На правах рукописи

ГУМЕРОВ Максим Маратович

Проектирование обратимых операций над объектами на основе шаблонов при разработке программного обеспечения

Специальность: 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ

диссертации на соискание учёной степени

кандидата технических наук

Уфа 2011

Работа выполнена
на кафедре вычислительной математики и кибернетики
Уфимского государственного авиационного технического университета

Научный руководитель	канд. техн. наук, доцент **Фридлянд Александр Михайлович** каф. вычислительной математики и кибернетики
Официальные оппоненты	д-р техн. наук, проф. **Мартынов Виталий Владимирович** каф. экономической информатики Уфимского государственного авиационного технического университета
	канд. техн. наук, доцент **Торшин Дмитрий Вячеславович** ЗАО «Университетский кластер» г. Москва
Ведущая организация	ФГБОУ ВПО «Уфимский государственный нефтяной технический университет»

Защита диссертации состоится «25» ноября 2011 г. в __10__ часов
на заседании диссертационного совета Д-212.288.07
при Уфимском государственном авиационном техническом университете
по адресу: 450000, Уфа, ул. К. Маркса 12

С диссертацией можно ознакомиться в библиотеке университета

Автореферат разослан «24» октября 2011 г.

Ученый секретарь
диссертационного совета
д-р техн. наук, проф. **С.С. Валеев**

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы

Установлено, что поиск и формализация шаблонных проектных решений при разработке программного обеспечения (ПО) на этапе проектирования позволяет сократить количество неизбежных ошибок, допускаемых при проектировании, и сократить его длительность.

В работе исследована задача проектирования приложений для конечного пользователя, содержащих функции редактирования различного рода документов, состоящих из связанных простейших элементов, или примитивов.

Связанные примитивы содержатся в разнообразных видах гипертекстовых документов, или схем, или векторной графики, или моделей, – применяемые в различных областях – от представления Web-страниц и редактирования компьютерной графики до наполнения систем поддержки принятия решений.

Как и в любых приложениях, подразумевающих существенный объем операций редактирования данных пользователем, в рассматриваемом классе программных продуктов необходимым условием эффективной работы пользователя является возможность отмены выполненных операций. Функция отмены операций реализована во многих программных продуктах; известны и научные исследования, посвященные различным сторонам отмены операций.

Опубликованы результаты исследований в трех основных направлениях – отмена транзакций в СУБД, в многопользовательских приложениях и в классических однопользовательских. Исследуются, как правило, общая архитектура и теоретические основы отмены операций, но не отмена конкретных классов действий пользователя.

К недостаткам существующих решений следует отнести в первую очередь то, что предлагаемые решения часто инженерные: они разрабатываются для конкретного приложения и подходят именно для него, в результате чего в каждом новом проекте к проектированию отмены операций нужно приступать заново, и затраты времени на это могут составить 20-25% от общего времени проектирования редактора. Именно поэтому в данной работе акцент делается на постановку научной задачи и поиск ее общего решения, подходящего для использования в различных приложениях.

Цель работы. Целью диссертационной работы является повышение эффективности разработки приложений-редакторов на основе разработки шаблона обратимой операции удаления для создания механизма отмены операций в этих приложениях.

Задачи исследования

Для достижения поставленной цели требуется решить следующие задачи:

1) Разработка формальной модели операции удаления примитива.

2) Разработка метода проектирования обратимой операции удаления примитива (метод формулируется в виде шаблона проектирования). Разработка алгоритмического обеспечения для созданного шаблона.

3) Разработка модели качества программного обеспечения, подходящей для обобщенных повторно используемых фрагментов архитектуры.

4) Разработка метода оценки качества внедрения предложенного подхода в программных продуктах.

5) Реализация предложенного шаблона в виде новых фрагментов программ и изменений существующего кода. Анализ эффективности шаблона с помощью предложенной модели качества.

Методы исследования

В основе исследования лежит подход к проектированию приложений, основанный на выделении и повторном использовании шаблонов проектирования.

При моделировании операции удаления примитива применялось объектно-ориентированное проектирование и язык моделирования UML. Для формализации последовательности действий оператора по осуществлению тестирования также применялся UML в сочетании с методологией нисходящего проектирования.

Процесс разработки и практической реализации предлагаемых шаблонов следовал методологии экстремального программирования. Контроль качества проектного решения проводился при помощи метода, предлагаемого моделью качества SQuaRE.

