Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт управления бизнес-процессами и экономики
Кафедра экономики и информационных технологий менеджмента

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.А. Ступина
подпись
«____» _____ 2017г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Методика расчета стоимости реализации аппаратно-программного комплекса отказоустойчивых систем управления

09.04.03 Прикладная информатика

09.04.03.02 Реинжиниринг бизнес-процессов

Научный руководитель _____ профессор, А.А. Ступина
подпись, дата д-р техн. наук

Выпускник _____ К.В. Ярков
подпись, дата

Рецензент _____ доцент Д.В. Сорокин
подпись, дата канд.техн.наук

Красноярск 2017

РЕФЕРАТ

Выпускная квалификационная работа содержит 88 страниц текстового документа, 3 приложения, 62 использованных источников, 3 таблицы, 17 рисунков.

СТОИМОСТЬ, ОЦЕНКА ЗАТРАТ, АППАРАТНО-ПРОГРАММНЫЙ КОМПЛЕКС, ОКТАЗОУСТОЙЧИВОСТЬ, СИСТЕМЫ УПРАВЛЕНИЯ, ИНФОРМАЦИОННАЯ СИСТЕМА, IDEF0, БИЗНЕС-ПРОЦЕСС, ЖИЗНЕННЫЙ ЦИКЛ, МОДЕРНИЗАЦИЯ, МОДЕЛЬНО-АЛГОРИТМИЧЕСКОЕ ОБЕСПЕЧЕНИЕ, НАДЕЖНОСТЬ, КРИТИЧЕСКИ ВАЖНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ.

Объектом исследования в данной работе являются аппаратно-программные комплексы отказоустойчивых систем управления.

Предметом исследования являются бизнес-процесс оценки затрат на разработку и модернизацию программного обеспечения.

Целью настоящей работы является теоретическое обоснование и разработка методических инструментов расчета стоимости реализации аппаратно-программного комплекса отказоустойчивых систем управления.

В диссертации проведен анализ моделей оценки стоимости программного обеспечения, обоснована необходимость в разработке модельно-алгоритмического обеспечения оценки затрат, разработан методический инструмент оценки затрат при многоэтапной реализации аппаратно-программного комплекса, проведена апробация методики при модернизации аппаратно-программного комплекса «Calculation of workspace» обеспечения охранной и пожарной безопасности, применяемого в ООО «Автоматические системы пожаротушения».

REVIEW

The final thesis contains 88 pages of a text document, 7 applications, 62 used sources, 7 tables, 18 sheets of graphic material.

COST, EVALUATION COSTS, HARDWARE AND SOFTWARE COMPLEX, OCTOASTIC RESISTANCE, CONTROL SYSTEMS, INFORMATION SYSTEM, IDEF0, BUSINESS PROCESS, LIFE CYCLE, MODERNIZATION, MODEL-ALGORITHMIC SUPPORT, RELIABILITY, CRITICALLY IMPORTANT INFORMATION SYSTEMS.

The object of my research in this work is a hardware-software complex of the fault-tolerant control systems.

The subject of my research is the business process for estimating the costs of developing and software upgrade.

The purpose of this work is the theoretical justification and development of the methodological tools for estimating the cost of implementation of the hardware and software complex for fault-tolerant control systems.

In the thesis, there was made an analysis of software cost estimation models, substantiated the need for the development of model-algorithmic support for cost estimation, provided a methodical tool for estimating costs for multi-stage implementation of the hardware and software complex. In addition to that, there was an approbation of the methodology for upgrading the hardware and software complex "Calculation of workspace" for providing security and fire Safety, that applied in "ASP" company.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность темы. При подготовке технических заданий на создание или доработку аппаратно-программного комплекса систем управления объектами в основу оценки сложности и стоимости проекта закладываются неформализованные представлений клиентов о требуемых функциях и характеристиках качества. Возможны существенные ошибки при определении трудозатрат, показателей качества, сложности, продолжительности и стоимости проекта. Наибольшую проблему составляют АПК, которые требуют повторной доработки. Особенно остро стоит вопрос в тех случаях, когда сменилась команда разработчиков или лояльность новой команды по отношению предыдущей версии низка.

Опрос экспертов и анализ статистических данных показал, что большая часть проектов выходит за рамки ранее спланированных бюджетов и календарных планов, половина из оставшихся вообще не могут быть закончены и лишь 25% завершаются в установленные сроки и полном объеме. В качестве причин называются следующие:

• нечетко формализованы функциональные требования к конечному продукту, что не позволяет разработчикам предусмотреть работы и пакеты работ по созданию процессов, обеспечивающих требуемое качество и производительность разрабатываемой системы;

• при заключении контрактов часто происходит занижение необходимого бюджета, сроков и ресурсов, т.к. заказчик старается сэкономить, а у исполнителя недостаточно аргументированных доказательств, а это влияет на снижение качества разрабатываемых программных комплексов.

При невозможности выполнения работ изначально заданного уровня качества некоторые компании – разработчики вынуждены уменьшать заявленный функционал. Это приводит к дополнительным убыткам

компании разработчика, если такие ситуации не были предусмотрены в контракте с заказчиком.

Таким образом, умение достоверно рассчитать риски проекта на этапе инициирования, проанализировать возможные сценарии развития событий, правильно оценить кадровый потенциал проектной команды и уровень сложности предстоящих работ позволит избежать финансовые потери компании- разработчика.

При разработке крупных ИТ - систем необходимо не только снизить зависимость качества результатов от субъективных факторов с помощью внедрения инновационных технологических методов оценки ресурсов, но и максимально автоматизировать данный процесс.

Вопросами оценки стоимости разработки программного обеспечения занимались в разное время Меламед А.Я., Глазова М.А., Сидоров Н.А., Барри В. Боэм, Крис Абтс, Макконнелл, Нельсон (одно из первых исследований в области оценки стоимости), Волвертон (модель Волвертона), Сунит Девнани-Чулани (модель COCOMOII), Лоуренс Путнама (концепция функциональных точек, модель SLIM). Проблемам повышения надежности аппаратно - программного комплекса, создания отказоустойчивых систем управления посвящены работы Ковалева И.В., Терскова В.А., Антамошкина А.Н., Липаева В.В., Брауде Э.Д., Липаева В.В., Позднякова Д.А. и других ученых. Несмотря на значительную проработку данных вопросов, анализ исследовательских работ показал, что слишком мало внимания уделено оценке стоимости модернизации и доработки АПК критически важных систем управления.

Необходимо провести анализ существующих моделей оценки стоимости программных продуктов и предложить алгоритм для оценки стоимости модернизации и доработки АПК критически важных систем управления.

Объектом исследования в данной работе являются аппаратно-программные комплексы отказоустойчивых систем управления.

Предметом исследования являются бизнес-процесс оценки затрат на разработку и модернизацию программного обеспечения.

Целью настоящей работы заключается в теоретическом обосновании и разработке методических инструментов расчета стоимости реализации аппаратно-программного комплекса отказоустойчивых систем управления.

Для достижения поставленной цели решались следующие задачи:

– анализ существующих методик, моделей и алгоритмов оценки затрат на разработку программного обеспечения систем управления;

– разработка методики оценки затрат на модернизацию аппаратно-программного комплекса (АПК);

– реализация программной системы для автоматизации расчетов по оценке затрат на разработку АПК и ее тестирование при решении практических задач.

В качестве примера аппаратно- программного комплекса отказоустойчивой системы управления выступают критически важные информационные системы обеспечения охранной и пожарной безопасности, применяемые в ООО «Автоматические системы пожаротушения».

Научная новизна исследований заключается в создании методики оценки затрат на разработку аппаратно-программного комплекса с учетом возможности введения программной избыточности при проектировании критически важных информационных систем, с учетом различного уровня оплаты труда персонала на разных этапах жизненного цикла разработки ПО, а также с учетом применения процессоров разного типа.

Научная значимость. Результаты, полученные при выполнении диссертационной работы, создают теоретическую основу для разработки моделей, методов и алгоритмов, направленных на повышение эффективности процессов разработки и модификации аппаратно-программных комплексов систем управления.

Практическая значимость. Разработанная в результате работы над диссертацией система оценки затрат на разработку аппаратно-программного

комплекса «*Calculation of workspace»* позволяет автоматизировать расчеты по оценке затрат на модернизацию программного обеспечения.

Методы исследования. При выполнении работы использовались методы оценки стоимости разработки программного обеспечения, структурный анализ, методы повышения отказоустойчивости программного обеспечения.

Апробация работы. Диссертационная работа в целом обсуждалась на научных семинарах кафедры экономики и информационных технологий менеджмента Сибирского федерального университета (2015-2017 гг.), результаты докладывались на научно-практических конференциях «Молодежная наука. Проспект Свободный – 2016», «Молодежная наука. Проспект Свободный – 2017», XIX межвузовской научно-практической конференции «Проблемы и перспективы развития железнодорожного транспорта», Международный студенческий научный вестник.

Разработанный программный продукт прошел опытное тестирование и внедрение в ООО «Автоматические системы пожаротушения».

Реализация результатов работы. Разработанная программная система прошла экспертизу и зарегистрирована в реестре программ для ЭВМ (№ гос. регистрации 2017615181), что делает ее доступной широкому кругу специалистов по системному анализу и разработке программного обеспечения.

Опубликовано 7 статей в научных журналах.

Достоверность полученных результатов подтверждается тестированием и оценкой результатов применения разработанной системы в реальных проектах, согласованностью расчетных и экспериментальных данных.

Структура и объем работы. Диссертация состоит из введения, трех глав, заключения, списка использованной литературы, списка публикаций автора по теме диссертации и приложения.

В первой главе данной работы приведены основные теоретические направления оценки затрат на разработку программного обеспечения отказоустойчивых систем, описаны неалгоритмические и алгоритмические модели оценки стоимости программного обеспечения, показаны принципы обеспечения надежности программного обеспечения, проанализированы различные подходы к созданию устойчивости к программным сбоям, обоснован выбор мультиверсионного подхода к решению проблемы.

Во второй главе предложено решение существующей проблемы, описаны этапы оценки затрат на программное обеспечение, предложены параметры для расчетов затрат, составлен алгоритм применения методики.

Третья глава является проектной частью, в которой предложены проектные решения по информационному, программному и аппаратному обеспечению, создана концепция информационной системы, функциональная архитектура, представлен интерфейс, произведено тестирование созданного программного обеспечения информационной системы.

1 ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ НАПРАВЛЕНИЯ ОЦЕНКИ СТОИМОСТИ РЕАЛИЗАЦИИ АППАРАТНО-ПРОГРАММНОГО КОМПЛЕКСА ОТКАЗОУСТОЙЧИВЫХ СИСТЕМ УПРАВЛЕНИЯ

1.1 Оценка стоимости реализации АПК: сущность и содержание

При проведении оценочных работ в рыночной экономике возникают различные методологические и практические проблемы, поэтому необходимо, прежде всего, правильно определить объекты и субъекты стоимостной оценки, выявить совокупность факторов, влияющих на величину стоимости и выбрать методы их анализа.

В качестве объекта оценки аппаратно-программный комплекс (АПК) представляет собой совокупность технических средств и программного обеспечения, которое предназначено для совместного решения задач. Это могут быть интеллектуальные системы хранения данных, современные офисные АТС, комплексы пожарно-охранной сигнализации, контроля доступа, видеонаблюдения, всевозможные решения для промышленной автоматизации и многое другое.

Разработчик реализуемого аппаратно-программного комплекса очень часто не знает заранее, где именно, на основе какой технической базы будет установлена предлагаемая информационная система, как она будет функционировать. В большинстве случаев предполагается один или нескольких типовых сценариев применения комплекса в составе системы, что неизбежно еще на первом этапе планирования закладывает необходимость оценки АПК в условиях модернизации и должно отражается на стоимости.

Стоимость является предметом сделки, основной составляющей рыночной цены. Рыночная цена – денежное выражение стоимости, результат договоренности между продавцом и покупателем.

Оценка стоимости представляет собой целенаправленный упорядоченный процесс определения количественной величины стоимости объекта в денежном выражении с учетом влияющих на нее факторов, в конкретный момент времени в условиях конкретного рынка.

Процесс оценки стоимости включает в себя ряд обязательных этапов, представленных на рисунке 1.1.

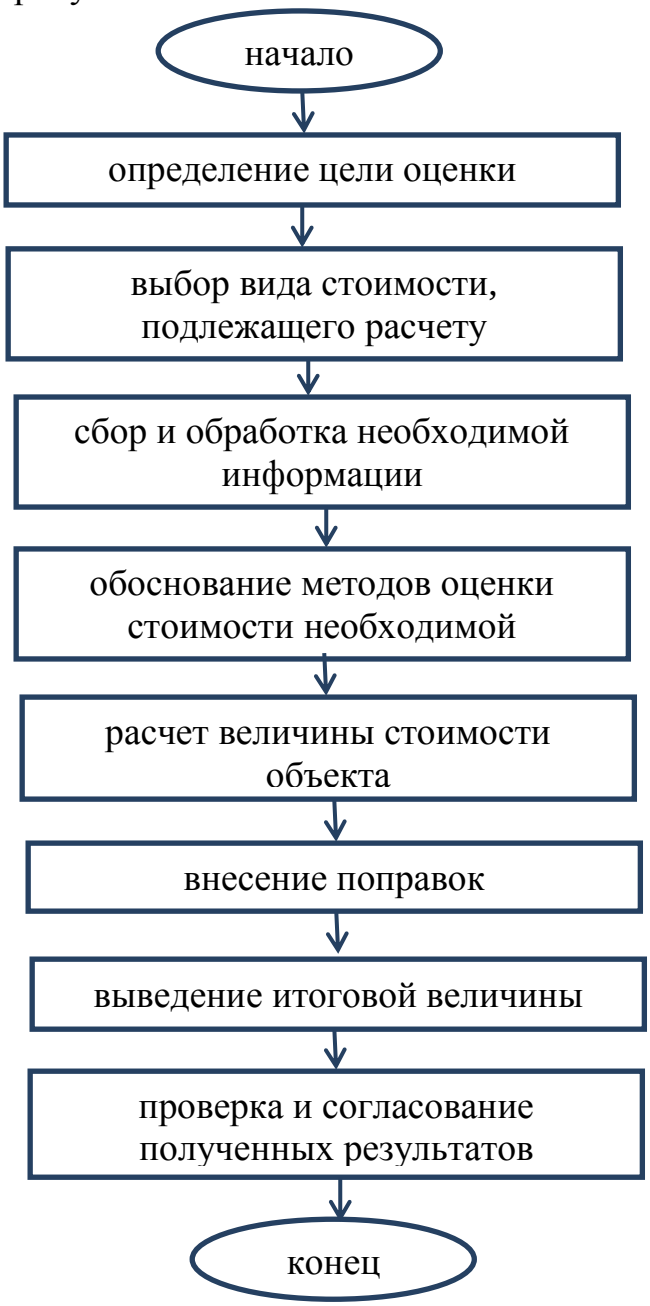


Рисунок 1.1 – Этапы оценки стоимости реализации АПК

В эпоху быстро развивающихся информационных технологий, растущего числа высокобюджетных проектов в области разработки программного обеспечения, очень важным становится умение оценить на ранних этапах возможные выгоды и убытки от проекта, проанализировать возможные сценарии развития событий. Ошибка, недооценка сложностей, с которыми предстоит столкнуться в процессе разработки, переоценка сил команды разработчиков, непринятие в расчет тех или иных фактором часто приводит к финансовым потерям и даже банкротству ИТ-компаний. По статистике только четверть всех начатых проектов завершается своевременно и в полном объеме, четверть вообще отменяется, и половина проектов завершается с превышением бюджетных затрат или с опозданием.[5, 21, 44]

Согласно ежегодно публикуемому обзору группы *The Standish Group* в среднем превышение стоимости проектов от первоначальной сметы составляет 178% для крупных компаний, 182% для средних компаний и 214% для небольших компаний.[2]

У части завершенных с превышением сроков и стоимости проектов частично урезан функционал, который был указан в техническом задании. Это приводит к дополнительным убыткам компании разработчика, если такие ситуации не были предусмотрены в контракте с заказчиком.

С целью минимизации рисков при разработке сложных программных систем, которые, как правило, входят в состав корпоративных информационных систем, необходимо снизить зависимость качества результатов от таких субъективных факторов, как квалификация исполнителей, их опыт, организация процесса разработки. Для этого требуются промышленные технологические методы оценки ресурсов, требуемых для разработки программного обеспечения.[3, 17, 35, 57]

Анализ причин возможных ошибок оценки при планировании IT-проектов показал, что наиболее значимыми и частыми причинами ошибок оценки являются:

– неопределённость на ранних стадиях проектирования;

– несовершенство организации процесса разработки;

– неполные технические требования;

– оптимистическая и субъективная оценка трудоемкости задач.

Поэтому менеджер проекта в компании разработчика нуждается в инструменте, позволяющем как можно точнее оценить трудозатраты и стоимость проекта, а при большом количестве проектов данный процесс должен быть автоматизирован.

Получить точные оценки путем несложных расчетов позволяет сделать алгоритм модели оценки стоимости. Модель оценки стоимости используется в следующих случаях:

– принятие решения о вложении инвестиции в проект или иные финансовые решения, касающиеся разработки программного обеспечения;

– контроль над бюджетом и расписанием проекта (количество людей для реализации определенного этапа разработка, количество финансовых и трудовых затрат на разработку);

– решение о снижении затрат, времени, функциональности, качества продукта при ограничении ресурсов на разработку;

– распределение задач по исполнителям или решение о покупке готовых компонентов разрабатываемой системе.

Существует множество различных моделей оценки стоимости. При выборе для конкретной компании наиболее адекватной модели, необходимо определить критерии, по которым можно оценить предлагаемые модели.

1.2 Модели оценки стоимости программного обеспечения

Одним из первых исследователей в области оценки стоимости программ был Нельсон, который в 1965-1966 годах провел статистический анализ затрат и факторов, характеризующих процессы разработки программ Он сделал расчеты по оценке трудозатрат и опубликовал результаты в «Настольной книге по управлению оценкой стоимости затрат на

программирование». В конце 60-х - начале 70-х стали появляться и другие работы, первые модели оценки - Delphi, Wolverton, разработана концепция функциональных точек, которая позволила оценивать сложность и предполагаемый размер программного продукта на ранних стадиях разработки. В 1977 году вышел первый коммерческий программный продукт по оценке стоимости, получивший название PRICE-S. Вслед за ним появились система SLIM, SEER, Estimacs и ряд других.

В 1981 году Барри Боэм создал принципиально новую модель COCOMO, которая в дальнейшем стала одной из наиболее надежных и полно документированных моделей, с полностью открытыми алгоритмами подсчетов. Исследователи продолжили активно работать, совершенствуя уже существующие модели и создавая программные продукты.

В итоге сложились целые направления оценки стоимости программного обеспечения, которые представлены в таблице 1.1.

