1 ВВЕДЕНИЕ

2 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

2.1 Организационно-экономическая характеристика предметной области (*анализ бизнеса «КАК ЕСТЬ»)*

**2.1.1 Организационная структура объекта автоматизации**

**2.1.2 Описание информационного** и технического обеспечения объекта автоматизации

**2.1.3 Структурно-функциональная диаграмма организации бизнеса «КАК ЕСТЬ» и её описание**

2.2 Экономическая сущность комплекса задач

**2.2.1 Общие сведения о задачах**

**2.2.2 Декомпозиция комплекса задач**

2.2.3 Способы решения задачи

2.3 Обоснование проектных решений по автоматизированному решению экономико-информационных задач

2.3.1 Обоснование необходимости использования вычислительной техники и создания АРМ для решения данного комплекса задач

**2.3.2 Анализ существующих аналогов автоматизированных систем для решения данного комплекса задач**

2.3.3 Обоснование проектных решений по информационному обеспечению

**2.3.4 Обоснование выбора программной среды разработки проектируемой автоматизированной системы**

**2.3.5 Постановка задачи *(Выбор стратегии автоматизации «КАК ДОЛЖНО БЫТЬ»)***

3 ПРОЕКТНАЯ ЧАСТЬ

3.1 Информационное обеспечение комплекса задач

3.1.1 Инфологическая или информационная модель (*схема данных*) и ее описание.

3.1.2 Используемые классификаторы и системы кодирования.

3.1.3 Характеристика входной информации *(нормативно-справочная информация и входная оперативная информация)*

3.1.4 Характеристика результатной (*выходной*) информации

3.2 Технологическое обеспечение.

3.2.1 Организация технологии сбора, передачи, обработки и выдачи информации

3.2.2 Схема технологического процесса сбора, передачи, обработки и выдачи информации

3.3 Выбор и обоснование технических средств

3.4 Программное обеспечение комплекса задач

3.4.1 Общие положения (*дерево функций и сценарии диалогов*)

3.4.2 Структурная схема пакета *(дерево вызовов процедур и программ)*

3.4.3 Описание программных модулей

3.5 Описание контрольного примера реализации

**4 ТЕХНИКО - ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ЭФФЕКТИВНОСТИ ВКР**

5 ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

ПРИЛОЖЕНИ А КОНЦЕПТУАЛЬНАЯ МОДЕЛЬ

Оглавление

[1 ВВЕДЕНИЕ 4](#_Toc325692043)

[2 ОПИСАНИЕ КОМПАНИИ ОАО «СКАЙТЕК» 6](#_Toc325692044)

[2.1 Деятельность компании и оказываемые услуги 6](#_Toc325692045)

[2.2 Организационная структура компании 6](#_Toc325692046)

[2.3 Предметная область деятельности компании ОАО «СкайТек» 12](#_Toc325692047)

[2.4 Структурно-функциональная диаграмма организации процесса тестирования компании ОАО «СкайТек» 18](#_Toc325692048)

[3 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 25](#_Toc325692049)

[**3.1 Описание проблемы и путей ее устранения** 25](#_Toc325692050)

[**3.2 Общие сведения о задачах** 28](#_Toc325692051)

[3.3 Декомпозиция комплекса задач 32](#_Toc325692052)

[*3.3.1 Планирование тестирования* 32](#_Toc325692053)

[*2.2.3.2 Оценка качества тестирования* 44](#_Toc325692054)

[*2.2.3.3 Оценка надежности ПО по ГОСТу 28195–99* 44](#_Toc325692055)

[*4.6.1 Система управления проектами DevProm. Подсистема «Управление тестированием продуктов»* 52](#_Toc325692056)

[*4.6.2 TestRail* 53](#_Toc325692057)

[*4.6.3 TestLink* 55](#_Toc325692058)

[*4.6.4 informUp* 55](#_Toc325692059)

[*4.6.5 SilkCentralTM (TestManager)* 56](#_Toc325692060)

[*4.6.6 QA Complete* 56](#_Toc325692061)

[*4.5.3 Выбор языка программирования* 75](#_Toc325692062)

[4.2 Потоки информации 78](#_Toc325692063)

[4.1 Обоснование целесообразности разработки проекта 81](#_Toc325692064)

[4.2 Оценка конкурентоспособности в сравнении с аналогом 81](#_Toc325692065)

[4.3 Планирование комплекса работ по разработке темы и оценка трудоемкости 84](#_Toc325692066)

[4.4 Расчет затрат на разработку проекта 89](#_Toc325692067)

[4.5 Расчет эксплуатационных затрат 94](#_Toc325692068)

[4.6 Расчет показателя экономического эффекта 98](#_Toc325692069)

[4.7 Маркетинговое сопровождение разрабатываемого продукта 100](#_Toc325692070)

[ПРИЛОЖЕНИЕ А 103](#_Toc325692071)

[ПРИЛОЖЕНИЕ Б 105](#_Toc325692072)

[ПРИЛОЖЕНИЕ В 109](#_Toc325692073)

[ПРИЛОЖЕНИЕ Г 111](#_Toc325692074)

[ПРИЛОЖЕНИЕ Д 113](#_Toc325692075)

# 1 ВВЕДЕНИЕ

Тестирование программного обеспечения (ПО) часто рассматривается как тяжелый, неконтролируемый, зачастую слишком длительный и дорогостоящий процесс, не способствующий повышению качества продукции. Тем не менее, при использовании определенного подхода к организации процесса тестирования, оно может привнести значительный вклад в процесс разработки ПО.

Краеугольный камень эффективной программы тестирования ПО — планирование тестирования. Планирование тестирования включает в себя обзор всех работ по тестированию и проверку того, что процессы, методы, технические приемы, персонал, инструменты и оборудование (аппаратное, промежуточное и программное обеспечение) упорядочены и применяются эффективно. Какими бы ни были проекты, в которых предполагается проведение тестирования, необходимо определить цели и задачи тестирования, зафиксировать требования к тестированию, разработать стратегии, обеспечивающие выполнение установленных требований к тестированию. Основные события и виды деятельности по тестированию должны быть включены в план-график проекта.

Значительная роль в проведении эффективного тестирования принадлежит процессам управления тестированием. На этом этапе ключевыми задачами являются отслеживание дефектов и управление требованиями, совместно с поддержкой количественного анализа и отчетов о ходе тестирования.

Задача планирования и управления тестированием ПО на сегодняшний день представлена широким спектром программных продуктов, которые значительно повышает эффективность работы группы тестирования на этапе подготовки плана тестирования и создания отчетов по итогам проводимых тестовых испытаний. Тем не менее, при обзоре аналогов ПО для управления тестированием было выявлено, что большинство таких программных продуктов не реализует весь возможный потенциал для решения поставленной задачи.

Целью данной выпускной квалификационной работы является проектирование и разработка прототипа информационной системы для планирования и управления процессом тестирования (в дальнейшем ИС «Управление процессом тестирования»), реализующей расширенный функционал для оценки результатов тестирования, не поддерживаемые аналогами ПО в области тестирования. В рамках поставленной задачи необходимо:

* составление набора ключевых показателей для управления процессом тестирования;
* разработка алгоритма построения плана-графика тестирования на основе исходных данных о проекте и данных результатов тестирования;
* разработка моделей для проектируемой ИС «Управление процессом тестирования»;
* разработка прототипа ИС «» Управление процессом тестирования».

Настоящая работа выполнялась применительно к деятельности компании ОАО «СкайТек».

# 2 ОПИСАНИЕ КОМПАНИИ ОАО «СКАЙТЕК»

## 2.1 Деятельность компании и оказываемые услуги

Компания ООО «СкайТек» была создана в 2005 году для работы на рынке заказной разработки программного обеспечения и аутсорсинга. Компания специализируется на разработке Web-ориентированных корпоративных систем, систем принятия решений, аналитических систем и баз данных. На рынке компания работает под коммерческим обозначением «SibEDGE».

До 2009 года компания, главным образом, занималась аутсорсингом разработки программного обеспечения, работая с заказчиками на российском и европейском рынках. С середины 2009 года было принято решение о расширение деятельности, диверсификации форм бизнеса, интенсивного и экстенсивного роста. На текущий момент, компания насчитывает чуть более 25 сотрудников, ведет разработку на нескольких технологических платформах и пропагандирует современные методы к организации процесса производства. В конце первого полугодия 2011 года компания приняла решение о создании линейки собственных продуктов на базе накопленного опыта.

В текущий момент компания активно ищет способы и возможности для развития на рынке информационных технологий и продолжает улучшать качество создаваемых продуктов и услуг. Одним из инструментов является привлечение талантливых, молодых специалистов, способных привнести академический взгляд и коррективы в процессы компании.

{примеры список разрабатываемых проектов, краткое описание (заказчики, назначение систем)}

## 2.2 Организационная структура компании

Под организационной структурой управления предприятием понимается состав, соподчиненность, взаимодействие и распределение работ по подразделениям и органам управления, между которыми устанавливаются определенные отношения по поводу реализации властных полномочий, потоков команд и информации.

[<http://dvsneg.ru/structura_predpriyatiya/organizacionnaya_structura_upravlenia.html>]

В рамках структуры протекает управленческий процесс, между участниками которого распределены функции и задачи управления. С этой позиции организационная структура – это форма разделения и кооперации управленческой деятельности, в рамках которой происходит процесс управления, направленный на достижение целей организации.

Организационная структура управления (ОСУ) ОАО «СкайТек» представлена на рисунке 2.1.

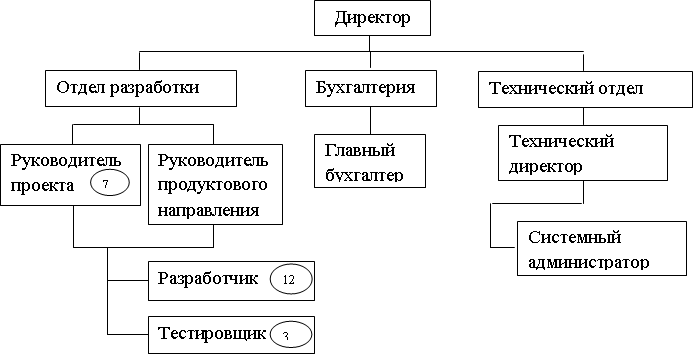


Рисунок 2.1 – ОСУ ОАО «СкайТек»

Во главе компании ОАО «» СкайТек» стоит директор. Круг его задач составляют стратегическое и оперативное управление, разработка позиционирования, маркетинговой стратегии продвижения, коммерческой и ассортиментной политик, бюджетов, оптимизация налоговой политики, постановка управленческого учёта, развитие инфраструктуры.

ООО «СкайТек» состоит из трех подразделений: отдела разработки, технического отдела и бухгалтерии.

Отдел разработки представляет собой основной функциональный орган компании. Главная цель отдела – автоматизация бизнес-процессов компаний-заказчиков с использованием программного обеспечения, как собственной разработки, так и сторонних разработчиков.

Приоритетными направлениями в деятельности отдела являются:

* проектирование, разработка, внедрение и сопровождение информационных систем компаний-заказчиков;
* проведение обзора, анализа и внедрения новых информационных технологий и методов информационного обеспечения на основе передового мирового и отечественного опыта, по мере необходимости, разработка собственного программного обеспечения в этой области.

Сотрудники этого подразделения отвечают за оперативную работу с заказчиками, разработку качественного программного обеспечения в установленные сроки. Текущую работу с заказчиками (обсуждение, уточнение, согласование требований, составление технического задание, распределение задач между сотрудниками) осуществляют руководители проектов или руководитель продуктового направления. Разработчики и тестировщики занимаются решением назначенных на них задач. В свою очередь, руководитель проекта контролируют выполнение задач и отслеживают текущее состояние проекта по средствам регулярной отчетности о проделанной работе.

Технический отдел находится в ведении технического директора и отвечает за выполнение таких задач, как:

* выработка и проведение единой технической политики в компании;
* организация и участие в разработке основных направлений совершенствования технических ресурсов компании;
* организация и участие в проведении работ по техническому обслуживанию, развитию, внедрению передовых технологий, поддержанию в исправном состоянии действующих технических средств компании и т.д.

Системный администратор является исполнителем принимаемых решений в части поддержания ежедневной деятельности сотрудников компании.

В круг типовых задач системного администратора обычно входят:

* подготовка и сохранение [резервных копий](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B7%D0%B5%D1%80%D0%B2%D0%BD%D0%B0%D1%8F_%D0%BA%D0%BE%D0%BF%D0%B8%D1%8F) данных, их периодическая проверка и уничтожение;
* установка и конфигурирование необходимых [обновлений](http://ru.wikipedia.org/wiki/%D0%90%D0%BF%D0%B4%D0%B5%D0%B9%D1%82) для [операционной системы](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) и используемых [программ](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0);
* установка и конфигурирование нового [аппаратного](http://ru.wikipedia.org/wiki/%D0%90%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) и [программного обеспечения](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5);
* создание и поддержание в актуальном состоянии [пользовательских](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C) [учётных записей](http://ru.wikipedia.org/wiki/%D0%A3%D1%87%D1%91%D1%82%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C);
* ответственность за информационную безопасность в компании;
* устранение неполадок в системе;
* планирование и проведение работ по расширению сетевой структуры предприятия;
* документирование всех произведенных действий и т.д.

В обязанности главного бухгалтера входят: (Функциональные обязанности Главного бухгалтера определены на основе и в объеме квалификационной характеристики по должности Главного бухгалтера)

* организация ведения бухгалтерского учета хозяйственно-финансовой деятельности;
* формирование в соответствии с законодательством о бухгалтерском учете учетной политики;
* подготовка рабочего плана счетов, форм первичных учетных документов;
* формирование и своевременное представление полной и достоверной бухгалтерской информации о деятельности предприятия, его имущественном положении, доходах и расходах;
* обеспечение законности, своевременности и правильности оформления документов, составление экономически обоснованных отчетных калькуляций себестоимости продукции, выполняемых работ (услуг), расчеты по заработной плате, правильное начисление и перечисление налогов и сборов в федеральный, региональный и местный бюджеты, страховых взносов в государственные внебюджетные социальные фонды, платежей в банковские учреждения, средств на финансирование капитальных вложений, погашение в установленные сроки задолженностей банкам по ссудам и т.д.

На данный момент компания испытывает нехватку штатных сотрудников (в частности, разработчиков), что приводит к необходимости привлечения сторонних лиц. Помимо этого, в штате компании нет дизайнера, что также приводит к необходимости пользоваться услугами внештатных сотрудников.

Проектная форма организации работ над разрабатываемыми проектами определяет бригадную (кросс-функциональную) структуру команды. Основой этой структуры управления является организация работ по рабочим группам (бригадам). Основными принципами такой организации управления являются [<http://corpsite.ru/Encyclopedia/Consulting/Structure/Brigade.aspx?AspxAutoDetectCookieSupport=1>]:

* автономная работа рабочих групп (бригад);
* самостоятельное принятие решений рабочими группами и координация деятельности по горизонтали;
* замена жестких управленческих связей бюрократического типа гибкими связями;
* привлечение для разработки и решения задач сотрудников разных подразделений.

На рисунке 2.2 представлена схема бригадной формы управления.

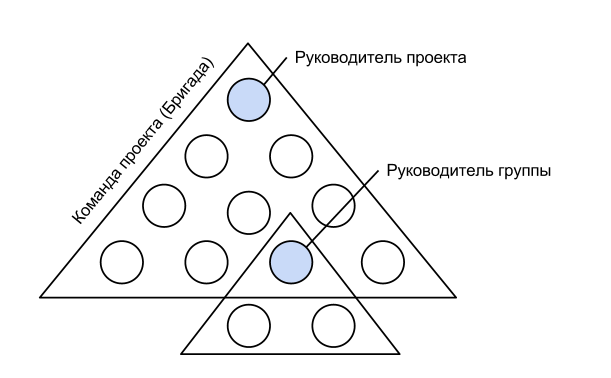


Рисунок 2.2 – Бригадная структура команды для работы над проектом разработки ПО

К достоинствам бригадной структуры следует отнести:

* сокращение управленческого аппарата и повышение эффективности управления;
* гибкое использование кадров, их знаний и компетентности;
* создание условий для самосовершенствования;
* возможность применения эффективных методов планирования и управления;
* сокращение потребности в специалистах широкого профиля.

Недостатки бригадной структуры:

* усложнение взаимодействия
* сложность в координации работ отдельных бригад;
* потребность в высокой квалификации и ответственности персонала и высокие требования к коммуникациям.

## 2.3 Предметная область деятельности компании ОАО «СкайТек»

Как уже было отмечено выше, разработка программного обеспечения на заказ является основным видом деятельности компании «СкайТек». Следует остановиться на нескольких теоретических аспектах, касающихся разработки ПО.

Жизненный цикл программного обеспечения (ЖЦ ПО) — период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации. Этот цикл — процесс построения и развития ПО.

[<http://ru.wikipedia.org/wiki/>Жизненный\_цикл\_программного\_обеспечения]

Нормативной базой на всех этапах ЖЦ являются международные и отечественные стандарты в области информационных технологий, и, прежде всего стандарты ИСО/МЭК, стандарты IEEE, стандарты OMG, стандарты ГОСТ Р, стандарты организации-заказчика.

В качестве первоочередных целесообразно использовать следующие нормативные документы:

* *в части регламентации процессов предприятия:*
* IEEE Std 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology;
* ГОСТ Р ИСО МЭК 12207-99 “Информационные технологии. Процессы жизненного цикла программного обеспечения”;
* IEEE 1074 – Жизненный цикл разработки программных средств;
* ИСО/ТО 10006:1997 (R) “Менеджмент качества. Руководство качеством при административном управлении проектами”;
* ISO 15846, ISO 10007 - стандарты по менеджменту конфигурации программных средств;
* ISO 9000 – 2000; группы ГОСТ Р 9000х;
* ISO/IEC TR 15504 – Оценка процессов жизненного цикла ПО (Information technology – Software process assessment).
* *в части порядка разработки и документирования ИС и ПО:*
* ГОСТ 34.ххх “Информационная технология. Комплекс стандартов и руководящих документов на автоматизированные системы”;
* ГОСТ 19.ххх “Единая система программной документации”;
* IEEE 1063-1987 Standard for Software User Documentation;
* IEEE 830-1994 “Рекомендуемая практика формирования спецификаций программного обеспечения”;
* IEEE 829 – Планирование тестирования программных средств;
* DoD STD 2167A “Разработка программного обеспечения оборонных систем”;
* *в части качества программных средств:*
* ГОСТ 28806 “Качество программных средств. Термины и определения”;
* ГОСТ 28195 “Оценка качества программных средств. Общие положения”;
* ГОСТ 9126 “Информационная технология. Оценка программного продукта. Характеристики качества и руководящие указания по их применению”.

!!!!!!обратить внимание на список стандартов, добавить ссылки

Модель жизненного цикла ПО — структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении жизненного цикла. Модель жизненного цикла зависит от специфики, масштаба и сложности проекта и специфики условий, в которых система создается и функционирует.

Стандарт ГОСТ Р ИСО/МЭК 12207-99 не предлагает конкретную модель жизненного цикла. Его положения являются общими для любых моделей жизненного цикла, методов и технологий создания ИС. Он описывает структуру процессов жизненного цикла, не конкретизируя, как реализовать или выполнить действия и задачи, включенные в эти процессы.

Модель ЖЦ ПО включает в себя:

стадии (часть процесса создания ПО, ограниченная определенными временными рамками);

* результаты выполнения работ на каждой стадии (модели, программные компоненты, документация);
* ключевые события — точки завершения работ и принятия решений.

<http://citforum.ru/database/case/glava1_2.shtml>

К настоящему времени наибольшее распространение получили каскадная и спиральная модели ЖЦ.

Каскадный подход хорошо зарекомендовал себя при построении ИС, для которых в самом начале разработки можно достаточно точно и полно сформулировать все требования. В эту категорию попадают сложные расчетные системы, системы реального времени и другие подобные задачи. Основной характеристикой каскадной модели является разбиение всей разработки на этапы. Переход с одного этапа на следующий происходит только после того, как будет полностью завершена работа на текущем. Каждый этап завершается выпуском полного комплекта документации.

Положительные стороны применения каскадного подхода заключаются в следующем:

* на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
* выполняемые в логичной последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

В процессе использования этого подхода обнаружился ряд его недостатков:

* реальный процесс создания ПО никогда полностью не укладывался в такую жесткую схему, постоянно возникала потребность в возврате к предыдущим этапам и уточнении или пересмотре ранее принятых решений;
* существенное запаздывание с получением результатов, требования к ИС «заморожены» в виде технического задания на все время ее создания.

Спиральная модель ЖЦ, делающая упор на начальные этапы ЖЦ (анализ и проектирование), лишена вышеизложенных недостатков. На этих этапах реализуемость технических решений проверяется путем создания прототипов. Каждый виток спирали соответствует созданию фрагмента или версии ПО, на нем уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Таким образом, конкретизируются детали проекта и, в результате, выбирается обоснованный вариант, который доводится до реализации. Разработка итерациями отражает объективно существующий спиральный цикл создания системы.

Основная проблема спирального цикла - определение момента перехода на следующий этап. Для ее решения необходимо ввести временные ограничения на каждый из этапов жизненного цикла. Переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. План составляется на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

Сократить, добавить про современные модели

В компании ОАО «СкайТек» разработка ПО ведется согласно спиральному процессу разработки . На рисунке 2.3 представлен обобщенный сценарий процесса разработки ПО в компании ОАО «СкайТек».

Функциональный дизайн

Ключевым моментом разработки информационных систем является Функциональные Требования (ФТ). Для написания функциональных требований компания использует сценарии использования (usecases). Благодаря этому написанные требования легко читаемы и наиболее полно и просто способны отражать пожелания заказчика. Функциональные требования пишутся либо заказчиком самостоятельно, либо совместно с исполнителем.

Разработка ТЗ

На основе функциональных требований компания-разработчик разрабатывает Техническое Задание (ТЗ). Отсутствие ТЗ, согласованного до того, как начнется разработка программного кода, является одним из опаснейших источником проблем. ТЗ также полезно писать на языке сценариев использования и обязательно на языке, понятном заказчику. Это важно, так как ТЗ является основной спецификацией, прикладываемой к контракту. Если оно будет не понятно заказчику, то в дальнейшем практически неизбежны разочарования, когда после окончания всех работ и сдачи проекта выяснится, что заказчик не сможет использовать разработанное ПО. По необходимости разработчики всегда могут написать для своего внутреннего использования другие технически углубленные спецификации.

На основе технического задания разработчики разрабатывают программное обеспечение, а тестировщики - тестовые сценарии. Очень важно чтобы тестовые сценарии были написаны на основе того, что написано в ФТ и ТЗ, а не на основе программных спецификаций, так как тесты должны проверить, насколько качественно выполнено задание заказчика, а не просто, работает ли программа.

В случае сложной системы выполняется модульное тестирование, позволяющее снизить риски сборки системы из полуфабрикатов, и провести затем комплексное тестирование за минимальное количество времени и итераций.

Разработка ПО и тестирование

Разработка корпоративных информационных систем, и особенно разработка приложений, критических для бизнеса, требует, как правило, модульного, нагрузочного, регрессионного и интеграционного тестирования. Создание информационных систем узконаправленной функциональности так же нуждается в адекватном тестировании. Также в разработке новых приложений важно использовать промежуточные демонстрации заказчику работающих прототипов пользовательского интерфейса на как можно более ранних этапах разработки ПО. Это позволит снять возможное непонимание исполнителем замыслов заказчика.