Основные научные результаты, выносимые на защиту:

1. Формальная модель операции удаления примитива редактирования в приложении-редакторе, позволяющая идентифицировать нетривиальные составляющие этой операции с точки зрения обеспечения ее обратимости – в виде статической структурной диаграммы UML.

2. Метод проектирования обратимой операции удаления примитива редактирования. Метод представлен шаблоном проектирования, включающим формальную модель и алгоритмы работы ее составляющих, в том числе: алгоритм поддержания посредника в рабочем состоянии при удалении указуемого объекта, алгоритм выполнения удаления объекта с запоминанием его связей, алгоритм отмены удаления с восстановлением связей.

3. Обобщение модели качества SQuaRE, пригодное для случая обобщенных повторно используемых фрагментов архитектуры.

4. Метод оценки качества разрабатываемого ПО, основанный на интеграционном тестировании и дополняющий его приемами модульного тестирования с целью ускорения покрытия приложения тестами.

5. Модули поддержки отмены выполняемых операций в программных продуктах «BNView» и «NGT Smart». Результаты анализа эффективности работы разработанного и внедренного программного модуля поддержки отмены выполняемых операций в рамках.

Научная новизна результатов

1. Новизна предложенной формальной модели операции удаления примитива состоит в том, что она позволяет выделить составляющие состояния приложения, которые необходимо учитывать при восстановлении состояния приложения в ходе отмены осуществленной операции – тем самым, модель позволяет разбить задачу разработки шаблона на связанные подзадачи, что *позволяет*, в свою очередь, разрабатывать различные архитектурные решения задачи обратимой операции удаления.

2. Новизна разработанного метода проектирования обратимой операции удаления,– на основе построенной формальной модели операции,– заключается в том, что, *в отличие от известных* шаблонов, учитываются связи между примитивами как часть состояния приложения, подлежащего восстановлению при отмене операции. Шаблон *позволяет* при разработке конкретных программных продуктов сокращать и повышать качество этапа проектирования за счет использования обобщенного и эффективного проектного решения. В качестве составных частей в шаблоне используются несколько шаблонов общего назначения

3. Новизна построенной *специализации существующего* шаблона «посредник» состоит в поддержании посредника в рабочем состоянии при удалении идентифицируемого им объекта. Этот вариант шаблона *может использоваться* как в обратимой операции удаления, так и для решения других задач – например, поддержки перемещения программного кода между компьютерами (code mobility).

4. Новизна предложенного обобщения модели качества SQuaRE состоит в том, что, *в отличие от известных моделей,* это обобщение позволяет применять критерии модели SQuaRE к анализу обобщенных повторно используемых фрагментов архитектуры. Конечной целью было проведение информативного анализа качества разработанного шаблона.

5. Новизна разработанного метода тестирования программного обеспечения, *основанного* на интеграционном тестировании, состоит в широком использовании удачных практик модульного тестирования, что *позволяет* при проектировании тестовых каркасов достичь более эффективной динамики покрытия функций приложения во времени. Метод разрабатывался для оценки качества программных систем, созданных на основе предложенного проектного решения.

Практическая значимость

– Предложенный метод оценки качества *может использоваться* для анализа качества программного обеспечения и поиска ошибок как в процессе проектирования ПО, так и в процессе сопровождения.

– Критерии и метрики предложенной модели качества, *применимой* для анализа обобщенных архитектурных решений, *могут использоваться* для сравнения различных архитектур с целью выбора наиболее подходящей, а также для выявления потенциальных проблем анализируемой архитектуры.

− Программные модули для ПП «NGT Smart» и «BNView», написанные с применением предложенного подхода к проектированию команды удаления примитива.

Внедрение результатов, полученных в работе, осуществлено в ООО «Уфимский научно-технический центр». Разработанное в процессе проведения экспериментальной части исследования программное обеспечение (комплексы «NGT Smart» и «BNView») зарегистрировано в реестре программ для ЭВМ Российского агентства по патентам и товарным знакам (свидетельства N 2010614274 и N 2011613115). Внедрение осуществлено в ОАО «НК Башнефть» и ООО «Газпромнефть-Хантос».