Таблица 1.1 – Основные направления оценки стоимости ПО

№	Направление	Описание
1	Модельные методы	основанные на математической модели, с помощью которой в соответствии с входными параметрами вычисляется стоимость проекта
2	Экспертные методы	опираются на точку зрения экспертов, имеющих значительный опыт в разработке программного обеспечения в заданной сфере
3	Нейронные сети	Методы оценки с возможностью обучения по аналогии с помощью техники искусственного интеллекта
4	Динамические методы	используют динамику изменения таких показателей как затраты, навыки, усилия
5	Регрессионные методы	используются в сочетании с методами оценки, основанными на моделях, включают различные виды регрессии
6	Композитные методы	являются комбинацией двух и более методов для формирования наиболее подходящей системы оценки

Рассмотрим модельные методы оценки стоимости ПО: неалгоритмические методы (*Price-to-win*, оценка по Паркинсону, экспертная оценка, оценка по аналогии) и алгоритмические модели (*SLIM* и *COCOMO*).

Сущность неалгоритмических методов состоит в том, что при оценке стоимости ПО используются определенные схемы и принципы, а не математические формулы. Рассмотрим некоторые из них.

Метод *Price-to-win* состоит в том, что независимо от предполагаемых реальных затрат на разработку проекта, оценка стоимости ПО корректируется в соответствии с пожеланиями заказчика. *Price-to-win* фактически является политикой проведения переговоров с клиентом, поэтому часто применяется компаниями, не имеющими средств для качественной оценки проектов. Применение метода может привести к нехватке ресурсов для выполнения проекта, невыполнению сроков сдачи проекта и как результат – потеря контракта или банкротство компании разработчика [22, 33, 51, 63].

Метод оценки по Паркинсону описывает природу взаимодействия бюрократической системы в административных институтах, отображая процесс неэффективного использования ресурсов. При использовании закона Паркинсона в разработке программного обеспечения для того чтобы повысить производительность труда разработчика, уменьшают время, отведённое на разработку.

Метод экспертной оценки применяется в проектах использующих новые технологии, новые процессы или решающих инновационные задачи. К процессу оценки привлекаются инженеры-разработчики, которые сами оценивают курируемую ими часть проекта. Результаты отдельных оценок интегрируются в единую, целостную систему.

Предположения, на которых основывалась оценка отдельных экспертов, заносятся в протокол и открыто обсуждаются. При опросе экспертов используются Дельфийская или расширенная Дельфийская методика, ориентированная на приведение экспертов к консенсусу. В результате

достигается баланс оценки при интеграции отдельных компонентов в общую систему. Далее следует очередная стадия покомпонентной оценки, и по мере увеличения количества итераций точность оценки увеличивается.

Оценка по аналогии является разновидностью экспертной оценки и часто выделяется в отдельный метод. Оценка по аналогии, как и алгоритмические модели, использует эмпирические данные о характеристиках завершенных проектов. Ключевое различие состоит в том, что алгоритмические модели используют эти данные косвенным образом, например, для калибровки параметров моделей, а метод оценки по аналогии с помощью эмпирических данных позволяет отобрать схожие проекты. На первом этапе осуществляется сбор данных по разрабатываемому проекту. В рамках жизненного цикла программного обеспечения оптимальными формами для этого являются анализ требований и проектирование. На основе экспертной оценки производится отбор характеристик, по которым будут сравниваться проекты. Выбор характеристик зависит от типа приложения, среды разработки и набора известных параметров приложения. Следующий этап включает в себя поиск и анализ проектов, аналогичных, по выбранным характеристикам, разрабатываемому проекту. Результатом данного этапа является, как правило, несколько проектов имеющих наименьшие различия в численных значениях характеристик оценки.

Алгоритмические модели оценки стоимости ПО представляет собой одну или несколько функций, которые описывают зависимость между характеристиками проекта и затратами на его реализацию. По типу используемых функций модели разделяют на линейные, мультипликативные, степенные, эмпирические и аналитические. Наиболее часто реализуемыми и хорошо документированными моделями являются модель Путнэма *SLIM* (степенная, аналитическая) и модель *COCOMO* (степенная, эмпирическая).[10, 18]

Модель *SLIM*, созданная Лари Путнэмом из компании *Quantitative Software Mangement*, основана на анализе жизненного цикла программного

продукта, распределения персонала проекта во времени. Она поддерживает наиболее популярные методы измерения размера кода, в том числе и функциональные точки. Это позволяет использовать кривую для оценки графика проекта и его дефектов. Для изменения формы кривой используется коэффициент наращивания мощностей *MBI* (*Manpower Buildup*) и технологический коэффициент фактора продуктивности *PF* (*Productivity factor*). В программной реализации *SLIM* можно сохранять и анализировать информацию, полученную от других, завершенных проектов, и затем использовать ее для калибровки модели, или же если такая информация отсутствует, тогда пользователь может ответить на ряд наводящих вопросов, чтобы рассчитать значения *MBI* и *PF* на основе существующей базы данных.

В *SLIM* продуктивность используется, чтобы связать простую модель распределения мощностей Рейлайха с характеристиками размера и технологии, используемыми в проекте.

Продуктивность рассчитывается по формуле 1.1:

$$P = \frac{S}{E}, \tag{1.1}$$

где P – продуктивность в единицах измерения программного кода на человеко месяц;

S – размер программного продукта рассчитанный по одной из методик подсчета размера кода (функциональные точки, количество строк);

E – объем усилий, направленных на разработку, оцениваемый в человека месяцах.

Кривая Рейлайха, которая используется для определения распределения усилий, моделируется выражением 1.2:

$$m(t) = 2 * K * a * t * e^{-at^2}, \tag{1.2}$$

где $m(t)$ – коэффициент потребности в персонале в момент времени t;

K – общие трудозатраты проекта в человека годах;

a – фактор ускорения.

Фактор ускорения вычисляется по формуле 1.3:

$$a = \frac{1}{2t_d^2}, \tag{1.3}$$

где t_d – время разработки.

Делается предположение, что пиковый уровень персонала в кривой рейлайха соответствует времени разработки t_d. Далее строится линия основного тренда и определяется, попадает ли проект в область досягаемости. Если это так, то проект можно выполнить, в противном случае это не возможно.

Для использования модели в программных продуктах было выведено уравнение 1.4:

$$S = \frac{C}{K^{1/3} + t_d^{4/3}}, \tag{1.4}$$

где S – количество строк в программном обеспечении;

C – фактор среды, зависящий от технологии;

K – общие трудозатраты для всего проекта в чел.-годах;

t_d – ограничения времени разработки программного продукта согласно графику.

Фактор среды вычисляется по формуле 1.5:

$$C = \frac{S}{K^{1/3} + t_d^{4/3}}, \tag{1.5}$$

где коэффициенты K и t_d определяются на базе данных, полученных от старых проектов с размером S. Значение C калибруется и может применяться для последующих оценок.

Модель *COCOMO,* созданная Барри Боэмом, первоначально основывалась на результатах анализа 63 программных преоктов различных типов. При этом оценивался фактический размер кода, трудозатраты, а также фактическая длительность разработки.

В модели используются 3 режима, с помощью которых различается сложность системы и среды разработки – органический режим, сбокированный режим и внедренный режим. Органический режим применим к небольшим программным продуктам, сблокированный к проектам средней величины, требующим небольших инноваций. Внедренный режим характеризуется высокой сложностью, большим объемом инноваций и большой командой разработчиков.

Точность оценки зависит от уровня детализации, который может быть базовым (для выполнения оценки используется только значение размера кода и сведения о текущем режиме), промежуточным (добавляются значения 15 дополнительных переменных, которые получают оценку по 6-бальной шкале) и детализированным (добавляются дополнительные переменные).

Для определения значения дополнительных переменных (драйверов затрат) используются специальные рейтинговые таблицы, содержащие значения варьирующиеся от условий разработки. Для выбора нужного значения менеджеру проекта нужно лишь выбрать те условия, которые, на его взгляд, наиболее сопоставимы с реальной ситуацией. Базовая формула оценки трудозатрат по модели *COCOMO* выглядит следующим образом 1.6:

$$E = a * S^b, \tag{1.6}$$

где a и b – константы, определяемые на этапе регрессионного анализа (в зависимости от проекта);

E – трудозатраты в чел.-мес; S – размер кода в тыс. строк.

Модель включает в себя этапы разработки, кодирования и тестирования, остальные этапы исключаются. Возможно выполнение калибровки модели для конкретной компании, где на основе нескольких проектов проверяют точность коэффициентов и, если требуется, корректируют их.[21] У модели есть ряд недостатков, одним из которых является ее основанность на тысячах условных строк кода, т.е. на размере самого кода программного комплекса.

Композиционные модели призваны объединить преимущества двух-трех методов оценки в единое целое, и тем самым увеличить точность расчетов.

С течением времени и ростом требованиям к системам, модель *COCOMO* оказалась устаревшей в значительной своей части.

Непосредственно по этой причине и ряд других немаловажных проблем была разработана модель *COCOMO II*, впервые опубликованная в 1999 году.[1]

Преимуществом данной модели в том, что она позволяет исследователю использовать фактическую информацию и экспертную оценку.

Модель COCOMO II для расчета стоимости вместо оценки размера программного продукта по количеству строк кода использует объектные указатели - количество экранных форм, отчетов, модулей. Каждый объектный указатель соответствует определенному уровню сложности: простой, средний и сложный.[1]

В рамках этой модели оценки трудоемкости проекта и времени, требующегося на его выполнение, определяются тремя разными способами на разных этапах проекта. На этапе проектирования системы трудоемкость рассчитывается как (1.7):

$$Effort = A * Size, \tag{1.7}$$

где *Size* – оценка размера форм, отчетов, компонентов и модулей будущей системы (каждый элемент оценивается с коэффициентом от 1 до 10 в зависимости от сложности),

A – коэффициент, который учитывает возможное повторное использование части компонентов и производительность разработки, зависящую от опыта команды разработчиков и применяемых инструментов и оценивается числом от 4 до 50.[6]

В таблице 1.2 приведен сравнительный анализ наиболее успешно применяемых моделей *SLIM* и *COCOMO II*.[21]

Таблица 1.2 – Сравнительный анализ моделей

Группа факторов	Факторы	SLIM	COCOMO II
Атрибуты размера	Количество инструкций в коде	+	+
	Функциональные точки	+	+
	Объектно-ориентированные метрики	+	+
Атрибуты программы	Тип	+	-
	Сложность	+	+
	Язык	+	+
	Повторное использование	+	+
	Требуемая надежность	-	+
Атрибуты компьютера	Ограничение ресурсов	+	+
	Устойчивость платформы	-	+
Атрибуты персонала	Возможности персонала	+	+
	Текучесть кадров	-	+
	Опыт персонала	+	+
Атрибуты проекта	Инструментарий и техники	+	+
	Распределенность	+	+
	Ограничение расписания	+	+
	Зрелость проекта	+	+
	Сплоченность команды	+	+
	Запросы безопасности	-	-
	Множественная разработка	-	+
Этапы	Начало работ	+	+
	Уточнение требований	+	+
	Исполнение	+	+
	Сопровождение	+	+

По результатам анализа можно сделать вывод, что преимущество отдается модели *COCOMO II*. Среди ее сильных сторон можно выделить следующие:

– позволят оценить трудоемкость исходя из разных уровней определенности требований;

– легка в использовании;

– поддерживает современный процесс разработки программного обеспечения;

Основным недостатком модели являются необходимость производить калибровку модели, что требует большого количества сведений о предыдущих проектах.

Кроме того, особое место в решении вопроса оценки стоимости занимают критически важные системы управления и обеспечение их надежности.

1.3 Подходы к обеспечению отказоустойчивости информационно-управляющих систем

Для обеспечения отказоустойчивой работы корпоративной информационной системы в целом, необходимо обеспечить достаточный уровень отказоустойчивости составляющих ее узлов. Таким образом, необходимо обеспечить:

– надежность программного комплекса;

– надежность рабочих станций;

– надежность периферийного оборудования;

– непрерывную подачу электропитания и соответствие ее предельно допустимым нормам;

– отказоустойчивую и помехозащищенную структурированную кабельную систему;

– достаточный уровень квалификации персонала использующего АПК;

– регулярное техническое обслуживание АПК;

– быструю и квалифицированную техническую поддержку.

Обеспечение надежности программного обеспечения информационной системы является наиболее сложной задачей, т.к. оно отличается от аппаратного обеспечения следующим:

– элементы ПО не стареют от износа;

– число способов контроля ПО значительно больше числа способов контроля аппаратуры;

– в ПО значительно больше объектов для контроля, чем в аппаратуре;

– в ПО гораздо проще вносить исправления и дополнения, чем в аппаратуру, но это трудно делать корректно и безошибочно.

Разработка отказоустойчивого программного обеспечения (ПО) – отдельный аспект разработки надежных информационно-управляющих систем (ИУС), так как системная надежность зависит от надежности как аппаратных, так и программных компонент. Как правило, надежностное проектирование ИУС фокусируется на критичных частях аппаратного обеспечения системы. Однако в таких областях как производство, транспорт, финансы, оборона и медицина сбой в работе программного обеспечения может привести к катастрофическим последствиям. Поэтому одной из основных задач при разработке программного обеспечения является создание таких алгоритмов или методов разработки ПО, которые обеспечивали бы устойчивость системы к программным и аппаратным сбоям.

На практике существует два дополняющих друг друга подхода, которые используются при разработке надежного программного обеспечения ИУС [7]:предотвращение сбоев и устойчивость к сбоям.

Для предотвращения сбоев в процессе проектирования и реализации программных систем используются такие технологии разработки ПО, которые сводят к минимуму ошибки оператора и помогают находить системные ошибки до того, как система будет запущена в эксплуатацию. Предотвращение сбоев, фактически, означает поставку заказчику программных систем, свободных от ошибок и сбоев. Это можно сделать двумя способами: с помощью статических и динамических методов тестирования, которые обнаруживают эти ошибки и позволяют исправить их до начала эксплуатации системы. Однако с уменьшением ошибок в системе

стоимость их обнаружения возрастает экспоненциально. Это значит, что при усложнении системы обеспечить достаточный уровень ее надежности только за счет тестирования становится практически невозможным.

Для обеспечения устойчивости к сбоям система проектируется таким образом, чтобы можно было обнаружить и исправить сбои, устраняя непредвиденное поведение системы до того, как это приведет к ее отказу. Устойчивость к сбоям подразумевает наличие в системе возможности исправления ошибок отдельных модулей в процессе их выполнения.

Проведенный анализ подходов к обеспечению отказоустойчивости и обзор моделей оценки стоимости, позволяет сделать следующие выводы:

a) рассмотренные модели оценки стоимости реализации АПК не учитывают трудозатраты компании - разработчика на разных этапах жизненного цикла программного продукта (при проектировании, разработке, тестировании и документировании);
b) большинство существующих моделей при оценке трудозатрат исходят прежде всего из размера программного кода, что не соответствует фактической трудоемкости в условиях современных инструментов проектирования;
c) несмотря на огромный объем ежегодно разрабатываемого ПО, по прежднему проектирование на ранних этапах осуществляется в условиях неопределенности, отсутствия необходимых исходных данных для расчета параметров, что увеличивает вероятность ошибки оценивания стоимости реаализации АПК;
d) следует учитывать потребность в доработке компонентов системы или всей системы, глубину такой доработки, сложность архитектурных решений, так как изменение одного элемента ПО с современной сложной иерархической структурой неизбежно повлечет за собой и другие изменения;
e) проанализированные модели оценки стоимости рассматривают каждую новую доработку как отдельный новый проект;

f) для повышения надежности и отказоустойчивости критически важных систем управления необходимо применять особые подходы к проектированию, которые существенно осложняют оценку стоимости реализации таких ПО и влияют на его цену.

Следовательно необходима разработка методики оценки трудозатрат на модернизацию уже существующего программного продукта при условии повышения надеждости программного обеспечения, которая снизит вероятность ошибки, уменьшит время оценивания и в целом риски по проекту.

2 МЕТОДИКА ОЦЕНКИ ЗАТРАТ НА МОДЕРНИЗАЦИЮ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ОТКАЗОУСТОЙЧИВЫХ СИСТЕМ УПРАВЛЕНИЯ

2.1 Мультиверсионное проектирование как инструмент обеспечения устойчивости АПК к программным сбоям

Одной из наиболее перспективных и уже положительно зарекомендовавших себя методологий обеспечения высокой надежности и отказоустойчивости программного обеспечения является мультиверсионное проектирование. Согласно данной методологии предполагается, что возникновение сбоя в функционально эквивалентных модулях на одних и тех же входных данных происходит в различных точках исполнения. Таким образом, вводится программная избыточность, отличающаяся от аппаратной тем, что простое копирование программных компонентов приводит к копированию ошибок, т.к. программные сбои в отличие от аппаратных зависят от внутренних дефектов, а не внешних факторов.

Создание функционально-эквивалентных, но, тем не менее, разных модулей может быть достигнуто с помощью разнообразия при разработке версий одного модуля. Разнообразие применяют для разработки компонентов, к которым происходит наиболее частое обращение, или результаты работы которого участвуют в критических циклах управления. Разнообразие может быть в следующем:

– образование и опыт разработчиков;

– алгоритмы и структуры данных;

– языки программирования;

– инструментальные средства программирования и среды (включая компиляторы);

– методы тестирования.

Существует два основных подхода к реализации мультиверсионности программного модуля. Первый подход называют *NVP* (*N-version programming* – мультиверсионное программирование) предложен ученым Авижиенисом в 1985 г. При этом подходе версии выполняются параллельно (рисунок 2.1). Результат их работы определяется с помощью какого-либо алгоритма голосования.

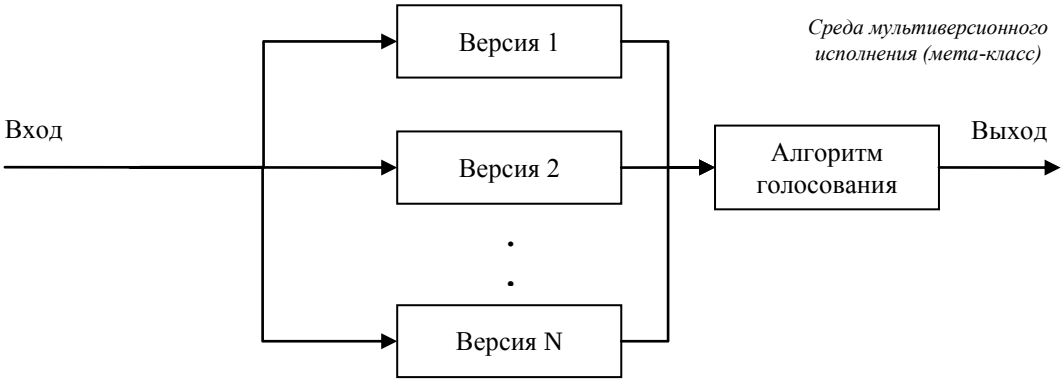


Рисунок 2.1 – Схема работы при *NVP*

Алгоритмы голосования могут быть разными в зависимости от задачи. Обычно используется выбор результата по абсолютному большинству (эквивалентных выходов больше половины) или по согласованному большинству (самая большая группа эквивалентных выходов). При этом выходные значения являются идентичными при заданной погрешности.

Второй подход называют *RB* (*Recovery Block* – блок восстановления) предложен Брайеном Ренделом в 1975 г. В этом случае каждый критичный программный компонент содержит множество версий вычислительного модуля; тест, проверяющий его работу; и подпрограмму, которая по результатам выполнения теста либо принимает результаты вычисления, либо запускает их повторно, но уже с помощью другой версии вычислительного модуля (рисунок 2.2).