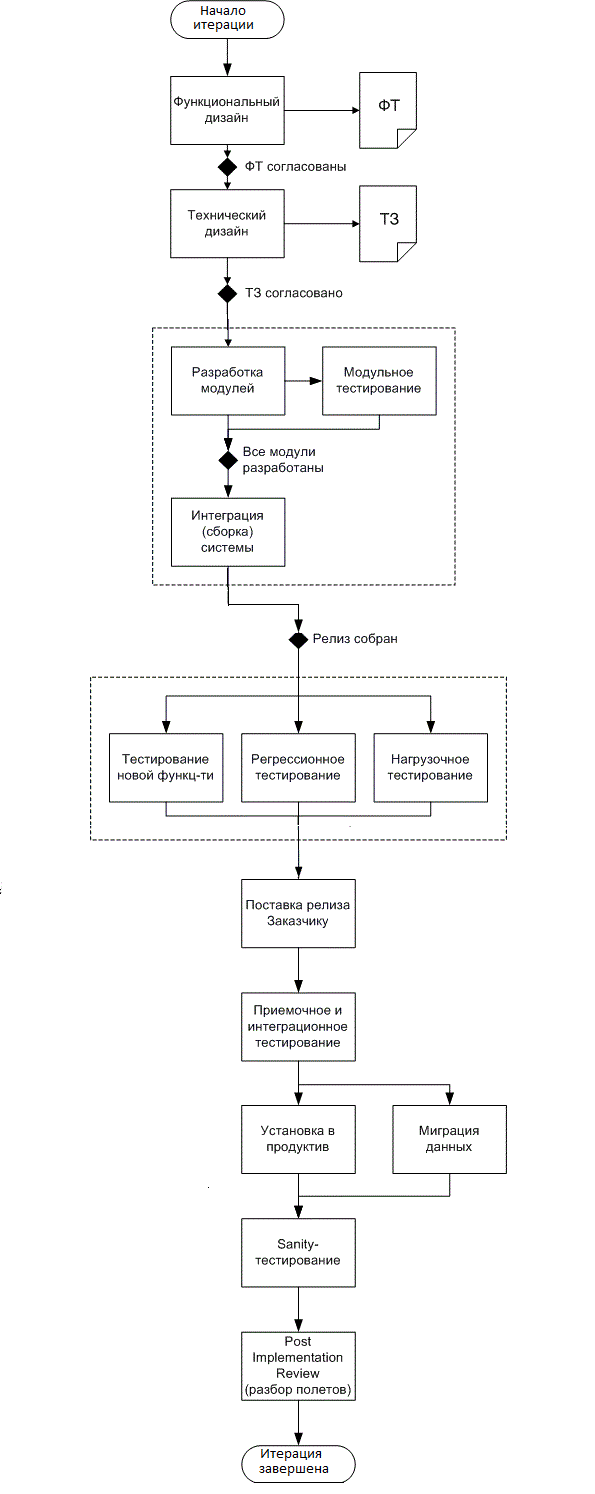


Рисунок 2.3 – Сценарий процесса разработки компании ОАО «СкайТек»

## 2.4 Структурно-функциональная диаграмма организации процесса тестирования компании ОАО «СкайТек»

В рамках представленной выпускной квалификационной работы освещена деятельность сотрудников отдела тестирования компании в процессе разработки программного обеспечения (ПО).

На рисунке 2.5 изображена функциональная модель организации процесса тестирования ПО в компании ООО «СкайТек». На схеме более отражены стадии разработки ПО, касающиеся процесса тестирования (напрямую или косвенно).

Заказчик ПО

Тестировщик

Разработчик

Руководитель проекта

Отдел разработки

Требования к ПО (1)

Функционал на тестирование (5)

Техническое задание ПО (3)

Задача на разработку функционала (4)

Консультация, согласование требований и бюджета (2)

Отчет о тестировании функционала (6а - успешно)

Отчет о тестировании функционала (6б – не успешно)

Отчет о затраченном времени на тестировании функционала (7)

Релизная версия ПО (8)

Оценка заказчика (9)

Рисунок 2.4 – Функциональная модель организации процесса тестирования ПО

Разработка ПО начинается с согласования требований между заказчиком и командой разработчиков под началом руководителя проекта. на разработку ПО между заказчиком и компанией заказчик излагает свои требования назначенному руководителю проекта (1). Руководитель в свою очередь консультирует заказчика по интересующим его вопросам в области технологий разработки, согласовывает требования, макеты дизайна интерфейса для проекта, выясняет неточности и неоднозначные моменты в требованиях, согласовывает бюджет (общий и на отдельные задачи) (2).

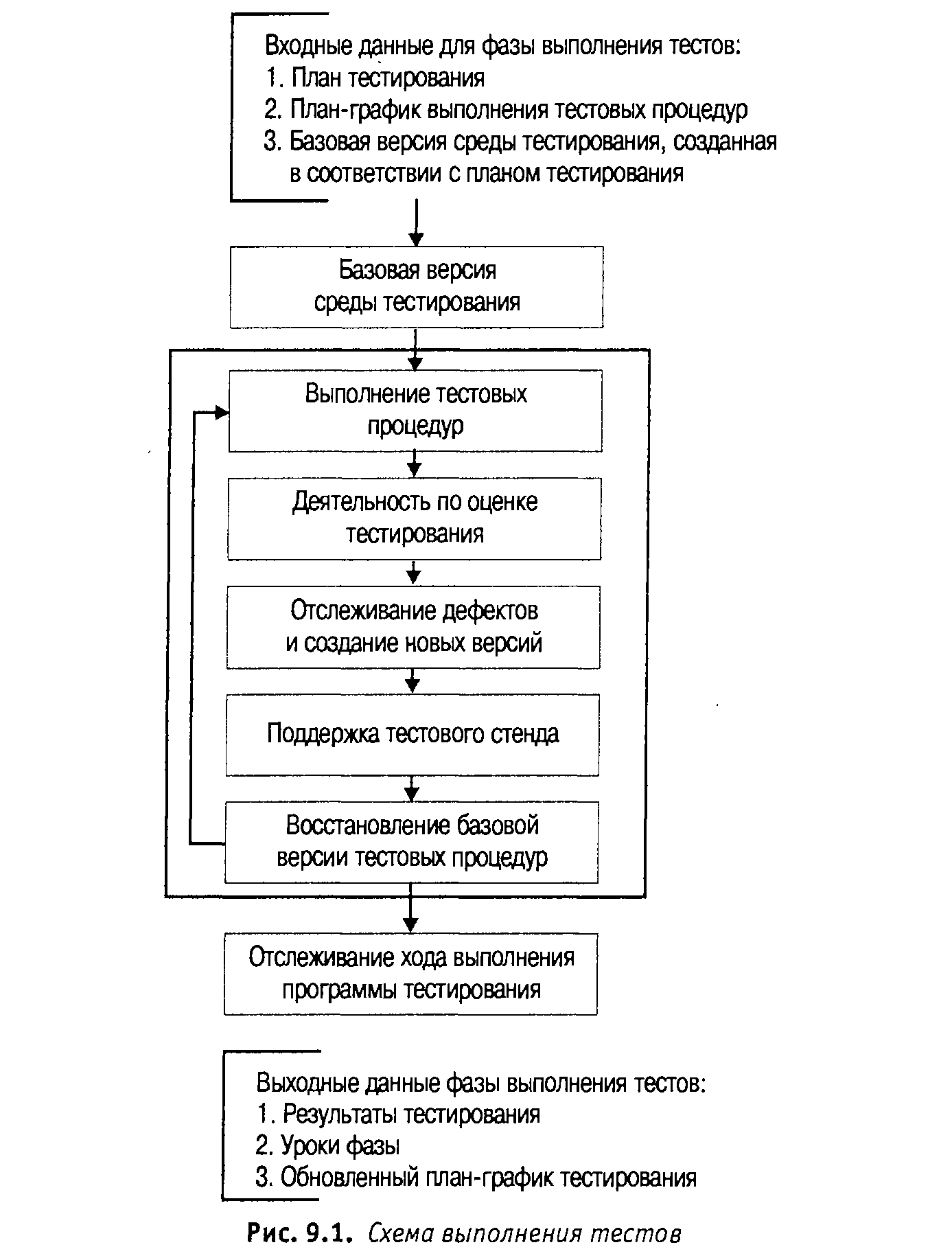
Затем руководителем составляется техническое задание (ТЗ), необходимые шаблоны и файлы, которые публикуются в репозитории проекта и доступны всем участникам проекта (3). На основе ТЗ и приоритетности того или иного функционала руководитель составляет задачи и распределяет их на участников команды (4).

После того, как функционал частично или полностью реализован разработчиком, формируется задача на тестирование этого функционала (5). Тестировщик проводит все необходимые виды тестов и формирует отчет о тестировании и затраченном времени на тестирование. Если функционал успешно проходит испытания, отчет с пометкой «Задача решена» направляется на руководителя (6а), а задача разработчика по реализации этого функционала закрывается. В противном случае – отчет о тестировании с приведенными замечаниями и пометкой «Задача переоткрыта» направляется на разработчика (6б). Разработчику необходимо поправить выявленные замечания (в случае их адекватности и актуальности) и вновь направить задачу на тестирование. Отчет о затраченном времени на тестирование функционала (с учетом всех итераций тестирования в случае обнаружения замечаний) передается руководителю (7).

После того, как все функциональные задачи закрываются, формируетсярелизная версия ПО, которая подвергается регрессионному тестированию и приемочным тестам. Составляются отчеты, устраняются замечания.

После этого в назначенный срок релизная версия передается заказчику (8). На стороне заказчика также осуществляется приемочное тестирование, после чего формируются и направляются руководителю проекта оценки версии, пожелания и замечания (9).

На рисунке 2.4 показана детальная схема проведения тестирования.



На рисунке 2.5 изображена информационная схема организации процесса тестирования. Наиболее приоритетным является документирование всех диалогов и полученных сведений по процессу разработки. При этом общение между заказчиком и руководителем происходит по средством электронной почты и других подобных ресурсов. Часто возникают ситуации общения разработчика и тестировщика напрямую с заказчиком с целью уточнения требований, выяснения специфичных моментов в бизнес-процессах заказчика. Заказчик может выходить на контакт с участниками процесса разработки ПО с целью высказывания дополнительных пожеланий или текущих замечаний.

Отчет о затраченном времени на тестировании функционала (7)

Заказчик ПО

Тестировщик

Разработчик

Руководитель проекта

Отдел разработки

Требования к ПО (1)

Задача на тестирование функционала(5)

Техническое задание ПО (3)

Задача на разработку функционала (4)

Согласованные требования, бюджет (2)

Отчет о тестировании функционала (6а - успешно)

Отчет о тестировании функционала (6б – не успешно)

Релизная версия ПО (8)

Оценка заказчика (9)

письменная информация

устная информация

Пожелания/ замечания к функционалу

Уточнение требований к функционалу

Рисунок 2.6 – Информационная модель процесса тестирования

В компании в процессе разработки формируется большой объем документации по проекту. Некоторые из документов используются на всех этапах разработки ПО, в том числе и на этапе тестирования ПО (например, техническое задание проекта, шаблоны файлов и макеты дизайна интерфейса). Некоторые документы формируются только на этапе тестирования и передаются другим участникам процесса разработки проекта для формирования полного пакета документов по проекту.

В таблице 2.1 приведен перечень внутренних форм документов компании, формируемых на этапе тестирования.

Таблица 2.1 – Внутренние формы документов

|  |  |  |
| --- | --- | --- |
| Наименование документа | Реквизиты документа | Описание |
| Техническое задание |  |  |
| Тест-комплект | Уникальный номер тест-комплекта; автор; приоритет; разработчик; описание (что проверяет тест-комплект) | Совокупность тест-кейсов для тестирования конкретного функционала ПО |
| Тест-кейс | Уникальный идентификатор; приоритет; идея; подготовительная часть; история редактирования (дата, время, автор, что, зачем, почему было изменено); шаги; ожидаемый результат | Описание тестируемой ситуации и ее ожидаемого результата |
| Отчет о тестировании | Уникальный идентификатор задачи; краткое описание; описание и шаги для воспроизведения проблемы (если обнаружена); приложение; автор; дата и время составления; держатель отчета; компонент ПО; версия ПО; билд; комментарии; серьзность ошибки (если обнаружена); приоритет ошибки (с точки зрения бизнес-логики; если обнаружена); список для оповещения; история изменений; тип ошибки; резолюция | Учетная карточка задачи по тестированию функционала с результатами тестирования |
| Отчет о затраченном времени на тестирование | Уникальный идентификатор задачи; деятельность; затраченное время; комментарий | Учетная карточка задачи по тестированию функционала со свежениями о затраченном времени на тестирование |

Тест-кейсы, объединяемые в тест-комплекты для конкретного функционала, составляются преимущественно на основе ТЗ. На рисунке 2.7 представлена форма документа для описания тест-комплекта. Рисунок 2.8 отражает форму документа для описания тест-кейса.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Автор спека: | ID спека: | Приоритет: | Менеджер проекта: | Разработчик: |
| Описание: | | | | |
| Дополнительная информация: | | | | |

Рисунок 2.7 – Внутренняя форма документа «Тест-комплект»

|  |  |  |  |
| --- | --- | --- | --- |
| ID тест-кейса: | Приоритет: | |  |
| Идея: | | | |
| Дополнительная информация: | | | |
| История изменений | | | |
| Дата создания: | Автор: | | Изменения: |
| Выполнение | | | |
| Шаги для выполнения: | | Ожидаемый результат: | |
|  |  | |  |

Рисунок 2.8 – Внутренняя форма документа «Тест-кейс»

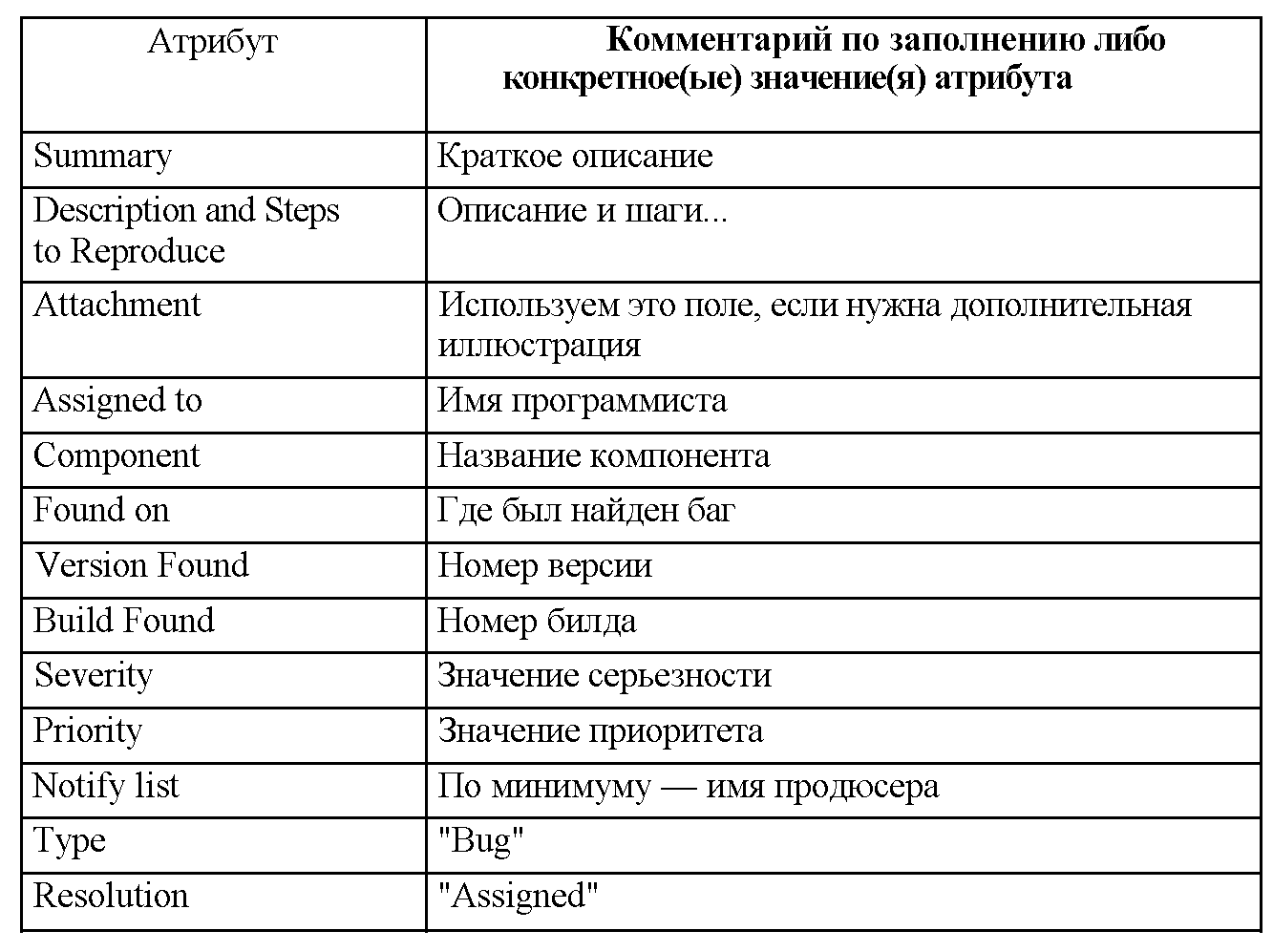


Рисунок 2.9 – Внутренняя форма документ «Отчет о тестировании»

Для организации и ведения процесса тестирования в компании выделено четыре бизнес-процесса:

* формирование тест-комплектов и тест-кейсов для функционала проекта;
* тестирование функционала;
* формирование отчета о тестировании.

НУЖНЫ СХЕМЫ!!!!

3 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

**3.1 Описание проблемы и путей ее устранения**

Современные менеджеры и разработчики программного обеспечения стремятся осуществлять подготовку своих продуктов в минимальные сроки с минимальными ресурсами. Более 90% разработчиков срывают даты поставки. Нарушение сроков носит регулярный характер для 67% разработчиков. Кроме того, в 91% случаев приходилось удалять в цикле разработки ключевую функциональность, чтобы уложиться в срок [1]. В отчете Стендиш-группы содержатся аналогичные данные [2].

В связи получением крупных заказов на разработку и увеличением объемов работ компания ООО «СкайТек» столкнулась с рядом проблем:

* высокая загруженность сотрудников в командах по разработке проектов;
* выход за рамки установленных сроков релизов;
* недостаточное тестирование релизных продуктов и т.д.

Эти и ряд других трудностей приводят к нареканиям со стороны заказчиков, и в конечном итоге могут значительно подорвать конкурентоспособность компании на рынке разработки ПО.

В связи с обозначенными проблемами было принято решение проанализировать все этапы разработки ПО в компании для определения причин возникшей ситуации и поиска решений по устранению последствий этой ситуации, а также предотвращению ее возникновения в будущем. После тщательного анализа были выявлены наиболее проблемные области на этапах разработки и тестирования ПО.

В рамках данной ВКР рассматривается подход, нацеленный на решение найденных проблем в области тестирования ПО. Как уже было упомянуто, в последнее время для ряда проектов было проведено недостаточное по объему и эффективности тестирование, вследствие чего заказчики остались не удовлетворены качеством релизных продуктов.

Для решения обнаруженной проблемы были изучены теоретические основы, применяемые практики в области тестирования. В качестве базового подхода было предложено использование методологии ATML (методология жизненного цикла автоматизированного тестирования), нацеленной на повышение производительности тестирования.

Методология ATLM предназначена для того, чтобы гарантировать успешную реализацию автоматизированного тестирования. Она включает в себя шесть первичных процессов (компонентов):

* принятие решения об автоматизации;
* выбор инструментальных средств тестирования;
* внедрение автоматизированного тестирования;
* планирование, проектирование и разработка тестирования;
* выполнение и управление процессом тестирования;
* оценка и усовершенствование процесса.

Данная ВКР охватывает два этапа этой методологии – планирование (документирование плана тестирования, анализ требований к тестированию, проектирование тестов) и управление процессом тестирования (отслеживание дефектов, отслеживание хода тестирования, оценка тестирования) (не включая разработку и выполнение тестов). Это обосновано тем, что выделенные этапы успешно могут быть реализованы и для проектов, для которых автоматизация тестирования не предусмотрена, но, тем не менее, требуется повысить эффективность тестирования разрабатываемого продукта.

Независимо от принятой методологии процесса разработки ПО тестирование играет в нем ключевую роль, решая следующие задачи:

* выявление дефектов (ошибок, недоработок, неполноты и пр.) различных артефактов разработки ПО (требований, проектных решений, документации или кода);
* выявление наиболее критичных и наиболее подверженных ошибкам частей создаваемой или сопровождаемой системы;
* контроль и оценка качества ПО во всех его аспектах;
* предоставление всем заинтересованным лицам (руководителям, заказчикам, пользователям и пр.) информации о текущем состоянии проекта и характеристиках его результатов;
* предоставление руководству проекта и разработчикам информации для планирования дальнейших работ, а также для принятия решений о продолжении проекта, его прекращении или передаче результатов заказчику.

На сегодняшний день тестирование - гораздо больше, чем просто фаза в конце цикла разработки; оно имеет собственный цикл. В течение всего ЖЦ разработки ПО должны проводиться тщательные и обоснованные испытания. Это означает, что процесс тестирования начинается с фазой разработки требований и продолжается параллельно всему процессу разработки ПО [3].

Раннее включение в работы по проекту позволяет команде тестировщиков лучше понять потребности заказчика, подлежащие реализации, что поможет при разработке архитектуры для создания соответствующей среды тестирования и более тщательного проекта тестирования. Это также обеспечивает ранее обнаружение ошибок и предотвращает их миграцию из спецификации требований в проект и, следовательно, из проекта в код. Это снижает стоимость, минимизирует повторное выполнение работ и сьереает время. [avtotestpo]

ИТ-организации стремятся сбалансировать качество, стоимость и график разработки ПО в целом и для отдельных этапов разработки в частности. В общем случае при планировании и организации тестирования следует искать компромиссное решение, учитывающее два противоречивых требования: обеспечение максимальной достоверности обобщенной оценки качества ПО и выполнение испытаний в ограниченное время с использованием ограниченных ресурсов.

Таким образом, ключевыми средствами для достижения поставленной цели компании – повышение качества разрабатываемого ПО (на этапе тестирования ПО) – должны стать:

* планирование процесса тестирования (фаза планирования тестирования);
* сбор метрик тестирования (фаза управления тестированием);
* сбор метрик ПО (для оценки качества ПО) (фаза управления тестированием).

**3.2 Общие сведения о задачах**

Фаза планирования тестирования включает в себя обзор долговременных работ по планированию тестирования. На этом этапе команда тестировщиков определяет стандарты и основные направления создания процедур тестирования; аппаратные, программные и сетевые средства, необходимые для поддержки тестовой среды; требования к данным для тестирования; предварительный план-график тестирования; требования по оценке производительности; процедуру управления конфигурацией и средой тестирования; процедуру отслеживания дефектов и средства для его проведения.

На фазе проектирования тестирования определяются количество тестов, которые нужно выполнить, стандарты проектирования тестов и условия тестирования, которые должны соблюдаться.

На фазе управления должны оцениваться итоги выполнения тестирования таким образом, чтобы избежать неправильных положительных или отрицательных оценок. [avtotestpo]

Управление процессом тестирования сопряжено с регулярной подготовкой следующих типов документов (таблица 3.1).

