 3 дефекта, низкий – 15 дефектов. Формулы 3 – 6 показывают процентный показатель устранения дефектов.

DFTP1= $\frac{1}{22} × 100 \% ≈4,55 \%$ (3)

DFTP2=$\frac{3}{22} × 100 \% ≈13,6 \% $ (4)

DFTP3=$\frac{3}{22} × 100 \% ≈13,6 \%$ (5)

DFTP4=$\frac{15}{22} × 100 \% ≈68,2 \%$ (6)

*Рис. 2 «Общее устранение дефектов»*



 Процентный показатель важности устранения дефектов обратно пропорционален его количеству.

# 3. ОБЗОР МОДЕЛЕЙ ОЦЕНКИ СТОИМОСТИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## 3.1 Основанная на стоимости разработка программного обеспечения

### 3.1.1 Эффективность автоматизированного тестирования

Большая часть компаний-разработчиков ПО в настоящее время нейтрально относится к стоимостной части вопроса разработки. Для них каждое требование, объект, прецедент и дефект одинаково важны. Однако большинство исследований критических факторов успеха, различающих успешный от неудавшихся проектов программного обеспечения, находит, что основные критические факторы успеха лежат в области стоимости.

Большинство неудач проектов разработки программного обеспечения вызвано отсутствием ориентации на стоимость, такими как отсутствие пользовательских инвестиций, неполные требования, быстро изменяемые требования, отсутствие ресурсов, нереалистичных ожиданий, неясных целей, и нереалистичные временные рамки. [22]

Также трудно принять финансово ответственные решения, используя методы, которые не основаны на стоимости. Разберем это на примере.

Предположим, что разработка и внедрение плагина «Отчёт Х» оценивается в стоимость 2 миллиона рублей. Менеджер предлагает ввести в проект автоматизированное тестирование (automated test generation (ATG)), далее просто ATG.

Прогнозируется, что данная программа сократит затраты на тестирование ПО в половину. Затраты на тестирование составляют примерно 50% совокупных затрат на разработку, или 1 миллион рублей. Предположим, что ATG составляет 30% от общего процесса тестирования, то есть 300 тысяч рублей. При использовании данной программы, вы экономите 50% затрат на тестирование, то есть 500 тысяч рублей – следовательно выгода составляет 200 тысяч рублей.

Необходимо выяснить, выгодно ли переходить на автоматизированное тестирование.

Образованный разработчик ПО оценил бы данное решение с технической точки зрения и с точки зрения управления проектом. Данный вопрос не новый, но несомненно актуален в нынешнее время. Экономисты Персон и Йилмэзтерк в 2004 году выявили 34 проблемы, почему инструмент ATG не экономит затраты на тестирование. [18] На основе их исследования проведём анализ эффективности введения автоматизированного тестирования в проект по плагину «Отчёт Х».

 По мнению специалистов, ATG не является эффективным по ряду причин, в которые входят нетипичное тестовое покрытие, слишком большое количество выходной информации, отсутствие валидных тестовых критериев, недостаточная тестовая документация, нестабильность из-за быстрых изменений функциональности, отсутствие чётких обязательств руководителей, отсутствие подготовки и опыта автоматизации тестирования.

 Метод автоматизированного тестирования предполагает, что каждое требование, прецедент и дефект одинаково важны.

Однако типичную ситуацию можно представить как распределение Парето, в котором 80% стоимости проекта прибывают от 20% компонентов программного обеспечения. Данные в рисунке 3 являются хорошей иллюстрацией этого явления. [9]

Рис. 3 «Распределение Парето 80:20 стоимости тестовой документации»[[1]](#footnote-1)



Каждый протестированный тип пользователей приводит к улучшению начальной выручки с 75% до 90% и намного больше снижает показатель неудовлетворённость пользователей. И на одного из 15 типов клиентов приходится около 50 % доходов. Прямая линия на рисунке 3 является обычным результатом ATG, в котором у каждого следующего теста будет низкая или высокая деловая стоимость с одинаковой вероятностью.

В Таблице 10 показаны относительные объёмы инвестиционных расходов, бизнес преимущества и коэффициент окупаемости инвестиций для автоматизированного тестирования, основанного на нейтральной стоимости, и основанной на стоимости стратегии тестирования по Парето. [21] Коэффициент ROI рассчитывается по формуле 7.