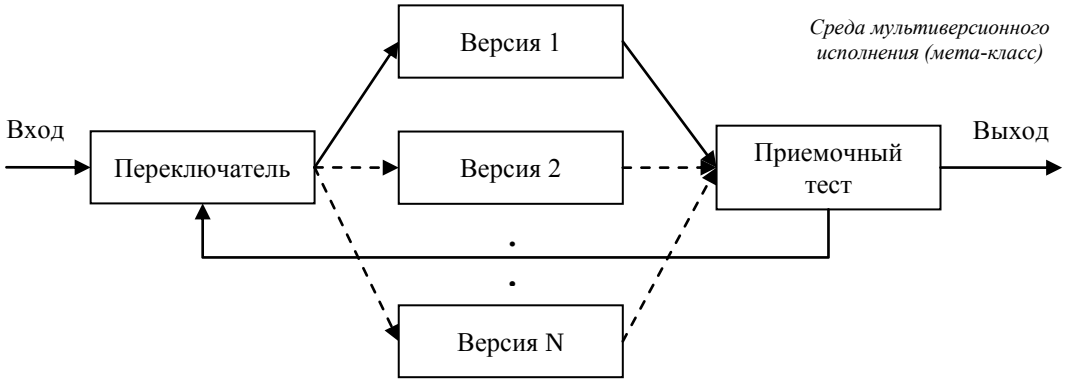


Рисунок 2.2 – Схема работы при *RB*

Также применяются гибридные методы[8]:

1. CRB – согласованный блок восстановления (*consensus RB*). Предложенный ученым Китом Скоттом в 1987 г. Если происходит сбой в алгоритме голосования или алгоритм голосования не может выбрать результат, то модуль продолжает работу как Блок восстановления.

2. *NSCP* (*N-self checking programming*) или *NVP* с самоконтролем – предложен Жан-Клодом Лапри в 1987 г. Приемочный тест играет роль фильтра. Используется соответствующий алгоритм голосования учитывающий, что в некоторых версиях результат может быть не приемлемым.

Надежность отказоустойчивого модуля зависит от глубины программной избыточности (количества версий), из чего вытекают следующие недостатки применения данного подхода:

- увеличение затрат на разработку каждой версии, реализация алгоритмов голосования или приемочного теста, среды исполнения версий, также увеличивается время тестирования и необходима проработка спецификаций для каждой версии модуля;
- сложность реализации независимой работы версий из-за обращения к одной базе данных и другим модулям;

— увеличение времени выполнения модуля;
— увеличение требований к производительности для параллельного выполнения нескольких версий.

Но, тем не менее, данный подход имеет явные достоинства:

— снижение вероятности сбоя модуля, выполняющего важную функцию в автоматизированной системе;
— возможность горячей замены версии, при продолжении работы всей системы, если каждая версия реализована отдельным файлом;
— есть возможность анализировать сбой в отказавшей версии, пока система продолжает функционировать. Уже при плановом обновлении системы, версия может быть заменена.

Мультиверсионный подход, и в частности *NVP* доказал свою эффективность при разработке программного обеспечения для космических спутников (космический шаттл *NASA*), в автоматизированной системе управления дорожным движением (*COMTRAC*), системах сигнализации на железных дорогах, в системах контроля работы за ядерным реактором, а также в системе управления полетом боинга 777 и системах управления американскими водохранилищами.[11]

Подход *RB* используется в бортовых системах самолетов *Airbus* A320/A330/A340. Например, при эксплуатации одной из военных систем в США было зафиксировано 222 программных сбоя, из которых 165 были замаскированы, и не проявились как сбой системы.[8]

Таким образом, применение мультиверсионного подхода влечет за собой дополнительные трудовые затраты, что существенно может сказаться на стоимости реализации аппаратно-программного комплекса в целом, но может быть вполне оправдано для разработки наиболее критичных программных компонентов отказоустойчивых систем управления высокой степени надежности.

2.2 Алгоритм применения методики оценки затрат на аппаратно-программный комплекс

При разработке высокобюджетных систем управления большую роль играет оценка затрат на аппаратное и программное обеспечение.

Расхождение планируемых затрат с фактическими может привести к серьезным финансовым потерям и даже к банкротству компании разработчика. Существующие модели и, основанные на них, методики оценки будущей стоимости проекта в совокупности имеют недостатки.

В существующих моделях не учитываются затраты на повышение надежности наиболее важных компонентов путем введения программной избыточности. Снижение вероятности сбоя получают за счет одновременной работы нескольких версий одного и того же компонента, реализованных разными командами разработчиков или с использованием разных языков программирования. Таким образом снижается вероятность того, что на одних и тех же входных данных откажу все версии компонента. Для одновременной работы нескольких версий необходима разработка среды их исполнения. В среду исполнения входит алгоритм голосования, который выбирает наиболее приемлемые выходные данные от всех версий.[13] Введение программной избыточности необходимо для обеспечения безопасности функционирования критически важных информационных систем, связанных с управлением технологическими процессами или финансовыми расчетами.

Следовательно, необходима разработка методики оценки трудозатрат на модернизацию уже существующего программного продукта, которая поможет менеджерам проектов уменьшить ошибку оценки и снизить финансовые риски компании.

При расчете размера оцениваемой системы будут учитываться объектные указатели (компоненты) разрабатываемой системы аналогично модели *COMOMO* II.

Рассмотрим основные этапы методики оценки затрат на модернизацию программного обеспечения критических по надежности систем:

1. **Сбор и анализ технических требований заказчика.** Заказчик передает свои задокументированные требования на доработку системы компании разработчика для анализа, или аналитики выявляют требования заказчика в ходе интервью. Цель этой работы – определить основные требования бизнеса (исходные данные, истинные цели, которым должен служить продукт и проблемы, которые нужно решить с помощью разрабатываемого продукта). После проведенного анализа формируется проект технического задания на модернизацию системы.

2. **Декомпозиция задач, дизайн архитектуры.** На этапе проектирования архитектуры системы, задачи модернизации декомпозируются на множество простых. Затем задачи сопоставляются с доработкой уже существующих компонентов системы или разработкой новых. Системный архитектор определяет общую структуру каждого архитектурного представления, декомпозицию представлений и интерфейсы взаимодействия элементов. Таким образом, происходит разбиение большой системы на более мелкие части (модули), в соответствии с определенным уровнем абстракции. Поэтому архитектурный компонент может быть определен по-разному в зависимости от архитектурного подхода и степени подробности описания архитектуры.

3. **Группировка компонентов.** Компоненты, которые необходимо разработать или модернизировать, группируются на типы по аналогичному назначению и сложности.

4. **Определение избыточности программных компонентов.** Модульность построения приводит к использованию иерархической структуры взаимодействия модулей программы. Иерархическая схема, отражая функции модулей, одновременно показывает структуру связей между ними. Иерархические структуры системы характеризуются, с одной стороны, вертикальным управлением, когда модули верхнего уровня имеют

право вмешательства и координирования работы модулей нижнего уровня. С другой стороны, действия модулей верхнего уровня зависят от информации, полученной в результате функционирования модулей нижних иерархических уровней. Таким образом, сверху вниз идут в основном управляющие воздействия, а снизу вверх – информация о соответствующих решениях и переменных. Число архитектурных уровней в модели архитектуры ПО зависит от частного проекта системы.

Известно, что чем позже будет обнаружена ошибка проектирования, тем дороже обойдется ее исправление разработчику программного продукта. Поэтому для соблюдения требований к надежности разрабатываемой программной системы необходимо уже на стадии дизайна архитектуры тщательно прорабатывать связи между компонентами и устанавливать иерархию их взаимодействия. Компоненты, которые наиболее часто используются или архитектурно связанные с множеством других компонентов, оказывают наибольшее влияние на надежность системы. Зависимости компонентов позволяют неисправности распространяться из компонента, в котором она происходит, к другим компонентам. Обнаружение отказов во время разработки зависит от тщательности этапа тестирования. Также для уменьшения вероятности сбоя в наиболее важных компонентах определяется количество версий, которые будут реализованы разными командами разработчиков, или с использованием различных технологий. Также проводится оценка трудозатрат на разработку алгоритма голосования для данного компонента.

5. **Оценка трудозатрат на разработку компонентов.** Вычисляется среднее время разработки компонента каждого типа, на основании предыдущих работ или с помощью экспертной оценки. Если тип компонентов ранее не разрабатывался, то время определяется экспертно, исходя из анализа подобных типов компонентов.

6. **Определение глубины модернизации компонентов.** Определяется глубина доработки компонентов (весовые коэффициенты) в процентах.

Глубина модернизации может быть рассчитана из отношения размера кода, который предполагается изменить, к текущему размеру кода компонента. Если компонент разрабатывается вновь, то глубина модернизации равна единице.

7. **Определение количества процессоров.** Определяется количество типов процессоров и количество процессов каждого типа.

8. **Вычисление трудоемкости разработки программного обеспечения.** Суммируется трудоемкость разработки компонентов каждого типа. Если компонент подлежит модернизации, то трудоемкость разработки этого компонента умножается на весовой коэффициент. Если компонент реализуется с применением программной избыточности, то суммируются затраты на разработку всех его версий и среды их исполнения (мета-класса среды исполнения и алгоритма голосования или приемочного теста).

9. **Определение затрат на этапы жизненного цикла программного обеспечения.** На основании предыдущих работ вычисляется соотношение среднего времени, затраченного на другие этапы жизненного цикла к среднему времени разработки. Вычисленные коэффициенты (соотношения) умножаются на трудоемкость этапа разработки. Таким образом, получается трудоемкость этапов анализа или проектирования, разработки, тестирования и работ по организации процесса.

10. **Определение затрат на покупку компонентов аппаратного обеспечения.** Определяется стоимость процессора каждого типа и стоимость других переферийных устройств, обеспечивающих функционирование программного обеспечения с приемлемым уровнем производительности.

11. **Вычисление финансовых затрат на реализацию АПК.** Трудоемкость каждого этапа умножается на среднюю норму оплаты труда сотрудника, работающего на данном этапе жизненного цикла. Затем полученные суммы складываются с затратами на компоненты аппаратной части.

В результате проведения всех этапов методики, на основе изученных и систематизированных требований, аналитик вместе с командой экспертов получает концепцию и границы будущего проекта, которые должны содержать ориентировочные сроки и бюджет.

На рисунке 2.3 представлена блок-схема алгоритма применения методики в жизненном цикле программного обеспечения.

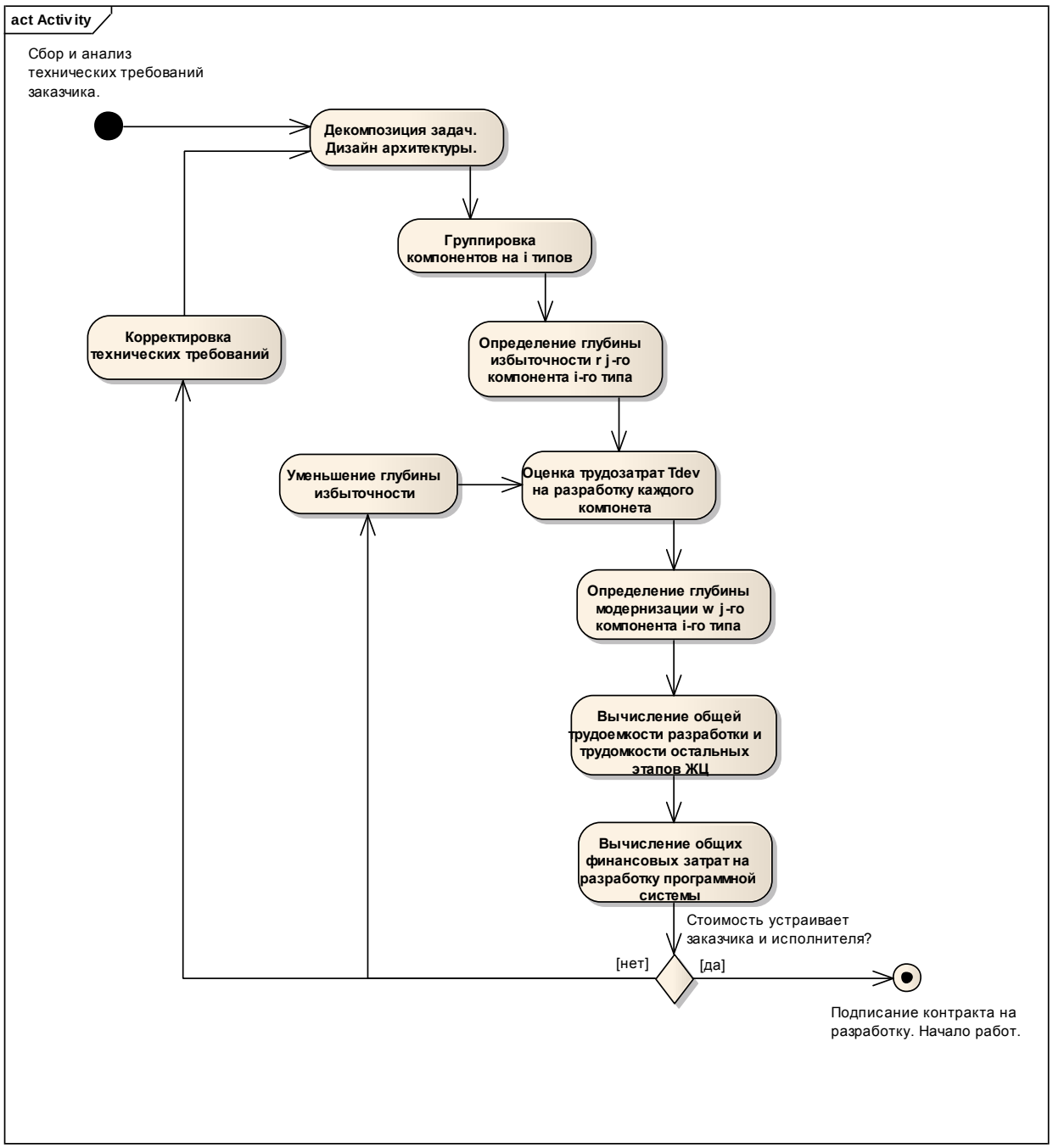


Рисунок 2.3 – Алгоритм применения методики

Сначала происходит сбор и анализ технических требований заказчика. Затем декомпозиция задач, группировка компонентов, определение глубины избыточности, оценка затрат трудовых и финансовых ресурсов.

Предполагаемая стоимость работ обсуждается с заказчиком и, если его и компанию разработчика стоимость устраивает, то подписывается контракт на разработку. Если же заказчик считает, что стоимость завышена, то компания разработчик может вернуться к этапу определения глубины избыточности и уменьшить ее, что может снизить затраты. Также можно откорректировать технические требования заказчика и выбрать альтернативное архитектурное решение, затраты на реализацию которого будут меньше.

2.3 Параметры для расчетов затрат

Для проведения расчетов по приведенной методике используются следующие параметры:

M – множество типов программных компонентов;

i – тип компонента, $i \in M$;

N – множество новых и подлежащих доработке компонентов i-го типа;

j – номер компонента i-го типа, $j \in N$;

T_i – трудоемкость разработки нового или уже имеющегося компонента i-го типа, чел.-час;

w_{ij} – весовой коэффициент модернизации существующего j-го компонента i-го типа (для новых компонентов равен $w_{ij} = 1$, если нет j компонента i-го типа, то $w_{ij} = 0$);

r_{ij} – количество версий в компоненте, если вводится программная избыточность (если программная избыточность не вводится, то $r_{ij} = 1$);

TR_{ij} – трудоемкость разработки алгоритма голосования для j-го компонента i-го типа (если программная избыточность не вводится, то $TR_{ij} = 0$);

TM – общие трудозатраты на управление и организацию проекта, чел.-час;

WM – весовой коэффициент, определяющий процент трудоемкости организации, от трудоемкости разработки;

CM – стоимость оплаты одного чел.-часа менеджера проекта, руб;

TA – общая трудоемкость работы аналитиков и проектировщиков, чел.-час;

WA– весовой коэффициент, определяющий процент трудоемкости анализа и проектирования от трудоемкости разработки, чел.-час;

CA – стоимость одного чел.-часа аналитика или проектировщика, руб;

TDEV – общая трудоемкость этапа разработки, чел.-час;

CDEV – стоимость оплаты одного чел.-часа разработчика, руб;

TT – общая трудоемкость этапа тестирования;

WT – весовой коэффициент, определяющий процент трудоемкости анализа, от трудоемкости разработки;

CT – стоимость оплаты одного чел.-часа тестировщика, руб;

TD – общая трудоемкость документирования доработок системы, чел.-час;

WD – весовой коэффициент, определяющий процент трудоемкости документирования, от трудоемкости разработки;

CT – стоимость оплаты одного чел.-часа специалиста по технической документации (технического писателя), руб;

TP – общая трудоемкость разработки ПО, чел.-час;

PTYPE – множество типов процессоров, шт;

ptype – номер типа процессора, $ptype \in PTYPE$;

P_{ptype} – количество процессоров типа *ptype*, шт;

p – номер процессора типа *ptype*, $p \in P$;

CPR_{ptype} – стоимость процессора типа *ptype*, руб.;

CS – стоимость других периферийных устройств, руб.

CH – стоимость аппаратной части, руб.

CP – общая стоимость проекта, руб.

Трудоемкость разработки ПО рассчитывается по матрице w_{ij}, и с учетом введения программной избыточности, следующим образом (2.1):

$$TDEV = \sum_{i=1}^{M}\sum_{j=1}^{N}\left(w_{ij}T_i + \left(r_{ij} - 1\right) * T_i + TR_{ij}\right) \tag{2.1}$$

Трудоемкости остальных этапов жизненного цикла рассчитываются по соответствующим весовым коэффициентам из трудоемкости разработки:

$TM = WM * TDEV$, $TA = WA * TDEV$, $TT = WT * TDEV$, $TD = DW * TDEV$

Общая трудоемкость рассчитывается по формуле 2.2

$$TP = \left(1 + WM + WA + WT + WD\right) * TDEV \tag{2.2}$$

Стоимость аппаратного обеспечения рассчитывается по формуле 2.3

$$CH = \sum_{ptype=1}^{PTYPE}\left(P_{ptype} * CPR_{ptype}\right) + CH \tag{2.3}$$

Формула расчета общей стоимости в детализированном виде выглядит следующим образом (2.4):

$$CP = \left(WM * CM + WA * CA + CDEV + WT * CT + WD * CD\right) * \\ * \sum_{i=1}^{M}\sum_{j=1}^{N}\left(w_{ij} * T_i + \left(r_{ij} - 1\right) * T_i + TR_{ij}\right) + \sum_{ptype=1}^{PTYPE}\left(P_{ptype} \cdot CPR_{ptype}\right) + CH \tag{2.4}$$

$$CP = \left(WM * CM + WA * CA + CDEV + WT * CT + WD * CD\right) * TDEV + CH \tag{2.5}$$

Предлагаемая модель и алгоритм оценки стоимости позволит рассчитывать затраты на разработку и модернизацию аппаратно-программного комплекса с учетом обеспечения критически важного уровня надежности различных систем управления. Однако, большой объем параметров необходимых для расчета и сложность вычисления требуют автоматизации данного процесса, путем разработки специального программного продукта для менеджеров разрабатываемых программных систем. Поэтому в работе предложена программная система, с помощью которой можно рассчитывать затраты на проект.