Таблица 3.1 – Документация на фазе управления тестированием

|  |  |
| --- | --- |
| Документ программы тестирования | Описание |
| План тестирования | Документ, содержащий информацию о планировании тестирования |
| Матрица отслеживания требований | Матрица, показывающая покрытие требований тестовыми процедурами, определяющая метод проверки для каждого требования |
| Тестовые сценарии | Описание шагов для тестирования определенной функциональности |
| *Отчет о ходе разработки тестов* | *Отчеты, показывающие состояние хода работ по созданию тестовых процедур* |
| *Отчет о готовности к тестированию* | *Отчет, показывающий готовность к проведению приемо-сдаточных испытаний* |
| Отчеты о ходе тестирования | Отчеты, описывающие ход исполнения тестов |
| Отчеты об отслеживании дефектов | Отчеты, описывающие число и серьезность дефектов |
| Отчет о состоянии функциональных характеристик | Отчеты, показывающие, насколько система удовлетворяет заданным требованиям |
| Отчет о тестировании | Отчет, содержащий результаты тестирования |

Уровень формализации каждой из стадий тестирования зависит от способа организации группы тестирования или от спецификаций, определенных в требованиях заказчика или конечного пользователя. Группа тестирования должна четко определить всю необходимую документацию программы тестирования в плане тестирования, а затем обеспечить разработку документации согласно плану-графику.

Краеугольный камень эффективной программы тестирования — планирование тестирования. Элемент методологии ATLM, связанный с планированием тестирования, включает в себя обзор всех работ по тестированию. Какими бы ни были проекты, в которых предполагается тестирование, необходимо определить цели и задачи тестирования, зафиксировать требования к тестированию, разработать стратегии, обеспечивающие выполнение установленных требований к тестированию. Основные события и виды деятельности по тестированию должны быть включены в план-график проекта.

На проведение испытаний ПО приходится затрачивать значительные трудовые и материальные ресурсы. Сроки проведения испытаний всегда ограничены. Поэтому перед испытателями всегда стоит задача поиска путей минимизации затрат (материальных, трудовых и временных ресурсов) для достижения цели испытания. Для реализации этой задачи необходимо установить критерии завершенности испытаний, которые могут служить основой для принятия решения о завершении испытаний.

Чаще используют критерий, основанный в значительной степени на здравом смысле и информации о количестве ошибок, полученных в процессе тестирования. Для этого строят график зависимости количества ошибок и времени их появления. По форме полученной кривой можно определить, стоит продолжать тестирование или нет. Если с увеличением времени тестирования число ошибок растет, то естественно, что тестирование необходимо продолжать. Если в процессе тестирования в определенный момент наступило снижение числа выявленных ошибок, постепенно стремится к нулю или достигло нуля, то процесс тестирования можно завершать [1].

Более прогрессивным может считаться подход, при котором для определения критерия завершенности тестирования используются количественные показатели надежности, рассчитываемые по моделям надежности.

Современная программная индустрия за полвека исканий накопила внушительную коллекцию моделей и метрик, оценивающих отдельные производственные и эксплуатационные свойства ПО. Однако погоня за их универсальностью, неучет области применения разрабатываемого ПО, игнорирование этапов жизненного цикла программного обеспечения и, наконец, необоснованное их использование в разноплановых процедурах принятия производственных решений, существенно подорвало к ним доверие разработчиков и пользователей ПО [6].

Сбор и оценка метрик — это вид деятельности, связанный с выполнением программы тестирования и составлением отчетов о ходе тестирования. Успех любого процесса по созданию ПО, в котором контроль за параметрами выполняется вручную и требует больших трудозатрат, будет ограниченным. В современном процессе разработки наиболее важными параметрами ПО являются простые и объективные измерения того, как продукт и проект изменяются с различных точек зрения.

Метрики ПО – меры, которые используются для количественной характеристики программного обеспечения, ресурсов разработки программного обеспечения, и /или процесса разработки программного обеспечения. Это включает в себя элементы, которые являются непосредственно измеримыми, например как количество строк кода, а также предметы, которые рассчитывается на основании измерений.

Необходимо иметь возможность измерять качество программных продуктов и прогресс на пути к достижению целей проекта на протяжении всего цикла разработки ПО. Целью метрик ПО является обеспечение команды разработчиков и управляющей команды:

* точной текущей оценкой прогресса;
* уровнем качества изменяющегося программного продукта;
* основой для приблизительной оценки затрат и времени, необходимых для получения конечного продукта, с учетом того, что точность оценки увеличивается во времени.

На сегодняшний день не существует универсального набора метрик, подходящего для любого проекта ПО. Трудности заключается в определении, какие показатели имеют важное значение для компании и тестируемого программного продукта, и какие процедуры наиболее эффективны для сбора и использования этих показателей.

## 3.3 Декомпозиция комплекса задач

### *3.3.1 Планирование тестирования*

Фаза планирования тестирования сосредотачивает внимание тестировщика прежде всего на подготовке документации по проведению тестирования, на планировании работ по достижению целей и решению задач тестирования, на подготовке среды тестирования, а также на создании документа, фиксирующего план тестирования.

План тестирования содержит большой объем информации, включая необходимые документы по тестированию в проекте. В плане тестирования определяются роли и ответственность сотрудников группы тестирования, план-график проекта, работы по проектированию тестов, подготовка тестовой среды, риски и непредвиденные обстоятельства, связанные с тестированием, и приемлемый уровень глубины тестирования. Приложения к плану тестирования могут содержать описания тестовых процедур, соглашений по именованию и матрицу соответствия требований и тестовых процедур.

В приложении ??? приведен эскиз плана тестирования, отражающий основные разделы этого документа. Детальное описание того, на основе каких данных формируются разделы, изложено во главе ????.

### *3.3.2 Оценка результатов*

В фазе управления группе тестирования после выполнения тестовых процедур необходимо оценить их результаты. Сотрудники рассматривают и анализируют результаты комплексного и системного тестирования, а также приемо-сдаточных испытаний. В качестве базовых, входных данных для каждой фазы тестирования выступает соответствующий комплект тестовых процедур. На выходе каждой фазы тестирования — достигнутые или модифицированные критерии завершения тестирования, как определено в плане тестирования. На каждой фазе тестирования составляют отчеты о проблемах. Исправление ошибок разработчиками сопровождается документирование внесенных исправлений в соответствующем отчете о проблеме.

## 3.4 Способы решения задачи

### *3.4.1 Планирование тестирования*

Важным элементом ЖЦ тестирования является его планирование. Планирование тестирования включает в себя обзор всех работ по тестированию. Основные события и виды деятельности по тестированию должны быть включены в план-график проекта.

Какими бы ни были проекты, в которых предполагается тестирование, необходимо определить цели и задачи тестирования, зафиксировать требования к тестированию, разработать стратегии, обеспечивающие выполнение установленных требований к тестированию.

Для эффективного применения средств тестирования необходимо осуществить планирование и подготовку тестирования. План тестирования содержит большой объем информации, включая необходимые документы по тестированию в проекте. В плане тестирования определяются роли и ответственность сотрудников группы тестирования, план-график проекта, работы по проектированию тестов, подготовка тестовой среды, риски и непредвиденные обстоятельства, связанные с тестированием, и приемлемый уровень глубины тестирования. Приложения к плану тестирования могут содержать описания тестовых процедур, соглашений по именованию и матрицу соответствия требований и тестовых процедур [4].

Разделы плана тестирования:

1. вводный раздел - содержит описание цели составления плана, истоки проекта, описание системы, диаграмму организации проекта. Перечисляются все документы, относящиеся к проведению тестирования (спецификации требований и архитектуры, руководство пользователя, руководство по режимам работы системы, стандарты графических интерфейсов, стандарты кодирования, требования по сертификации системы и другая информация о проекте). Источники: план разработки приложения, план развития системы, планы миграции данных, план управления разработкой системы и план управления проектом.
2. Оценка и выбор инструмента для поддержки тестирования - фиксируются причины выбора одного или нескольких инструментов тестирования, их функции, назначение в проекте и причины выбора.
3. Анализ процесса тестирования - уточнение целей, задач и стратегий тестирования.
4. Рамки работ по тестированию - выстраивают структуру разбиения работ, в которой на одном уровне определяются категории работ по тестированию, а на другом уровне — подробные описания работ. Структура разбиения работ обычно используется в сочетании с хронометражем для определения времени выполнения каждого из этапов тестирования.
5. Результаты работы по оценке затрат на тестирование. Оценка затрат может определять число сотрудников группы тестирования в проекте в часах или в количестве людей, если на выполнение определенного объема работ выделяется некоторый известный срок. По возможности в план тестирования помещаются такие оценки затрат, как планируемое число тестовых процедур и тестовых скриптов.
6. Качества и навыки сотрудников, необходимые для проведения тестирования.
7. Роли и ответственность сотрудников группы тестирования.
8. Требования к тестированию. Документация для проведения тестирования.
9. Допущения, предварительные условия и риски тестирования - включаются все события, действия или обстоятельства, которые могут помешать выполнению тестирования в срок, например поставка тестового оборудования с опозданием или задержка в получении приложения для тестирования.
10. Функции, разработка которых имеет большое значение для успеха проекта, и функций, разработка которых связана с наибольшим риском.
11. Матрица соответствия тестовых процедур и требований - позволяет группе тестирования проследить покрытие требований тестовыми процедурами. Каждому системному требованию ставится в соответствие метод проверки.
12. Требования к аппаратному, сетевому и программному обеспечению - это позволяет создать тестовую среду, являющуюся зеркальным отражением среды приложения, предназначенного для тестирования.
13. Общий подход к проектированию тестов.
14. Описания всех необходимых тестовых процедур. Описание тестовых процедур включает в себя определение комплекта тестовых процедур, который будет разрабатываться и выполняться в ходе тестирования. На этапе проектирования производится объединение тестовых процедур в логические группы и определение соглашений по именованию для комплекта процедур. При составлении плана тестирования определяют требования к тестовым данным и средства для их получения, генерации или разработки.
15. График разработки/выполнения тестов - составляется группой тестирования с целью определения временных рамок разработки и выполнения различных тестов. При составлении графика учитываются любые зависимости между тестами и включаются работы по настройке тестов, последовательностей тестовых процедур и по очистке среды.
16. Руководства по разработке тестов, которые будут использоваться на фазе разработки тестов. Руководства должны быть подготовлены как для тестовых процедур, создаваемых вручную, так и для автоматизированных тестовых процедур.
17. В плане тестирования должны быть отражены работы по выполнению тестов, в том числе разработка плана-графика и переход от одной фазы тестирования к другой.
18. Метрики тестирования, которые будут собираться во время жизненного цикла тестирования. Метрики тестирования являются ключевыми индикаторами покрытия тестами требований, хода и качества тестирования.
19. Предварительный план-график тестирования. После завершения анализа работ по тестированию и детализации планов тестирования необходимо уточнить и расширить план-график. Группа тестирования обязана уточнять любые обновления плана-графика разработки для обеспечения согласованности с графиком тестирования.

После того как план тестирования составлен и уточнен так, что он полностью документирует подходы к проведению тестирования, он становится руководящим средством, обеспечивающим успех тестирования.

Планирование тестирования — не единовременный акт, а процесс. План тестирования — это документ для проведения тестирования от начала до конца, и необходимо, чтобы он отражал все изменения в проекте [4].

*3.4.2 План график тестирования*

Основные события, работы и документы должны быть представлены в плане-графике, содержащем рубежи программы тестирования (рисунок 4.1). Работы на этом уровне распределяются на основе их логического порядка следования.

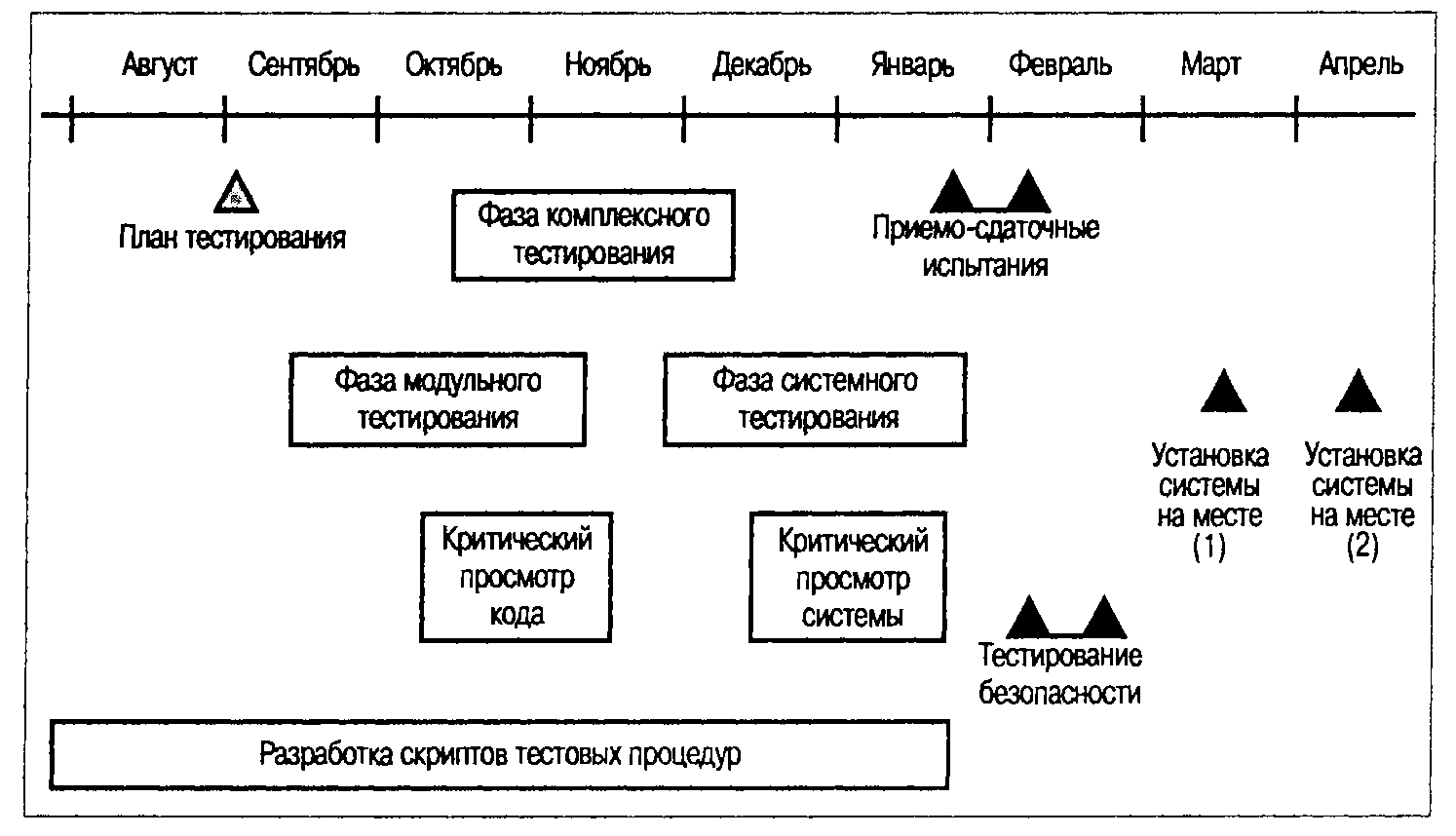


Рисунок 4.1 - План-график, содержащий основные рубежи программы тестирования

Каждая задача этого уровня декомпозируется на составные части - подзадачи. Например, в таблице 4.2 приводится пример плана- графика приемо-сдаточных испытаний.

Таблица 4.2 - План-график тестирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Описание задания | Продолжитель-ность (день) | Начало | Окончание | Ответственный |
| 1 | Распределение обязанностей тестировщиков при проведении приемо-сдаточных испытаний | 1 | 11.01 | 12.01 | Т1 |
| 2 | Определение методов критического просмотра и отчетности | 1 | 12.01 | 13.01 | Т1 |
| ………. | | | | | |
| 12 | Разработка тестовых процедур для приемо-сдаточных испытаний | 35 | 13.01 | 2.03 | Т2, Т4, Т5 |
| ………. | | | | | |

При этом учитывается приоритеты подзадач задач с точки зрения пользователя и приоритеты выявленных дефектов. Каждой подзадаче присваивается исполнитель. На основе прогнозных данных (оценка эксперта (в роли него может выступать проект-менеджер, ведущий тестировщик, руководитель отдела тестирования/ отдела качества), оценка методом UCP (глава ?????)) устанавливается продолжительность выполнения подзадач.

После проведения тестовых испытаний, план-график дополняется фактическими данными о продолжительности и датах выполнения задач и подзадач.

Для построения плана-графика тестирования было принято использовать следующий алгоритм: (блок схема)

Вход

Наименование задачи, суть задачи

Временные рамки задачи (устанавливаются совместно на этапе согласования с заказчиком+можно 2.2.3.3.)

Приоритет функционала задачи (с точки зрения бизнеса)

Назначение ответственного за функционал

Разбиение на подзадачи, суть подзадачи

Приоритет функционала подзадачи (с точки зрения бизнеса)

Если был выявлен дефект – Приоритет дефекта (с точки зрения разрабатываемого проекта)

Временные рамки

*Назначение исполнителя на подзадачу (с учетом занятости тестировщика)*

*3.4.3 Показатели для оценки продолжительности (трудозатрат) тестирования ПО*

На практике нет выделенных подходов для оценки трудозатрат на этапе тестирования ПО. Представленные методики являются заимствованными из методов оценки трудозатрат ПО в целом [8].

Были проанализированы следующие подходы для оценки трудозатрат на разработку ПО:

* экспертная оценка;
* оценки с помощью моделей:
* оценка по аналогии;
* UseCasePoints (UCP);
* FunctionPoints (FP);
* FastFunctionPoints (FFP);
* EarlyFunctionalPoints (EFP).

Для оценки трудозатрат на этапе тестирования наиболее эффективными для применения (с учетом практики тестирования в компании ОАО «СкайТек») были выделены следующие подходы:

* экспертная оценка;
* оценки с помощью моделей:
* оценка по аналогии;
* UseCasePoints (UCP).

*Экспертная оценка*

Методика «Экспертной оценки» предназначена для оценки тестирования проектов с опорой на знания и опыт экспертов. Суть методики состоит в том, что оценка проекта в целом или его отдельных частей проводится экспертами (в их роли обычно выступают проект-менеджеры или тестировщики высокого уровня подготовки). Результаты оценки выражаются в трудозатратах или длительности процесса тестирования и определяется как процент от общих трудозатрат/ длительности создания ПО. Преимуществами такой оценки являются универсальность и применимость к проектам любого типа. Ограничения применимости данной методики складывается из зависимости от наличия и квалификации экспертов и «человеческого фактора» эксперта.

В рамках «Экспертной оценки» может быть использован PERT анализ (Program, Evaluation, and Review Technique). Это техника оценки ожиданий в отношении длительности задач проекта, проводимая на основе определения среднего весового значения трех оценок длительности - пессимистической, оптимистической и ожидаемой (то есть наиболее вероятной, при первичной оценке).

Для каждого ПО даются три оценки трудозатрат тестирования: *Оi* – оптимистичная (минимально возможные трудозатраты); *Рi* – пессимистичная (максимально возможные трудозатраты); *Mi* – наиболее вероятная (при текущем уровне понимания задачи, квалификации исполнителей и т.п.).

Тогда оценка средней трудоемкости работы равна:

*Еi = (Pi + 4Mi + Oi)/6.* (4.1)

Суммарная трудоемкость равна (если тестирование разбито на этапы):

, (4.2)

где *n –* количество этапов тесттирования.

*Оценка по аналогии*

Методика предусматривает оценку проекта на основании исторических данных. По сути – это автоматизированная версия экспертной методики.

Суть методики:

* оценка проекта на основании его «измерения» в формах, отчетах, подсистемах, сущностях и т.п.;
* пересчет результатов измерений в трудозатраты и т.п. согласно накопленной статистики;
* выходные данные - трудозатраты.

Плюсы:

* устраняет недостатки экспертной методики;
* при накоплении исторической информации точность растет.

Ограничения применимости:

* необходимо наличие исторических данных и их обработка;
* слабо применима для новых проектов, новых бизнес-областей;
* при смене технологических платформ или команды требует переоценка накопленных статистических данных.

*UseCasePoints*

Методика предназначена для оценки проектов, для которых применяется определение требований с помощью сценариев использования (UseCases) [16].

Суть методики:

* выявление акторов (actors) и сценариев (прецедентов) использования, их «взвешивание» (оценка сложности);
* выходные данные – трудозатраты.

Плюсы:

* хорошо подходит для стандартных информационных систем (много пользовательского интерфейса, мало сложных алгоритмов);
* легко подстраивается под производительность команды или среднюю производительность по компании.

Актор (actor) – это любая внешняя по отношению к моделируемой системе сущность (человек, программная система, аппаратное устройство), которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач.

Сценарий использования (usecase) - некоторая функция, реализуемая системой (и имеющая ценность для конечного пользователя); набор возможных последовательностей взаимодействия между системой и одним из акторов (действия актора и отклики системы), которое переводит систему из одного стабильного состояния в другое, так что разница ощутима для одного или нескольких акторов.

Суть методики:

* оценка сложности акторов, взаимодействующих с системой;
* оценка сложности функциональных требований к системе;
* учет влияния нефункциональных требований к системе;
* учет влияния условий выполнения проекта;
* расчет трудозатрат на реализацию проекта.

Сложность акторов:

* простой - взаимодействует с системой посредством API;
* средний - взаимодействует с системой по определенному протоколу (TCP/IP, HTTP, SOAP и т.п.) или является накопителем данных (datastore) (файл, СУБД);
* сложный - взаимодействует с системой посредством GUI.

Сложность сценариев использования: простой - не более 3 шагов; средний - 4-7 шагов; сложный - более 7 шагов. Если шагов сценария больше, то формально оценка может быть экстраполирована на большее количество шагов. Но делать это не рекомендуется (это может быть признаком недостаточной детальности описания требований). Лучше сделать декомпозицию сценария.

Учет влияния нефункциональных требований производится оценкой 13 «технических факторов» (TCF): распределенность системы; время отклика; эффективность работы пользователя; сложные внутренние расчеты / обработка данных; повторное использование кода; простота установки; удобство использования; переносимость, кросс-платформенность; простота изменений работы системы пользователями; доступ со стороны одновременно работающих пользователей; безопасность; прямой доступ из внешних приложений; наличие специальной функциональности для обучения пользователей.

Учет влияния условий выполнения проекта производится оценкой 8 «факторов условий выполнения» (EF): знакомство с процессом разработки; знакомство с приложением (бизнес-доменом, предметной областью); владение технологиями разработки; уровень ведущего аналитика; мотивация команды; стабильность требований; персонал, работающий в режиме parttime (частичной занятости в проекте); опыт команды в языке программирования.

Формулы дляTF EF

UCP определяется по формуле:

*UCP = (сложность акторов + сложность сценариев) TCF EF.* (4.3)

Трудозатраты определяются по формуле:

*Трудозатраты (в человеко-часах) = UCP PF,* (4.4)

где *PF* (PerformanceFactor, количество человеко-часов на 1 UCP).

Автор методики рекомендует использовать PF = 20 чел/часов на 1 UCP. Этот показатель можно варьировать в пределах [15;30] чел/часов на 1 UCP в зависимости от оценки квалификации программиста. Чтобы перенести эту модель оценки на тестирование, было принято использовать среднее значение PF тестировщика равным 2 чел/часов на 1 UCP. Данные взяты на основе статистики компании «СкайТек». В зависимости от квалификации, это значение может варьироваться в пределах [0,5; 3].