$ROI=\frac{(benefits-costs)}{costs}$, (7)

где benefits – инвестиционные расходы, costs – стоимость.

Таблица 10 «Сравнение тестирования по Парето и автоматизированного»

|  |  |
| --- | --- |
| % тестовых прогонов | Автоматизированное тестирование |
| Инвест. Расходы, ₽ | Стоимость, ₽ | Чистая стоимость, ₽ | ROI |
| 0 | 1 300 | 0 | -1 300 | -1 |
| 10 | 1 350 | 400 | -950 | -0,7 |
| 20 | 1 400 | 800 | -600 | -0,43 |
| 40 | 1 500 | 1 600 | 100 | 0,07 |
| 60 | 1 600 | 2 400 | 800 | 0,5 |
| 80 | 1 700 | 3 200 | 1 500 | 0,88 |
| 100 | 1 800 | 4 000 | 2 200 | 1,22 |
|   | Тестирование по Парето |
|  | Инвест. Расходы, ₽ | Стоимость, ₽ | Чистая стоимость, ₽ | ROI |
| 0 | 1 000 | 0 | -1 000 | -1 |
| 10 | 1 100 | 2 560 | 1 460 | 1,33 |
| 20 | 1 200 | 3 200 | 2 000 | 1,67 |
| 40 | 1 400 | 3 840 | 2 440 | 1,74 |
| 60 | 1 600 | 3 968 | 2 368 | 1,48 |
| 80 | 1 800 | 3 994 | 2 194 | 1,21 |
| 100 | 2 000 | 4 000 | 2 000 | 1 |

Рисунок 4 обеспечивает графическое сравнение получающихся ROI. Анализ основан на следующих предположениях:

* 1 миллион рублей затрат на разработку инвестировали в биллинговую пользовательскую систему к началу тестирования.
* Инструмент ATG будет стоить 300 тысяч рублей и уменьшит затраты на тестирование на 50%, как обещано.
* Экономическое обоснование для проекта - 4 миллиона рублей – стоимость проекта, инвестиционные затраты – 2 миллиона рублей.
* Экономическое обоснование для проекта - распределение 80:20 для оставшихся 14 пользовательских типов.

Рис. 4 «VBSE для ATG и Парето тестировании»[[2]](#footnote-2)


Как замечено в Таблице 10 и в рисунке 4, модель VBSE для ATG действительно достигает снижения затрат, и более высокий коэффициент ROI - 1.22 в 100% тестовом прогоне. Но основанный на стоимости подход к тестированию по Парето имеет более высокий коэффициент возврата инвестиций – 1.74, который может быть достигнут прогоном около 40% наиболее важных тестов. Кроме того, оставшиеся 600 тысяч рублей инвестиций будут генерировать только 160 тысяч рублей стоимости проекта, что является неэффективным вложением дефицитных ресурсов.

Так же отметим, что:

* Полная фокусировка на сокращении затрат может привести к низкому коэффициенту возврата инвестиций.
* Большая часть прибыли от стратегии ATG прибывает в менее ценные будущие денежные потоки.
* Комбинация двух этих подходов может привести к высокому коэффициенту возврата инвестиций.

Несомненно, выявление оставшихся тестовых случаев (60%), которые приводят к отрицательному ROI было бы весьма полезным, так как потенциал снижения расходов для инвестиций в стоимостное тестирование весьма велик.

### 3.1.2 Преимущества разработки ПО основанной на стоимости

На рисунке 5 изображена схема преимуществ VBSE. Она концентрирует в себе инициативы, вклады и результаты развития программного обеспечения в сочетании с информационными технологиями на (SW/IT) уровне. [6] К её общим задачам относится развитие фундаментальных знаний и практических методов, совершенствование которых позволит повысить ценность разрабатываемого ПО и информационно-технологических проектов. Рассматривая модель с конца схема идентифицирует сеть важных промежуточных результатов, зависимость между ними, а так же пути обратной связи, с помощью которых модели и методы анализа будут улучшены с течение времени.

Рис. 5 «Схема реализации преимуществ модели VBSE»



В нижней части диаграммы располагаются тактические проблемы, такие как повышение стоимости и оценка выгод от программных проектов, в то время как верхняя часть отражает стратегические интересы, такие как рассуждения о синергии между проектными и программными элементами больших портфелей, а также возможные пути для улучшения программного обеспечения с технической и информационной-технологической стороны.