Связь исследования с научными программами

Исследования проводились в рамках НИР НК «Роснефть» 1980908-0067Д «Создание методологии информационного обеспечения интегрированного проектирования», а также грантов РФФИ №08-07-00495-а «Технологии распределённого искусственного интеллекта при поддержке принятия решений в задачах календарного планирования» (2008-2010 гг.) и № 06-07-89228-а «Система поддержки коммуникативных процессов при выполнении проектов фундаментальных исследований сложных систем на основе интеллектуальных мультиагентов» (2006-2008 гг.).

Апробация и публикации

Основные результаты диссертации докладывались на следующих конференциях и семинарах:

1. VII Международная научно-техническая конференция «Новые информационные технологии и системы» (Пенза, 2006).
2. The 9th International Workshop on Computer Science and Information Technologies CSIT’2007 (Уфа, 2007).
3. X Международная научно-практическая конференция «Наука и современность – 2011» (Новосибирск, 2011).

Список публикаций по теме диссертации содержит 8 работ, в том числе 2 статьи в рецензируемых журналах из списка ВАК и 2 программы для ЭВМ.

Благодарности

Автор выражает благодарность главному специалисту управления развития банковских систем ОАО «БАНК УРАЛСИБ» Кузьмину Илье Владимировичу за ценные замечания, а также начальнику отдела развития систем проектирования ООО «РН-УфаНИПИнефть» Якупову Рустему Назировичу за консультации.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Во введении к диссертации обрисовывается круг рассматриваемых проблем и обосновывается их актуальность, формулируются цель и задачи исследования, перечисляются подходы и методы решения задач, характеризуются результаты, выносимые на защиту, отмечается их научная новизна и практическая значимость. Приводятся сведения о внедрении результатов и публикациях.

В первой главе приведен анализ процесса проектирования программного обеспечения, эффективных подходов к ускорению и повышению

качества проектирования, выделен класс приложений, находящихся в центре внимания данной работы. Проанализированы актуальные для этого класса задачи проектирования. Отмечено слабо исследованное направление в области отмены операций, приведена содержательная постановка задачи проектирования операции удаления примитива редактирования, показана невозможность использования существующих научных разработок в качестве готовых шаблонных решений задачи. Проанализирована современная широко распространенная модель качества ПО SQuaRE, показана ее неприменимость для оценки качества обобщенных архитектурных решений. Рассмотрены разнообразные подходы к автоматическому тестированию приложений и сформулирован набор требований к эффективному методу разработки каркаса тестовых методов.

Предметом настоящего исследования является обращение операции удаления, применяемой к примитиву редактируемого документа. Выбор именно этой операции обусловлен той важной особенностью операции удаления, что от способа ее работы зависит возможность обращения уже выполненных до нее операций, как-либо затрагивающих удаляемый примитив (см. рисунок 1).

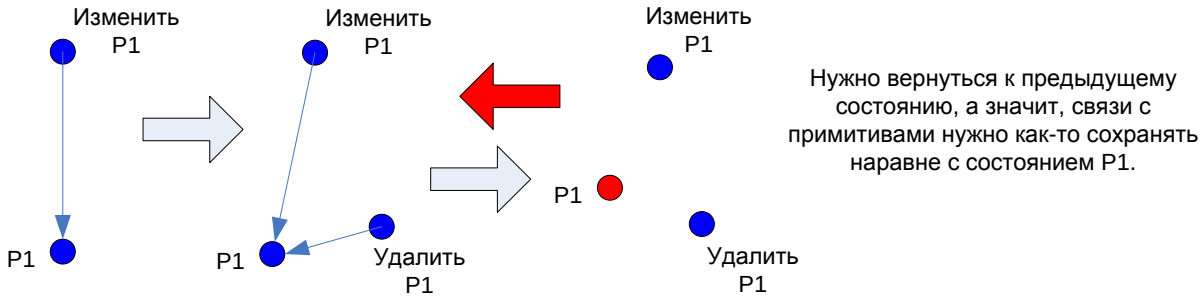

Рисунок 1 – Проблема, возникающая при удалении примитива.