Ограничения применимости:

* для проведения оценки нужен аналитик;
* требуется выделение «стандартных» (относительно простых) сценариев использования;
* оценка сложности выполняется экспертным путем;
* точность зависит от опыта аналитика и корректности коэффициента пересчета UseCasePoints в человеко-часы;
* не все системы можно оценивать по UCP (сложные алгоритмы и т.п.).

Данная методика была выбрана основной для автоматизации оценки трудозатрат на тестирование ПО в силу того, что в компании применяется методика написания тестов на основе сценариев.

### *3.4.4 Оценка качества тестирования*

Для оценки качества тестирования формируется отчет о ходе тестирования. Его основу составляет расчет вторичных метрик тестирования, которые предоставляют возможность количественно оценить прогресс тестирования. Расчет показателей – вторичных метрик – должен производиться на основе необходимых входных данных (первичные метрики, т.е. показатели, получаемые непосредственно в процессе тестирования – количество найденных дефектов, затраченное время и т.д.)

В приложении Б представлены различные метрики тестирования, которые следует использовать в ходе тестирования методами «черного ящика». Каждая метрика относится к одной из трех категорий: к покрытию, ходу работ по тестированию или качеству [3]. Эта группа метрик была выбрана по причине того, что в компании отдел тестирования отвечает за проведение только тестирования методами черного ящика.

Для полноценной картины о качестве и результатах проведения тестирования необходимо накапливать расчет по вторичным метрикам на каждой итерации тестирования, тем самым обеспечивая возможность сравнения полученных метрик в динамике.

Если на проекте внедрена автоматизация тестирования, важно проводить измерения, показывающие стоимость автоматизации, особенно в тех случаях, когда в проекте впервые прибегают к автоматизации тестирования. Группа тестирования должна измерить, сколько времени было потрачено на разработку и выполнение автоматизированных скриптов, и сопоставить это с полученными результатами. Например, группа тестирования может сравнить количество часов, потребовавшихся на разработку и проектирование тестовых процедур, с количеством документированных дефектов, которые, вероятнее всего, не были бы выявлены при проведении тестирования вручную.

### *3.4.5 Оценка надежности ПО*

*Оценка надежности ПО по ГОСТу 28195–99*

Под показателем качества продукции в ГОСТ 28195–99 подразумевается количественная характеристика одного или нескольких свойств продукции, составляющих ее качество, рассматриваемая применительно к определенным условиям ее создания и эксплуатации или потребления. Основу описываемого метода оценки надежности составляет четырехуровневая иерархическая модель надежности [7].

Терминология для показателей качества каждого уровня:

* уровень 1 - факторы качества (в терминологии, принятой в международных стандартах, соответствуют характеристикам качества, в данном случае это относится к надежности));
* уровень 2 - критерии качества (в международной терминологии – подхарактеристики качества, в данном случае надежности);
* уровень 3 - метрики (соответствует международной терминологии);
* уровень 4 - оценочные элементы или единичные показатели (данный уровень в международных стандартах отсутствует).

Для фактора надежности составляется четырехуровневая иерархическая модель, отражающая ее взаимосвязь с критериями, метриками и оценочными элементами. Вид данной модели зависит от фазы жизненного цикла ПС.

В рисунке 4.1 приведены три верхних уровня иерархической модели фактора надежности для фазы реализации, тестирования, изготовления и сопровождения. Номера на рисунке соответствуют номерам метрик данного фактора.



Рисунок 4.1 - Модели фактора надежности для фазы реализации, тестирования, изготовления и сопровождения

Оценка надежности программного средства проводится в следующей последовательности:

1. Для показателей надежности на всех уровнях принимается единая шкала оценки (от 0 до 1).

2. В процессе оценки надежности на каждом уровне (кроме уровня оценочных элементов) проводится вычисление двух величин - абсолютного показателя и относительного показателя , где *j* – порядковый номер показателя данного уровня для *i*-го показателя вышестоящего уровня.

Относительный показатель является функцией показателя и его базового значения и определяется по формуле:

. (4.5)

ГОСТ 28195–99 содержит таблицу, содержащую базовые значения для показателей надежности второго уровня (критериев). Данные значения определяются подклассом ПС. Базовые значения для надежности и ее показателей 3-го уровня формируются методом экспертного опроса с учетом назначения ПС или на основании показателей существующих аналогов или расчетного эталонного ПС. Значения базовых показателей ПС должны соответствовать значениям показателей, отражающих современный уровень качества и прогнозируемый мировой уровень.

3. Каждый показатель надежности второго и третьего уровней характеризуется двумя параметрами - количественным значением и весовым коэффициентом .

Сумма весовых коэффициентов всех показателей некоторого уровня, относящихся к показателю вышестоящего уровня, постоянна и равна 1:

, (4.6)-

где *J* – общее количество всех показателей *j*-го уровня, относящихся к *i*-му показателю вышестоящего уровня, определенных в стандарте.

ГОСТ 28195–99 содержит таблицы, содержащие перечни весовых коэффициентов для показателей второго и третьего уровней (критериев и метрик). Количественные величины весовых коэффициентов зависят от фазы жизненного цикла ПС и подкласса ПС.

4. Определение усредненной оценки оценочного элемента по нескольким его значениям (измерениям) осуществляется по следующей формуле (формула для вычисления значений показателей надежности 4-го уровня):

, (4.7)

где *k* – порядковый номер метрики;

*q* – порядковый номер оценочного элемента;

*T* – число значений (измерений) оценочного элемента;

*t* – номер значения оценочного элемента.

5. Итоговая оценка *k*-ой метрики *j*-го критерия определяется по формуле (формула для вычисления значений показателей надежности 3-го уровня):

, (4.8)

где *М* – признак метрики;

*Q* – число оценочных элементов, реально используемых при оценке *k*-й метрики.

6. Абсолютные показатели *j*-го критерия надежности вычисляются по формуле (формула для вычисления значений показателей надежности 2-го уровня):

, (4.9)

где *n* – число метрик, относящихся к *j*-му критерию, реально используемых при оценке;

*К* – признак критерия.

7. Относительные значения *j*-го критерия надежности по отношению к базовому значению определяются по формуле:

. (4.10)

8. Абсолютные и относительные значения надежности определяются по формулам:

, (4.11)

(4.12)

где Ф – признак фактора;

*N* – число критериев надежности, реально используемых при оценке.

В приложении В представлены оценочные элементы фактора «Надежность».

*Модель надежности программных средств по стандарту ISO/IEC 9126-1:2001*

В стандарте ISO/IEC 9126-1:2001 определены две части модели качества. Это модель внутреннего и внешнего качества и модель качества в использовании. Модель внутреннего и внешнего качества используется в процессе разработки ПС (программного средства). При этом в модели внутреннего качества применяются внутренние метрики качества, в модели внешнего качества – внешние метрики. На двух верхних уровнях (уровне характеристик и подхарактеристик) различия между внутренним и внешним качеством в модели отсутствуют [7].

Надежность ПС является одной из шести основных характеристик, регламентированных в модели внутреннего и внешнего качества. В модели качества в использовании характеристика Надежность не определена. В модели внутреннего и внешнего качества Надежность (Reliability) определяется как способность программного продукта поддерживать заданный уровень качества функционирования при его использовании в заданных условиях. Ограничения надежности в процессе эксплуатации вызваны ошибками в требованиях, проектировании и кодировании.

Надежность включает четыре подхарактеристики - завершенность, устойчивость к ошибке, восстанавливаемость, соответствие надежности.

Для внутренней и внешней моделей надежности отличия наблюдаются на уровне применяемых метрик.

Внутренние метрики надежности используются во время разработки программного средства для предсказания того, удовлетворяет ли ПС заявленным потребностям в надежности.

Внутренние метрики устойчивости к ошибке определяют набор атрибутов для оценки способности программного продукта поддерживать желаемый уровень качества функционирования в случае ошибок выполнения или нарушения заданного интерфейса.

Внутренние метрики восстанавливаемости определяют набор атрибутов для оценки способности программного продукта восстанавливать соответствующий уровень качества функционирования и данные, поврежденные в случае отказа.

Внутренние метрики соответствия надежности определяют набор атрибутов для оценки способности программного продукта соответствовать стандартам, соглашениям или нормативам организации пользователя, связанным с надежностью.

Приложение Г содержит внутренние метрики подхарактеристик надежности ПС, рекомендованные стандартом ISO/IEC TR 9126–3:2003 [7].

Внешние метрики надежности должны измерять свойства, связанные с поведением системы, содержащей ПС, во время тестирования, чтобы показать степень надежности ПС в этой системе в процессе эксплуатации. Внешние метрики завершенности должны позволять измерять такие атрибуты, как отсутствие отказов в ПС, вызываемых ошибками, существующими в самом ПС.

Внешние метрики устойчивости к ошибке должны быть связаны со способностью ПС поддерживать заданный уровень качества функционирования в случаях ошибок выполнения или нарушения заданного интерфейса.

Внешние метрики восстанавливаемости должны позволять измерять такие атрибуты, как способность ПС в системе восстанавливать соответствующий уровень качества функционирования и данные, поврежденные в случае отказа.

Внешние метрики соответствия надежности должны позволять измерять такие атрибуты, как количество функций или случаев с проблемами соответствия, вызывающими неудовлетворение требований стандартов, соглашений или нормативов, связанных с надежностью.

Приложение Д содержит внешние метрики подхарактеристик надежности, рекомендуемые стандартом ISO/IEC TR 9126–2:2003 [7].

*2.3 Обоснование проектных решений по автоматизированному решению экономико-информационных задач*

*включают обоснование выбора задач, входящих в комплекс, обоснование необходимости использования вычислительной техники и создания АРМ для решения данного комплекса задач, анализ существующих аналогов автоматизированных систем для решения данного комплекса задач.*

*Затем необходимо указать какой тип платформы будет задействован для решения данного комплекса задач, обосновав при этом экономическую целесообразность использования вычислительной техники. При рассмотрении недостатков, присущих существующему состоянию дел на предприятии, целесообразно акцентировать внимание на тех из них, устранение которых предполагается осуществить в проекте.*

*Наиболее распространенными недостатками являются:*

* *невозможность расчета показателей, необходимых для управления объектом, из-за сложности вычислений или чрезмерного объема информации;*
* *большая трудоемкость обработки информации (привести объемно- временные параметры);*
* *низкая оперативность, снижающая качество управления объектом;*
* *невысокая достоверность результатов решения задачи из-за дублирования потоков информации;*
* *несовершенство организации сбора и регистрации исходной информации;*
* *несовершенство процессов сбора, передачи, обработки и выдачи информации.*

2.3.1 Обоснование необходимости использования вычислительной техники и создания АРМ для решения данного комплекса задач

2.3.2 Анализ существующих аналогов автоматизированных систем для решения данного комплекса задач

В качестве аналогов программного обеспечения разрабатываемой системы были рассмотрены инструменты класса управления испытаниями, используемые при тестировании для протоколирования работы тестируемой системы.

Программное обеспечение для управления тестированием используется командами обеспечения качества ПО, группами тестирования, менеджерами проектов для управления процессом тестирования. Такие программные средства поддерживают функции по ведению плана тестирования, отслеживанию результатов тестирования и их сопоставлению с плановыми ожиданиями.

### *4.6.1 Система управления проектами DevProm. Подсистема «Управление тестированием продуктов»*

Производитель – DevProm [22]

Основное назначение ПП - планирование функционального тестирования – разработка разделов тестовой документации, отчетов по тестированию продукта. При разработке тестовой документации существует возможность формировать иерархию разделов тестовой документации, настройки жизненного цикла тестовой документации (этапы согласования тестовой документации), автоматическая привязка разработанных разделов тестовой документации к требованиям тестируемого ПО. Последний аспект обеспечивает легкость контроль того, что реализуемый функционал будет протестирован.

Контроль за изменениями в требованиях – журнал обсуждений, журнал изменений.

Формирование графика подготовки тестовой документации.

Разработка отчетов о выполнении тестов. В отчете о выполнении теста указываются версия тестируемого приложения, окружение, в котором выполнялось тестирование (операционная система, тип браузера). Для каждого тестового сценария отмечается результат выполнении, могут быть добавлены поясняющие файлы (скриншоты, логи) или добавлен текстовый комментарий. Каждый тестовый сценарий связывается с требования.

Отображение результатов тестирования по релизу или итерациям - отчет о результатах тестирования отображает результаты всех тестов выполненных для различных версий продуктов. Возможна группировка, фильтрация результатов отчета по версиям, окружениям, сценариям и т.п.

Подходит для проведения регрессионного, смоук и приемочного тестирования.

Распространяется бесплатнаяtrial-версия с ограниченным функционалом. Стоимость приобретения пакета для установки на сервере вашей компании – 6750 рублей за пользователя. Хостинг «Облако проектов» - 750 рублей/ месяц за одного пользователя.

### *4.6.2 TestRail*

Производитель - GurockSoftware

GurockSoftware специализируется в производстве инструментов для разработки программного обеспечения групп и отделов качества ПО. TestRail является популярным программным обеспечением для управления тестированием, отслеживания результатов и взаимодействия с командой проекта [23].

Веб-управление тестированием – поддерживает управление, организацию и отслеживание выполнения тестирования через удобный и высокопроизводительный пользовательский интерфейс.

Организация проектов – поддерживает управление всеми текущими тестируемыми проектами (детализация по этапам тестирования, активным тестам, результаты тестирования).

Возможность организовать всю соответствующую информацию в структурированном виде. Предоставляет богатый функционал по форматированию текста и настройке полей, возможность прикреплять файлы.

TestRail позволяет вводить и отслеживать результаты испытаний, включая подробное описание дефектов и конфигурацию/ условия проведения тестов.

Основная функциональность – планирование предстоящих мероприятий по тестированию. TestRailпредоставляет информативные отчеты, которые позволяют отслеживать прогресс тестирования и помогут в принятии обоснованных решений на основе результатов тестирования.

Отслеживание и сохранение истории изменения тестов/ проектов.

С помощью TestRail можно назначать пользователям (группе тестирования) задания. Такое назначение эффективно для улучшения организации рабочего процесса всех членов команды тестирования. Автоматическое информирование членов команды тестирования о новых заданиях, о вносимых изменениях в конфигурацию проектов, плану тестирования по электронной почте.

TestRail позволяет настраивать роли и настраивать ограничения прав доступа к конкретным проектам или версиям проекта.

Требования к аппаратному и программному обеспечению:

Для Windows-сервера: ОС: Windows Server 2003 or 2008/2008 R2; *БД*: SQL Server 2005/ 2008.*Web-сервер*: IIS сFastCGI / PHP; PHP: 5.1.2 - 5.3.x (с curl, mbstringиsqlsrv/mysqlрасширениями). НаличиерасширенияionCube PHP Loader. Web-браузеры:Internet Explorer 7, 8, 9; Firefox 3.5 ивыше; Chrome, Safari, Webkit; Adobe Flash Player 9.x ивыше.

TestRail для сервера компании – 239$ для 1 пользователя. Стоимость приобретения для 10 пользователей составляет 2149$. Такая лицензия включает один год бесплатной поддержки пользователей и обновлений.

TestRailHosted – вид лицензии, предоставляемой за ежемесячную абонентскую плату (лицензия, хостинг, обновления продуктов и поддержка пользователей). Абонентская плата за одного пользователя в месяц составляет 25$.

### *4.6.3 TestLink*

TestLink - это веб-инструмент управления тестированием. Приложение обеспечивает создание спецификаций испытаний, планов тестирования и выполнения тестов, создание отчетности, требований по спецификациям. TestLink принадлежит и поддерживается открытым сообществом тестеровщиков со всего мира. Лицензия - [GNUGeneralPublicLicense (GPL)](http://sourceforge.net/directory/license:osi-approved-open-source/gnu-general-public-license-gpl/). Поддержка 17 языков, в том числе русского. Тип пользовательского интерфейса - [Web](http://sourceforge.net/directory/environment:web-based/) [24].

Поддерживаемыеконфигурации (браузер+ OC): • Firefox; InternetExplorer + (Windows) 5+ (Mac); Netscape + (Windows); Opera + (Windows + Linux).

Особенности:

* управление требованиями - определение требований, изменения требований, хранение различных версий требований;
* спецификации испытаний - определение тестов, группировка тестов в различные тестовые наборы;
* планирование выполнения тестов;
* выполнение тестов - выполнение тестов для свободного определяемых платформы, сборки и тестирования планов
* протоколы испытаний, метрик, графики, контроль выполнения тестов.

### *4.6.4 informUp*

informUp – это программа, которая обеспечивает простое управление жизненным циклом тестирования приложений и предоставляет следующие возможности [25]:

* управление требованиями и тестовыми сценариями;
* отслеживание дефектов / ошибок, задач;
* формирование стандартных отчетов по результатам тестирования, настройка собственных отчетов;
* управление процессом тестирования;
* автоматические уведомление по электронной почте о конкретных событиях, таких как: изменения в данных по проекту, требования, тестовым сценариям.

Веб-интерфейс. Полнаяподдержка Internet Explorer, Firefox и Chrome.

Распространяется бесплатная 30-дневная trial-версия с ограничением на количество пользователей (максимум – 20 клиентов). Стоимость приобретения пакета informUp для 5 пользователей 375$; для 10 пользователей - 600$. Плата за хостинг ПП informUp - 7$ за пользователя в месяц.

### *4.6.5 SilkCentralTM (TestManager)*

SilkCentralTM (TestManager) представляет собой мощное программное обеспечение управления тестированием. Он обеспечивает комплексную основу для повышения производительности процесса тестирования через отслеживание и контроль результатов тестирования [26].

SilkCentral TM позволяет команде качества/ группе тестирования постоянно синхронизировать требования ПОс планами тестирования. Возможность визуализировать результаты тестирования. SilkCentral снижает затраты на планирование и выполнение тестов в среде с большим количеством конфигураций для тестирования. Регистрация прогресса тестирования, результатов тестирования.

### *4.6.6 QA Complete*

QAComplete – ПП для управления тестами, тестовой средой, дефектами и задачи тестирования проекта. QA Complete является всеобъемлющим решением, обеспечивающим наглядностью в управлении процессом тестирования и обеспечения высокого качества программного обеспечения [27].

Особенности:

управление тестовыми сценариями – создание описаний тестовых требований, особенностей, тестовых сценариев;

* управление тестовой средой – описание конфигурации тестирования;
* управление дефектами – регистрация результатов тестирования, отслеживание состояния результатов во времени и прогресса тестирования;
* автоматизации тестирования - интеграция с инструментами автоматизированного тестирования;
* управление проектом тестирования – полный контроль над проектом от этапа создания плана тестирования до составления отчетов о результатах тестирования;
* управление требованиями - определение требований для каждой версии тестируемого ПО, сохранение истории изменения требований, отслеживание охвата всех требований тестами.

В сочетании этих особенностей с совместным использованием создаваемой документации, календарей (план-график) происходит рост эффективности работы команды тестирования - предоставляют решений в более короткие сроки с более высоким уровнем качества.

Стоимость приобретения 399$.

В приведенной таблице 4.5 выделены ключевые характеристики для сравнения функционала рассмотренных ПП.

Таблица 4.5 – Сравнение функционала инструментов управления тестированием

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Характеристика | «Управление тестированием продуктов» | TestRail | TestLink | InformUP | SilkCentralTM | QA Complete |
| Расчетные метрики оценки качества тестирования | - | - | - | - | - | - |
| Метрики оценки надежности ПО | - | - | - | - | - | - |
| Прогноз трудозатрат на тестирование ПО | - | - | - | - | - | - |
| Визуализация данных по проекту (графики, отчеты) | + | + | + | + | + | + |
| Иерархическое представление проектов | + | + | + | + | + | + |
| Настройка дизайна представления информации | - | - | - | + | + | - |
| Назначение пользователям потока задач (план-график) | + | + | + | + | + | + |
| Рассылка уведомления | + | + | + | + | + | + |
| Сохранение истории изменений | + | + | + | + | + | + |

Таким образом, следует отметить, что выбранные инструменты для сравнения по ключевым функциональным характеристикам имеют мало различий. Существенные вариации наблюдаются в дополнительных функциях, таких как возможность настройки пользовательского представления информации (цветовое оформление, расположение на странице), фильтрация и мощные средства поиска. На основе проведенного сравнения было сделано заключение, что проектируемый прототип ИС должен реализовывать описанный в таблице 4.5 функционал. Мощный функционал для управление пользовательским представлением, поиском, фильтрацией для прототипа ИС в рамках данной ВКР разрабатываться не будет.

*2.3.3 Обоснование проектных решений по информационному обеспечению*

* *основные принципы проектирования информационного обеспечения комплекса задач;*
* *состав и содержание результатных массивов и выходных документов;*
* *обоснование состава, формы представления исходной информации в первичных документах и на машинных носителях;*
* *обоснование требований к системам классификации и кодирования информации.*

*В данном разделе также необходимо уделить внимание обоснованию методов организации информационной базы. Здесь следует рассмотреть следующие вопросы:*

* *обоснование выбора формы хранения данных (база данных или совокупность локальных файлов);*
* *обоснование выбора модели логической структуры базы*

*данных (иерархической, сетевой, реляционной);*

* *обоснование методов организации информационных массивов (прообразов файлов), ключей упорядочения и т.д.*

*При выборе ИО создаваемой системы наиболее важными являются следующие альтернативные решения:*

* *определение целесообразности использования интегрированной базы данных;*
* *выбор СУБД;*
* *выбор структуры автономных файлов;*
* *использование диалога.*

*По каждому из названных узлов выбора альтернативных решений необходимо определить следующие основные факторы, влияющие на этот выбор:*

*1) Определение целесообразности использования интегрированной базы данных (БД): сложность информации; разнообразие запросов; объем информации; объем корректировок; возможности ЭВМ (память, программное обеспечение, надежность).*

*2) Использование диалога: требования пользователя; разнообразие запросов; объемы информации; возможности ЭВМ; надежность; время реакции на запрос; простота работы пользователя.*

*3) Выбор структуры автономных файлов: потребная память; время на корректировку; надежность; время решения задачи.*

*4) Выбор СУБД: структура информации; возможности ЭВМ; наличие программного обеспечения; широта программного окружения СУБД; наличие сети ЭВМ.*

Система управления базами данных – это комплекс программных средств, предназначенных для создания и хранения базы данных на основе некоторой модели данных, обеспечение логической и физической целостности данных, предоставление к ней санкционированного доступа приложений и пользователей, а также поддержка функций администратора.

1. *MySQL*

MySQL [17] — свободная система управления базами данных (СУБД). Тип - Реляционная СУБД. Разработчик - - MySQL AB (дочерняя компания Oracle). Написана на - C, C++. ОС - Кроссплатформенное ПО. Последняя версия - - 5.5.17 (19 октября 2011). Тестовая версия - 5.6 (25 июля 2011). Лицензия - GNU GeneralPublicLicense или коммерческая лицензия.

MySQL является решением для малых и средних приложений. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей.

MySQL импортирована на большое количество платформ: AIX, BSDi, FreeBSD, HP-UX, Linux, MacOSX, NetBSD, OpenBSD, OS/2 Warp, SGIIRIX, Solaris, SunOS, SCOOpenServer, SCOUnixWare, Tru64, Windows 95, Windows 98, WindowsNT, Windows 2000, WindowsXP, WindowsServer 2003, WinCE, WindowsVista и Windows 7.