Главная цель схемы преимуществ – поддержание промежуточного результата: дизайнеры, программисты и менеджеры должны принимать здесь и сейчас решения, которые были бы лучше тех, которые были отличным вариантом 5 минут назад. Основанные на стоимости решения являются сущностью продуктового и процессного планирования, структуры и активного управления крупными программами. Принятие лучших решений – это ключ к возможному увеличению добавленной стоимости продукта.

Принятие решений в свою очередь зависит от многих других факторов.

Во-первых, количество возможных альтернативных вариантов, с которыми сталкиваются руководители и проектировщики, должно быть достаточно широким. В некоторой степени это определяется структурой IT рынка, то есть где фирмы существуют и что они производят. Эта структура в свою очередь подвергается влиянию огромного числа факторов, в том числе, не только в рамках конкретного государства, но и мира в целом. Структура рынка определяет тот продукт, который компаниям выгодно разрабатывать, его свойства и возможные пути реализации.

Во-вторых, необходимо лучше понимать связи между техническими механизмами проекта, его архитектурой, содержанием и ценностями для конечного потребителя для того, чтобы принятие решений в той или иной ситуации оказалось наиболее эффективным. Непосредственно от руководителей проекта зависит то, насколько верно разработчики понимают взаимосвязь отдельных факторов производимого продукта.

В-третьих, люди, принимающие решения должны понимать, как за счёт технической базы повысить ценность продукта. В частности, они лично должны иметь лучшее понимание источников ценности продукта для потребителя, а также связи между технической составляющей и конечной стоимостью реализации.

В-четвёртых, динамические инструменты мониторинга и контроля в большей степени необходимы для лиц, понимающих решения из большого количества альтернативных вариантов с целью снижения конечной стоимости продукта.

Анализируя автоматизированную систему тестирования ПО, можно выявить, что данный инструмент в большей степени вынуждает компанию затрачивать существенное количество ресурсов, которые находятся в дефиците, на действия с отрицательными инвестиционными доходами. То есть написание тестовой документации для создание данной системы является совершенно невыгодным. [6]

Но значительные преимущества имеет основанная на стоимости модель разработки ПО (VBSE). Переход к основанной на стоимости разработке программного обеспечения несомненно значим, потому что не существует ещё таких пакетов, которые могли бы предоставить анализ преимуществ ПО или его стоимостную оценку. Как и всё, что связано с информационными технологиями, VBSE претерпевает значительные изменения. И те, кто использует данную модель уже сейчас, будут первыми, кто оценит в будущем положительное влияние данной системы.

## 3.2 Расширенная модель оценки стоимости разработки программного обеспечения с сертифицирующим уровнем надежности

Ошибки после релиза ПО занимают время и увеличивают затраты. Традиционная методология разработки программного обеспечения определяет отношение Написание документации:Кодирование:Тестирование (Design:Code:Test) как 40:20:40. [7] Исправление дефектов уже после внедрения продукта увеличивает стоимость разработки ПО по экспоненте. Методология разработки программного обеспечения «чистого помещения» управляет экспоненциальным ростом в стоимости, предотвращая ошибки после внедрения. Это значит, что переход от одного этапа разработки ПО к другому осуществляется после получения доказательства правильности. [19]

Этот новый подход минимизирует исправления и сокращает стоимость в экспоненциальном отношении. Вследствие изменения фазы тестирования COCOMO (Constructive Cost Model - конструктивная модель стоимости) используемая для традиционной разработки непосредственно не может быть применима в разработке программного обеспечения «чистого помещения». Оценки затрат, используемые для традиционного COCOMO, должны быть пересмотрены, то есть необходимо использовать расширенную версию модели COCOMO (Е-COCOMO), в которую включены некоторые новые затраты.

Конструктивная Модель Стоимости (COCOMO) является алгоритмической моделью оценки стоимости программного обеспечения, разработанной Барри Боемом. Модель использует параметры, которые получены из исторических данных и текущих особенностей проекта.