Каждый новый коллектив разработчиков в новом для себя проекте приложения такого рода решает эту задачу проектирования самостоятельно. На проектирование системы отмены операций приходится около 20-25% всего этапа проектирования той части приложения, которая непосредственно обеспечивает отображение и редактирование документа. Чтобы сэкономить это время, предлагается воспользоваться парадигмой шаблонов проектирования (design patterns) и поставить задачу разработки шаблона проектирования для таких операций π и π^{-1}. Шаблонами называют специальным образом оформленные обобщенные проектные решения; шаблон именует, абстрагирует и идентифицирует ключевые аспекты структуры решения, обеспечивая таким образом повторное применение этого решения в различных проектах.

Такой шаблон проектирования определяется моделью фрагмента архитектуры программного продукта, определяющей взаимодействующие сущности и их отношения, и обобщенным алгоритмом их взаимодействия. Алгоритм должен определять по текущему состоянию документа D’ и последней выполненной операции π – являющейся операцией удаления

Delete(Pi) – исходное состояние D, в котором документ находился перед выполнением π.

Поскольку исследование посвящено разработке шаблона проектирования, для оценки результата необходимо выбрать модель качества. Шаблоны представляют собой не готовую программу и даже не конкретный алгоритм, а типовую архитектуру. Из-за этого многие разработанные на сегодняшний день метрики оценки качества программного обеспечения к ним неприменимы либо же недостаточно информативны.

Ситуация такова, что среди широко применяемых моделей качества нет такой, которая бы без каких-либо изменений подходила для этой задачи; есть лишь отдельные пригодные составляющие различных моделей. В результате, как правило, и приводимый авторами отдельных шаблонов и их доработок анализ качества их решений – односторонен.

Ввиду отсутствия моделей, пригодных для оценки качества обобщенных решений, для сколь-нибудь полного анализа качества решения, предлагаемого в рамках настоящей работы, необходимо также разработать подходящую модель качества.

Помимо этапа проектирования, в разработке присутствует и этап тестирования. В настоящей работе исследуется, как часть работы по ускорению разработки ПО, возможность усовершенствования одного из подходов к автоматическому тестированию – функциональному или регрессионному – так называемого интеграционного тестирования, состоящего, по наиболее общему определению, в вынесении заключения о наличии или отсутствии ошибок в программе по ее функционированию в целом, а не по проверке ее элементарных частей (что характерно для модульного тестирования). Предлагается оформить каркас интеграционных тестов так же, как обычно оформляются тесты модульные.

В рамках данного исследования был разработан метод, развивающий идеи, изложенные в книге M. Feathers, посвященной вопросам, связанным со спецификой уже существующего крупного программного проекта. Рассматривается проблема разработки метода создания каркаса тестов, который: 1) изначально был бы в состоянии выявлять ошибки времени выполнения, 2) мог бы быть быстро реализован в первом приближении и в таком состоянии уже позволял выявлять многие ошибки, 3) мог бы быть по частям, без нарушения его работоспособности, усовершенствован до желаемого уровня детальности и изолированности тестов, вплоть до постепенного перехода к модульному тестированию.

Во второй главе приведена формальная постановка задачи, предложен подход к проектированию, формальные модели и методы решения задач. Приводится алгоритм работы спроектированной операции.

Редактируемый документ состоит из примитивов и связей между ними, каждый из примитивов имеет набор неких атрибутов, которые могут представлять собой целочисленные, текстовые или какие-либо еще пометки.

Тогда документ D = <P, A, R>, где

– P – множество примитивов (в зависимости от приложения – символов, слоев, микросхем, разделов, правил, ...)

– R – множество пар $<P_i, P_j>$, определяющих связь между примитивами P_i и P_j.

– A – множество пометок-атрибутов, которыми помечаются примитивы (например: размер, цвет, текст); A есть множество пар $<P_i, a_i>$, где a_i – вектор атрибутов примитива P_i.

– Таким образом, документ есть граф $<P, R>$ с вершинами, маркированными значениями атрибутов примитивов.

Здесь вершины P_i представлены объектами, описывающими соответствующие примитивы документа P_i; вершины P_i и P_j инцидентны, если $PiPj \in R$. Видно, что атрибуты вершин меняются во времени, как меняется и сам набор вершин. Моменты времени можно отсчитывать по числу выполненных команд. Момент t наступает после операции Cm_t.

Cm_i соответствуют последовательности выполненных в сеансе работы команд, формирующих документ; команда Cm_i выполняется i-ой по счету. На рисунке показано состояние документа после 3-ей команды, когда Cm_4 еще не выполнена. Согласно шаблону Undo/Redo, у документа в текущий момент времени t есть предистория (очередь Undo) и будущее (очередь Redo). Множество команд, выполненных не раньше момента времени t, составляет очередь Undo, прочие – очередь Redo.