3 ПРОЕКТИРОВАНИЕ СРЕДЫ ДЛЯ РАСЧЕТОВ ЗАТРАТ НА МОДЕРНИЗАЦИЮ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В ходе исследования были рассмотрены наиболее крупные представители систем по оценке стоимости программных проектов: SLIM Tools, KnovvledgePlan и Charismatek's Function Point WORKBENCH™ (FPW), Cost XPert, Borland CaliberRM, На основании анализа их функциональных возможностей автором были выработаны требования к программному продукту по оценке стоимости, исходя из потребностей современных ИТ-предприятий. Согласно этим требованиям, продукт должен обеспечивать:

- ввод, хранение с полной историей изменений и возможностью повторного использования информации по проектам и оценкам проектов;
- возможность гибкого, наглядного предоставления выходной информации по результатам оценок и сравнительного анализа;
- возможность прогнозирования, анализа «что если»;
- учет в оценке разных языков программирования, методов оценки величины программного кода, разных моделей жизненного цикла;
- возможность подстройки системы под нужды конкретного предприятия;
- наличие системы разграничения доступов пользователей по проектам оценки и операциям;
- возможность совместной работы группы пользователей над одним проектом в рамках системы;
- возможность двухстороннего взаимодействия с системой управления проектами.

3.1 Техническое задание на разработку

Разработанная программная система имеет наименование «*Calculation of workspace*» и предназначена для автоматизации расчета затрат на

модернизацию программного обеспечения критически важных информационных систем.

Чтобы представить общую функциональность и поведение необходимой для автоматизации системы, а также определить границы проекта, воспользуемся *UML* (англ. *Unified Modeling Language* – унифицированный язык моделирования) – язык графического описания для объектного моделирования в области разработки программного обеспечения. *UML* является языком широкого профиля, это открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой *UML*-моделью. *UML* был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. *UML* не является языком программирования, но в средствах выполнения *UML*-моделей как интерпретируемого кода возможна кодогенерация.

Использование *UML* не ограничивается моделированием программного обеспечения. Его также используют для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

UML позволяет также разработчикам программного обеспечения достигнуть соглашения в графических обозначениях для представления общих понятий (таких как класс, компонент, обобщение (generalization), объединение (aggregation) и поведение), и больше сконцентрироваться на проектировании и архитектуре.

На начальной стадии проектирования построим диаграмму вариантов использования. Для этого воспользуемся программным средством *Sparx Systems Enterprise Architect v7.1.834 Corporate Edition* – мощным набором *UML*-инструментов для профессионалов, занимающихся разработкой, тестированием и внедрением программного обеспечения. С помощью широкого выбора опций и без лишних расходов, *Enterprise Architect* может

устроить всех специалистов организации: аналитиков, испытателей, проект-менеджеров, персонал проверки качества.

Enterprise Architect объединяет в себе силу языка *UML* 2.1 с высоко-эффективным, понятным интерфейсом. Данная программа дает возможность расширенного моделирования на рабочем столе, разработки и созданию групп.

Corporate Edition включает все особенности *Desktop* и *Professional* версий, с добавленной возможностью использовать SQL Server, MySQL, *Oracle* 9*i* и 10*g*, *PostgreSQL*, *MSDE*, *Adaptive Server*, *MS Access*, *Firebird* как модель хранения. Возможность установки и управления пользовательской защитой для модели. Это издание - для больших групп, которые требуют хорошего управления по пользовательскому доступу (и ограничениям) к элементам и редактирующей части в модели.

Enterprise Architect поддерживает многие популярные языки программирования: *ActionScript* 2.0, *Java*, *C#*, *C++*, *VB.Net*, *Delphi*, *Visual Basic*, *Python* и *PHP*.

Диаграммы вариантов использования описывают функциональное назначение системы или то, что система должна делать. Разработка диаграммы преследует следующие цели:

- определить общие границы и контекст моделируемой предметной области;
- сформулировать общие требования к функциональному поведению проектируемой системы;
- разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
- подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Суть диаграммы вариантов использования состоит в следующем. Проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью вариантов

использования. При этом актером (*actor*) или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик. Вариант использования служит для описания сервисов, которые система предоставляет актеру. Диаграмма вариантов использования может дополняться пояснительным текстом, который раскрывает смысл или семантику составляющих ее компонентов.

Построенная диаграмма вариантов использования приведена на рисунке 3.1.

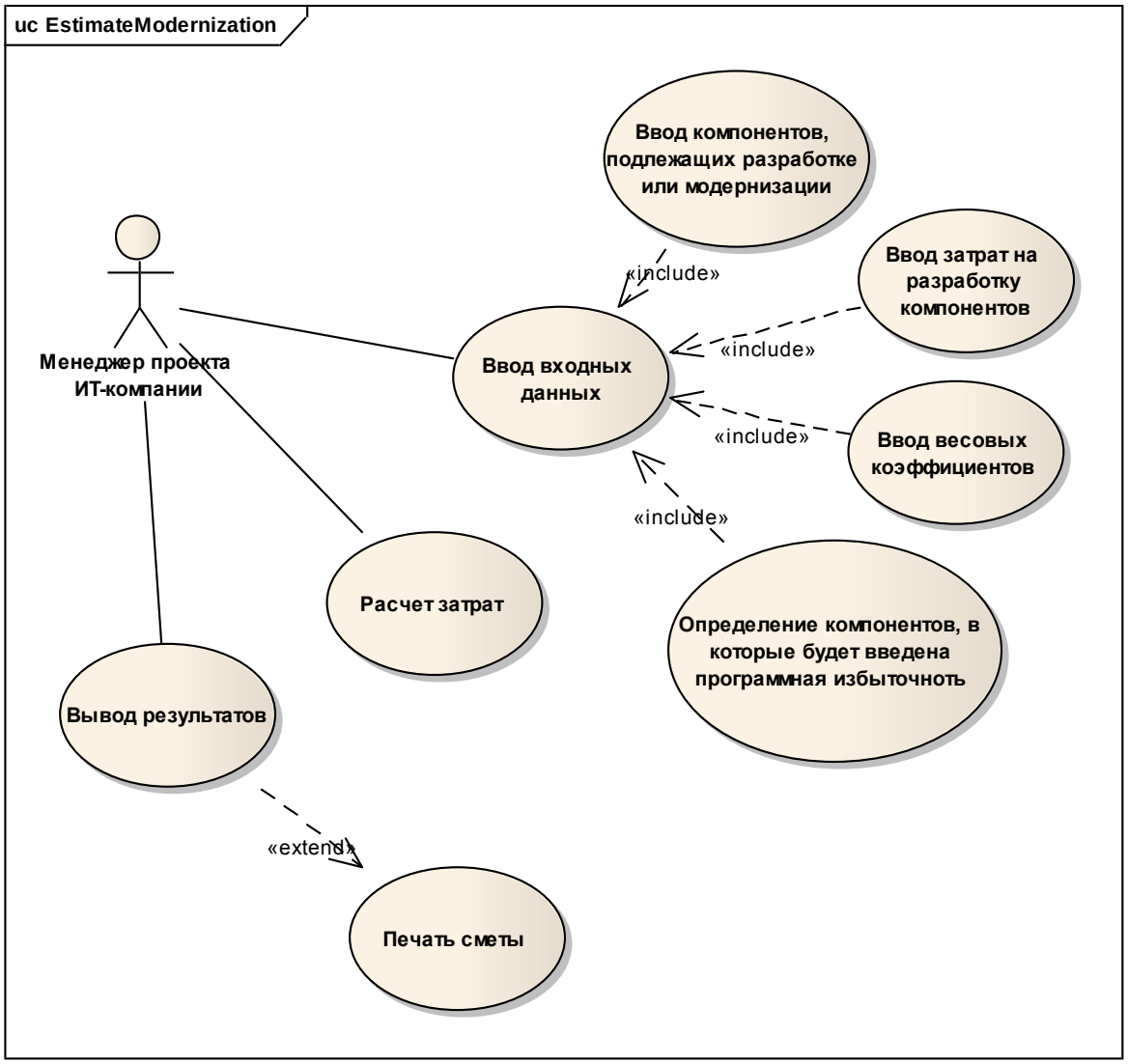


Рисунок 3.1 – Диаграмма вариантов использования программной системы

На диаграмме можно увидеть, что основные функции, которыми должна обладать система, следующие:

– ввод входных данных;
– расчет затрат на выполнение работ;
– вывод результатов и печать сметы.

Далее следует определить требования к надежности, техническому и программному обеспечению.

Программная система должна осуществлять контроль вводимой информации на соответствие типу. Должен быть реализован интуитивно понятный интерфейс. Во время работы с базой данных должна быть обеспечена целостность хранимой информации.

Система должна работать на компьютерах с процессором 1 ГГц и выше и с объемом ОЗУ 1 Гб и более под операционной системой *Windows XP* и выше. Для работы программы необходима установка пакета *Microsoft Excel* 2010 и выше, а также установка драйвера к БД – *Microsoft Office Access Database Engine* 2010.

Должна быть подготовлена следующая документация:

– руководство программиста;
– руководство пользователя.

3.2 Описание программной системы

Первый этап заключается в проектировании базы данных. Для хранения данных необходимо спроектировать их структуру. Проектирование и реализация базы данных будет проведено с помощью пакета *Microsoft Access* 2010.

Основные преимущества данной СУБД: обладает высокой устойчивостью данных, проста в освоении, может использоваться непрофессиональным программистом, позволяет готовить отчеты из баз данных различных форматов. СУБД *Access* предназначена для создания

отчетов произвольной формы на основании различных данных и разработки некоммерческих приложений по обработке локальной информации.

Схема спроектированной БД изображена на рисунке 3.2.

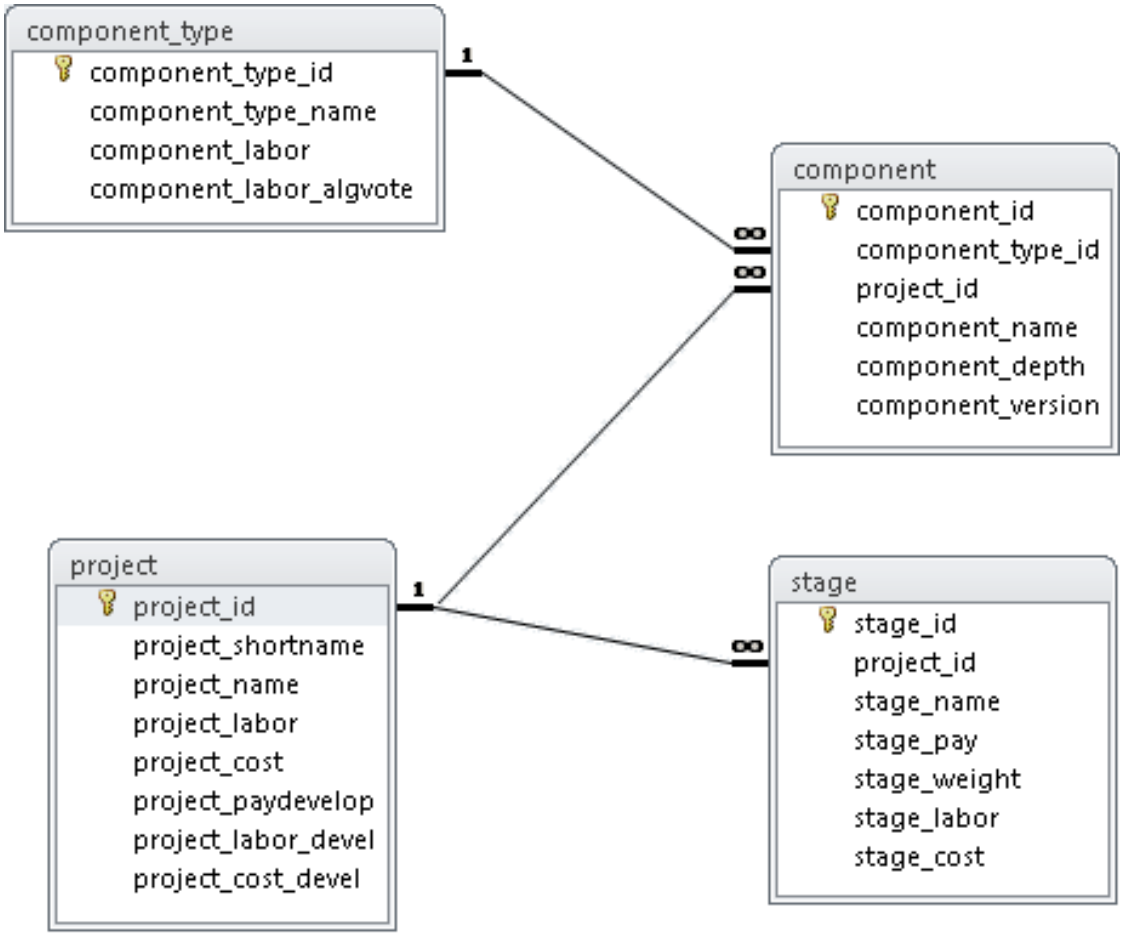


Рисунок 3.2 – Схема БД

Целостность данных и контроль корректности вводимых значение осуществляется с помощью инструментов базы данных.

На втором этапе следует разработать приложения. Необходимо реализовать функции программной системы, согласно определенным функциональным требованиям для реализации расчетов затрат, приведенных в постановке задачи.

Для реализации логики программы использован *C++Builder* из пакета *RAD Studio* 2010 *Architect*.

C++Builder 2010 – средство быстрой разработки приложений для *Windows* на *C++*. *C++Builder* предоставляет разработчику программного обеспечения лучшее, что есть в двух технологиях: мощь языков и библиотек *C* и *C++* в сочетании с продуктивностью быстрой визуальной разработки приложений.

Среда разработки C++*Builder* включает расширенный редактор, отладчик, средства тестирования модулей и моделирования и мощный компилятор с опережающей поддержкой будущих стандартов. В данной среде разработки наряду с библиотеками существует визуальный редактор и сотни других компонентов, которые позволяют быстро создавать пользовательские интерфейсы и разрабатывать приложения для взаимодействия с различными СУБД. *C++Builder* 2010 реализует полную поддержку *Unicode*, благодаря чему приложения могут выполняться на любой языковой версии *Windows*. Применение *Unicode* гарантирует, что приложения будут одинаково выглядеть и функционировать во всех языковых версиях *Windows* и безукоризненно поддерживать как *Unicode*-, так и *ANSI*-строки. Кроме того, новые усовершенствованные средства локализации помогают переводить приложения на различные языки.

C++Builder 2010 обеспечивает эффективный и быстрый доступ ко всем популярным системам управления базами данных. С помощью надёжной и мощной платформы доступа к базам данных разработчики могут легко получать доступ и просматривать данные без написания кода. Создание, чтение, обновление и удаление данных можно легко выполнять с помощью визуальных элементов управления или кода. Многоуровневая архитектура *DataSnap* позволяет создавать в среде быстрой разработки высокопроизводительные, масштабируемые *middleware*-приложения для баз данных. Профессиональное средство моделирования *ER/Studio*, входящее в состав редакции *Architect*, позволяет проектировать схемы баз данных и выводит на новый уровень интеграцию разработки приложений и разработки баз данных. *C++Builder* 2010 обеспечивает полное двустороннее

взаимодействие модели со средствами моделирования *LiveSource*, в результате чего изменения модели отражаются в исходном коде, а изменения исходного кода - в модели *UML*.

Каким образом преимущества *C++Builder* 2010 следущие:

– полная поддержка *Unicode*. Приложения могут выполняться на любой языковой версии *Windows*;

– наличие расширенных компонентов *VCL* для создания современного внешнего вида пользовательского интерфейса;

– поддержка стандартов и библиотек *C++*;

– улучшенная совместимость с *Delphi*;

– возможность проектирования и разработки баз данных благодаря входящему в состав редакции *C++Builder Architect* профессиональномму средству моделирования *Embarcadero – ER/Studio*.

Структура разработанного в *C++Builder* 2010 исполняемого файла приведена на рисунке 3.3.

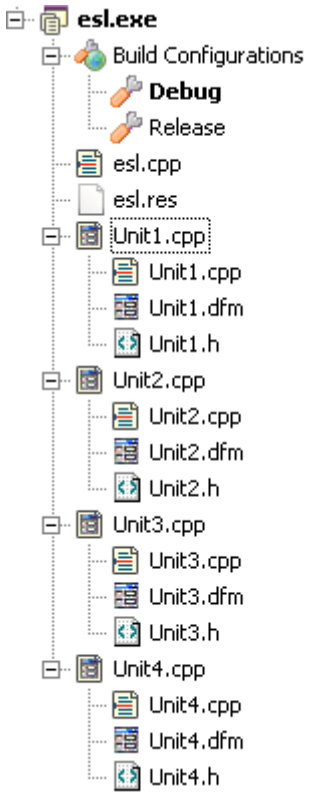


Рисунок 3.3 – Структура исполняемого файла

Разработанный пакет документаций включает в себя руководство программиста. Для работы программы необходима установка драйвера доступа к БД *Access* 2010 – *AccessDatabaseEngine.exe*.

Программная система «Calculation of workspace» состоит из следующих компонентов:

– *cow.exe* (исполняемый файл программы);
– *cow.accdb* (файл БД компонентов);
– *estimate.xls* (шаблон сметы работ);
– *exitexcel.bat* (файл закрытия документов Excel);
– *killexcel.bat* (файл принудительного закрытия документов Excel);
– *manual.pdf* (руководство пользователя).

Исполняемый файл *cow.exe* был скомпилирован в пакете *C++Builder* 2010 *Architect* из 4-х модулей. Листинг программного кода модулей *Unit1.cpp*, *Unit2.cpp*, *Unit3.cpp*, *Unit4.cpp* приведен в приложении А.