### 3.2.1 Методология «Чистого помещения»

 Харлан Миллз и его коллеги из IBM разработали методологию CSE (Cleanroom Software Engineering (Разработка программного обеспечения методом “Чистого помещения”)) в начале 1980-х. Методология CSE позволяет обнаруживать и устранять ошибки перед выпуском и непосредственно внедрением программного обеспечения, реализуя очевидную идею, что предотвращение дефекта более экономически эффективно, чем устранение его последствий. Для этого необходимо контролировать и вести статистику затрат. [10]

Первой попыткой разработки ПО при статическом контроле качества является модель «чистого помещения», которая характеризуется стратегией непрерывного улучшения процесса разработки. Для достижения данной цели в CSE применяются математические модели разработки, которые позволяют повысить точность планирования и эффективность тестирования, чтобы корректно сертифицировать надежность программного обеспечения.

Разработка программного обеспечения методом «чистого помещения» отличается от других объектно-ориентированных методов:

* Использованием статистического контроля качества
* Проверкой спецификации планирования
* Статистическими тест-кейсами для выявления ошибок верхнего уровня

### 3.2.2 COCOMO

Базовую модель называют COCOMO 81. В 1997 году разработана COCOMO II, которая была опубликована в 2000 году в книге «Оценка стоимости программного обеспечения». COCOMO II является приемником базовой версии и лучше подходит для оценки современных проектов разработки программного обеспечения. Она обеспечивает дополнительную поддержку для современных процессов разработки программного обеспечения и обновленную базу проекта.

COCOMO состоит из иерархии трех уровней:

1. Базовый – подходит для быстрой, грубой оценки величины расходов на написание программного обеспечения, но его точность ограничена из-за отсутствия факторов для учета разницы в атрибутах проекта (драйверы стоимости)
2. Средний уровень учитывает драйверы стоимости в расчете проекта
3. Детальный уровень - дополнительно принимает во внимание влияние отдельных фаз проекта

Базовый уровень рассчитывает затраты на разработку программного обеспечения как функцию от размера программы, который выражается в количестве исходных строк кода. СОСОМО распространяется на три класса программных проектов (см. Таблицу 11).

Таблица 11 «Классы программных проектов»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс проекта | Размер проекта, тыс. строк | Природа проекта | Дедлайн | Среда разработки |
| Органический |  5-20 | "маленькие" команды с хорошим опытом работы и "менее жесткими" требованиями | не жесткий | простая |
| Полуразделен-ный |  50-300 | "средние" команды со смешанным опытом работы и «средними/жесткими» требованиями | средний | средняя |
| Внедренный |  Более 300 | разработанные в рамках "жестких" ограничений (аппаратные средства, программное обеспечение, эксплуатационные модели) | жесткий | комплексная (интегрированная) |

Считается, что базовый уровень COCOMO хорош для быстрой оценки затрат программного обеспечения. Однако, он не дает возможности учитывать различия в ограничениях аппаратных средств, качестве персонала и опыте, использовании современных инструментов и методов, и т.д.

Средний уровень COCOMO вычисляет затраты на разработку программного обеспечения как функцию размера программы и множества «факторов стоимости», которые включают субъективную оценку тестирования продукта, характеристики проекта, аппаратных средств и персонала. У каждого фактора существуют свои дополнительные признаки.

Каждый из 15 факторов экспертно оценивается по важности и стоимости в рейтинге из шести пунктов, от «очень низкого» до «экстра высокого» Далее значения рейтинга заменяются множителями трудоёмкости. Произведение данных множителей является поправочным коэффициентом EAF, который может принимать значения от 0.9 до 1.4 (см. Приложение 1).

Детальный уровень COCOMO описан в книге "Software Engineering Economics in 1981" Барри Боема. Детальный уровень включает в себя все особенности среднего уровня с оценкой влияния факторов стоимости на каждом этапе (анализа, планирования, и т.д.) процесса разработки программного обеспечения. Данный уровень помогает построить оценку программного обеспечения с помощью введения еще двух факторов. [5]

### 3.2.3 Е-COCOMO (Extended Cost Constructive Model)

Как уже было написано, в среднем уровне COCOMO для того, чтобы вычислить EAF, существует 15 факторов затрат в традиционной разработке программного обеспечения. Но так как мы рассматриваем методологию «чистого помещения», нам необходимы некоторые новые факторы оценки. В данную модель необходимо включить “Formal Method Knowledge Capability (FMKC)”, что означает установление знаний, полученных опытным путем, в формальный метод и формальный язык спецификации. [15]