MySQL имеет API для языков Delphi, C, C++, Эйфель, Java, Лисп, Perl, PHP, PureBasic, Python, Ruby, Smalltalk, Компонентный Паскаль и Tcl библиотеки для языков платформы .NET, а также обеспечивает поддержку для ODBC посредством ODBC-драйвера MyODBC.

*Основные возможности версии MySQL 5.5:*

* практически полная реализация ANSISQL-99, SQL-2003 плюс расширения;
* межплатформенная совместимость;
* независимые типы таблиц (MyISAM для быстрого чтения, InnoDB для транзакций и ссылочной целостности);
* транзакции;
* кэширование запросов;
* репликация: один головной сервер на одного подчинённого, много подчинённых на одного головного;
* полнотекстовая индексация и поиск с использованием типа таблиц MyISAM;
* внедрённая библиотека базы данных;
* таблицы InnoDB, обеспечивающие соответствие требованиям ACID;
* встроенный сервер, позволяющий включать MySQL в автономные приложения.
* вложенные запросы и производные таблицы;
* хранимые процедуры и функции;
* обработчики ошибок;
* курсоры;
* триггеры;
* информационная схема (так называемый системный словарь, содержащий метаданные);
* сегментирование — возможность разбить одну большую таблицу на несколько частей, размещенных в разных файловых системах, основываясь на определенной пользователем функции. При определенных условиях это может дать серьезное увеличение производительности и, кроме того, облегчает масштабирование таблиц;
* дополнительный набор функций для обработки XML, реализация поддержки XPath;
* новые средства диагностики проблем и утилиты для анализа производительности.

*Технические характеристики*

Максимальные размеры таблиц - MySQL 3.22: до 4 Гб, MySQL 3.23+: До 8 миллионов терабайт. (2 ^ 63). Размер таблицы ограничен её типом. В общем случае тип MyISAM ограничен предельным размером файла в файловой системе операционной системы. В NTFS этот размер теоретически может быть до 32 эксабайт. В случае InnoDB одна таблица может храниться в нескольких файлах, представляющих единое табличное пространство. Размер последнего может достигать 64 терабайт.

1. *Firebird*

Firebird (FirebirdSQL) [18] — компактная, кроссплатформенная, свободная система управления базами данных (СУБД), работающая на Linux, MicrosoftWindows и разнообразных Unix платформах. Тип - Реляционная СУБД. Разработчик - -Сообщество Firebird. Написана на - C++. ОС - Кроссплатформенное программное обеспечение. Последняя версия - 2.5.1 (4 октября 2011). Лицензия - Лицензия IDPL, InterBasePublicLicense, MozillaPublicLicense 1.1.

В качестве преимуществ Firebird можно отметить многоверсионную архитектуру, обеспечивающую параллельную обработку оперативных и аналитических запросов (это возможно потому, что читающие пользователи не блокируют пишущих), компактность (дистрибутив 5Mb), высокую эффективность и мощную языковую поддержку для хранимых процедур и триггеров.

Среди недостатков: отсутствие кеша результатов запросов, полнотекстовых индексов.

Firebird является сервером баз данных. Один сервер Firebird может обрабатывать несколько сотен независимых баз данных, каждую с множеством пользовательских соединений. Он является полностью свободным от лицензионных отчислений даже для коммерческого использования.

*Основные характеристики:*

* соответствие требованиям ACID;
* версионная архитектура;
* хранимые процедуры;
* события;
* генераторы;
* базы данных только для чтения;
* полный контроль за транзакциями;
* резервное копирование на лету;
* триггеры;
* внешние функции;
* декларативное описание ссылочной целостности;
* наборы символов;
* соответствие стандарту SQL;

*Требования к аппаратному обеспечению*

Firebird работает на 32- и 64-разрядных версиях Windows, Linux, также на MacOS X, HP-UX, FreeBSD, и др., на аппаратных платформах x86, x64 и PowerPC, Sparc и многих других, и поддерживает легкий переход между этими платформами. Может использоваться даже не очень мощное оборудование, особенно под Linux.

*Доступные операционные системы*

Наиболее широко используемыми для серверов Firebird операционными системами являются Linux и MicrosoftWindows (включая NT) (включая терминальные сервисы MS и Citrix). Другие поддерживаемые платформы — Mac OS X, Solaris, FreeBSD и HP-UX.

*Средства доступа к серверу*

Firebird поддерживает множество способов доступа, включая: собственные наборы компонент для C/C++, Delphi, классы для ADO, ODBC, JDBC (Jaybird), драйверы для Python, PHP, драйвер OLE DB, dbExpress, провайдер данных .NET и прямой доступ с использованием клиентской библиотеки сервера (fbclient.dll или GDS32.dll)

*Физические ограничения*

Firebird поддерживает большие базы данных. Базы данных могут быть расположены в нескольких файлах, предельный размер которых зависит от операционной системы. Теоретический предел в настоящее время составляет 64TB для одного файла базы данных, таким образом, главные ограничения накладываются файловой системой и местом на жёстком диске. Практически база данных объемом 1 ТБ стабильно работает, что доказывает проведенный тест. Максимальная длина одной записи (суммарно все поля за исключением полей с типом BLOB) равна 64 КБ.

1. *Microsoft Office Access*

Microsoft Office Access [19] — реляционнаяСУБДкорпорации Microsoft.Имеет широкий спектр функций, включая связанные запросы, связь с внешними таблицами и базами данных. Благодаря встроенному языку VBA, в самомAccess можно писать приложения, работающие с базами данных. Тип – РСУБД. Разработчик - -Microsoft. ОС - MicrosoftWindows. Последняя версия - -2010 (14.0.5128.5000) (28 апреля 2010). Лицензия - Проприетарное программное обеспечение EULA.

Основные компоненты MS Access:

* построитель таблиц;
* построитель экранных форм;
* построитель SQL-запросов (язык SQL в MS Access не соответствует стандарту ANSI);
* построитель отчётов, выводимых на печать.

Они могут вызывать скрипты на языке VBA, поэтому MS Access позволяет разрабатывать приложения и БД практически «с нуля» или написать оболочку для внешней БД.

MS Access является файл-серверной СУБД и потому применима лишь к маленьким приложениям. Отсутствует ряд механизмов, необходимых в многопользовательских БД, таких, например, как триггеры.

Существенно расширяет возможности MS Access по написанию приложений механизм связи с различными внешними СУБД: «связанные таблицы» (связь с таблицей СУБД) и «запросы к серверу» (запрос на диалекте SQL, который «понимает» СУБД). Также MS Access позволяет строить полноценные клиент-серверные приложения на СУБД MS SQL Server. При этом имеется возможность совместить с присущей MS Access простотой инструменты для управления БД и средства разработки.

*Требования к аппаратному обеспечению*

Процессор с тактовой частотой 500 мегагерц (МГц) или выше.

Память - 256 или более мегабайт (МБ) ОЗУ.

Жесткий диск - 1,5 ГБ; часть дискового пространства после установки можно освободить, удалив установочный файл с жесткого диска.

Оптический привод - устройство для чтения компакт- или DVD-дисков/

Операционная система - Windows XP с пакетом обновления (SP3) (32-разрядная), WindowsVista с пакетом обновления 1 (32-разрядная или 64-разрядная), WindowsServer 2003 с пакетом обновления 2 (SP2) и MSXML 6.0 (только для 32-разрядных версий Office), WindowsServer 2008 с пакетом обновления 2 (32-разрядная или 64-разрядная), Windows 7 (32-разрядная или 64-разрядная). Поддерживаются и подсистема WindowsonWindows (WOW) (позволяющая устанавливать 32-разрядные версии Office 2010 в 64-разрядных операционных системах).

Для использования аппаратного ускорения графики необходим видеоадаптер, совместимый с технологией DirectX 9.0c и оснащенный как минимум 64 МБ видеопамяти. Требования к системе и функциональность продукта могут изменяться в зависимости от аппаратной конфигурации и операционной системы.

Стоимость приобретения - $140,00

По данным проведенных сравнений ощутимое предпочтение отдается СУБД MySQL и Firebird; в наиболее критичных моментах для проектируемой БД MS OfficeAccess поигрывает своим конкурентам (таблица 4.3).

Таблица 4.3 - Сравнение СУБД с точки зрения критичных аспектов проектируемой ИС

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Параметр | MySQL | Firebird | | MS OfficeAccess |
| Размер базы данных – гигабайты | + | + | - | |
| Количество одновременных пользователей – до 10 пользователей | + | + | + | |
| Цена БД – полностью бесплатно | + | + | - | |
| Платформа - Windowsonly | + | + | + | |
| Тип программы - маленький web-сервер | + | + | - | |
| Защита данных - сильная | + | + | - | |

При выборе СУБД были учтены как характеристики проектируемой ИС, так и пожелания заказчика. В качестве СУБД будет использована MySQL. На данном этапе ее технические характеристики и функциональные возможности удовлетворяют требованиям проектируемой БД (представлены в таблице 4.9). Немаловажным преимуществом выбора данной СУБД является значительный опыт работы в этой среде у будущих пользователей создаваемой информационной системы.

2.3.4 Обоснование выбора программной среды разработки проектируемой автоматизированной системы

заключается в формировании требований к системному, специальному и прикладному программному обеспечению. Целесообразно:

* обосновать выбор соответствующего инструментального средства (языки программирования, специализированные библиотеки, **СУБД**, системы автоматизированного проектирования, системы класса CASE и др.) и среды, в которой предполагается использование разрабатываемой ЭИС;
* определить цель проектирования рациональной внутримашинной технологии обработки на основе выбранных инструментальных средств (например, сокращение времени обработки по сравнению с тем, что существует в настоящий момент за счет улучшенных механизмов поиска и сортировок, которые обеспечивает выбранный инструментарий; минимизация затрат на разработку и дальнейшее сопровождение ПО; обеспечение надежности ЭИС и защиты информации и т.д.);
* раскрыть сущность методов проектирования рациональной внутримашинной технологии (например, сокращение числа сортировок, использование эффективных методов поиска информации, процедурно-ориентированных подходов к выделению модулей и т.д.);
* определить функции управляющей программы;
* выработать требования к оформлению экранных и печатных форм, интерфейсу программного обеспечения.

Характеризуя программное обеспечение для создания и эксплуатации вашей ЭИС необходимо аргументировать, чем данное ПО лучше подобных сред разработки ЭИС, существующих на Российском и зарубежном рынке.

Современные средства разработки включают в себя все новейшие технологии программирования, работы с данными взаимодействия с другими информационными объектами. Они должны отвечать ряду критериев:

* являться компиляторами (т.е. на любой стадии разработки создает настоящий машинный код);
* работать и создавать исполнимые файлы для платформы Win 32;
* давать возможность отладки (просмотр произвольных областей памяти, точки останова, просмотр и модификация значений переменных, дизассемблирование);
* иметь интегрированную среду разработки и отладки;
* работать с базами данных SQL-серверного типа;
* поддерживать компонентный подход в разработке (как с точки зрения использования готовых компонентов, так и с точки зрения создания собственных).

Определяющим при выборе нами языка программирования стали другие критерии, тем более что далеко не все новейшие технологии будут использованы при создании автоматизированной системы. В частности, не последнюю роль играют опыт работы и репутация продукта.

1. *Embarcadero RAD Studio*

Embarcadero RAD Studio [20] — среда быстрой разработки приложений ([RAD](http://ru.wikipedia.org/wiki/RAD_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))) для [MicrosoftWindows](http://ru.wikipedia.org/wiki/Microsoft_Windows)фирмы[EmbarcaderoTechnologies](http://ru.wikipedia.org/wiki/Embarcadero_Technologies). Текущая версия Embarcadero RAD Studio XE2 объединяет [Delphi](http://ru.wikipedia.org/wiki/Delphi) XE2 и [C++ Builder](http://ru.wikipedia.org/wiki/C%2B%2B_Builder) XE2, [DelphiPrism](http://ru.wikipedia.org/wiki/Delphi_Prism) XE2 и[RadPHP](http://ru.wikipedia.org/w/index.php?title=RadPHP&action=edit&redlink=1) XE2 в единую интегрированную среду разработки.

В Delphi версии XE2 появилась возможность компилировать под различные платформы, в частности кроме win32 появилась возможность компилировать под win64, и под Mac OS X (визуальный интерфейс предлагается создавать на базе новой кроссплатформенной библиотеки [FireMonkey](http://ru.wikipedia.org/wiki/FireMonkey)). При компиляции под Mac OS X автоматически используются [FreePascal](http://ru.wikipedia.org/wiki/FreePascal) и [XCode](http://ru.wikipedia.org/wiki/XCode). Дополнительная поддержка в [DelphiPrism](http://ru.wikipedia.org/wiki/Delphi_Prism)фреймворка[Mono](http://ru.wikipedia.org/wiki/Mono) обеспечивает возможность создания кросс-платформенных .NET приложений, которые работают под операционными системами [Windows](http://ru.wikipedia.org/wiki/Windows), [Linux](http://ru.wikipedia.org/wiki/Linux) и [Mac OS X](http://ru.wikipedia.org/wiki/Mac_OS_X). [<http://ru.wikipedia.org/wiki/Embarcadero_RAD_Studio>]

Более подробно остановимся на DelphiXE2.

Требования Delphi XE2 к системе:

* ОЗУ 1 ГБ (рекомендуется 2 ГБ и больше);
* От 3 до 7 ГБ свободного места на жестком диске в зависимости от выпуска и конфигурации;
* DVD-привод (если установка будет выполняться с DVD-диска MediaKit);
* базовый графический процессор любого поставщика, класса DirectX 9.0 или выше (PixelShaderLevel 2);
* монитор с разрешением 1024x768 или выше;
* процессор IntelPentium или совместимый с частотой не менее 1,6 ГГц (рекомендуется 2 ГГц и выше);
* MicrosoftWindows 7 (32- и 64-разрядная), MicrosoftWindowsVista™ SP2 (32- и 64-разрядная, требуются права администратора), MicrosoftWindowsXPHome или Professional (32- и 64-разрядная) с SP2 или SP3, MicrosoftWindowsServer® 2003 SP1 или 2008 (32- и 64-разрядная).

Наибольшее количество функциональных изменений в новой версии продукта касаются IDE (возможность привести Delphi к «классическому виду», новая система поиска в редакторе кода, доработка многих диалогов). Некоторые изменения претерпел и редактор кода. В частности, появилась возможность автоматически форматировать код. Так же в IDE DelphiXE2 появилась еще одна интересная возможность – фоновая компиляция. В Architect версии DelphiXE2 появились средства анализа кода - QA Аудит и QA Метрика. Данные инструменты позволяют выявить QA характеристики кода. В DelphiXE2 появились и принципиально новые возможности. Прежде всего, это новый механизм RTTI (RunTimeTypeInformation) - очень эффективное средство, позволяющее получать информацию о типах в режиме исполнения (RunTime). В DelphiXE2 в функциональность языка были добавлены атрибуты, подобно тому, как это работает в .Net. По сути, атрибуты хранят в себе некую мета-информацию о типах. В контексте RTTI, естественно, были внесены дополнения и в VCL (поддержка механизма естественного ввода - альтернативного механизма ввода данных, с помощью устройств основанных на прикосновениях (Gesturing) (классические примеры таких устройств – точпад, точскрин или световое перо)).

В наборе DBExpress появилась поддержка InterBase 2009, MS SQL Server 2008. MySQL 5.1 и Firebird.

Обратной стороной медали является увеличение размера исполняемых файлов. Пустая форма с кнопкой компилируется в исполняемый файл размером 811 Kb.

Подводя итог вышесказанному, следует отметить следующее о DelphiXE2:

* более быстрая, надежная и удобная IDE;
* радикально обновленный RTTI, со значительно более широкими возможностями, чем в предыдущих версиях;
* развитие технологий работы с БД;
* попытка охватить интересы не только кодеров, но и других участников процесса разработки ПО (UML, QA инструменты);
* новые возможности VCL, позволяющие использовать альтернативные устройства ввода данных.

Выход DelphiXE2, с моей точки зрения, не принес какой-то революционной функциональности. Упор скорее сделан на производительность IDE и ее удобство.

Но цена продукта по-прежнему высока для отечественных разработчиков (от 26 340 рублей (Именная лицензия Professional, 1 пользователь)**.**

1. *MicrosoftVisualStudio 2010 Ultimate*

MicrosoftVisualStudio 2010 Ultimate [21] предоставляет интегрированную инструментальную среду и серверную инфраструктуру, упрощающую полный цикл разработки приложений.

Составляющие пакета VisualStudio 2010 и языки программирования, которые они используют:

1. Visual Basic (Visual Basic 2010 Express Edition);
2. C# (Visual C# 2010 Express Edition);
3. C++ (Visual C++ 2010 Express Edition);
4. Длясоздания Web-приложений (Visual Web Developer 2010 Express Edition).

*Характеристики платформы:*

* все продукты, входящие в группу ExpressEdition, отличаются простым интерфейсом, малым объемом (порядка 50 Мбайт), наличием большого числа примеров и обучающих материалов;
* наличие визуального дизайнера (WindowsFormsDesigner), с помощью которого можно создавать интерфейсы приложений простым перетаскиванием визуальных компонентов;
* визуальные компоненты GridView, ToolStrip и WebBrowser, позволяющие быстро создавать пользовательские интерфейсы любой сложности;
* поддержка технологии SmartTags, что дает возможность быстро настраивать основные свойства компонентов;
* поддержка технологии IntelliSenseCodeSnippets (более 200 законченных фрагментов кода, выполняющего часто используемые операции) и наличие пространства имен My (облегченный доступ к некоторой функциональности .NET Framework), использование которых значительно сокращает объем кода, создаваемого непосредственно разработчиками;
* механизм подсказок по исправлению ошибок AutoCorrect, позволяющий быстро освоить написание правильного кода;
* фоновая компиляция, отслеживающая малейшие изменения в коде;
* возможность модификации и исправления кода при выполнении программы – EditandContinue;
* упрощенные средства отладки, в том числе средства визуализации данных, расположенных в наборах данных DataSets, HTML, XML, а также текстовых данных – Datatips и Visualizers;
* поддержка работы с базами данных за счет использования SQL Server 2010 ExpressEdition и автоматическая генерация интерфейса для связи с источниками данных.

*Недостатки:*

* для работы приложений созданных в VisualStudio 2010 требуется установить на компьютере пользователя библиотеки классов и средства управления ими в виде Microsoft .NET Framework 2 и MicrosoftFramework 3, которые использовались при разработке этих приложений;
* повышенная требовательность VisualStudio 2010 к системным ресурсам в момент разработки и, в частности, отладки приложения. Это обусловлено перечисленными выше возможностями «горячего» исправления кода, фоновой компиляции и др. Кроме того, очень объемная служба помощи MSDN;
* нет поддержки работы с СУБД, кроме SQL Server 2010 ExpressEdition.

*Системные требования:*

Операционные системы:Windows 7, WindowsServer 2003 R2 (32-Bitx86), WindowsServer 2003 R2 x64 editions, WindowsServer 2003 ServicePack 2, WindowsServer 2008 R2, WindowsServer 2008 ServicePack 2, WindowsVistaServicePack 2, WindowsXPServicePack 3; WindowsXP (x86) с пакетом обновления 3 (SP3) - все выпуски, кроме Starter; WindowsVista (x86 &x64) с пакетом обновления 2 (SP2) - все выпуски, кроме Starter; Windows 7 (x86 и x64); MicrosoftWindowsServer 2003 (x86 &x64) с пакетом обновления 2 (SP2) - все выпуски.

Пользователи должны установить MSXML6, если этот компонент отсутствует. Поддерживаемые архитектуры: 32-разрядная (x86), 64-разрядная (x64) (WOW).

*Аппаратные требования:*

* компьютер с процессором с тактовой частотой 1,6 ГГц или большей;
* 1024 МБ ОЗУ (1,5 ГБ при выполнении на виртуальной машине)
* 3 ГБ доступного пространства на жестком диске;
* жесткий диск с частотой вращения 5400 об/мин;
* видеоадаптер, соответствующий стандарту DirectX 9 и поддерживающий разрешение экрана 1024 x 768 или выше.
* наличие манипулятора мышь;
* привода CD или DVD не требуется.

Обе среды разработки (EmbarcaderoDelphi XE2 и MicrosoftVisualStudio 2010 Ultimate) поддерживают технологию .Net. .NET Framework — [программная платформа](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0), выпущенная компанией [Microsoft](http://ru.wikipedia.org/wiki/Microsoft) в [2002 году](http://ru.wikipedia.org/wiki/2002_%D0%B3%D0%BE%D0%B4). Основой платформы является исполняющая среда [CommonLanguageRuntime (CLR)](http://ru.wikipedia.org/wiki/Common_Language_Runtime), способная выполнять как обычные программы, так и серверные веб-приложения. NET Framework поддерживает создание программ, написанных на разных языках программирования. Основной идеей при разработке .NET Framework являлось обеспечение свободы разработчика за счёт предоставления ему возможности создавать приложения различных типов, способные выполняться на различных типах устройств и в различных средах. Вторым принципом стало ориентирование на системы, работающие под управлением семейства операционных систем MicrosoftWindows.

Обе фирмы стремились облегчить разработку веб- и B2B- приложений, что создали визуальные компоненты для эффективной работы с ними. Они преуспели в этом, но Microsoft, как всегда впереди - для своих ASP.NET серверов нет ничего удобнее в разработке, чем VisualStudio.

Удобство обоих сред на высоком уровне. Важное преимущество Delphi - большой выбор бесплатных компонент. Поэтому значительно легче построить приложение в TRM (True RAD Manner). MS VisualStudio - более серьезная и денежная среда. Ее компоненты тяжеловеснее и дороже. Но и приложения можно строить на порядок более сложнее.

Delphi RAD - среда, подходящая для разработки прототипов и средних проектов. Важное в ней - легкость разработки и доступность бесплатных компонент. Тем не менее, в ней еще достаточно много недоработок и ошибок, что значительно ухудшает удобство ее использования. VisualStudio - ни разу не выдавала ошибок, но компоненты дороже. Бесплатных почти нет, что делает ее не так привлекательной для начинающих и продвинутых программистов.

Для реализации пользовательского интерфейса в ИС «Планирование процесса тестирования» будет использоваться среда разработки MicrosoftVisualStudio 2010 Ultimate. Решающее влияние в выборе данного программного продукта оказало отсутствие других альтернативных сред разработки в компании. При этом компания является обладателем лицензии на данный программный продукт. Все проекты, разрабатываемые компанией, реализовываются в этой среде. Поэтому, аспект значительного опыта работы с этим продуктом у будущих пользователей системы сыграл немаловажную роль в выборе среды разработки.

### *4.5.3 Выбор языка программирования*

Для написания алгоритмов обработки данных и выполнения расчетов был выбран объектно-ориентированный язык программирования C#. C# входит в новую версию VisualStudio - VisualStudio.NET. КромеC# вVisualStudio.NETвходятVisualBasic.NETиVisualC++. Другие языки были созданы до появления платформы .NET, язык же C# создавался специально под эту платформу и не несет с собой груза совместимости с предыдущими версиями языков.

2.3.5 Постановка задачи *(Выбор стратегии автоматизации «КАК ДОЛЖНО БЫТЬ»)*

ИС «Управление процессом тестирования» должна предоставлять пользователю следующий функционал:

* ведение базы ресурсов тестирования ПО (тестируемое ПО, команда тестирования, конфигурация компьютера и т.д.);
* ведение базы планов тестирования ПО;
* ведение базы результатов тестирования ПО;
* оценка качества процесса тестирования;
* оценка надежности тестируемого ПО;
* оценка трудозатрат процесса тестирования;
* создание отчетов, отражающих прогресс тестирования;
* создание плана-графика работ по тестированию ПО.