Ввод входных данных осуществляется пользователем, при создании объектов модели оценки затрат используется следующий паттерн заполнения данных по умолчанию (на примере создания проекта):

```
UnicodeString new_project_shortname = "новый проект", new_project_name = "описание нового проекта";
 if (InputQuery("Calculation of workspace ",
                "Введите наименование нового проекта:", new_project_shortname))
  {
 if (InputQuery("Calculation of workspace ",
                "Введите описание нового проекта:", new_project_name))
  {
                TableProject->Append();
                TableProject->FieldByName("project_shortname")->AsAnsiString = new_project_shortname;
                TableProject->FieldByName("project_name")->AsAnsiString = new_project_name;
                TableProject->Post();
  }
 }
```

Расчет затрат осуществляется на основании введенных пользователем данных. Результаты расчетов также сохраняются в БД:

```
//расчеты затрат на разработку
 ADOTable1->Close();
 ADOTable1->TableName = "labor_develop";
 ADOTable1->Open();
 ADOTable1->Filtered = false;
 ADOTable1->Filter = "project_id="+TableProject->FieldByName("project_id")->AsAnsiString;
 ADOTable1->Filtered = true;
 ADOTable1->First();
 float labor_devel = 0;

 if (TableComponent->RecordCount==0) {Abort();}

 for (int i = 0; i < TableComponent->RecordCount; i++)
 {
  float component_version = ADOTable1->FieldByName("component_version")->AsFloat,
                component_depth = ADOTable1->FieldByName("component_depth")->AsFloat,
                component_labor = ADOTable1->FieldByName("component_labor")->AsFloat,
                component_labor_algvote = ADOTable1->FieldByName("component_labor_algvote")-
>AsFloat;

  if (component_version==1)
        {
         labor_devel = labor_devel +
                (component_depth*component_labor);
        }
  else
        {
         labor_devel = labor_devel +
                (component_depth*component_labor) +
                (component_version-1)*component_labor + component_labor_algvote;
        }


  ADOTable1->Next();
 }
 TableProject->Edit();
 TableProject->FieldByName("project_labor_devel")->AsFloat = labor_devel;
 TableProject->FieldByName("project_cost_devel")->AsFloat = TableProject-
>FieldByName("project_paydevelop")->AsFloat * labor_devel;
```

```
TableProject->Post();
//расчет затрат на другие этапы ЖЦ и весь проект

TableStage->First();
float project_labor = labor_devel,
project_cost = TableProject->FieldByName("project_paydevelop")->AsFloat * labor_devel;

for (int i = 0; i < TableComponent->RecordCount; i++)
 {
 TableStage->Edit();
 TableStage->FieldByName("stage_labor")->AsFloat =
                    TableStage->FieldByName("stage_weight")->AsFloat *
                    TableProject->FieldByName("project_labor_devel")->AsFloat;

 TableStage->FieldByName("stage_cost")->AsFloat =
                    TableStage->FieldByName("stage_pay")->AsFloat *
                    TableStage->FieldByName("stage_labor")->AsFloat;

 TableStage->Post();

 project_labor = project_labor + TableStage->FieldByName("stage_labor")->AsFloat;
 project_cost = project_cost + TableStage->FieldByName("stage_cost")->AsFloat;

 TableStage->Next();
 }

TableProject->Edit();
TableProject->FieldByName("project_labor")->AsFloat = project_labor;
TableProject->FieldByName("project_cost")->AsFloat = project_cost;
TableProject->Post();
```

Результаты расчета выводятся на главную экранную форму программной системы. Также результаты расчетов могут быть выведены в документ *Excel*:

```
MessageBox(NULL,(char*)("Для выгрузки сметы необходимо сохранить и закрыть все документы Excel. Для продолжения нажмите OK."),(char*)(" Calculation of workspace "),MB_OK+64);
 ShellExecute(Form1->Handle,NULL,"exitexcel.bat",NULL,NULL,SW_HIDE);

 MessageBox(NULL,(char*)("Несохраненные документы Excel будут закрыты принудительно. Для продолжения нажмите OK."),(char*)(" Calculation of workspace "),MB_OK+64);
 ShellExecute(Form1->Handle,NULL,"killexcel.bat",NULL,NULL,SW_HIDE);
```

```
if (SaveDialog1->Execute())
 {
        AnsiString exist_path = ExtractFilePath(Application->ExeName)+"\\estimate.xls";
        const char *ex_p = exist_path.c_str();
        AnsiString save_path = SaveDialog1->FileName+".xls";
        const char *sp = save_path.c_str();
        if (!CopyFile(ex_p,sp,1))
         {
          ShowMessage(GetLastError());
         }

OleInitialize(NULL);
iStartExcel(save_path.c_str());

toExcelCell(2,1,TableProject->FieldByName("project_name")->AsString);
toExcelCell(5,2,TableProject->FieldByName("project_labor_devel")->AsString);
toExcelCell(5,3,TableProject->FieldByName("project_cost_devel")->AsString);

int i=6;
TableStage->First();
for (int count=1; count < TableStage->RecordCount+1; count++)
 {
        toExcelCell(i,1,TableStage->FieldByName("stage_name")->AsString);
        toExcelCell(i,2,TableStage->FieldByName("stage_labor")->AsString);
        toExcelCell(i,3,TableStage->FieldByName("stage_cost")->AsString);
        TableStage->Next();
        i++;
 }

        toExcelCell(i,1,"Итого");
        toExcelCell(i,2,TableProject->FieldByName("project_labor")->AsString);
        toExcelCell(i,3,TableProject->FieldByName("project_cost")->AsString);

 Border("A6:C"+IntToStr(i),1);
}
```

Второй не менее важный документ – руководство пользователя. В нем говориться, что программная система «Calculation of workspace» предназначена для автоматизации расчета трудозатрат и стоимости

разработки/модернизации программного обеспечения критически важных систем.

Работа с программой начинается с запуска файла COW.exe. Открывается главное окно программы (рисунок 3.4).

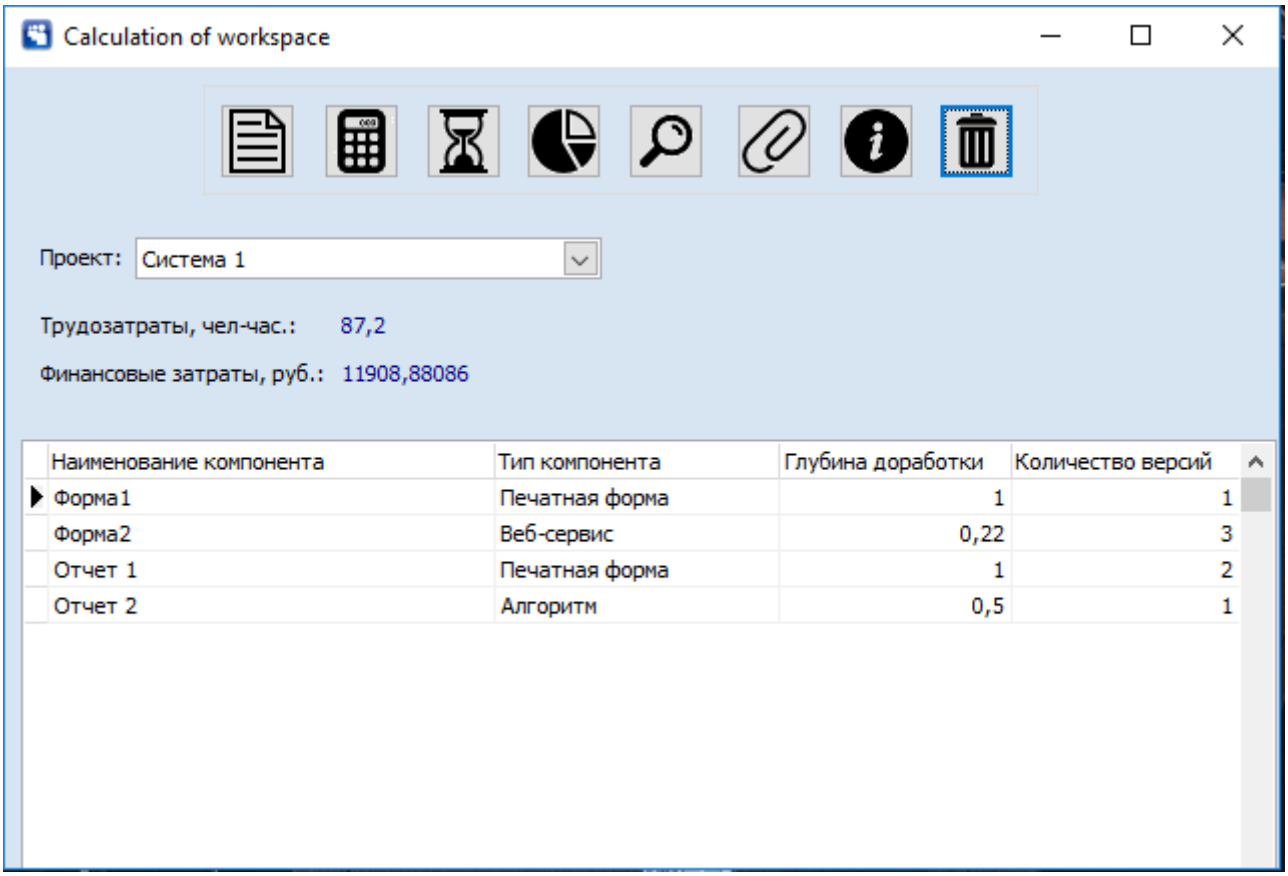


Рисунок 3.4 – Главное окно Calculation of workspace

Для начала ввода данных по проекту его необходимо создать, выбора в главном меню пункт «Проект/Создать новый». После этого запрашивается наименование проекта, которое впоследствии будет отображаться в общем списке проектов, на главной форме (рисунок 3.5).

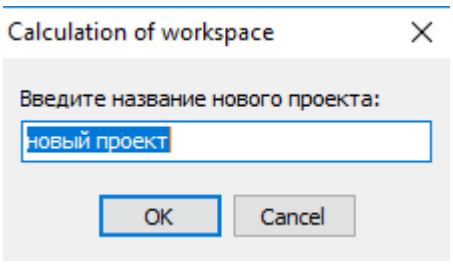


Рисунок 3.5 – Запрос наименования нового проекта

Для определения этапов жизненного цикла проекта необходимо выбрать пункт меню «Проект/Этапы ЖЦ и нормы оплаты труда» (рисунок 3.6). Здесь необходимо ввести для каждого этапа нормы оплаты труда (финансовые затраты сотрудника за час работы на этапе работ), и весовой коэффициент, на который будет умножаться трудоемкость этапа разработки, для определения трудоемкости каждого этапа.

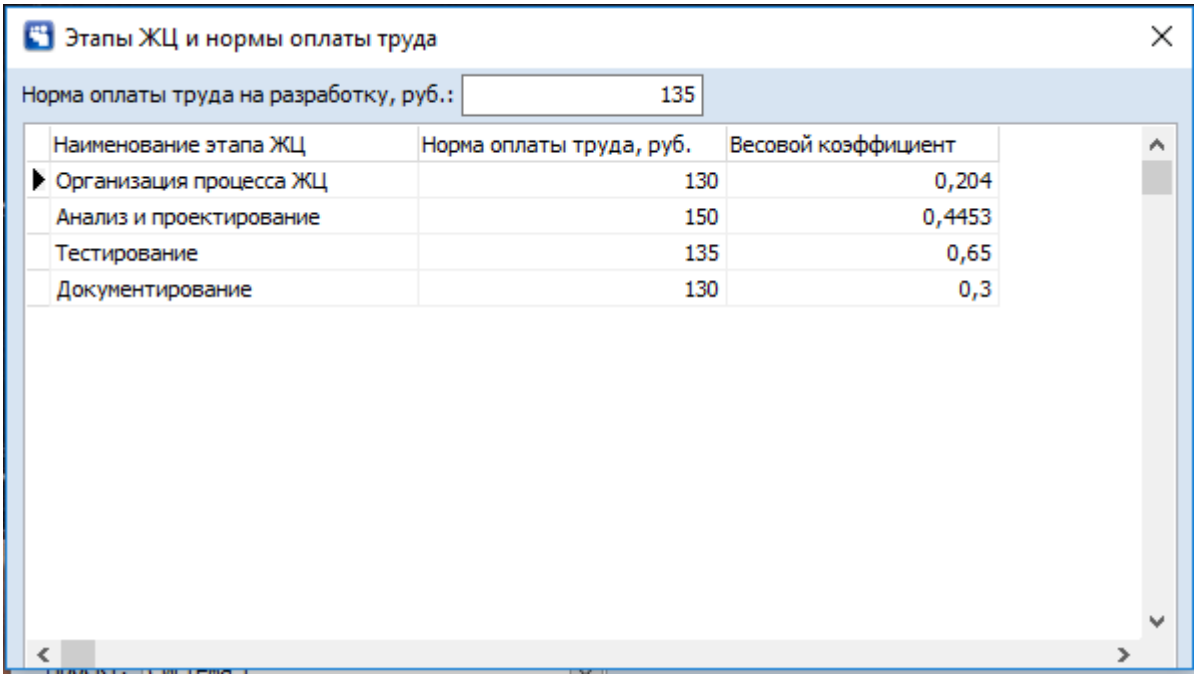


Рисунок 3.6 – Окно «Этапы ЖЦ и нормы оплаты труда»

Выбрав пункт меню «Настройки/Типы компонентов» открывается форма (рисунок 3.7). В ней необходимо определить типы компонентов, которые существуют в системе или необходимо разработать. Также здесь определяются трудозатраты на разработку компонента каждого типа и трудозатраты на разработку среды исполнения, если вводится программная избыточность.

После ввода данных необходимо на главной форме создать компоненты и задать их характеристики (тип компонента, глубина доработки, количество версий).

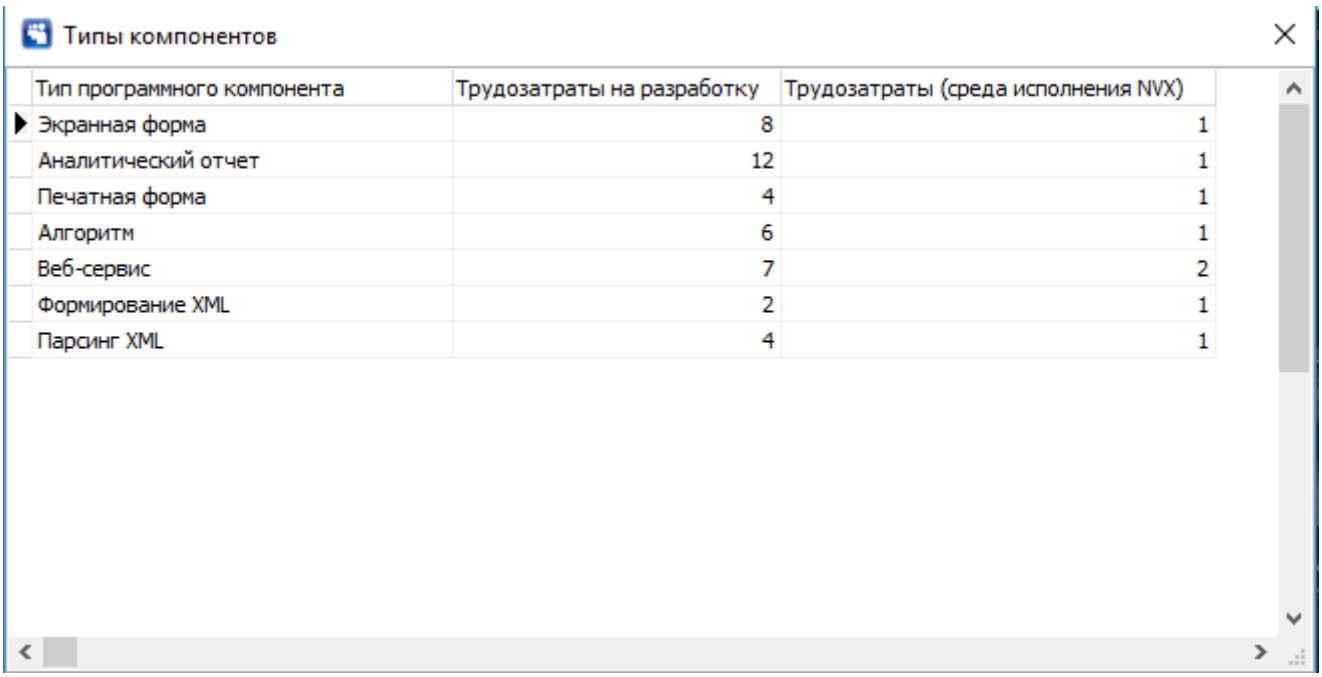


Рисунок 3.7 – Окно «Типы компонентов»

Для расчета трудоемкости необходимо нажать кнопку «Рассчитать смету». Рассчитываются показатели трудоемкости и стоимости проекта. Для более детальной информации расчетов необходимо нажать кнопку «…». Открывается форма, на которой отражены расчеты стоимости каждого этапа работ (рисунок 3.8).

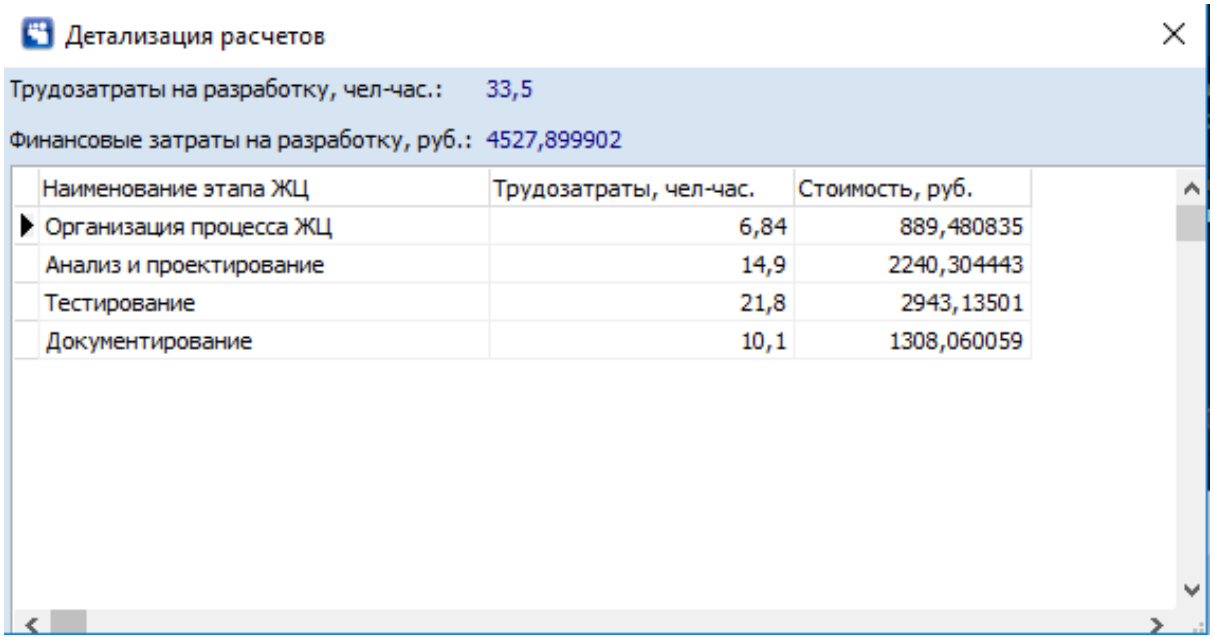


Рисунок 3.8 – Окно «Детализация расчетов»

Для выгрузки сметы в документ Microsoft Excel необходимо выбрать пункт меню «Проект/Экспорт сметы в Excel». На диске сохраняется файл Excel с выводом расчетов (рисунок 3.8).

	A	B	C
1	Для разработки/модернизации программного обеспечения системы		
2	**новый проект**		
3	необходимо выполнение следующих работ:		
4	**Наименование этапа ЖЦ ПО**	**Трудозатраты, чел-час.**	**Стоимость работ, руб.**
5	Разработка	35,94000244	48519,00391
6	Организация процесса ЖЦ	7,331760406	9531,288086
7	Анализ и проектирование	16,00408363	24006,125
8	Тестирование	23,36100006	31630,79492
9	Документирование	10,7820015	14016,60156
10	Итого	104,2008514	141720,4219

Рисунок 3.8 – Смета работ по разработке/модернизации ПО

В результате работы программы «Calculation of workspace» менеджеру проекта будет предоставлена полная информация о трудозатратах по каждому этапу жизненного цикла программного продукта и расчет стоимости выполнения работ по этим этапам.