Формальный метод подразумевает метод «чистого помещения» в разработке ПО. Данный метод осуществляется поэтапно, поэтому в модели E-COCOMO фазы, используемые в детальной модели COCOMO, расширяются на планирование требований, разработку продукта, детальное проектирование, кодирование, модульное тестирование, интеграцию и полное тестирование. [13]

На примере ГК Скаут, рассмотрим два варианта временных затрат на реализацию проекта:

1. Проект не предусматривает написание подробной тестовой документации, ограничиваясь лишь общими фактами о взаимодействии с ПО
2. Проект предусматривает написание подробной тестовой документации, углубляясь в отдельные аспекты работы ПО и пути его совершенствования

В каждом варианте значения отдельных факторов равны, за исключением фактора «написание тестовой документации». В первом случае значение поправочного коэффициента будет минимальным (0,93), так как проект предусматривает написание лишь общих фактов об использовании ПО. Во втором же случае, команда разработчиков совместно с тестировщиками уделяет большое внимание написанию тестовой документации, следовательно значение поправочного коэффициента ставим на высокий уровень – 1,02. В плагине «Отчёт Х» примерное количество строк кода составляет 5000.

Подставим исходные данные в формулу 8.

$E=a\_{i}(KLOC)^{b\_{i}}×EAF$ (8)

где Е – трудоёмкость, измеряемая в человеко-месяцах; KLOC – предполагаемое количество тысяч строк кода; EAF – поправочный коэффициент; Коэффициенты ai, bi, представлены в таблице 12.

 Таблица 12 «Классы проекта»

|  |  |  |
| --- | --- | --- |
| Класс проекта | ab | bb |
| Органический | 3.2 | 1.05 |
| Подразделённый | 3.0 | 1.12 |
| Внедрённый | 2.8 | 1.20 |

1. $(3×5)^{1.12}×0.93=19.3 $человеко/месяцев при минимальной разработке тестовой документации
2. $(3×5)^{1.12}×1.04=21.2$ человеко/месяцев при углубленной разработке тестовой документации

Далее, с учётом того что над проектом трудятся 4 человека, оценим трудозатраты на каждого из них, и представим результаты в человеко/днях для более наглядного представления. В месяце – 20 рабочих дней.

1. $\frac{19.3×20}{4}≈97$ на разработку ПО без написания тестовой документации
2. $\frac{21.2×20}{4}≈106$ на разработку ПО с написанием тестовой документации

### 3.2.4 COCOMO для Agile

С течением времени методы разработки программного обеспечения изменяются, поэтому должны изменяться и методы оценки трудозатрат на разработку. Расширенная модель COCOMO адаптирована под методологию «чистого помещения». Но данной модели не хватает некоторых факторов, чтобы использовать ее при таких методах разработки ПО, как Agile, объектно-ориентированная разработка и компонентная разработка. [8]

Потребность гибкого подхода к разработке ПО в COCOMO II обусловливается следующими признаками:

* проектирование оценки программного обеспечения по аналогии размер-производительность-скорость;
* использование алгоритма оценки проекта, как «сегодня будет точно так же, как вчера»;
* простота оценки трудозатрат в модели Agile относительно COCOMO II, требующего установления 20 параметров.

Причина использования методологии COCOMO II для Agile – быстрое выполнение точных расчетов стоимости, учитывая опыт прошлых проектов. COCOMO II для Agile – это интернет ресурс, который рассчитывает стоимость разработки того или иного компонента[24].

Терминология COCOMOII для Agile:

* Analogy parameter (параметр аналогии) : основания для сравнения прошлого проекта с новым. Например, общее достижение в человеко-месяцы, общая стоимость в Долларах
* Baseline value (базовая стоимость) : стоимость определённого параметра прошлого проекта. Например, 40 человеко-месяцев
* Cost Driver (стоимость факторов) : стоимость изменённых факторов модели
* Scale Factor (коэффициент пропорциональности) : коэффициент изменений между прошлым и новым проектами.