Точка зрения - проект-менеджер; участник группы тестирования.

Цели ИС [13]:

* обеспечение пользователя информацией о тестируемом ПО, об условиях проведения тестирования, о группе тестирования;
* обеспечение пользователя информацией о результативности процесса тестирования ПОдля принятия решения о завершении очередной итерации процесса тестирования;
* обеспечение пользователя прогнозной информацией о трудозатратах процесса тестирования ПО для учета этих трудозатрат в суммарных затратах по разработке ПО;
* обеспечения пользователя информацией о надежности тестируемого ПОдля принятия решения о завершении очередной итерации процесса тестирования;
* обеспечение пользователя планом-графиком проведения тестирования.

Задачи ИС [13]:

* преобразование входной информации о тестируемом ПО, первичных данных о результатах тестирования и выдача ее в удобном для пользователя виде (графики, таблицы);
* экономия временных, трудовых ресурсов при выполнении процессов преобразования информации;
* специальные задачи:
* проведение мероприятий по оценке надежности ПО;
* проведение мероприятий по оценке результатов тестирования ПО;
* проведение мероприятий по оценке трудозатрат на тестирование ПО;
* обеспечение необходимого уровня надежности ПО;
* обеспечение необходимого уровня качества тестирования ПО;
* обеспечение необходимого уровня трудозатрат на тестирование ПО;
* проведения мероприятий по составлению плана-графика проведения тестирования.

Функции ИС [13]:

* учет ошибок и других первичных показателей при тестировании ПО;
* учет трудозатрат на тестирование ПО;
* достижение заданного уровня надежности ПО;
* прогнозирование трудозатрат на процесс тестирования ПО;
* достижение заданного уровня качества тестирования;
* снижение трудозатрат на тестирование ПО;
* снижение трудоемкости планирования процесса тестирования, проведения тестирования ПО и оценки результатов тестирования;
* анализ надежности тестируемого ПО;
* анализ качества процесса тестирования ПО;
* анализ затрат на тестирование ПО;
* управление процессом тестирования;
* сбор сведений о проекте ПО, условиях и требованиях к процессу тестирования ПО, группе тестирования и ввод данных в ПК;
* обработка и ведение баз данных по тестируемому ПО, хранение и корректировка данных;
* выдача произвольных документов пользователю.

Задачи автоматизации:

* расчет показателей, характеризующих прогнозные трудозатраты на тестирование ПО;
* расчет показателей качества тестирования ПО;
* расчет показателей надежности тестируемого ПО;
* построение плана-графика тестирования;
* формирование плана тестирования и других форм отчетов.

## 4.2 Потоки информации

В качестве входных данных используются:

* описание тестируемого ПО (описание ПО можно получить из технического задания, описания ее назначения, сценариев использования системы.Описание ПО должно включать определение пользовательской среды, компьютерных платформ и потребительских свойств ПО, приоритеты функционала ПО);
* цели и задачи тестирования;
* используемые инструменты тестирования;
* параметры тестирования – границы работ по тестированию, критерии приемки системы, стратегии приемо-сдаточных испытаний, зависимости между различными видами, риски тестирования;
* применяемые методы верификации;
* требования к тестам (подробный перечень того, что должно быть протестировано; требования к тестам можно получить на основе бизнес-требований, функциональных требований к системе и сценариев использования системы);
* детализированные тестовые сценарии (тестовые процедуры, помещенные в отдельный файл или документ - могут содержать описание алгоритмов на псевдокоде, предварительное определение этапа тестирования или псевдокод программ автоматизированного тестирования);
* характеристика тестовых сценариев (количество тестовых сценариев и их веса (отражают сложность); количество ролей ПО; веса ролей; веса технических факторов веса факторов окружения);
* описание группы тестирования (состав, квалификация);
* результаты тестирования (первичные метрики).

Выходные данные группируются по четырем направлениям: прогнозные оценки трудозатрат на тестирование ПО, оценки качества тестирования ПО, оценки надежности тестируемого ПО, план-график тестирования.

Форма представления выходных данных:

* план тестирования ПО;
* матрица отслеживания требований;
* тестовые сценарии;
* отчет о ходе разработки тестов;
* отчет о готовности к тестированию;
* отчет о ходе тестирования;
* отчет об отслеживании дефектов;
* отчет о состоянии функциональных характеристик ПО;
* отчет о тестировании.

4 ПРОЕКТНАЯ ЧАСТЬ

3.1 Информационное обеспечение комплекса задач

3.1.1 Инфологическая или информационная модель (*схема данных*) и ее описание.

Приложение – мб уменьшить

* + 1. Используемые классификаторы и системы кодирования.

Справочникик – сотрудники, должности, статусы, приоритеты

* + 1. Характеристика входной информации *(нормативно-справочная информация и входная оперативная информация)*

3.1.4 Характеристика результатной (*выходной*) информации

3.2 Технологическое обеспечение.

3.2.1 Организация технологии сбора, передачи, обработки и выдачи информации

3.2.2 Схема технологического процесса сбора, передачи, обработки и выдачи информации

3.3 Выбор и обоснование технических средств

3.4 Программное обеспечение комплекса задач

3.4.1 Общие положения (*дерево функций и сценарии диалогов*)

3.4.2 Структурная схема пакета *(дерево вызовов процедур и программ)*

3.4.3 Описание программных модулей

3.5 Описание контрольного примера реализации

5 ТЕХНИКО - ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ЭФФЕКТИВНОСТИ ВКР

## 4.1 Обоснование целесообразности разработки проекта

В данной работе представлено технико-экономическое обоснование разработки информационной системы планирования и ведения процесса тестирования программного обеспечения в компании ООО «СкайТек».

Сегодняшний рынок программного обеспечения предлагает довольно широкий ассортимент программных продуктов и информационных систем, поддерживающих приемлемый для компании функционал по ведению процесса тестирования. Но их использование на данном этапе развития компании не является целесообразным, так как возникают достаточно высокие затраты на приобретение и поддержку такой системы.

В компании «СкайТек» действует информационная система RedMine, которая является инструментом поддержки коммуникации всех участников процесса разработки ПО, мониторинг выполнения задач. Но для данной организации возникает необходимость разработки автоматизированной системы по ведению тестирования ПО, так как RedMine не предоставляет необходимого функционала в области планирования предстоящих работ по тестированию и оценке результатов на тестирование.

Ее основными пользователями будут сотрудники отдела тестирования, а также менеджеры по проектам. Предполагается, что применение этой системы значительно облегчит труд сотрудников и сократит трудозатраты на выполнение работы сотрудниками.

## 4.2 Оценка конкурентоспособности в сравнении с аналогом

В качестве программы для сравнения при разработке проекта принята программа «TestRail» (производитель – компания «GurockSoftware»).

Эта разработка принята в качестве базового варианта исходя из следующих факторов:

1. соответствие требованиям технического задания проекта;
2. доступность для исследования и сравнения с разрабатываемым проектом trial-версии программы.

Для оценки конкурентоспособности разрабатываемого продукта необходимо провести анализ и сравнение с выбранным аналогом по функциональному назначению, основным техническим и эксплуатационным параметрам, областям применения. Подобный анализ осуществляется с помощью оценки эксплуатационно-технического уровня разрабатываемого продукта.

Эксплуатационно-технический уровень (ЭТУ) разрабатываемого продукта – это обобщенная характеристика его эксплуатационных свойств, возможностей, степени новизны, являющихся основой качества продукта. Для определения ЭТУ продукта можно использовать индекс эксплуатационно-технического уровня *J*ЭТУ*,* который рассчитывается как сумма частных индексов, куда входят показатели качества программного продукта. Для учета значимости отдельных параметров применяется балльно-индексный метод.

Тогда

|  |  |
| --- | --- |
| , | (4.1) |

где *J*ЭТУ *–* комплексный показатель качества продукта по группе показателей;

*n–* число рассматриваемых показателей;

*Вj* – коэффициент весомости *j*-го показателя в долях единицы, назначаемый в соответствии с потребностями организации-заказчика программного продукта;

*Xj –* относительный показатель качества, устанавливаемый экспертным путем по выбранной шкале оценивания

В таблице 4.1 представлены результаты расчета балльно-индексным методом при 5-ти балльной шкале оценивания.

Таблица 4.1 – Расчет показателя качества балльно-индексным методом

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Показатели качества | Коэффициент весомости, *Вj* | Проект | | Аналог | |
|
| *Xj* | *Вj*×*Xj* | *Xj* | *Вj*×*Xj* |
| 1.Удобство работы (пользовательский интерфейс) | 0,1 | 4 | 0,4 | 4 | 0,4 |
| 2.Соответствие требованиям заказчика | 0,1 | 4 | 0,4 | 2 | 0,2 |
| 3.Соответствие профилю деятельности заказчика | 0,15 | 5 | 0,75 | 5 | 0,75 |
| 4.Операционная система (многозадачность, графика) | 0,05 | 4 | 0,2 | 4 | 0,2 |
| 5. Надежность (защита данных) | 0,1 | 2 | 0,2 | 2 | 0,2 |
| 6.Скорость доступа к данным | 0,09 | 4 | 0,36 | 4 | 0,36 |
| 7.Гибкость | 0,05 | 2 | 0,1 | 3 | 0,15 |
| 8.Функции обработки информации | 0,25 | 5 | 1,25 | 3 | 0,75 |
| 9.Соотношение стоимость/возможности | 0,06 | 4 | 0,24 | 2 | 0,12 |
| 10. Время обучения персонала | 0,05 | 4 | 0,2 | 3 | 0,15 |
| Обобщенный показатель качества *J*ЭТУ | | *J*ЭТУ1=4 | | *J*ЭТУ2=2,55 | |

Отношение двух найденных индексов называют коэффициентом технического уровня *Аk*первого программного продукта по отношению ко второму:

|  |  |
| --- | --- |
| . | (4.2) |

Так как коэффициент больше 1, то разработка проекта с технической точки зрения оправдана.

## 4.3 Планирование комплекса работ по разработке темы и оценка трудоемкости

Трудоемкость работ определяется с учетом срока окончания работ, объема выполняемых функций, выбранной среды программирования.

Для определения ожидаемой продолжительности работы *Т*ож применяется формула

|  |  |
| --- | --- |
| , | (4.3) |

где *t*мин – кратчайшая продолжительность заданной работы (оптимистическая оценка);

*t*макс – самая большая продолжительность работы (пессимистическая оценка);

*t*нв – наиболее вероятная продолжительность работы (реалистическая оценка).

Оценка трудоемкости отдельных видов работ приведена в таблице 4.2.

Для разработки было задействовано два человека: руководитель проекта и исполнитель (инженер-программист).

Таблица 4.2 – Оценка трудоемкости отдельных видов работ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Виды | Оптимистическая оценка,*tmin* | Реалистическая оценка,*t*нв | Пессимистическая оценка,*tmax* | Ожидаемая |
| работ | продолжительность |
|  | работы,*Т*ож |
| 1.1 | 15 | 18 | 21 | 18 |
| 1.2 | 6 | 8 | 10 | 8 |
| 1.3 | 10 | 12 | 15 | 12 |
| 2.1 | 5 | 6 | 7 | 6 |
| 2.2 | 5 | 7 | 9 | 7 |
| 3.1 | 15 | 19 | 22 | 19 |
| 3.2 | 19 | 21 | 23 | 21 |
| 3.3 | 3 | 5 | 6 | 5 |
| 3.4 | 4 | 6 | 8 | 6 |
| 4.1 | 3 | 5 | 6 | 5 |
| 4.2 | 14 | 16 | 22 | 17 |

Руководитель выполняет постановку задачи, курирует ход работ и дает необходимые консультации при разработке системы. Исполнитель отвечает за проектирование информационного обеспечения, разработку структур баз данных, реализацию вычислительных алгоритмов в виде завершенного продукта, разработку интерфейсных блоков и отладку программы.

Выбор комплекса работ по разработке проекта производится в соответствии со стандартом «ГОСТ Р ИСО/МЭК 12207-99 Информационная технология. Процессы жизненного цикла программных средств», устанавливающим стадии разработки программных продуктов, и приведен в таблице 4.3.

Таблица 4.3 – Комплекс работ по разработке проекта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Содержание работ | Исполнители | Длительность, дни | Загрузка | |
| дни | % |
| **1. Подготовка процесса разработки и анализ требований** | | | | |
| 1.1 Исследование и обоснование разработки | | | | |
| 1.1.1 Постановка задачи | Руководитель | 3 | 1 | 33 |
| Программист | 3 | 100 |
| 1.1.2 Сбор исходных данных | Руководитель | 15 | 4 | 27 |
| Программист | 15 | 100 |
| 1.2 Поиск аналогов и прототипов | | | | |
| 1.2.1 Анализ существующих методов решения задачи и программных средств | Руководитель | 6 |  |  |
| Программист | 6 | 100 |
| 1.2.2 Обоснование принципиальной необходимости разработки | Руководитель | 2 | 1 | 50 |
| Программист | 2 | 100 |
| 1.3 Анализ требований | | | | |
| 1.3.1 Определение и анализ требований к проектируемой программе | Руководитель | 3 | 1 | 33 |
| Программист | 3 | 100 |
| 1.3.2 Определение структуры входных и выходных данных | Руководитель | 4 | 1 | 25 |
| Программист | 4 | 100 |
| 1.3.3 Выбор технических и программных средств реализации | Руководитель | 3 | 1 | 33 |
| Программист | 3 | 100 |
| 1.3.4 Согласование и утверждение технического задания | Руководитель | 2 | 1 | 50 |
| Программист | 2 | 100 |
| **Итого по этапу 1** | Руководитель | **38** | **10** |  |
| Программист | **38** |  |
| **2. Проектирование** | | | | |

Окончание таблицы 4.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2.1 Проектирование программной архитектуры | Руководитель | 6 |  |  |
| Программист | 6 | 100 |
| 2.2 Техническое проектирование компонентов программы | Руководитель | 7 |  |  |
| Программист | 7 | 100 |
| **Итого по этапу 2** | Руководитель | **13** |  |  |
| Программист | **13** |  |
| **3. Программирование и тестирование программных модулей** | | | | |
| 3.1 Программирование модулей в выбранной среде программирования | Руководитель | 19 |  |  |
| Программист | 19 | 100 |
| 3.2 Тестирование программных модулей | Программист | 21 |  |  |
| Программист | 21 | 100 |
| 3.3 Сборка и испытание программы | Руководитель | 5 | 2 | 40 |
| Программист | 5 | 100 |
| 3.4 Анализ результатов испытаний | Руководитель | 6 | 2 | 33 |
| Программист | 6 | 100 |
| **Итого по этапу 3** | Руководитель | **51** | **4** |  |
| Программист | **51** |  |
| **4. Оформление рабочей документации** | | | | |
| 4.1 Проведение экономических расчетов | Руководитель | 5 |  |  |
|  | Программист | 5 | 100 |
| 4.2 Оформление пояснительной записки | Руководитель | 17 | 4 | 24 |
| Программист | 17 | 100 |
| **Итого по этапу 4** | Руководитель | **22** | **4** |  |
| Программист | **22** |  |
| **Итого по проекту** | Руководитель | **124** | **18** |  |
| Программист | **124** |  |

На основе данных таблицы 4.3 разработан календарный график выполнения работ (таблица 4.4), показывающий последовательность и взаимосвязь выполнения комплекса работ.

Таблица 4.4 - Календарный график выполнения работ

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание работы | Исполнители | Длительность, дни | График работ |
| 1 Постановка задачи | Руководитель Программист | 1  3 | 16.01.12-16.01.12  16.01.12-18.01.12 |
| 2 Сбор исходных данных | Руководитель  Программист | 4  15 | 19.01.12-22.01.12  19.01.12-02.02.12 |
| 3 Анализ существующих методов решения задачи и программных средств | Программист | 6 | 03.02.12-08.02.12 |
| 4 Обоснование принципиальной необходимости разработки | Руководитель Программист | 1  2 | 09.02.12-09.02.12  09.02.12-10.02.12 |
| 5 Определение и анализ требований к программе | Руководитель Программист | 1  3 | 11.02.12-11.02.12  11.02.12-13.02.12 |
| 6 Определение структуры входных и выходных данных | Руководитель Программист | 1  4 | 14.02.12-14.02.12  14.02.12-17.02.12 |
| 7 Выбор технических средств и программных средств реализации | Руководитель Программист | 1  3 | 18.02.12-18.02.12  18.02.07-20.02.12 |
| 8 Согласование и утверждение технического задания | Руководитель Программист | 1  2 | 21.02.12 – 21.02.12  21.02.12 –22.02.12 |
| 9 Проектирование программной архитектуры | Программист | 6 | 23.02.12 –28.02.12 |
| 10 Техническое проектирование компонентов программы | Программист | 7 | 29.02.12– 06.03.12 |
| 11 Программирование модулей в выбранной среде программирования | Программист | 19 | 07.03.12 – 25.03.12 |
| 12 Тестирование программных модулей | Программист | 21 | 26.03.12 – 15.04.12 |
| 13 Сборка и испытание программы | Руководитель  Программист | 2  5 | 16.04.12 – 17.04.12  16.04.12 – 20.04.12 |
| 14 Анализ результатов испытаний | Руководитель  Программист | 2  6 | 21.04.12 – 22.04.12  21.04.12 –26.04.12 |
| 15 Проведение экономических расчетов | Программист | 5 | 27.04.12 – 01.05.12 |
| 16 Оформление пояснительной записки | Руководитель Программист | 4  17 | 02.05.12 – 05.05.12  02.05.12 – 18.05.12 |

На основе этих данных был сформирован график выполнения этапов работ, который приведен на рисунке 4.1.

Рисунок 4.1 - Календарный график выполнения работ

## 4.4 Расчет затрат на разработку проекта

Капитальные вложения, связанные с автоматизацией обработки информации рассчитываются по формуле

|  |  |
| --- | --- |
| К = Кп+ Кр, | (4.4) |

где Кп - капитальные вложения на проектирование, руб.;

Кр *-* капитальные вложения на реализацию проекта, руб.

Предпроизводственные затраты представляют собой единовременные расходы на разработку обеспечивающих или функциональных систем или элементов на всех этапах проектирования, а также затраты на их усовершенствование, т.е. на проведение обследования и обработку материалов исследования, разработку технического задания, разработку технического и рабочего проекта системы и ее опытного внедрения. Сюда включаются затраты на разработку алгоритмов и программ, стоимость разработок по привязке типовых проектных решений (ТПР) и пакетов прикладных программ (ППП) к конкретному объекту автоматизации.

Суммарные затраты на проектирование системы и ее разработку и отладку на компьютере определяются по формуле

|  |  |
| --- | --- |
| , | (4.5) |

где *m*– количество работников, участвующих в разработке проекта;

Зo*i –* затраты на основную заработную плату работника *i*-й категории, руб.;

*Wd*– коэффициент, учитывающий дополнительную заработную плату в долях к основной заработной плате (= 0,4 и состоит из коэффициента отпускных, равного 0,1, и районного коэффициента – 0,3 для Томска);

Wс – коэффициент, учитывающий отчисления на социальные нужды, в долях к сумме основной и дополнительной заработной платы разработчиков (=0,302: страховые взносы в Пенсионный фонд в долях единицы – 0,22, страховые взносы в ФСС – 0,029, страховые взносы в ФОМС – 0,051, страховые взносы на производственный травматизм – 0,002);

*W*н *–* коэффициент, учитывающий накладные расходы организации, в долях к основной заработной плате разработчиков (принимается по фактическим данным, Wн = 0,6);

*СM –* затраты на материалы;

*Мв* – затраты на использование машинного времени.

Затраты на основную заработную плату работника *i*-й категории:

|  |  |
| --- | --- |
| , | (4.6) |

где Здн*i* – среднедневная заработная плата работника *i*-й категории, руб./дн.;

*ti* – количество дней, отработанных работником *i*-й категории.

Затраты времени на разработку системы по каждому исполнителю принимаются, исходя из его загрузки по календарному графику выполнения работ (см. таблицу 4.4).

Расчет основной заработной платы разработчиков проекта приведен в таблице 1.5 из расчета, что в месяце в среднем 21 рабочий день.

Таблица 4.5 – Основная заработная плата разработчиков

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Должность | Должностной оклад, руб. | Средняя дневная ставка, руб. | Затраты времени на разработку, чел. дней | ОЗП, руб. |
| Руководитель Программист | 10840  2100 | 516,2  100 | 18  144 | 9291,5  14400 |
| Итого | | | | 23828,6 |

Ввиду того, что проектируемая информационная система должна быть запрограммирована и отлажена с помощью компьютеров, к суммарным затратам на разработку добавляются затраты на использование машинного времени, исчисляемые как:

|  |  |
| --- | --- |
| , | (4.7) |

где *t*мв – машинное время компьютера, необходимое для разработки программного продукта;*t*мв = 374 час.;

*S*мч – стоимость 1 часа машинного времени; *S*мч=30 руб./час.;

*К*м – коэффициент мультипрограммности (показывает долю машинного времени, отводимого непосредственно на работу над проектом); Км=1.

Материалы, приобретенные в процессе выполнения работы, и их стоимость приведены в таблице 4.6.

Таблица 4.6 - Затраты на материалы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Материалы | Единица измерения | Требуемое количество | Цена за единицу, руб. | Сумма, руб. |
| Тетрадь | шт. | 1 | 25 | 25 |
| Компакт-диск CD-RW | шт. | 1 | 35 | 35 |
| Тонер для лазерного принтера | шт. | 1 | 100 | 100 |
| Бумага офисная | пачка | 1 | 160 | 160 |
| Итого | | | | 320 |

Таким образом, капитальные вложения на проектирование равны:

Кп = (9291,5 + 14400) × ((1+0,4) × (1 + 0,302) + 0,6) + 224 × 12 ×1 + 320 руб. = 60407,8 руб.

Смета затрат на разработку представлена в таблице 4.7.

Таблица 4.7 – Затраты на разработку

|  |  |
| --- | --- |
| Статьи затрат | Сумма, руб. |
| Основная заработная плата | 23828,6 |
| Дополнительная зарплата | 9531,4 |
| Отчисления на социальные нужды | 7196,2 |
| Затраты на материалы | 320 |
| Затраты на машинное время | 11280 |
| Накладные расходы организации | 14297,2 |
| ИТОГО | 60407,8 |

Капитальные вложения на реализацию проекта:

|  |  |
| --- | --- |
| Кр= Ко + Кдд + Кпп +Ксв + Киб + Кпк , | (4.8) |

где Ко – затраты на основное и вспомогательное оборудование, руб.;

Кзд – затраты на строительство, реконструкцию здания и помещений, руб.;

Кпп – затраты на приобретение типовых разработок, пакетов, руб.;

Ксв – затраты на прокладку линий связи, руб.;

Киб – затраты на создание информационной базы, руб.;

Кпк – затраты на подготовку и переподготовку кадров, руб.

В связи с тем, что для внедрения системы, рассматриваемой в данном проекте, не было затрат связанных с прокладкой линии связи, затрат на основное и вспомогательное оборудование, затрат на реконструкцию и строительство зданий, то данные затраты для внедрения системы не учитывают. Также не принимаются в расчет затраты по подготовке и переподготовке кадров, затраты на создание информационной базы и затраты на приобретение типовых разработок.

Таким образом, при внедрении системы, рассматриваемой в данном проекте, затраты на его реализацию определяются затратами на оборудование и материалы. В оборудование и материалы входит компьютер стоимостью 24000 руб.