3.3 Тестирование и апробация разработанной программной системы на примере ООО «Автоматические Системы Пожаротушения»

Для проверки корректности реализации расчетов осуществим тестовый расчет с помощью программы и по формуле 2.4.

В разработанную программную систему были введены типы компонентов и их характеристики, приведенные на рисунке 3.9.

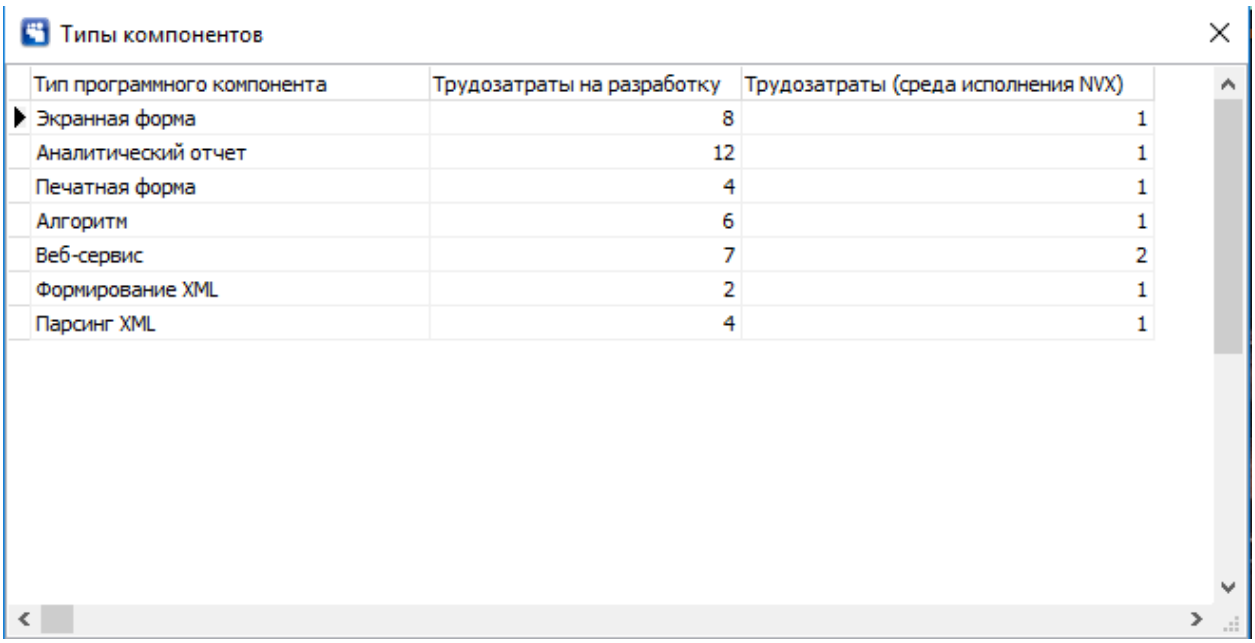


Рисунок 3.9 – Введенные типы компонентов

Был создан проект, по нему введены этапы ЖЦ и их характеристики (рисунок 3.10).

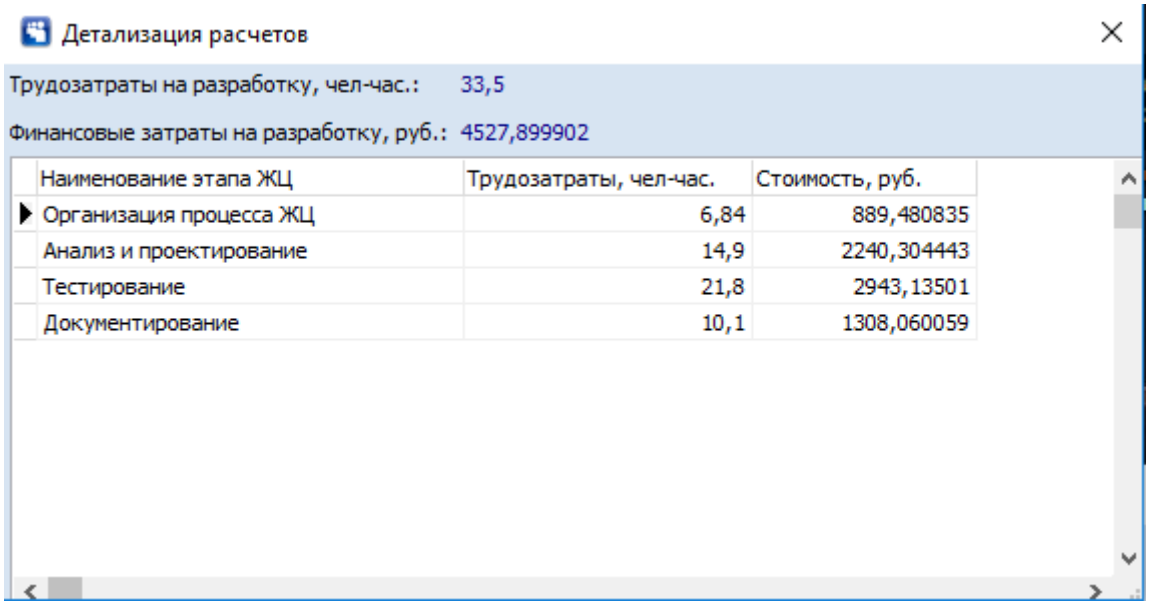


Рисунок 3.10 – Введенные этапы ЖЦ

Затем в проект были добавлены компоненты и введены их характеристики (рисунок 3.11).

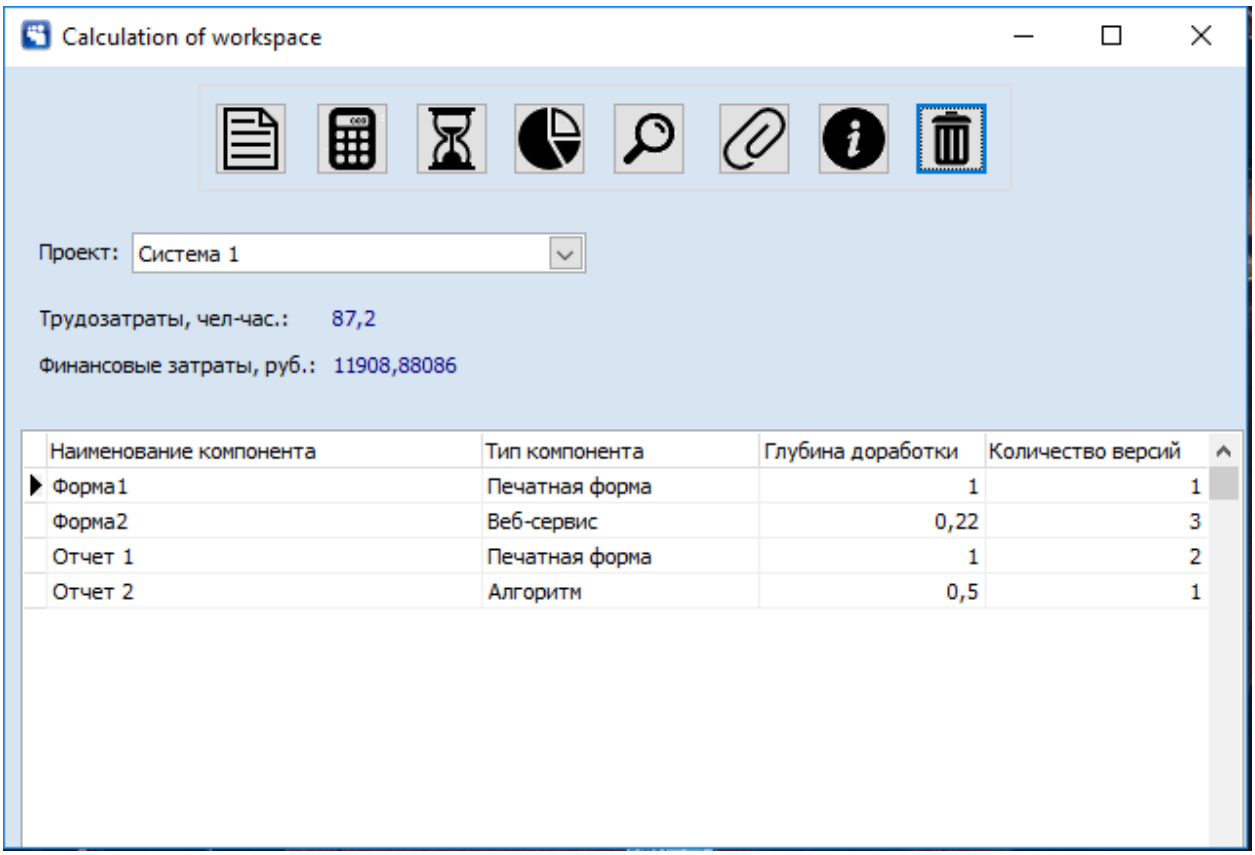


Рисунок 3.11 – Введенные компоненты

После ввода данных были осуществлены расчеты. Расчет по программной системе показал финансовые затраты равные 11908,88 руб. Расчет по формулам показал аналогичный результат.

Таким образом, можно сделать вывод о том, что программная система работает корректно и ее можно использовать для реальных расчетов по оценке затрат на модернизацию программного обеспечения.

Предлагаемые модель, алгоритм и программный продукт были апробированы на программном обеспечении информационной системы автоматических систем пожаротушений - программно-техническом комплексе «СУПРК», разработанном и активно применяемом в ООО «АСП». Программные средства комплекса обеспечивают автоматический контроль состояния технологических датчиков, протоколирование и наглядное отображение текущего состояния объекта, а также позволяет управлять системой в дистанционном режиме. Данная программа предназначена для

интерактивного общения с дежурным (диспетчером) при помощи интуитивно понятного интерфейса, за счет которого человек может получить представление о состоянии объекта в любой момент времени в зависимости от чего принять необходимые решения.

Данное программное обеспечение поддерживает ниже приведенные функции:

- контроль состояния технологических датчиков, шлейфа охранной сигнализации (контроль открытия/закрытия дверей в шкафах с электрооборудованием) и обработку поступившей информации;
- отображение плана охраняемого объекта при возникновении пожара с индикацией зоны, где возникла эта ситуация. При этом принята следующая индикация: нормальное состояние зоны обозначается белым цветом; зона, где обнаружено превышение фона по более чем по одному датчику – красным цветом;
- отображение общего плана территории, с отображением схематического обозначения охраняемого объекта с цветовой индикацией, где:
  - нормальное состояние объекта обозначается зеленым цветом;
  - превышение уровня «Опасность» по одному и более датчикам, если не больше чем один из датчиков превышает уровень «Пожар» – оранжевым цветом;
  - превышение уровня «Пожар» по двум и более датчикам– красным цветом;
  - неисправность хотя бы одного датчика – темно-серым цветом;
- отображение состояния и работы оборудования системы;
- перехват управления в дистанционный режим и ручное управление работой системы.

Аппаратно-программный комплекс СУПРК постоянно модернизируется, охватывая новые функции и задачи. Над модернизацией СУПРК работают специалисты компании ООО «Автоматические Системы Пожаротушения».

Общий размер кода системы более 500 тыс. строк, что позволяет отнести СУПРК к классу крупномасштабных проектов. Поэтому при сопровождении системы рекомендуется наиболее полное документирование всех этапов ее жизненного цикла.

Учитывая масштаб внедрения и сложность программной системы, к ее функционированию предъявляются высокие требования надежности. Для обеспечения высокой надежности требуется больший вклад трудовых и финансовых ресурсов со стороны исполнителя. Поэтому при модернизации системы необходима адекватная оценка и прогнозирование затрат трудовых и финансовых ресурсов.

При модернизации системы СУПРК был применен предложенный авторов методический инструмент оценки стоимости доработки АПК и программный продукт « ».

Для проверки корректности оценки по предложенной методике был проведен анализ оценочных и фактических данных на шести различных этапах модернизации программного обеспечения. Расхождение фактической и прогнозируемой стоимости вычислялось по формуле 3.1:

$$Расхождение = \left| \frac{Стоимость_{прогнозируемая} - Стоимость_{фактическая}}{Стоимость_{фактическая}} \right| \qquad (3.1)$$

На основании данных по расхождениям была произведена оценка показателя *PRED*(*L*). Он отражает процент оценок, отклонение которых от фактических значений меньше *L*.[1] Показатель рассчитывается по следующей формуле 3.2:

$$PRED(L) = \frac{100}{n} \sum_{i=1}^{n} \begin{cases} 1, Расхождение \le L, \\ 0 \end{cases}, \qquad (3.2)$$

где n – общее количество оценок.

В таблице 3.1 приведены значения *PRED*(*L*) для параметра *L*, равного 20%, 25% и 30%.

Таблица 3.1 – Значения *PRED*(*L*)

L	PRED(L)
PRED(20)	83%
PRED(25)	83%
PRED(30)	100%

Согласно этим данным, расчеты по предложенной методике достаточно точны. Также расхождения оценок по всем доработкам с применением методики меньше, чем без ее применения.

Таким образом, можно принять решение об использовании разработанной методики в ИТ-компаниях, занимающихся разработкой и поддержкой АПК сложных систем управления.

Разработанная методика оценки затрат на АПК отказоустойчивых систем управления позволяет:

– учитывать трудозатраты персонала на различных этапах жизненного цикла программного обеспечения;

– определять размер системы исходя из количества объектных указателей;

– учитывать глубину доработки компонентов архитектуры программной системы;

– использовать для расчета коэффициентов экспертную оценку, если нет актуальных данных по предыдущим работам в данной конкретной ИТ-компании;

– учитывать трудозатраты на введение программной избыточности;

– учитывать стоимость процессоров разного типа.

ЗАКЛЮЧЕНИЕ

Задумывая и начиная разработку IT проектов, руководители предполагают получить некоторый положительный эффект от их практического применения. Этот эффект может выражаться как экономическими категориями повышения дохода или прибыли, так и техническими: расширением назначения и функций или улучшением характеристик качества продуктов и систем. Однако, часто проекты не соответствуют исходному, декларированному назначению и первоначальным спецификациям требований к функциям и характеристикам качества, не укладываются в согласованные графики и бюджет разработки. Многие созданные информационные системы не способны выполнять полностью требуемые функциональные задачи с гарантированным качеством и их приходится долго и иногда безуспешно дорабатывать для достижения необходимого качества функционирования, затрачивая дополнительно большие ресурсы и время.

Этому способствует ограниченность ресурсов, особенно времени, необходимых для достижения и оценивания, требуемых значений характеристик комплекса программ, а также недостаточная точность и обоснованность процесса оценки стоимости реализации аппаратно – программных комплексов.

Отсутствие четкого понятного алгоритма оценивания вызывает конфликты между заказчиками и поставщиками. Поэтому создание методики оценивания стоимости АПК и автоматизация данного бизнес-процесса - ключевой фактор обеспечения достижения целей проекта.

Анализ теоретических и эмпирических подходов к существующим методам оценивания затрат на разработку программного обеспечения показал, что грамотное прогнозирование итоговой стоимости затрат по проекту является важным фактором успеха, так как позволяет менеджерам понимать и реализовывать поставленные перед ними задачи, оценивать

возможности их достижения в определенные сроки, своевременно вносить коррективы, а также обеспечивать производство нужными ресурсами.

Несмотря на достоинства существующих моделей оценки стоимости им также характерна излишняя для простого специалиста сложность, неспособность к адаптации, невозможность применения при разработки критических по надежности систем управления. Опыт оценивания информационных систем, в которых при проектировании применялся мультиверсионный подход на предприятиях часто неудачен.

Данная проблема послужили стимулом к разработке нового методического подхода к расчету стоимости реализации аппаратно-программного комплекса отказоустойчивых систем управления.

В результате диссертационного исследования были получены следующие результаты:

- проведен анализ существующих методик, моделей и алгоритмов оценки затрат на разработку программного обеспечения и выявлены их недостатки;
- проанализированы требования и подходы к обеспечению отказоустойчивости информационно-управляющих систем;
- определены достоинства и недостатки мультиверсионного подхода к проектированию;
- предложена методика оценки затрат на модернизацию программного обеспечения с учетом выявленных недостатков;
- реализована программная система для автоматизации расчетов по оценке затрат на разработку и модернизацию программного обеспечения;
- проведена апробация разработанной информационной системы на модулях обеспечения охранной и пожарной безопасности, применяемых в ООО «Автоматические системы пожаротушения»;

- осуществлена оценка затрат по предложенной методике на шести проектах;
- сделан вывод о пригодности методики для использования в промышленных задачах;
- опубликовано семь статей по теме диссертации;
- зарегистрирована программная система «*Calculation of workspace*».

Подведя итог, следует отметить, что разработанные рекомендации являются эффективными инструментами оптимизации системы бюджетирования в IT проектах.

Таким образом, научная задача точного прогнозирования затрат финансовых и трудовых ресурсов на разработку АПК отказоустойчивых систем управления решена.

В дальнейших исследованиях планируется изучение различия затрат на введение программной избыточности различными методами. Планируется ввести учет норм оплаты труда специалистов различной квалификации, а также зависимость количества процессоров от количества и сложности модулей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Boehm, Barry W. Software cost estimation with COCOMO II / B.W. Boehm. – Prentice Hall PTR, 2000.
2. Boehrn, Barry W. Software Development Cost Estimation Approaches [Электронный ресурс] / Barry W. Boehrn, Chris Abts, Sonita Chulani // University of Southern California, IBM Research. – Режим доступа : http://sunset.usc.edu/publications/TECHRPTS/2000/usccse2000-505/usccse2000-505.pdf (дата обращения: 29.09.2016).
3. Goodman, P. Software Metrics : Best Practices for Successful IT Management / P. Goodman. – Rothstein Associates Inc., Publisher, Brookfield, 2004.
4. Independent contractors in 2015 [Электронный ресурс] / U.S. Bureau of Labor Statistics. – Режим доступа : http://www.bls.gov/opub/ted/2005/jul/wk4/art05.htm (Дата обращения: 22.12.2016).
5. Jorgenson, D. Information Technology And The Japanese Economy / D. Jorgenson, K. Motohashi // Journal of the Japanese and International Economies. – 2005. – № 19.
6. Pfleeger, Sh. L. Software Cost Estimation And Sizing Methods : Issues And Guidelines / Sh. L. Pfleeger, F. Wu, R. Lewis. – Rand corporation, 2005.
7. RUP Rational Unified Process – технологический процесс разработки ПО (Программного обеспечения) [Электронный ресурс] // Режим доступа : http://www.rup-rus.ru. (Дата обращения: 12.07.2016)
8. The Standish Group Report [Электронный ресурс]. – Режим доступа : http://www.projectsmart.co.uk/docs/chaos-report.pdf (дата обращения: 29.09.2016).
9. Алиев, Х. Р. Эффективная модель оценки разработки программного обеспечения [Электронный ресурс] / Х. Р. Алиев. – Исследовано в России : Электронный научный журнал. – Т. 11. – Вып. 3 – 2008.– Режим доступа : http://zhurnal.ape.relarn.ru/articles/2008/030.pdf (дата обращения: 29.10.2016).