Система COCOMO II для Аджайл состоит из нескольких простых шагов:

* Один цикл, чтобы выбрать стоимость факторов или коэффициент пропорциональности
* Далее несколько шагов по циклу
* Указать параметр аналогии и его базовую стоимость
* Выбрать стоимость факторов/ коэффициент пропорциональности, которые будут изменены в данном цикле
* Представить прошлые и новые стоимости выбранных пунктов
* По мере необходимости указать величину, относящуюся к достижениям

На примере ГК Скаут, посчитаем затраты на написание документации (см. Рис 6)

Рис. 6 «Затраты на написание документации»



# ЗАКЛЮЧЕНИЕ

В выпускной квалификационной работе рассмотрена проблема оценки трудозатрат на написание тестовой документации. Проведенные оценки выявили ряд недостатков и позволили обосновать необходимость написания документации. С помощью линейного подхода оценивания было выявлено, что в рассматриваемом периоде расходы на написание и поддержание документации окупаются через 2 цикла разработки. Экономическая выгода составляет около 65 процентов.

Применение различных моделей оценки разработки программного обеспечения дало следующие результаты. Стоимостная оценка показала, что выбор автоматизированного тестирования, которое подразумевает под собой написание тестовой документации, помогает снизить некоторые затраты, но коэффициент возврата инвестиций тогда тоже снижается. ROI равный 1.74 может быть достигнут только при использовании 40% наиболее важных тест-кейсов. Автоматизированное тестирование требует дополнительных ресурсов, то есть написание тестовой документации для создания данной системы является совершенно невыгодным.

Расширенная алгоритмическая модель оценки стоимости не показала различий между двумя подходами к тестированию. Это является обоснованием того, что так же, как в линейной модели, в долгосрочной перспективе написание документации будет экономически выгодно. Модель COCOMO II для Agile показала, что затраты на разработку документации для плагина составили около 100 тысяч рублей.

Методы разработки программного обеспечения адаптируются под новые технологии, в следствие чего изменяются и методы оценки экономической эффективности. Произведенные мной оценки трудозатрат на написание тестовой документации и анализ разработки ПО через тестирование показали, что подход к тестированию TDD является экономически привлекательным для данной компании.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Куликов, С.С. Тестирование программного обеспечения. Базовый курс: практ. Пособие. – Минск: Четыре четверти, 2015. – 294 с.
2. Abran, A., Dumke, R.: COSMIC function points: Theory and Advanced Practices. CRC Press Taylor&Francis Group, 2016, pp 89
3. Glossary “Standard Glossary of Terms used in Software Testing” version 2.4 ISTQB (International Software Testing Qualifications Board), 2015.
4. ISTQB (International Software Testing Qualifications Board) Certification, 2015.
5. Boehm, B. W.: Software Engineering Economics (Prentice Hall, 1981)
6. Boehm, B. W., Sullivan, K., Software Economics: A Roadmap. In: The Future of Software Economics, ed by Finkelstein, A. (ACM Press, 2000), pp 319–343
7. Boehm, B. W. and Huang, L.: Value-Based Software Engineering: A Case Study. IEEE Computer (March 2003), pp 33–41
8. Bullock, J.: Calculating the Value of Testing. Software Testing and Quality Engineering (May/June 2000), pp 56–62
9. M. Wolak , Taking the Art out of  Software Development. An In-Depth Review of Cleanroom software Engineering by Chaelynne.
10. Linger, R.C., "Cleanroom Process Model," IEEE Software. March 1994, pp. 50–58.
11. Karlsson, J., Ryan, K.: A Cost-Value Approach for Prioritizing Requirements. IEEE Software (September–October, 1997), pp 67–74
12. Marschak, J.: Economic Information, Decision, and Prediction (3 volumes), 1974
13. Hevner, A.R. and H.D. Mills, “Box Structure Methods for System Development with Objects,” IBM Systems Journal, vol. 31, no.2, February 1993, pp. 232–251.
14. Henzinger, T.A., Jhala, R., Majumdar, R., Sanvido, M.A.: Extreme model check- ing. In: Verification: Theory and Practice: Essays Dedicated to Zohar Manna on the Occasion of His 64th Birthday. Volume 2772 of Lecture Notes in Computer  Science., Springer-Verlag (2004) 332–358
15. Kosmatov, N., Legeard, B., Peureux, F., Utting, M.: Boundary coverage criteria for  test generation from formal models. In: 15th International Symposium on Software Reliability Engineering (ISSRE’04), Saint-Malo, France, IEEE Computer Society (2004) 139–150
16. Linger, R.M. and H.D. Mills, “A Case Study in Cleanroom Software Engineering: The IBM COBOL Structuring Facility,” Proc. COMPSAC ’88, Chicago, October 1988.
17. Persson, C., Yilmazturk, N.: Establishment of Automated Regression Testing at ABB: Industrial Experience Report on ‘Avoiding the Pitfalls.’ Proc. ISESE 2004, IEEE, August 2004, pp 112–121
18. Stewart, R., Wyskida, R., Johaness, J.: Cost Estimator’s reference manual. A Wiley-Interscience Publication, 1995, pp 527
19. Sharma K., Database Systems Journal «E-COCOMO: The Extended COst Constructive MOdel for Cleanroom Software Engineering», #4, 2013
20. Sullivan, K., Cai, Y., Hallen B., Griswold, W.: The Structure and Value of Modularity in Software Design. Proceedings, ESEC/FSE, 2001, ACM Press, pp 99–108
21. Tockey, S.: Return on Software (Addison Wesley, 2004) (van Solingen, 2004) van Solingen, R.: Measuring the ROI of Software Process Improvement. IEEE Software, May/June 2004, pp 32–38
22. Webster’s Collegiate Dictionary, Merriam-Webster, 2002
23. Agile manifesto [Электронный ресурс] URL: http://www.agilemanifesto.org/iso/ru/ (Дата обращения 10.03.2016)