Тогда затраты на основное и вспомогательное оборудование составят

|  |  |
| --- | --- |
| , | (4.9) |

где *Cbj* – балансовая стоимость *j-*го вида оборудования, руб. (24000 руб.);

*Qj* – количество единиц *j*-гo оборудования, руб. (1 шт.);

*Yj* – коэффициент загрузки *j*-го вида оборудования при обработке информации по решению задач предметной области.

Коэффициент загрузки оборудования:

|  |  |
| --- | --- |
| , | (4.10) |

где Фэф*j –* эффективный годовой фонд времени работы технического средства *j*-го вида, час./год.

|  |  |
| --- | --- |
| , | (4.11) |

где *tкj*– трудоемкость однократной обработки информации по *к*-й задаче на *j*-м виде технических средств, часов машинного времени (*tкj*=6);

*Uк* – частота (периодичность) решения *к*-й задачи, дней /год (*Uк* =264).

Затраты на реализацию:

Кр= 24000 × 1 × 6 × 264/ (264 × 8) руб. = 18000 руб.

Таким образом, суммарные затраты на разработку проекта:

К= Кп + Кр= 60407,8 + 18000 руб. = 78407,77 руб.

Суммарные затраты, связанные с внедрением аналога складываются из следующих затрат:

* затраты на приобретение программного продукта (64470 руб. – покупка системы на 10 пользователей (4 - 5 тестировщика, 4 - 5 проект-менеджеров));
* затраты по оплате услуг на установку и сопровождение продукта (81000 руб.);
* затраты на основное и вспомогательное оборудование (24000 руб.);
* затраты на подготовку пользователей (12000 руб.).

Итого суммарные затраты, связанные с внедрением аналога составят117000 руб.

## 4.5 Расчет эксплуатационных затрат

К эксплуатационным затратам относятся затраты, связанные с обеспечением нормального функционирования проекта. Эти затраты называют также текущими затратами. Это могут быть затраты на ведение информационной базы, эксплуатацию комплекса технических средств, эксплуатацию систем программно-математического обеспечения, реализацию технологического процесса обработки информации по задачам, эксплуатация системы в целом.

Текущие затраты рассчитываются по формуле

|  |  |
| --- | --- |
| Зтек = ЗзП + Са + Зэ + Срем + Зм + Зн , | (4.12) |

где ЗзП – затраты на зарплату основную и дополнительную с отчислениями во внебюджетные фонды, руб.;

Са – амортизационные отчисления от стоимости оборудования и устройств системы, руб.;

Зэ – затраты на силовую энергию, руб.;

Срем – затраты на текущий ремонт оборудования и устройств системы, руб.;

Зм – затраты на материалы и машинные носители, руб.;

Зн – накладные расходы информационного отдела, руб.

Эксплуатацию разработанной системы осуществляют специалисты. Затраты на заработную плату основную и дополнительную с отчислениями на социальные нужды производственного персонала рассчитываются по формуле

|  |  |
| --- | --- |
| , | (4.13) |

где *ti* – время эксплуатации системы *i*-м работником, дни;

3*i* – среднедневная заработная плата *i*-го работника, руб./день.

Данные расчета заработной платы специалистов приведены в таблицах 4.8 и 4.9.

Таблица 4.8 – Данные по заработной плате специалистов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Должность | Должностной оклад, руб. | Средняя дневная ставка, руб./день | Затраты времени на эксплуатацию, чел.дней | Фонд заработной платы, руб. |
| Проект-менеджер | 12000 | 571,4 | 65 | 37141 |
| Тестировщик | 8000 | 381 | 95 | 36195 |
| Итого | | | | 73336 |

Сзп1 = (65 × 571,4 + 95 × 381) × 1,4 × 1,302 руб. = 133676,9 руб. (за год).

Таблица 4.9 – Данные по заработной плате специалистов

(для продукта-аналога)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Должность | Должностной оклад, руб. | Средняя дневная ставка, руб./день | Затраты времени на эксплуатацию, чел.дней | Фонд заработной платы, руб. |
| Проект-менеджер | 12000 | 571,4 | 75 | 42855 |
| Тестировщик | 8000 | 381 | 105 | 40005 |
| Итого | | | | 82860 |

Сзп2 = (75 × 571,4 + 105 × 381) × 1,4 × 1,302 = 151037,2 руб. (за год).

Сумма амортизационных отчислений рассчитывается следующим образом:

|  |  |
| --- | --- |
| , | (4.14) |

где *Cbj* – балансовая стоимость *j*-ro вида оборудования, руб.;

*tj* – время работы *j*-ro вида оборудования, час;

*F*эф*j* – эффективный фонд времени работы оборудования в год, час;

*aj* – норма годовых амортизационных отчислений для *j*-гo вида оборудования;

*gj* – количество единиц оборудования *j*-гo вида.

Эффективный фонд времени работы оборудования можно вычислить по формуле

|  |  |
| --- | --- |
| *Fэф =Dр×*Нэ, | (4.15) |

где *Dp* – количество рабочих дней в году. *Dp* = 249;

Нэ – норматив среднесуточной загрузки, час./день, Нэ = 8.

Таким образом, эффективный фонд времени работы оборудования составит

*Fэф*=249×8=1992 час.

Данные для расчета:

*aj*=0,2 (используется ускоренная амортизация);

*gj*=1;

*tj* (для проекта) = (65 + 95) × 8 = 1280 час.;

*tj*(для аналога) = (75 + 105) × 8 = 1440 час.;

*Cbj*= 24000 руб.

Сумма амортизационных отчислений для проекта составит

*Са*1= 0,01 × ((24000 × 20 × 1 × 1280)/ 1992) руб. = 3084 руб.

Сумма амортизационных отчислений для аналога составит

*Са*2=0,01 × ((24000 × 20 × 1 × 1440)/ 1992) руб. = 3470 руб**.**

Затраты на силовую энергию рассчитываются по формуле

|  |  |
| --- | --- |
| , | (4.16) |

где *Nj* – установленная мощность *j*-го вида технических средств, кВт;

*tj*– время работы *j*-го вида технических средств, час;

*gj*– коэффициент использования установленной мощности оборудования;

Тэ– тариф на электроэнергию, руб./кВт ч.

В настоящее время тариф на электроэнергию в г. Томск составляет 1,51 руб./кВт ч, установленная мощность для компьютера равна 0,2 кВт, таким образом затраты на силовую энергию для проекта составят Зэ = 0,2 × 1280 × 1,51 руб. =386,56 руб., для аналога составят Зэ = 0,2 × 1440 × 1,51 руб. = 434,88 руб.

Затраты на текущий ремонт оборудования рассчитываются по формуле

|  |  |
| --- | --- |
| , | (4.17) |

где *Cpi* - норматив затрат на ремонт (*Cpi* = 0,05).

Затраты на текущий ремонт оборудования составят:

* для проекта Зрем1*=* (0,05 × 24000 × 1280)/ 1992= 771,08 руб.,
* для аналога Зрем2 = (0,05 × 24000 × 1440)/ 1992= 867,47 руб.

Затраты на материалы, потребляемые в течение года, составляют 1 % от балансовой стоимости основного оборудования и равны 240 руб. (24000×0,01).

Накладные расходы включают затраты на содержание административного и управленческого персонала, на содержание помещения и т.д. Норматив накладных расходов составляет 20 % от прямых затрат, включающих первые пять статей затрат, представленных в таблице 4.10.

Накладные расходы для проекта:

Зн1= (133676,9 + 3084 + 386,56 + 771,08 + 240) × 0,2 = 27632 руб.

Накладные расходы для аналога:

Зн2 = (151037,2 + 3470 + 434,88 + 867,47 + 240) × 0,2 = 31209,91 руб.

Таблица 4.10 – Годовые эксплуатационные затраты

|  |  |  |
| --- | --- | --- |
| Статьи затрат | Затраты на проект, руб. | Затраты на аналог, руб. |
| Основная и дополнительная зарплата с отчислениями во внебюджетные фонды | 133676,9 | 151037,2 |
| Амортизационные отчисления | 3084 | 3470 |
| Затраты на электроэнергию | 386,56 | 434,88 |
| Затраты на текущий ремонт | 771,08 | 867,47 |

Окончание таблицы 1.10

|  |  |  |
| --- | --- | --- |
| Затраты на материалы | 240 | 240 |
| Накладные расходы | 27632 | 31209,91 |
| **Итого** | 165790,5 | 187259 |

## 4.6 Расчет показателя экономического эффекта

Оценка экономической эффективности вариантов проектных решений элементов АИС основывается на расчете показателей сравнительной экономической эффективности капитальных вложений. Годовой экономический эффект от использования разрабатываемой системы определяется по разности приведенных затрат на базовый и новый варианты в расчете на годовой объем выпуска:

|  |  |
| --- | --- |
| Э=(З1×*Ak*– З2) ×*N*, | (4.18) |

где З1,З2 *–* приведенные затраты на единицу работ, выполняемых с помощью базового и проектируемого вариантов процесса обработки информации, руб.;

*Ak –* коэффициент эксплуатационно-технической эквивалентности, или технического уровня (формула (1.2)), *Ak =* 1,57;

– объем работ, выполняемых с помощью разрабатываемого продукта (примем равным 1).

Приведенные затраты З*i* на единицу работ, выполняемых по базовому и разрабатываемому вариантам, рассчитываются по формуле

|  |  |
| --- | --- |
| 3*i=*C*i +* Eн×K*i*, | (4.19) |

где C*i –* себестоимость (текущие эксплуатационные затраты единицы работ), руб.;

Ен *–* нормативный коэффициент экономической эффективности (Ен *=*0,33);

K*i –* суммарные затраты, связанные с внедрением нового проекта.

Затраты на единицу работ по аналогу:

З1= 187259 + 0,33 × 117000 = 225869 руб.

Затраты на единицу работ по проекту:

З2= 165790,5 + 0,33 × 78407,77 = 191665,1 руб.

Экономический эффект от использования разрабатываемой системы:

Э= 225869 × 1,57 – 191665,1 = 162949 руб.

Сводные данные по расчету экономического эффекта приведены в таблице 4.11.

Таблица 4.11 – Экономический эффект

|  |  |  |
| --- | --- | --- |
| Характеристика | Значение | |
| продукт-аналог (базовый) | разрабатываемый продукт |
| Себестоимость (текущие эксплуатационные затраты), руб. | 187259 | 165790,5 |
| Суммарные затраты, связанные с внедрением проекта, руб. | 117000 | 78407,77 |
| Приведенные затраты на единицу работ, руб. | 225869 | 191665,1 |
| Экономический эффект от использования разрабатываемой системы, руб. | 162949 | |

После определения годового экономического эффекта необходимо рассчитать срок окупаемости затрат на разработку продукта по формуле

|  |  |
| --- | --- |
| Ток = К/ Э. | (4.20) |

Срок окупаемости составит: Ток = 78407,77/ 162949 = 0,48 года.

Затем рассчитаем фактический коэффициент экономической эффективности разработки (Еф) и сопоставим его с нормативным значением коэффициента эффективности капитальных вложений Ен*=*0,33*:*

|  |  |
| --- | --- |
| Еф=1/Tок.= 1/ 0,48 = 2,07. | (4.21) |

Фактический коэффициент экономической эффективности разработки получился больше, чем нормативный, поэтому разработка и внедрение разрабатываемого продукта является эффективной.

Таким образом, в ходе проделанной работы найдены все необходимые данные, доказывающие целесообразность и эффективность данной разработки. Приведем эти данные в сводной таблице (4.12).

Таблица 4.12 – Результаты экономического обоснования проекта

|  |  |
| --- | --- |
| Характеристика проекта | Значение |
| Затраты на разработку и внедрение проекта, руб. | 78407,77 |
| Общие эксплуатационные затраты, руб. | 165790,5 |
| Экономический эффект, руб. | 162949 |
| Коэффициент экономической эффективности | 2,07 |
| Срок окупаемости, лет | 0,48 |

## 4.7 Маркетинговое сопровождение разрабатываемого продукта

Автоматизированная система планирования и ведения процесса тестирования программного обеспечения была разработана для компании ООО «СкайТек».Основными пользователями продукта будут сотрудники компании – менеджеры проектов и тестировщики. Возможность его тиражирования и коммерческого распространения не рассматривается.

5 ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

# ПРИЛОЖЕНИЕ А

Эскиз плана тестирования

|  |  |  |
| --- | --- | --- |
| Разделпланатестирования | Заголовок | Содержание |
| 1.0 | Введение |  |
| 1.1 | Цель | Цельсозданияпланатестирования |
| 1.2 | Обзорсистемы | Описаниесистемы,функций,критичныхдлясистемы,ифункцийповышенногориска |
| 1.4 | Применимыедокументы | Документация,относящаясякпрограмметестирования |
| 1.5 | Графикосновныхработпотестированию | События,работы,поставляемыедокументы |
| 2.0 | Ролииответственность |  |
| 2.1 | Организацияпроекта | Диаграммаорганизациипроекта |
| 2.2 | Ролииответственностьвпроекте | Распределениеиответственностьролей |
| 2.3 | Декомпозицияработпотестированию | Задачитестированияидекомпозицияработ |
| 2.4 | Ресурсыгруппытестирования | Знанияинавыкитестировщиков |
| 3.0 | Программатестирования |  |
| 3.1 | Рамки | Краткоеописаниепокрытиятребованийсистемойтестов |
| 3.2 | Подходктестированию | Цели,задачииметодологияпроцесса;параметрыпрограммытестирования |
| 3.3 | Стратегиитестирования | Стратегиитестирования |
| 3.4 | Инструментытестирования | Описаниеиспользуемыхинструментовтестирования |
| 3.5 | Требованияктестам | Требованияктестам |
| 3.6 | Проектированиетестов | Проектированиетестов,соглашениепонаименованиютестов |
| 4.0 | Средатестирования |  |
| 4.1 | Конфигурациясредытестирования | Архитектурасредытестирования |
| 4.2 | Тестовыеданные | Созданиетестовыхданных |
| 5.0 | Выполнениетестов |  |
| 5.1 | Подготовкаотчетовоходетестирования | Подготовкаотчетовоходетестирования,метрики |
| 5.2 | Отслеживаниедефектов | Отслеживаниедефектов |
| 6.0 | Подробныйплан-графиктестирования | Подробныйплан-графиктестирования |
| А | Тестовыепроцедура | Тестовыепроцедурыприемо-сдаточныхиспытаний |

# ПРИЛОЖЕНИЕ Б

Переченьметриктестированияприиспользованиистратегии«черногоящика»

|  |  |
| --- | --- |
| Наименование метрики | Описание |
| 1 | 2 |
| Покрытие | |
| Тестовое покрытие | Общее количество созданных тестовых процедур деленное на общее количество установленных требований к тестам. Глубина тестового покрытия обычно определяется установленными критериями завершения тестирования |
| Анализ системного покрытия | Процент числа вызовов функций, которые проверяют тесты, относительно общего числа вызовов функций в системе. Количественно измеряется покрытие на уровне интерфейса ПО |
| Покрытие функциональными тестами | Количество требований к тестам, проверяемых тестовыми процедурами, деленное на общее число требований к тестам. Измерение тестового покрытия до поставки программного продукта заказчику - процент проверенного программного продукта в любой момент тестирования |
| Ход выполнения работ | |
| Статус выполнения тестовых процедур | Отношение количества уже проведенных тестовых процедур к общему числу запланированных процедур. Изучение этого значения позволяет группе тестирования определить число тестовых процедур, которые еще предстоит выполнить |
| Коэффициент обнаружения ошибок | Общее число документированных дефектов, деленное на количество выполненных тестовых процедур |
| Возраст дефектов | Оборотное время на исправление дефекта, т.е. временной промежуток между обнаружением дефекта и его разрешением. Используя данные о возрасте дефекта, группа тестирования может провести анализ тенденций. Например, в проекте было зарегистрировано 100 дефектов. В том случае, если предыдущий опыт документально показывает, что группа разработки способна справлять 20 дефектов в день, оборотное время на эти отчеты о проблемах может равняться одной рабочей неделе. Тогда возраст дефекта должен быть равен в среднем 5 дням. Если возраст дефекта лежит в диапазоне от 10 до 15 дней, более медленное внесение исправлений разработчиками может помешать тестировщикам уложиться в установленные в плане-графике сроки |
| Повторное тестирование исправленного дефекта | Время, прошедшее между исправлением дефекта в новой версии и его повторным тестированием. Показывает, проводит ли группа тестирования повторное тестирование исправленных фрагментов приложения в необходимом объеме |
| Анализ тенденции дефекта | Общее число найденных дефектов с количеством тестовых процедур, проведенных в течение определенного времени. Может способствовать определению тенденции обнаружения дефектов. Улучшается или ухудшается эта тенденция по мере приближения к окончанию фазы системного тестирования? |

|  |  |
| --- | --- |
| Качество | |
| 1 | 1 |
| Показатель успешности тестирования/ коэффициент качества | Отношение общего числа успешно завершившихся тестовых процедур к общему количеству выполненных процедур. Этот показатель помогает группе тестирования более точно оценить объем успешно работающей функциональности |
| Качество исправлений 1 | Общее количество повторно открытых дефектов, деленное на общее количество исправленных дефектов. Качество исправлений, произведенных для разрешения отчетов о проблемах. При высоком значении данного показателя группе тестирования, возможно, придется известить разработчиков о наличии проблемы |
| Качество исправлений 2 | Работающая ранее функциональность в сравнении с вновь возникшими ошибками. Этот показатель помогает группе тестирования определить, в какой степени исправления программного продукта негативно повлияли на работавшую ранее функциональность |
| Плотность дефектов | Отношение общего количества найденных дефектов к количеству тестовых процедур, выполненных для данной функциональности или сценария использования системы. Следует принимать во внимание приоритет отчетов о проблемах. Например, одно из требований к приложению может иметь 50 отчетов о проблемах с низким приоритетом, причем критерии завершения тестирования удовлетворены. Другое требование, возможно, имеет один открытый отчет о проблеме с высоким приоритетом, который препятствует выполнению критериев завершения тестирования |
| Анализ тенденции дефекта | Отношение общего количества найденных дефектов к количеству проведенных тестовых процедур. Если в начале тестирования выявлено большое количество ошибок и число вновь обнаруживаемых дефектов снижается после однократного проведения всех тестовых процедур, то тестировщик может сделать вывод о тенденции к улучшению |
| Эффективность тестов | Необходимо вести статистику эффективности тестов. В некоторых случаях по результатам тестирования проводится неадекватный анализ. Группе тестирования следует обратиться за помощью к лицам, имеющим опыт использования приложения, чтобы изучить результаты тестирования и определить их корректность |
| Отчет о проблеме | Количество отчетов об ошибках, классифицированных по уровню приоритета. Тестировщик должен выяснить, удовлетворяет ли проверяемое приложение критериям, установленным заказчиком и подтвержденным в плане тестирования |

# ПРИЛОЖЕНИЕ В

Оценочныеэлементыфактора«Надежность»

|  |  |  |
| --- | --- | --- |
| Наименование | Методоценки | Оценка |
| 1 | 2 | 3 |
| Наличиетребованийкпрограммепоустойчивостифункционированияприналичииошибоквовходныхданных | Экспертный | 0-1 |
| Возможностьобработкиошибочныхситуаций | Экспертный | 0–1 |
| Полнотаобработкиошибочныхситуаций | Экспертный | 0–1 |
| Наличиетестовдляпроверкидопустимыхзначенийвходныхданных | Экспертный | 0–1 |
| Наличиесистемыконтроляполнотывходныхданных | Экспертный | 0–1 |
| Наличиесредствконтролякорректностивходныхданных | Экспертный | 0–1 |
| Наличиесредствконтролянепротиворечивостивходныхданных | Экспертный | 0–1 |
| Наличиепроверкипараметровиадресовподиапазонуихзначений | Экспертный | 0-1 |
| Наличиеобработкиграничныхрезультатов | Экспертный | 0–1 |
| Наличиеобработкинеопределенностей | Экспертный | 0–1 |
| Наличиетребованийкпрограммеповосстановлениюпроцессавыполнениявслучаесбояоперационнойсистемы,процессора,внешнихустройств | Экспертный | 0–1 |
| Наличиетребованийкпрограммеповосстановлениюрезультатовприотказахпроцессора,операционнойсистемы | Экспертный | 0–1 |
| Наличиесредстввосстановленияпроцессавслучаесбоевоборудования | Экспертный | 0–1 |
| Наличиевозможностиразделенияповременивыполненияотдельныхфункцийпрограмм | Экспертный | 0–1 |
| Наличиевозможностиповторногостартасточкиостанова | Экспертный | 0–1 |
| Наличиецентрализованногоуправленияпроцессами,конкурирующимииз-заресурсов | Экспертный | 0–1 |
| Наличиевозможностиавтоматическиобходитьошибочныеситуациивпроцессевычисления | Экспертный | 0–1 |
| Наличиесредств,обеспечивающихзавершениепроцессарешениявслучаяхпомех | Экспертный | 0–1 |
| Наличиесредств,обеспечивающихвыполнениепрограммывсокращенномобъемевслучаеошибокилипомех | Экспертный | 0–1 |
| Вероятностьбезотказнойработы | Регистрационный  Расчетный | ,где*N*–числоэкспериментов;*Q*–числозарегистрированныхотказов |

# ПРИЛОЖЕНИЕ Г

ВнутренниеметрикинадежностиПС

|  |  |  |  |
| --- | --- | --- | --- |
| Названиеметрики | Формуладляоценкиметрики | Исходныеданныедлявычисленияметрикипосоответствующейформуле | Источникиданныхдляизмерения |
| Завершенность | | | |
| Обнаружениеошибок | Х=А/В | А–количествоошибок,обнаруженныхприпроверке;В–количествопредполагаемыхошибок,котороедолжнобытьобнаруженоприпроверке(используяпредыдущуюисториюилисоответствующуюмодель) | А:отчетопроверке;  В:базаданныхорганизации |
| Устранениеошибок | Х=А/В | А–количествоисправленныхошибокпроектирования/кодирования;В–общееколичествоошибок,обнаруженныхприпроверке | А:отчетобустраненныхошибках;В:отчетопроверке |
| Полнотатестирования | Х=А/В | А–количествотестовыхвариантов,спроектированныхвпланетестированияиподтвержденныхприпроверке;В–количествотребуемыхтестовыхвариантов | А:плантестирования;В:требования |
| Устойчивостькошибке | | | |
| Предотвращениеотказов | Х=А/В | А–количествотиповыхошибок,обработкакоторыхпредусмотренаприпроектировании/кодировании;В–количествотиповыхошибок,которыедолжныбытьучтены | А:отчетопроверке;В:спецификациятребований |
| Предотвращениенекорректныхдействий | Х=А/В | А–количествофункций,реализованныхспредотвращениемнекорректныхдействий;В–количествотипичныхнекорректныхдействий,котороедолжнобытьучтено | А:отчетопроверке;В:спецификациятребований |
| Восстанавливаемость | | | |
| Способностьквосстановлению | Х=А/В | А–количествореализованныхтребованийквосстановлению,подтвержденноеприпроверке;В–общееколичествотребованийквосстановлению,определенныхвспецификации | А:отчетопроверке;В:спецификациятребований |
| Эффективностьвосстановления | Х=А/В | А–количествореализованныхтребованийквосстановлению,удовлетворяющихзаданномувременномуограничению;В–количествотребованийквосстановлениюсзаданнымивспецификациивременными | А:отчетопроверке;В:спецификациятребований |
| Соответствиенадежности | | | |
| Соответствиенадежности | Х=А/В | А–количествокорректнореализованныхэлементов,связанныхссоответствиемнадежности,подтвержденноеприоценке;В–общееколичествоэлементовсоответствия | А:отчетопроверке;В:спецификациясоответствия,проект,исходныйкодисвязанныестандарты,соглашенияилинормативы |