10. Алиев, Х.Р. Комбинированная модель оценки разработки программного обеспечения на основе COCOMO II и Functional Point / Х.Р. Алиев // Научно-технические ведомости СПБГПУ. – 2009. – 117 с.
11. Алиев, Х.Р., Модель планирования и управления разработкой сложных программных систем на основе комбинированной методики оценки трудозатрат : дис. канд. экон. наук / Х.Р. Алиев. – СПб., 2010. – 206 с.
12. Ананьева, Т.Н. Стандартизация, сертификация и управление качеством программного обеспечения : учеб. пособие / Т.Н. Ананьева, Н.Г. Новикова, Г.Н. Исаев. — М. : ИНФРА-М, 2017. — 232 с.
13. Антамошкин, А.Н. Определение оптимальной структуры мультиверсионного программного обеспечения при ограничениях по времени и стоимости / А.Н. Антамошкин, И.В. Ковалев // Вестник САА. – 2000. – № 1.
14. Архипенков, С. Психология управления программными проектами / С. Архипенков. – Режим доступа : http://citforum.ru/SE/project/psychology/ . Дата обращения: 30.09.2011.
15. Богатырев С.Ю. Использование современных информационных систем в корпоративных финансах [Электронный ресурс]: учеб. пособие / С.Ю. Богатырев — М.: РИОР: ИНФРА-М, 2017. — 180 с.
16. Брауде, Э.Д. Технология разработки программного обеспечения / Э.Д. Брауде. – СПб. : Питер Принт, 2014. – 655 с.
17. Бриткин, А. И., Разработка методики и моделей управления рисками в проектах разработки программного обеспечения : Дис. канд. экон. наук / А.И. Бриткин. – М., 2009. – 129 с.
18. Гагарина, Л.Г. Введение в архитектуру программного обеспечения: Учебное пособие / Гагарина Л.Г., Федоров А.Р., Федоров П.А. - М.: ИД ФОРУМ, НИЦ ИНФРА-М, 2016. - 320 с.
19. Гагарина, Л.Г. Разработка и эксплуатация автоматизированных информационных систем: Учебное пособие / Л.Г. Гагарина - М.:ИД ФОРУМ, НИЦ ИНФРА-М, 2017. - 384 с.

20. Гагарина, Л.Г. Технология разработки программного обеспечения : учеб. пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Виснадул ; под ред. Л.Г. Гагариной. — М. : ИД «ФОРУМ» : ИНФРА-М, 2017. — 400 с.
21. Глазова, М.А., Моделирование стоимости разработки проектов в ИТ-компаниях : Дис. канд. экон. наук / М.А. Глазова. – М., 2008. – 205 с.
22. Долженко, А.И. Технологии командной разработки программного обеспечения информационных систем [Электронный ресурс] / А.И. Долженко – Электрон. текстовые данные. – М. : Интернет-Ун-т Информационных Технологий (ИНТУИТ), 2016. – 300 с. – Режим доступа: http://www.iprbookshop.ru/39569.html. – ЭБС «IPRbooks»
23. Емельянова, Н.З. Проектирование информационных систем : учебное пособие / Н.З. Емельянова, Т.Л. Партыка, И.И. Попов. — М. : ФОРУМ : ИНФРА-М, 2017. — 432 с.
24. Золотухина, Е.Б. Управление жизненным циклом информационных систем (продвинутый курс): Электронная публикация / Е.Б. Золотухина, С.А. Красникова, А.С. Вишня - М.:КУРС, НИЦ ИНФРА-М, 2017. - 119 с.
25. Иксанова, А.Х. Повышение эффективности управления инновационным процессом создания программного обеспечения с использованием модели разработки на заказ : Дис. канд. экон. наук / А.Х. Иксанова. – М., 2006. – 175 с.
26. Исаев, Г. Н. Управление качеством информационных систем / Г. Н. Исаев - М.:НИЦ ИНФРА-М, 2016. - 248 с.
27. Кибанов, А.Я. Экономика управления персоналом: Учебник/Кибанов А.Я., Митрофанова Е.А., Эсаулова И.А; под ред. А.Я. Кибанова - М.: НИЦ ИНФРА-М, 2016. - 427 с.
28. Ковалев, И. В. К вопросу реализации муравьиного алгоритма при выборе состава мультиверсионного программного обеспечения информационно-управляющих систем / И. В. Ковалев [и др.] // Приборы и системы : Управление, контроль, диагностика. – 2012 – № 2. – С. 1-4.

29. Колдовский В. Разработка ПО : метрики программных проектов / В. Колдовский // Компьютерное обозрение. – 2007. – №13 (581).
30. Кон, М. Scrum : Гибкая разработка ПО / М. Кон. – М. : Вильяме, 2011. – 576 с.
31. Липаев В.В. Экономика программной инженерии заказных программных продуктов: Учебное пособие. — М.: МАКС Пресс, 2014. —148 с.
32. Липаев, В.В. Проектирование и производство сложных заказных программных продуктов [Электронный ресурс] / В.В. Липаев – Электрон. текстовые данные. – М. : СИНТЕГ, 2011. – 398 с. – Режим доступа: http://www.iprbookshop.ru/27298.html. – ЭБС «IPRbooks»
33. Лич, Л Вовремя и в рамках бюджета: Управление проектами по методу критической цепи / Лич Л., - 3-е изд. - М.:Альпина Пабл., 2016. - 354 с.
34. Макконнелл, Н. С. Сколько стоит программный проект / Н. Макконнелл С. – М. : Русская Редакция ; Спб.: Питер, 2007. – 297 с.
35. Математические основы управления проектами : учебн. пособие / С.А. Баркалов, В.И. Воропаев, Г.И. Секлетова [и др.] ; под ред. В.Н. Буркова – М. : Высш. шк., 2005. – 423 с.
36. Меламед, А.Я. Методы оценки трудоемкости разработки программного обеспечения корпоративных информационных систем : Дис. канд. техн. наук / А.Я. Меламед. – М., 2006. – 137 с.
37. Меняев, М. Ф. Бизнес-проектирование инженерных разработок программного обеспечения / М.Ф. Меняев. – М. : Изд-во МГТУ им. Н.Э. Баумана, 2004.
38. Михалкина, Е.В. Экономика труда : учебник / Е.В. Михалкина, О.С. Белокрылова, Е.В. Фурса. — М. : РИОР : ИНФРА-М, 2017. — 273 с.
39. Назаров, С.В. Архитектура и проектирование программных систем : монография / С.В. Назаров. — 2-е изд., перераб. и доп. — М. : ИНФРА-М, 2017. — 374 с.
40. Насыров, И.Р. Мультиверсионное программирование в автоматических системах управления технологическими процессами / И.Р. Насыров, Д.В.

Сташков // Приоритетные научные направления: от теории к практике. – 2015. – № 16. – С. 75-78.

41. Новой, А.В. Система анализа архитектурной надежности программного обеспечения : Дис. канд. техн. наук / А.В. Новой. – Красноярск, 2011 – 131 с.
42. Орлов, С. Технологии разработки программного обеспечения : учебное пособие / С. Орлов. – 2-е изд. – Спб. : Питер, 2003. – 480 с.
43. Основная программа по администрированию // Издание Пенсионного Фонда Российской Федерации «Я работаю в ПФР». – 2011. [Электронный ресурс]. – Режим доступа : http://files.pfrf.ru/userdata/presscenter/gazeti/2011/gaz_co_201110.pdf (дата обращения : 01.12.2012).
44. Поздняков, Д. А. Архитектуры программных систем с применением мультиверсионной методологии / Д. А. Поздняков // Вестник университетского комплекса / НИИ СУВПТ, ВСФ РГУИТП. – 2005. – Вып. 6 (20). – С. 111-118.
45. Проектирование экономических информационных систем : учебник / Г.Н. Смирнова [и др.] ; под ред. Ю.Ф. Тельнова. – М. : Финансы и статистика, 2001. – 512 с.
46. Рухляда, И.В. Организация труда и формирование квалификационных требований к специалистам в сфере информационных технологий : Дис. канд. экон. наук / И.В. Рухляда. – М., 2011. – 147 с.
47. Савельева Е. А. Инжиниринг труда: проектирование трудовых процессов и систем: учеб. пособие / Е.А. Савельева. — М.: Вузовский учебник: ИНФРА-М, 2017. — 236 с.
48. Светлов, Н.М. Информационные технологии управления проектами: Учебное пособие / Н.М. Светлов, Г.Н. Светлова. - 2 изд., перераб. и доп. - М.: НИЦ ИНФРА-М, 2015. - 232 с.
49. Соммервилл, И. Инженерия программного обеспечения / И. Соммервилл. – 6-е изд. – М. : Вильяме, 2002. – 623 с.

50. Ступина, А. А. Технология надежностного программирования задач автоматизации управления в технических системах : монография / А. А. Ступина, С. Н. Ежеманская ; Сиб. федерал. ун-т, Ин-т. управления бизнес-процессами и экономики. – Красноярск: Сиб. федер. ун-т, 2011. – 164 с.
51. Суслин А.А. Экспериментальное исследование взаимосвязи значений метрик и показателей надежности программного обеспечения / А.А. Суслин // Молодой ученый : Научный журнал. – 2010. – № 6 (17). – С. 67-71.
52. Телекоммуникационные системы и сети : В 3 т. Т. 3. Мультисервисные сети : учеб. пособие / В.В. Величко и др. ; под ред. В.П. Шувалова. – 2-е изд. – М. : Горячая линия-Телеком, 2015. – 592 с. – (Специальность).
53. Тихомирова, О. Г. Управление проектом: комплексный подход и системный анализ: Монография / О. Г. Тихомирова - М.: НИЦ ИНФРА-М, 2016. - 300 с.
54. Троцкий, М. Управление проектами / М. Троцкий, Б. Груча, К. Огонек. – М. : Финансы и статистика, 2011. – 304 с.
55. Хелдман, К. Профессиональное управление проектом / К. Хелдман ; пер. с англ. А.В. Шаврина. – 6-е изд. – М. : БИНОМ. Лаборатория знаний, 2015. – 731 с. – (Проекты, программы, портфели).
56. Царев, Р.Ю. Мультиверсионное программное обеспечение / Р.Ю. Царев // Алгоритмы голосования и оценка надежности : монография / Р.Ю. Царев, А.В. Штарик, Е.Н. Штарик. – Красноярск : Сиб. федер. ун-т, 2013. – 116 с.
57. Царев, Р.Ю. Система многоатрибутивного формирования мультиверсионных программных средств отказоустойчивых систем управления : Дис. канд. техн. наук / Р.Ю. Царев. – Красноярск, 2003 143 с.
58. Шафер Д., Фатрелл Р., Шафер Л. Управление программными проектами: достижение оптимального качества при минимуме затрат: Пер. с англ. / Д. Шафер, Р. Фатрелл, Л. Шафер. – М. : Вильяме, 2003. – 1136 с.

59. Шеенок Д.А, Кукарцев В.В. Оценка затрат на модернизацию программного обеспечения критических по надежности систем. // Вестник СибГАУ. Вып. 5(45). – Красноярск, 2012. – с. 62-65.
60. Шеенок Д.А., Анализ существующих моделей оценки стоимости разработки программного обеспечения // Электронное научно-практическое периодическое издание «Экономика и социум» (ISSN 2225-1545). Вып. 5, 2012, с. 931-939.
61. Якимчук, С. MSF философия создания IT-решений или голые амбиции лидера [Электронный ресурс] / С. Якимчук // Softline Company. – Режим доступа : http://citforum.ru/SE/project/msf (Дата обращения: 13.11.2016)
62. Якобсон, А. Унифицированный процесс разработки программного обеспечения / А. Якобсон, Г. Буч, Дж. Рамбо. – СПб. : Питер, 2002. – 496 с.

СПИСОК ПУБЛИКАЦИЙ АВТОРА ПО ТЕМЕ ДИССЕРТАЦИИ

1. Ярков К.В. Бенчмаркинг как элемент повышения конкурентоспособности: исторические аспекты // Проспект Свободный-2015 : материалы науч. конф., посвященной 70-летию Великой Победы (15–25 апреля 2015 г.) [Электронный ресурс] / отв. ред. Е. И. Костоглодова. – Электрон. дан. – Красноярск : Сиб. федер. ун-т, 2015. - С. 86-89.
2. Ярков К.В., Вашко Н.С. Алгоритм создания клиентоориентированного IT- продукта // Проспект Свободный-2016 : материалы науч. конф., посвященной Году образования в Содружестве Независимых Государств (15-25 апреля 2016 г.) [Электронный ресурс] / отв. ред. А.Н. Тамаровская. – Электрон. дан. – Красноярск : Сиб. федер. ун-т, 2016.
3. Ярков К.В. К вопросу об инструментах совершенствования системы управления персоналом в организации// Проспект Свободный-2016 : материалы науч. конф., посвященной Году образования в Содружестве Независимых Государств (15-25 апреля 2016 г.) [Электронный ресурс] / отв. ред. А.Н. Тамаровская. – Электрон. дан. – Красноярск : Сиб. федер. ун-т, 2016. – С.62-64.
4. Ярков К.В., Коваленко В.С. К вопросу о моделировании процессов аппаратно-программных комплексов. // Проспект Свободный-2016 : материалы науч. конф., посвященной Году образования в Содружестве Независимых Государств (15-25 апреля 2016 г.) [Электронный ресурс] / отв. ред. А.Н. Тамаровская. – Электрон. дан. – Красноярск : Сиб. федер. ун-т, 2016.
5. Ярков К.В., Коваленко, В.С. Проектирование информационной системы управления научно-исследовательской работой в вузе // Международный студенческий научный вестник. – 2017. – № 4-2. С.

187-188. ; URL: https://eduherald.ru/ru/article/view?id=17054 (дата обращения: 16.06.2017).

6. Ярков К.В., Коваленко В.С. Особенности расчета трудовых затрат на разработку программного обеспечения отказоустойчивых систем // Международный студенческий научный вестник. – 2017. – № 4-2. С. 189-191.; URL: https://eduherald.ru/ru/article/view?id=17054 (дата обращения: 16.06.2017).
7. Ярков К.В., Машинец Е.Е. Модель оценки эффективности использования виртуальных и облачных технологий для организаций // Международный студенческий научный вестник. – 2017. – № 4-2. С. 188-189.; URL: https://eduherald.ru/ru/article/view?id=17054 (дата обращения: 16.06.2017).
8. Ярков К.В. Информационная система оценки стоимости разработки и модернизации программного обеспечения (Calculation of workspace): свидетельство о государственной регистрации программы для ЭВМ / К.В. Ярков, А.А. Ступина, Р.И. Кузьмич– М.: Реестр программ для ЭВМ, 2017. Номер гос. рег. № 2017615181.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А. Листинг программного кода