Agile COCOMO II [Электронный ресурс] URL: http://csse.usc.edu/csse/research/AgileCOCOMO/AgileCOCOMOII/Main.html (Дата обращения 20.05.2016)

SCRUM ALLIANCE, Inc. [Электронный ресурс] URL: https://www.scrumalliance.org (Дата обращения 10.03.2016)

ПРИЛОЖЕНИЕ 1

Таблица – Коэффициенты рейтинга

|  |  |
| --- | --- |
| Факторы стоимости | Рейтинг |
| Очень низкий | Низкий | Средний | Высокий | Очень высокий | Критический |
| Характеристики продукта |   |   |   |   |   |   |
| 1. Требуемая надёжность ПО | 0,75 | 0,88 | 1 | 1,15 | 1,4 |  - |
| 2. Размер БД приложения |   | 0,94 | 1 | 1,08 | 1,16 |  - |
| 3. Сложность продукта | 0,7 | 0,85 | 1 | 1,15 | 1,3 | 1,65 |
| Характеристики аппаратного обеспечения |   |   |   |   |   |   |
| 4. Ограничения быстродействия при выполнении программы | -  |  - | 1 | 1,11 | 1,3 | 1,66 |
| 5. Ограничения памяти |  - | -  | 1 | 1,06 | 1,21 | 1,56 |
| 6. Неустойчивость окружения виртуальной машины | -  | 0,87 | 1 | 1,15 | 1,3 | -  |
| 7. Требуемое время восстановления |  - | 0,87 | 1 | 1,07 | 1,15 | -  |
| Характеристики персонала |   |   |   |   |   |   |
| 8. Аналитические способности | 1,46 | 1,19 | 1 | 0,86 | 0,71 | -  |
| 9. Опыт разработки | 1,29 | 1,13 | 1 | 0,91 | 0,82 | -  |
| 10. Способности к разработке ПО | 1,42 | 1,17 | 1 | 0,86 | 0,7 |  - |
| 11. Опыт использования виртуальных машин | 1,21 | 1,1 | 1 | 0,9 | -  |  - |
| 12. Опыт разработки на языках программирования | 1,14 | 1,07 | 1 | 0,95 |  - |  - |
| Характеристики проекта |   |   |   |   |   |   |
| 13. Применение методов разработки ПО | 1,24 | 1,1 | 1 | 0,91 | 0,82 |  - |
| 14. Документация при разработке ПО | 0,93 | 0,97 | 1 | 1,02 | 1,08 |  - |
| 15. Требования соблюдения графика разработки | 1,23 | 1,08 | 1 | 1,04 | 1,1 | -  |

1. «Value-Based Software Engineering: Overview and Agenda» Barry Boehm [↑](#footnote-ref-1)
2. «Value-Based Software Engineering: Overview and Agenda» Barry Boehm [↑](#footnote-ref-2)