# ПРИЛОЖЕНИЕ Д

ВнешниеметрикинадежностиПС

|  |  |  |  |
| --- | --- | --- | --- |
| Названиеметрики | Формуладляоценкиметрики | Исходныеданныедлявычисленияметрикипосоответствующейформуле | Источникиданныхдляизмерения |
| 1 | 2 | 3 | 4 |
| Завершенность | | | |
| Оценочнаяплотностьскрытыхошибок | Х=(Abs(A1–А2))/B | А1–общееколичествопрогнозируемыхскрытыхошибоквпрограммномпродукте;А2–общееколичествореально  обнаруженныхвтечениеопределенногоиспытательногосрокаошибок;В–размерпродукта | А1:модельпредсказанияростанадежности;А2:отчетотестировании,отчетобэксплуатации,отчетопроблеме |
| Плотностьотказовпотестовымвариантам | Х=А/В | А–количествообнаруженныхвтечениеопределенногоиспытательногосрокаотказов;В–количествовыполненныхтестовыхвариантов | А,В:отчетотестировании,отчетобэксплуатации,отчетопроблеме |
| Устранениеотказов | Х=А/В | А–количествоустраненныхвтечениеопределенногоиспытательногосрокаотказов;В–общееколичествореальнообнаруженныхотказов | А,В:отчетотестировании,отчетобэксплуатации |
| Плотностьошибок | Х=А/В | А–количествоошибок,обнаруженныхвтечениеопределенногоиспытательногосрока;В–размерпродукта | А:отчетотестировании,отчетобэксплуатации,отчетопроблеме |
| Устранениеошибок | Х=А/ВилиY=А/С | А–количествоисправленныхошибок;В–общееколичествореальнообнаруженныхпритестированииошибок;С–общееколичествопрогнозируемыхскрытыхошибоквпрограммномпродукте | А,В:отчетотестировании;С:базаданныхорганизации |
| Среднеевремямеждуотказами | Х=Т1/АилиХ=Т2/А | Т1–времяэксплуатации;Т2–суммаинтерваловвременимеждупоследовательнымислучаяминаступленияотказов;А–общееколичествореальнообнаруженныхотказоввтечениеопределенногопериодаэксплуатации | Т1,Т2,А:отчетотестировании,отчетобэксплуатации |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Тестовоепокрытие | Х=А/В | А–количествореальновыполненныхтестовыхвариантов,представляющихсценарийэксплуатациивовремятестирования;В–количествотестовыхвариантов,которыедолжныбытьвыполнены,чтобыпокрытьтребования | А:отчетотестировании,отчетобэксплуатации;В:спецификациятребований,спецификациятестированияилируководствопользователя |
| Тестоваязавершенность | Х=А/В | А–количествопрогонявшихсятестовыхвариантоввовремятестированияилиэксплуатации;В–количествотестовыхвариантов,которыедолжныбытьвыполнены,чтобыпокрытьтребования | А:отчетотестировании,отчетобэксплуатации;В:спецификациятребований,спецификациятестовилируководствопользователя |
| Устойчивостькошибке | | | |
| Предотвращениеаварийныхотказов | Х=1–А/В | А–количествоаварийныхотказов;В–общееколичествоотказов | А,В:отчетотестировании,отчетобэксплуатации |
| Предотвращениеотказов | Х=А/В | А–количествослучаевпредотвращениятипичныхисерьезныхотказоввовремявыполнениятестовых | А,В:отчетотестировании,отчетобэксплуатациивариантов,направленныхнапроверкутипичныхошибок,которыемогутпривестикотказу;В–количествовыполненныхтестовыхвариантов,направленныхнапроверкутипичныхошибок,которыемогутпривестикотказу |
| Предотвращениенекорректныхдействий | Х=А/В | А–количествослучаевпредотвращения  критическихисерьезныхотказов;В–количествовыполненныхпритестированиитестовыхвариантов,направленныхнапроверкутиповыхнекорректныхдействий,которыемогутпривестикотказу | А,В:отчетотестировании,отчетобэксплуатации |

|  |  |  |  |
| --- | --- | --- | --- |
| Восстанавливаемость | | | |
| 1 | 2 | 3 | 4 |
| Готовность | У=А/В | А – общее число случаев успешного использования программного средства пользователем; В – общее число попыток использования ПС пользователем за время наблюдения | А, В: отчет о  тестировании, отчет  об эксплуатации |
| Среднее время  простоя | Х=Т/В | Т – общее время простоя;  В – количество отмеченных аварийных отказов | Т, В: отчет о тестировании, отчет об эксплуатации |
| Среднее время восстановления | Х = Sum(T)/B | Т – время восстановления неработающей системы программного обеспечения в каждом соответствующем случае; В – количество случаев, когда наблюдаемая система программного обеспечения восстановлена | Т, В: отчет о тестировании, отчет об эксплуатации |
| Способность к повторному запуску | Х=А/В | А – число повторных запусков, удовлетворяющих требуемому времени, во время тестирования или поддержки пользовательской эксплуатации; В – общее число повторных запусков во время тестирования или поддержки пользовательской эксплуатации | А, В: отчет о тестировании, отчет об эксплуатации |
| Способность к восстановлению | Х=А/В | А – количество случаев успешного восстановления;  В – количество случаев восстановления, протестированных согласно требованиям | А: отчет о тестировании, отчет об эксплуатации;  В: спецификация требований, спецификация тестов или руководство пользователя |
| Эффективность восстановления | Х=А/В | А – количество случаев успешного восстановления, удовлетворяющих заданному времени восстановления; В – количество выполненных случаев восстановления, в которых должно быть удовлетворено заданное время восстановления | А, В: отчет о тестировании, отчет об эксплуатации |
| Соответствие надежности | | | |
| Соответствие надежности | Х=1 – А/В | А – количество заданных элементов соответствия надежности, которые не были выполнены при тестировании; В – общее количество заданных | А: отчет о тестировании; В: описание соответствия продукта (руководства пользователя или спецификации), связанные стандарты, соглашения или нормативы, спецификация тестов |

обновить

ПРИЛОЖЕНИЕ Е

Концептуальное моделирование базы данных

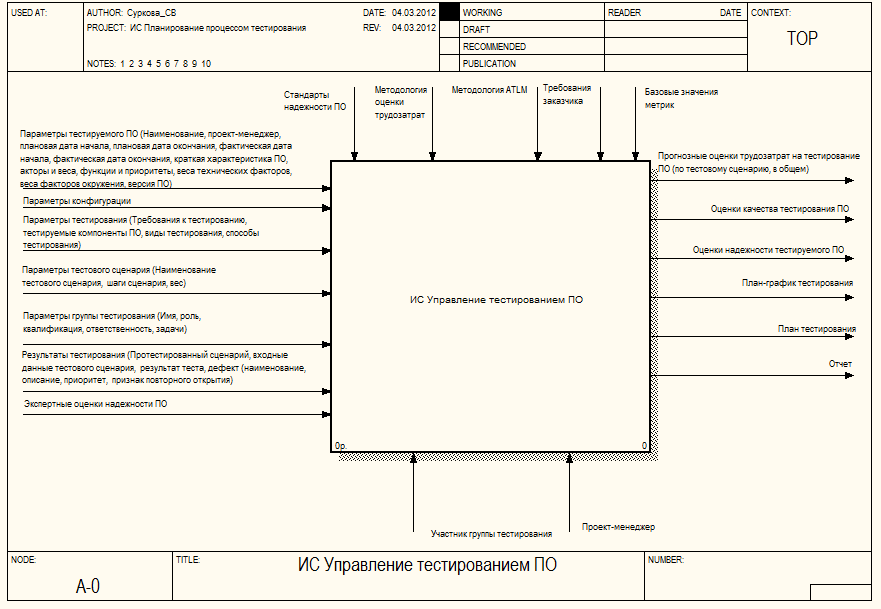


Рисунок E.1 – Концептуальная модель базы данных ИС «Управление процессом тестирования». ER-диаграмма

Рисунок E.2 – Концептуальная модель базы данных. FA-диаграмма

Таблица E.1– Глоссарий сущностей ОБНОВИТЬ

|  |  |
| --- | --- |
| Наименованиесущности | Определение |
| АКТОР | Любаявнешняяпоотношениюкмоделируемойсистемесущность(человек,программнаясистема,аппаратноеустройство),котораявзаимодействуетссистемойииспользуетеефункциональныевозможностидлядостиженияопределенныхцелейилирешениячастныхзадач |
| ВЕРСИЯ\_ПО | ВерсияПО(номер,номериособенности) |
| ВЕСОВОЙ\_КОЭФФИЦИЕНТ | Коэффициентсозначениямиот0до1 |
| ВТОРИЧНАЯ\_МЕТРИКА | Вычисленныезначенияпоказателейнаосновепервичныхметрик |
| ГРУППА\_ТЕСТИРОВАНИЯ | Командатестировщиков,назначенныхнатестированиеПО |
| ДЕФЕКТ | Ошибка,обнаруженнаявПОвовремятестирования |
| ЗАДАЧА | Задание,котороедолжнобытьвыполненоврамкахпрограммытестирования |
| НАБОР\_ВХОДНЫХ\_ДАННЫХ | Наборданных,которыйиспользуетсядлятестированияПО |
| ОЦЕНКА\_НАДЕЖНОСТИ\_ПО | Оценканадежности,вычисленнаянаосновеиспользуемойметодологииистандартов |
| ОЦЕНКА\_ТЕХНИЧЕСКОГО\_ФАКТОРА | ОтносительнаяважностьнефункциональныхтребованийкПО(требованиезаказчика) |
| ОЦЕНКА\_ФАКТОРА\_ОКРУЖЕНИЯ | Оценкадляучетавлиянияусловийвыполненияпроекта |
| ПЕРВИЧНАЯ\_МЕТРИКА | Количественныеизмерители,полученныенаосноверезультатовпроведениятестирования |
| ПОДЗАДАЧА | Минимальнаяединицазадания,полученнаяпутемдекомпозициизадачи |
| ПОДЗАДАЧА\_УЧАСТНИКА\_ГРУППЫ\_ТЕСТИРОВАНИЯ | Задача,назначеннаянаконкретногоучастникагруппытестирования |
| ПРИОРИТЕТ | Показательважности(от0до10) |
| ПРОГРАММА\_ТЕСТИРОВАНИЯ | РамкиработпотестированиюПО |
| ПРОЕКТ\_ТЕСТИРОВАНИЯ | Описаниепрограммногопродуктасточкизренияпроведенияпроцедурпланирования,реализациииоценкирезультатовтестирования |
| РОЛЬ\_УЧАСТНИКА | Справочникролейучастникагруппытестирования |
| СОТРУДНИК | Справочниксотрудников |
| СПОСОБ\_ТЕСТИРОВАНИЯ | Способ,которыйнеобходимоприменитьдлятестированиязадачи |
| ТЕСТИРУЕМОЕ\_ПО | ПО,длякоторогонеобходимосоставитьплантестирования |
| ТЕСТОВАЯ\_КОНФИГУРАЦИЯ | Конфигурация,идентичнаясредеиспользованияПО,илисокращеннаяверсиярабочейсредыиспользованияПО |
| ТЕСТОВЫЙ\_СЦЕНАРИЙ | Наборпоследовательностейвзаимодействиямеждусистемойиоднимизакторов(действияактораиоткликисистемы),котороепереводитсистемуизодногостабильногосостояниявдругое |
| ТИП\_ДЕФЕКТА | СправочниктиповдефектовПО |
| ТИП\_МЕТРИКИ | Справочникметриктестирования |
| ТИП\_ТЕХНИЧЕСКОГО\_ФАКТОРА | Справочниктиповтехническихфакторов |
| ТИП\_ФАКТОРА\_ОКРУЖЕНИЯ | Справочниктиповфакторовокружения |
| УЧАСТНИК\_ГРУППЫ\_  ТЕСТИРОВАНИЯ | Членкомандытестированиясопределеннойрольюиназначаемыминанегозадачами |
| ФАКТОР\_НАДЕЖНОСТИ | Справочникфакторовнадежности |
| ФУНКЦИЯ\_ПО | Возможность,которуюпредоставляетданноеПО |
| ЭКСПЕРТНАЯ\_ОЦЕНКА\_ФАКТОРА | ОценкафакторанадежностиПО,даннаяэкспертом |



ПРИЛОЖЕНИЕ Ж

Функциональное моделирование

ИС «Управление процессом тестирования»

Рисунок Ж.1 – SADT-модель ИС «Управление процессом тестирования». Уровень А-0

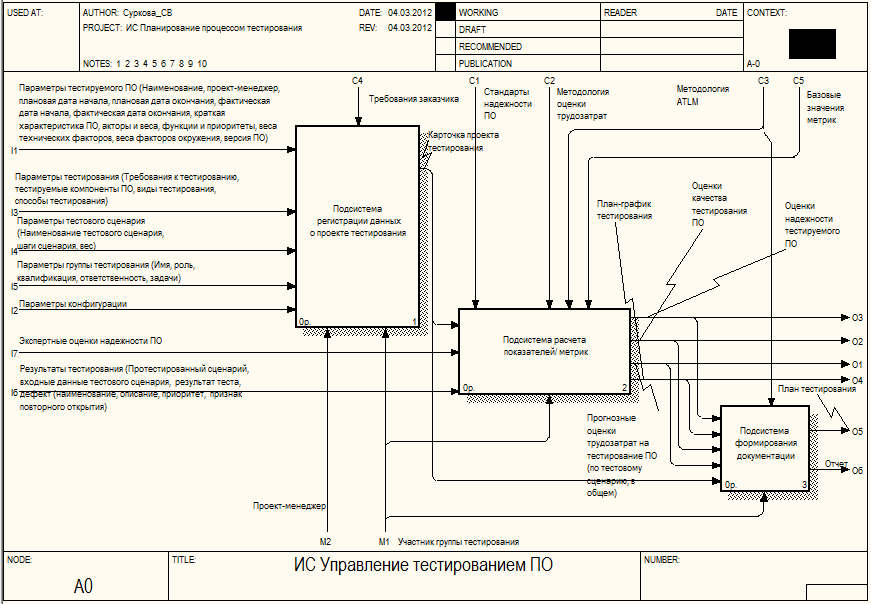


Рисунок Ж.2 – SADT-модель ИС «Управление процессом тестирования». Уровень А0

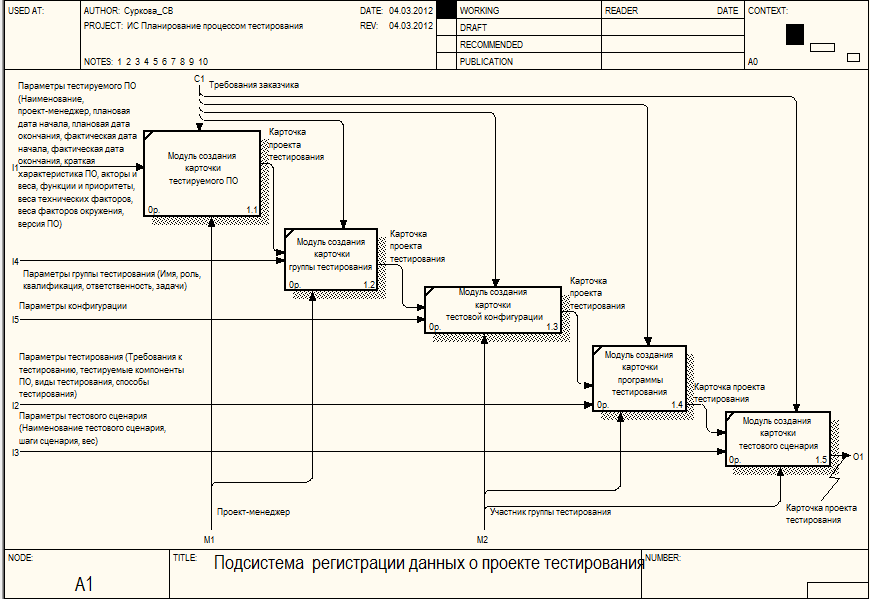


Рисунок Ж.3 – SADT-модель ИС «Управление процессом тестирования». Уровень А1 ОБНОВИТЬ

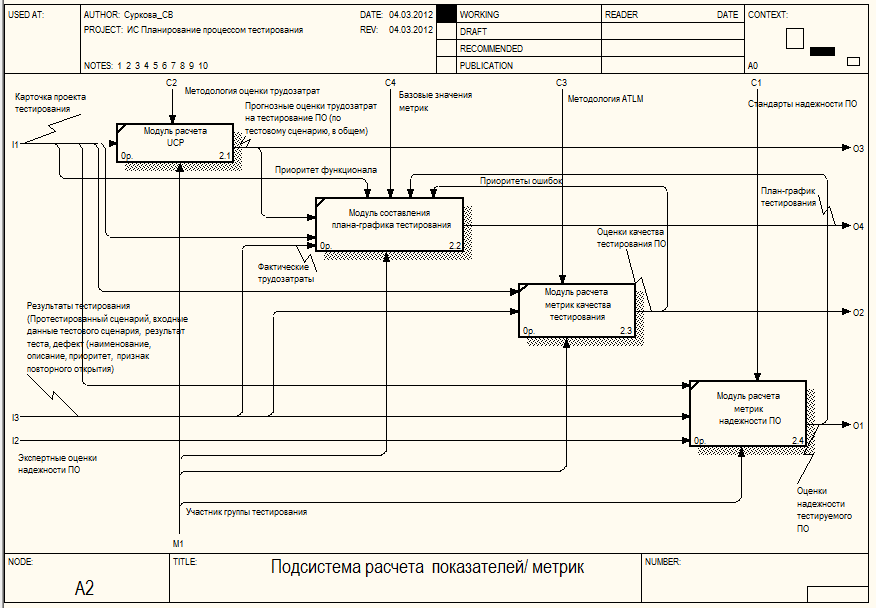


Рисунок Ж.4 – SADT-модель ИС «Управление процессом тестирования». Уровень А2

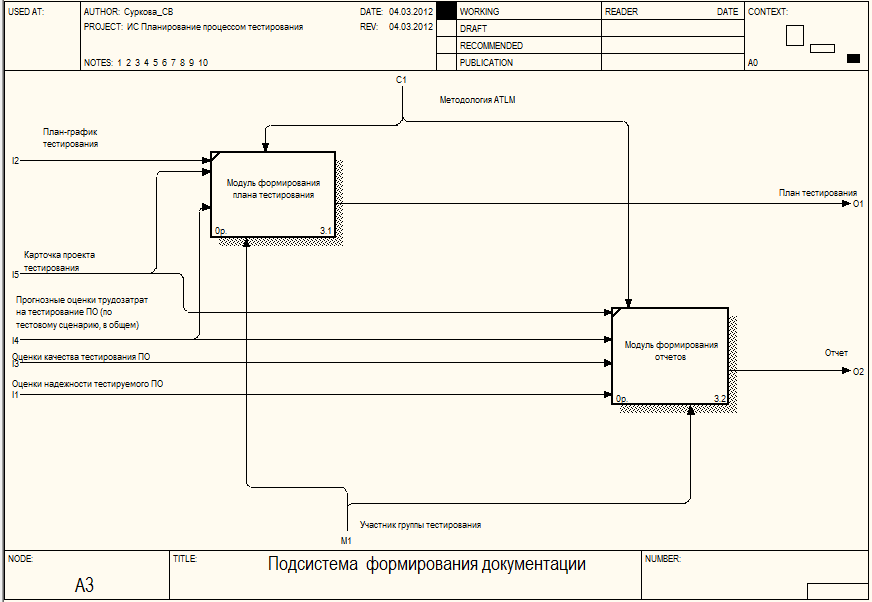


Рисунок Ж.5 – SADT-модель ИС «Управление процессом тестирования». Уровень А3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название | Производи-тель | Функции ИС | Цена | Платформа/ дополнительные требования | Недостатки  ПРИЛОЖЕНИЕ З  Аналоги ИС «Управление процессом тестирования» |
| |  |  | | --- | --- | | |  | | --- | | Testlink | | | |  |  | | --- | --- | | |  | | --- | | Testlink  Community | | | Описание требований, хранение различных версий требований. Описание тестов, группировка тестов в различные тестовые наборы. Планирование выполнения тестов. Протоколы испытаний, метрик, графики, контроль выполнения тестов | Demo – GNU GPL | Internet Explorer 4+ (Windows) 5+ (Mac). Mozilla Fox (Windows + Linux). Netscape 6.1+ (Windows). Opera 7+ (Windows + Linux) | В качестве метрик используется простые показатели, основанные на первичных данных, полученных по результатам тестирования  Высокая стоимость приобретения и поддержки |
| «Управление тестированием продуктов» | DevProm | Создание описания проекта. Участники и роли. История изменений в проекте. Назначение задач пользователям. Первичные метрики тестирования. [Управление тестированием](http://devprom.ru/docs/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D1%8F-Devprom-Team#13397) (план-график тестирования). [Тестовая документация](http://devprom.ru/docs/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D1%8F-Devprom-Team#2038) ([отчеты](http://devprom.ru/docs/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D1%8F-Devprom-Team#3225)). [Метрики по стадиям проекта](http://devprom.ru/docs/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D1%8F-Devprom-Team#3243). [Поиск](http://devprom.ru/docs/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D1%8F-Devprom-Team#13402). | 6750 руб. за пользователя/ 750 руб./мес. за пользователя | ОС – Windows, Linux/Unix  Браузер: Firefox, Chrome, Opera, Safari, IE |
| TestRail | GurockSoftware | Организация проектов – поддерживает управление всеми текущими тестируемыми проектами. Организация сопутствующей документации в структурированном виде. Регистрация результатов испытаний. Планирование предстоящих мероприятий по тестированию. Отчеты. Сохранение истории изменения тестов/ проектов. Назначение заданий пользователям. | 239$ для 1 пользователя/ 2149$ для 10 пользователей/ 25$ (1 польз-ль/ мес.) | ОС – Windows, Linux/Unix  Браузер: Internet Explorer 7, 8, 9; Firefox 3.5 и выше; Chrome, Safari, Webkit; Adobe Flash Player 9.x и выше  *БД*: SQL Server 2005/ 2008 |
| InformUP | informUpLtd | Управление требованиями и тестовыми сценариями. Отслеживание дефектов / ошибок, задач пользователей.  Отчеты по результатам тестирования. План-график тестирования.  Рассылка уведомлений. | 5 польз-й –  375$; 10  польз-й - 600$  хостинг  7$ (1 польз-ль/  мес.) | ОС – Windows, Linux/Unix  Браузер: Internet Explorer 7, 8, 9; Firefox 3.5 и выше; Chrome, Safari |
| QA Complete |  | Создание описаний тестовых требований, особенностей, тестовых сценариев. Описание конфигурации тестирования.  Регистрация результатов тестирования, отслеживание состояния результатов во времени и прогресса тестирования.  Полный контроль над проектом от этапа создания плана тестирования до составления отчетов о результатах тестирования. Определение требований для каждой версии тестируемого ПО. Сохранение истории изменения требований, отслеживание охвата всех требований тестами | 399$ - годовое обслуживание | ОС – Windows, Linux/Unix  Браузер: Internet Explorer 7, 8, 9; Firefox 3.5 и выше; Chrome, Safari |