Unit1.cpp

```
// ---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
#include <ComCtrls.hpp>
#include <objbase.h>
#include <ADODB.hpp>
#include <Classes.hpp>
#include <sysdyn.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <iostream.h>
#include <conio.h>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <windows.h>
#include <ComObj.hpp>
#include <utilcls.h>
// ---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

Variant App, vVarBooks, vVarBook, vVarSheets, Sh, vVarCell;

//---------------------------------------------------------------------------

void __fastcall toExcelCell(int Row,int Column, AnsiString data)
{
 vVarCell=Sh.OlePropertyGet("Cells").OlePropertyGet("Item",Row,Column);
```

```
 vVarCell.OlePropertySet("Value",data.c_str());
}


//---------------------------------------------------------------------


void __fastcall Border(AnsiString diapaz, int list)
{
  try
  {
   Variant range = App.OlePropertyGet("WorkSheets",list).OlePropertyGet("Range", diapaz.c_str());
   for (int z=1; z<=4; z++) range.OlePropertyGet("Borders").OlePropertyGet("Item",
z).OlePropertySet("LineStyle", 1);
  }
  catch(...) { ; }
}

//---------------------------------------------------------------------------

void __fastcall ExcelInit(String File, int lists)
{
if (lists > 0)
 {
  try {  App=Variant::GetActiveObject("Excel.Application");  }
  catch(...)
   {
        try { App=Variant::CreateObject("Excel.Application"); }
        catch (...)
         {
         }
   }

  try
  {
       if(File!="")
        {
        App.OlePropertyGet("WorkBooks").OleProcedure("Open",File.c_str());
        }
       else
        App.OlePropertySet("SheetsInNewWorkbook",lists);
```

```
            App.OlePropertyGet("WorkBooks").OleProcedure("add");
            Sh=App.OlePropertyGet("WorkSheets",1);
  }
  catch(...)
  {

  }
 }

}


void __fastcall  iStartExcel(AnsiString vasNameFile)
{
 //Создаем объект Excel.Application и загружаем файл бланка
 vasNameFile=vasNameFile.Trim();

  //Создаем объект Excel.Application
 try
  { App=GetActiveOleObject("Excel.Application");  }
 catch(...)
  { App=CreateOleObject("Excel.Application");  }
  App.OlePropertySet("Visible",true);
  vVarBooks=App.OlePropertyGet("Workbooks");
  vVarBooks.OleProcedure("Open",vasNameFile.c_str());
  vVarBook=vVarBooks.OlePropertyGet("Item",1);
  vVarSheets=vVarBook.OlePropertyGet("Worksheets");
  //На первый лист
  Sh=vVarBook.OlePropertyGet("Worksheets",1);
  Sh.OleProcedure("Activate");

}
// ---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner) : TForm(Owner) {
}
// ---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender) {
            AnsiString db_path = ExtractFileDir(Application->ExeName) + "\\esl.accdb";
            if (FileExists(db_path) == true) {
                        ADOConnectionMain->Connected = false;
```

```
                ADOConnectionMain->ConnectionString =
                "Provider=Microsoft.ACE.OLEDB.12.0;User ID=Admin;Data Source=" +
                db_path +
                ";Mode=Share Deny None;Jet OLEDB:System database=\"\";Jet OLEDB:Registry Path=\"\";Jet OLEDB:Database Password=\"\";Jet OLEDB:Engine Type=6;Jet OLEDB:Database Locking Mode=1;Jet OLEDB:Global Partial Bulk Ops=2;Jet OLEDB:Global Bulk Transactions=1;Jet OLEDB:New Database Password=\"\";Jet OLEDB:Create System Database=False;Jet OLEDB:Encrypt Database=False;Jet OLEDB:Don't Copy Locale on Compact=False;Jet OLEDB:Compact Without Replica Repair=False;Jet OLEDB:SFP=False;Jet OLEDB:Support Complex Data=False;";
                ADOConnectionMain->Connected = true;
                TableProject->Open();
                TableComponentType->Open();
                TableComponent->Open();
                TableStage->Open();
        }
        else {
                MessageBox(this->Handle, (char*)(
                        "Файл БД esl.accdb в директории исполняемого файла не найден. Приложение будет закрыто."),
                        (char*)("Evaluation System Labor"), MB_OK + 16);
                Application->Terminate();
        }
}
// ---------------------------------------------------------------------------
void __fastcall TForm1::N13Click(TObject *Sender) {
        if (MessageBoxA(this->Handle, (char*)(
                        "Вы действительно хотите удалить компонент?"),
                (char*)("Evaluation System Labor"),
                MB_YESNO + 32) == IDNO) {

        }
        else {
                Form1->TableComponent->Delete();
        }

}
// ---------------------------------------------------------------------------
void __fastcall TForm1::N16Click(TObject *Sender)
{
 Form2->ShowModal();
}
```

```
// ---------------------------------------------------------------------------
void __fastcall TForm1::DBLookupComboBox1Click(TObject *Sender)
{
        TableComponent->Filtered = false;
        TableComponent->Filter = "project_id=" + TableProject->FieldByName("project_id")->AsAnsiString;
        TableComponent->Filtered = true;

        TableStage->Filtered = false;
        TableStage->Filter = "project_id=" + TableProject->FieldByName("project_id")->AsAnsiString;
        TableStage->Filtered = true;
}
// ---------------------------------------------------------------------------

void __fastcall TForm1::FormShow(TObject *Sender)
{
        DBLookupComboBox1->KeyValue = 1; // по умолчанию первый проект из списка
        TableComponent->Filtered = false;
        TableComponent->Filter = "project_id=" + TableProject->FieldByName("project_id")->AsAnsiString;
        TableComponent->Filtered = true;

        TableStage->Filtered = false;
        TableStage->Filter = "project_id=" + TableProject->FieldByName("project_id")->AsAnsiString;
        TableStage->Filtered = true;
}
// ---------------------------------------------------------------------------
void __fastcall TForm1::N3Click(TObject *Sender)
{
        if (MessageBoxA(this->Handle, (char*)("Вы действительно хотите удалить проект?"),
                        (char*)("Evaluation System Labor"),
                        MB_YESNO + 32) == IDNO) {

        }
        else
        {
                Form1->TableProject->Delete();
        }
}
// ---------------------------------------------------------------------------
void __fastcall TForm1::N12Click(TObject *Sender)
```

```
	{
		UnicodeString new_component_name = "новый компонент";
		if (InputQuery("Evaluation System Labor",
			"Введите наименование нового компонента:", new_component_name))
			{
		TableComponent->Append();
		TableComponent->FieldByName("component_name")->AsAnsiString = new_component_name;
		TableComponent->FieldByName("component_depth")->AsAnsiString = "1";
		TableComponent->FieldByName("component_version")->AsAnsiString = "1";
		TableComponent->FieldByName("component_type_id")->AsAnsiString = "1";
		TableComponent->FieldByName("project_id")->AsAnsiString =	TableProject-
>FieldByName("project_id")->AsAnsiString;
		TableComponent->Post();
	    }
	}
	// ---------------------------------------------------------------------------

	void __fastcall TForm1::N17Click(TObject *Sender) {
		Form4->ShowModal();
	}
	// ---------------------------------------------------------------------------
	void __fastcall TForm1::N10Click(TObject *Sender)
	{
		MessageBox(this->Handle, (char*)("Данная программная система создана для автоматизации
расчета затрат на разработку/модернизацию программного обеспечения. Авторы: Шеенок Дмитрий
Александрович (СибГАУ)"), (char*)("Evaluation System Labor"), MB_OK + 64);
	}
	// ---------------------------------------------------------------------------
	void __fastcall TForm1::N6Click(TObject *Sender)
	{
		if (MessageBoxA(this->Handle, (char*)("Вы действительно хотите выйти из системы?"),
				(char*)("Evaluation System Labor"),
				MB_YESNO + 32) == IDNO) {

		}
		else {
			Close();
		}
	}
	// ---------------------------------------------------------------------------
```

```
void __fastcall TForm1::N20Click(TObject *Sender)
{
 Form3->ShowModal();
}
// ---------------------------------------------------------------------------
void __fastcall TForm1::SpeedButton2Click(TObject *Sender)
{
 Form4->ShowModal();
}
//---------------------------------------------------------------------------


void __fastcall TForm1::FormResize(TObject *Sender)
{
 DBGrid1->Height = Form1->Height - (375-223);
 DBGrid1->Width = Form1->Width - (593-568);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::SpeedButton3Click(TObject *Sender)
{
//проверка корректности данных на стороне БД

//расчеты затрат на разработку
 ADOTable1->Close();
 ADOTable1->TableName = "labor_develop";
 ADOTable1->Open();
 ADOTable1->Filtered = false;
 ADOTable1->Filter = "project_id="+TableProject->FieldByName("project_id")->AsAnsiString;
 ADOTable1->Filtered = true;
 ADOTable1->First();
 float labor_devel = 0;

 if (TableComponent->RecordCount==0) {Abort();}

 for (int i = 0; i < TableComponent->RecordCount; i++)
  {
   float component_version = ADOTable1->FieldByName("component_version")->AsFloat,
                        component_depth = ADOTable1->FieldByName("component_depth")->AsFloat,
                        component_labor = ADOTable1->FieldByName("component_labor")->AsFloat,
                        component_labor_algvote  =  ADOTable1->FieldByName("component_labor_algvote")-
>AsFloat;
```

```
if (component_version==1)
        {
        labor_devel = labor_devel +
                (component_depth*component_labor);
        }
else
        {
        labor_devel = labor_devel +
                (component_depth*component_labor) +
                (component_version-1)*component_labor + component_labor_algvote;
        }


 ADOTable1->Next();
 }
TableProject->Edit();
TableProject->FieldByName("project_labor_devel")->AsFloat = labor_devel;
TableProject->FieldByName("project_cost_devel")->AsFloat = TableProject-
>FieldByName("project_paydevelop")->AsFloat * labor_devel;
TableProject->Post();
//расчет затрат на другие этапы ЖЦ и весь проект

TableStage->First();
float project_labor = labor_devel,
project_cost = TableProject->FieldByName("project_paydevelop")->AsFloat * labor_devel;

for (int i = 0; i < TableComponent->RecordCount; i++)
 {
 TableStage->Edit();
 TableStage->FieldByName("stage_labor")->AsFloat =
                TableStage->FieldByName("stage_weight")->AsFloat *
                TableProject->FieldByName("project_labor_devel")->AsFloat;

 TableStage->FieldByName("stage_cost")->AsFloat =
                TableStage->FieldByName("stage_pay")->AsFloat *
                TableStage->FieldByName("stage_labor")->AsFloat;

 TableStage->Post();

 project_labor = project_labor + TableStage->FieldByName("stage_labor")->AsFloat;
```

```
 project_cost = project_cost + TableStage->FieldByName("stage_cost")->AsFloat;

 TableStage->Next();
 }

TableProject->Edit();
TableProject->FieldByName("project_labor")->AsFloat = project_labor;
TableProject->FieldByName("project_cost")->AsFloat = project_cost;
TableProject->Post();



}
//---------------------------------------------------------------------------
void __fastcall TForm1::N4Click(TObject *Sender)
{
//выгрузка в эксель сметы

 MessageBox(NULL,(char*)("Для выгрузки сметы необходимо сохранить и закрыть все документы Excel. Для продолжения нажмите ОК."),(char*)("Evaluation System Labor"),MB_OK+64);
 ShellExecute(Form1->Handle,NULL,"exitexcel.bat",NULL,NULL,SW_HIDE);

 MessageBox(NULL,(char*)("Несохраненные документы Excel будут закрыты принудительно. Для продолжения нажмите ОК."),(char*)("Evaluation System Labor"),MB_OK+64);
 ShellExecute(Form1->Handle,NULL,"killexcel.bat",NULL,NULL,SW_HIDE);

 if (SaveDialog1->Execute())
  {
        AnsiString exist_path = ExtractFilePath(Application->ExeName)+"\\estimate.xls";
        const char *ex_p = exist_path.c_str();
        AnsiString save_path = SaveDialog1->FileName+".xls";
        const char *sp = save_path.c_str();
        if (!CopyFile(ex_p,sp,1))
         {
          ShowMessage(GetLastError());
         }

 OleInitialize(NULL);
 iStartExcel(save_path.c_str());

 toExcelCell(2,1,TableProject->FieldByName("project_name")->AsString);
 toExcelCell(5,2,TableProject->FieldByName("project_labor_devel")->AsString);
```

```
toExcelCell(5,3,TableProject->FieldByName("project_cost_devel")->AsString);

int i=6;
TableStage->First();
for (int count=1; count < TableStage->RecordCount+1; count++)
 {
        toExcelCell(i,1,TableStage->FieldByName("stage_name")->AsString);
        toExcelCell(i,2,TableStage->FieldByName("stage_labor")->AsString);
        toExcelCell(i,3,TableStage->FieldByName("stage_cost")->AsString);
        TableStage->Next();
        i++;
 }

        toExcelCell(i,1,"Итого");
        toExcelCell(i,2,TableProject->FieldByName("project_labor")->AsString);
        toExcelCell(i,3,TableProject->FieldByName("project_cost")->AsString);

 Border("A6:C"+IntToStr(i),1);
}


}
//---------------------------------------------------------------------------
void __fastcall TForm1::N2Click(TObject *Sender)
{
 UnicodeString new_project_shortname = "новый проект", new_project_name = "описание нового проекта";
 if (InputQuery("Evaluation System Labor",
                "Введите наименование нового проекта:", new_project_shortname))
  {
        if (InputQuery("Evaluation System Labor",
                "Введите описание нового проекта:", new_project_name))
         {
                TableProject->Append();
                TableProject->FieldByName("project_shortname")->AsAnsiString = new_project_shortname;
                TableProject->FieldByName("project_name")->AsAnsiString = new_project_name;
                TableProject->Post();
         }
        }
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm1::N11Click(TObject *Sender)
{
AnsiString s = ExtractFileDir(Application->ExeName)+"\\manual.pdf";
 if (FileExists(s))
  {
   ShellExecuteA(NULL,"open",s.c_str(),NULL,NULL,SW_MAXIMIZE);
// вместо SW_SHOWNORMAL пишем SW_MAXIMIZE чтобы было развернуто на веьс экран
  }
 else
  {
   MessageBoxA(this->Handle,(char*)("Файл справки help.pdf в корневой папке программы не найден."),(char*)("ИС ЦТО Европринт"), MB_OK+16);
  }
}
//---------------------------------------------------------------------------
```

Unit2.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Unit2.h"
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm2::N2Click(TObject *Sender)
{
        if (MessageBoxA(this->Handle, (char*)(
                        "Вы действительно хотите удалить тип компонента? Все компоненты данного типа будут также удалены из всех проектов"),
                        (char*)("Evaluation System Labor"),
                        MB_YESNO + 32) == IDNO) {
```

```
		}
		else {
			Form1->TableComponentType->Delete();
		}
	}
	//---------------------------------------------------------------------------
	void __fastcall TForm2::N1Click(TObject *Sender)
	{
		UnicodeString new_type_name = "новый тип компонента";
		if (InputQuery("Evaluation System Labor",
			"Введите наименование нового типа компонентов:", new_type_name))
			{
		Form1->TableComponentType->Append();
		Form1->TableComponentType->FieldByName("component_type_name")->AsAnsiString = new_type_name;
		Form1->TableComponentType->FieldByName("component_labor")->AsAnsiString = "0";
		Form1->TableComponentType->FieldByName("component_labor_algvote")->AsAnsiString = "0";
		Form1->TableComponentType->Post();
	    }
	}
	//---------------------------------------------------------------------------
```

Unit3.cpp

```
	//---------------------------------------------------------------------------
	#include <vcl.h>
	#pragma hdrstop

	#include "Unit3.h"
	#include "Unit1.h"
	//---------------------------------------------------------------------------
	#pragma package(smart_init)
	#pragma resource "*.dfm"
	TForm3 *Form3;
	//---------------------------------------------------------------------------
	__fastcall TForm3::TForm3(TComponent* Owner)
		: TForm(Owner)
	{
	}
	//---------------------------------------------------------------------------
```

```
void __fastcall TForm3::N2Click(TObject *Sender)
{
        if (MessageBoxA(this->Handle, (char*)(
                                "Вы действительно хотите удалить этап ЖЦ из этого проекта?"),
                        (char*)("Evaluation System Labor"),
                        MB_YESNO + 32) == IDNO) {

        }
        else {
                Form1->TableStage->Delete();
        }
}
//---------------------------------------------------------------------------
void __fastcall TForm3::N1Click(TObject *Sender)
{
        UnicodeString new_stage_name = "новый этап ЖЦ";
        if (InputQuery("Evaluation System Labor",
                "Введите наименование нового этапа ЖЦ:", new_stage_name))
          {
        Form1->TableStage->Append();
        Form1->TableStage->FieldByName("stage_name")->AsAnsiString = new_stage_name;
        Form1->TableStage->FieldByName("project_id")->AsAnsiString = Form1->TableProject-
>FieldByName("project_id")->AsAnsiString;
        Form1->TableStage->FieldByName("stage_pay")->AsAnsiString = "0";
        Form1->TableStage->FieldByName("stage_weight")->AsAnsiString = "1";
        Form1->TableStage->Post();
      }
}
//---------------------------------------------------------------------------
void __fastcall TForm3::FormResize(TObject *Sender)
{
 DBGrid1->Height = Form3->Height - (326-264);
 DBGrid1->Width = Form3->Width - (600-569);
}
//---------------------------------------------------------------------------
void __fastcall TForm3::FormClose(TObject *Sender, TCloseAction &Action)
{
 if (Form1->TableProject->Modified == true )
  {
   Form1->TableProject->Edit();
   Form1->TableProject->Post();
```

```
  }
}
//---------------------------------------------------------------------------
```

Unit4.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "Unit4.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm4 *Form4;
//---------------------------------------------------------------------------
__fastcall TForm4::TForm4(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm4::FormResize(TObject *Sender)
{
 DBGrid1->Height = Form4->Height - (294-216);
 DBGrid1->Width = Form4->Width - (576-561);
}
//---------------------------------------------------------------------------
```

ПРИЛОЖЕНИЕ Б. Свидетельство и регистрации программ для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2017615181

Calculation of workspace

Правообладатель: ***Федеральное государственное автономное образовательное учреждение высшего образования «Сибирский федеральный университет» (СФУ) (RU)***

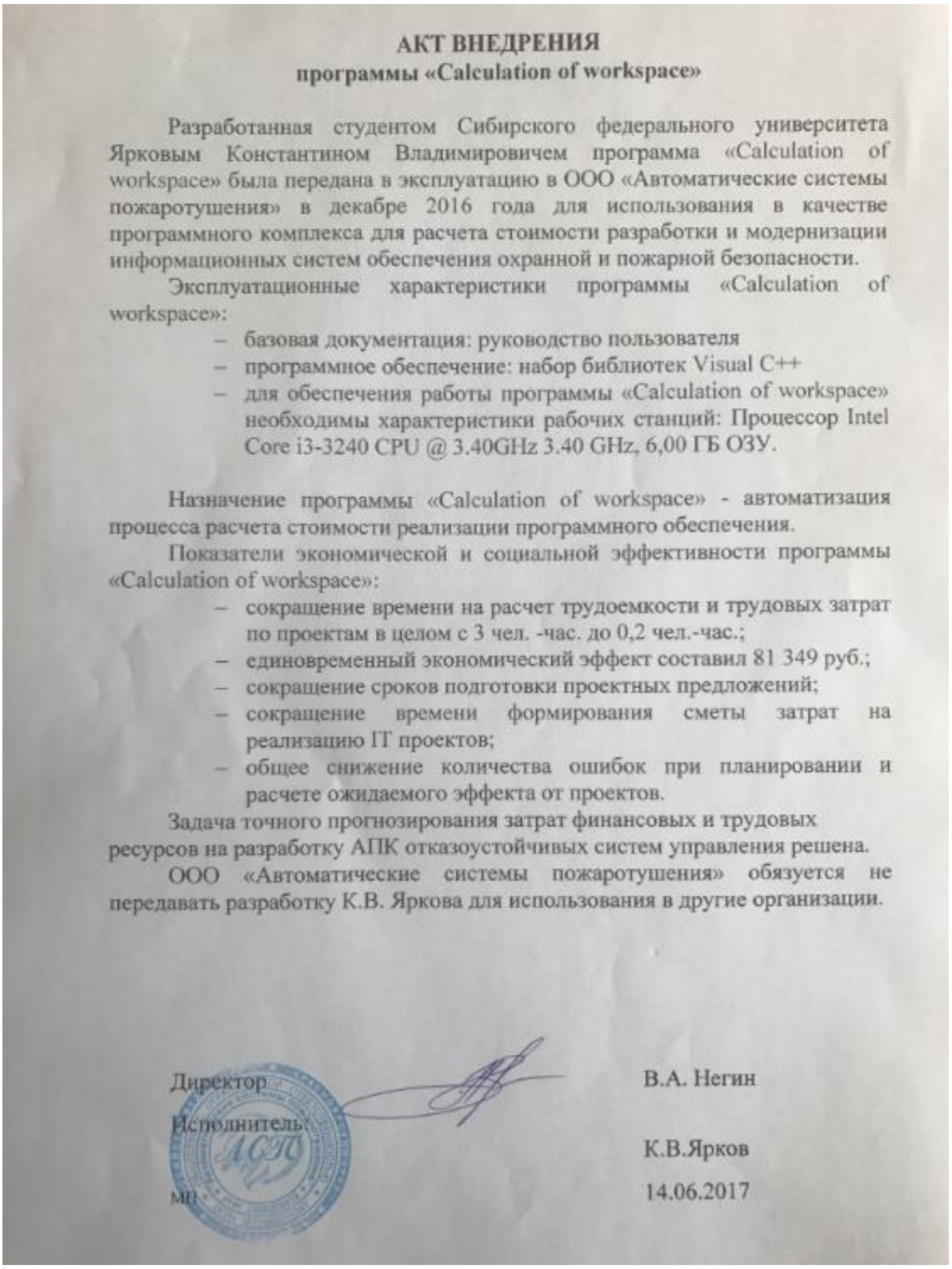
Авторы: ***Ярков Константин Владимирович (RU), Ступина Алена Александровна (RU), Кузьмич Роман Иванович (RU)***

Заявка № **2017612336**
Дата поступления **23 марта 2017 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***04 мая 2017 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

ПРИЛОЖЕНИЕ В. Акт внедрения результатов ВКР



АКТ ВНЕДРЕНИЯ
программы «Calculation of workspace»

Разработанная студентом Сибирского федерального университета Ярковым Константином Владимировичем программа «Calculation of workspace» была передана в эксплуатацию в ООО «Автоматические системы пожаротушения» в декабре 2016 года для использования в качестве программного комплекса для расчета стоимости разработки и модернизации информационных систем обеспечения охранной и пожарной безопасности.

Эксплуатационные характеристики программы «Calculation of workspace»:

- базовая документация: руководство пользователя
- программное обеспечение: набор библиотек Visual C++
- для обеспечения работы программы «Calculation of workspace» необходимы характеристики рабочих станций: Процессор Intel Core i3-3240 CPU @ 3.40GHz 3.40 GHz, 6,00 ГБ ОЗУ.

Назначение программы «Calculation of workspace» - автоматизация процесса расчета стоимости реализации программного обеспечения.

Показатели экономической и социальной эффективности программы «Calculation of workspace»:

- сокращение времени на расчет трудоемкости и трудовых затрат по проектам в целом с 3 чел. -час. до 0,2 чел.-час.;
- единовременный экономический эффект составил 81 349 руб.;
- сокращение сроков подготовки проектных предложений;
- сокращение времени формирования сметы затрат на реализацию IT проектов;
- общее снижение количества ошибок при планировании и расчете ожидаемого эффекта от проектов.

Задача точного прогнозирования затрат финансовых и трудовых ресурсов на разработку АПК отказоустойчивых систем управления решена.

ООО «Автоматические системы пожаротушения» обязуется не передавать разработку К.В. Яркова для использования в другие организации.

Директор В.А. Негин

Исполнитель К.В.Ярков

МП 14.06.2017