В конкретном сеансе работы в современных объектно-ориентированных приложениях документ представляется в памяти, как правило, графом G(t) следующего вида, показанного на рисунке 2, слева.

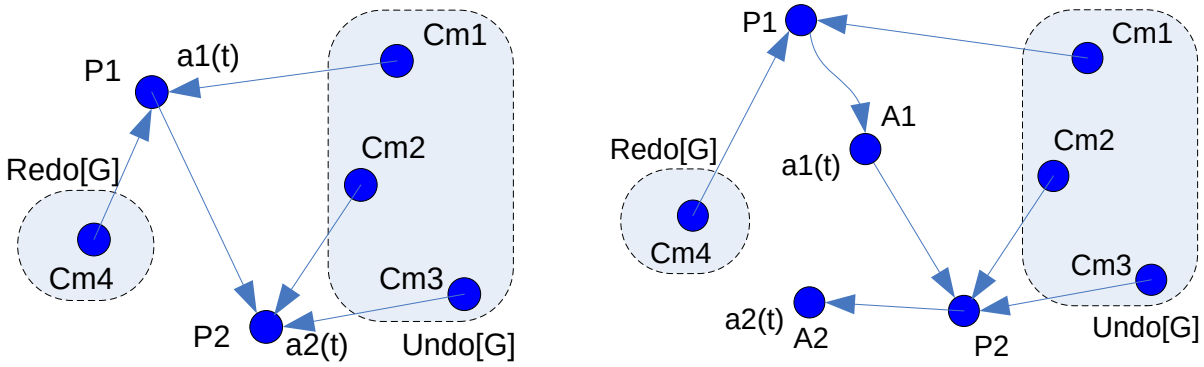

Рисунок 2 – Граф G(t): слева – простой вариант, справа – отражающий различимость Pi только благодаря их атрибутам

В G(t) $\{Cm_i\}$= Undo[G]∪Redo[G], где Undo(G)=$\{Cm_i: i \le t\}$. Так, в графе на рисунке мощность Undo[G(t)] равна 3, из чего видно, что t=3.При перемещении на один шаг в прошлое граф G(t) превращается в граф G(t-1), в котором команда Cmt, бывшая в Undo[G(t-1)], переходит в Redo[G(t)]. Вопрос в том, какой индекс тогда должна получить следующая выполняемая команда после того, как сделан возврат в прошлое. Согласно шаблону Undo/Redo, после возврата в прошлое (Undo), в граф G(t-1), возможны:

– либо шаг обратно в будущую историю документа (Redo), т.е. к графу G(t), в результате чего новых команд в документе не появится, Cm_t выполнится снова, а ее вершина в графе перейдет из Redo в Undo;

– либо выполнение какой-то новой команды, которой в графе еще нет; в этом случае вся будущая история документа (все команды множества Redo) уничтожается, тем самым уничтожается, как ее часть, вершина Cm_t, а значит, команду, с которой теперь начинается момент времени t, в новом графе G(t) можно обозначить тем же освободившимся теперь индексом t.

Таким образом, постоянно выполняется соглашение по именованию Cm_t вершины, с которой начинается момент времени t.

Важно, что сами по себе, без учета их атрибутов и связей, примитивы неразличимы: поменяв в графе местами вершины P_i и P_j, и перенаправив ребра, входящие в P_i, так, чтобы они вели в P_j, и наоборот (и то же самое для исходящих ребер), а также обменяв местами атрибуты a_i и a_j, можно получить граф, который будет с точки зрения содержимого документа идентичен исходному. Для пользователя ничего не изменится, если заменить в документе какой-то примитив на «другой точно такой же» - с теми же атрибутами и связями.

С учетом этого, можно предложить изменить представление таким образом, чтобы было удобнее эквивалентность графов документа определять только атрибутами и связями между их носителями. Для этого ту «часть» примитива P_i, которая различима и отпределяется атрибутами, можно вынести в отдельную вершину A_i, в которую исходит ребро из P_i. Атрибутная пометка в таком случае перемещается на вершину A_i, а исходящие ребра P_iV становятся ребрами A_iV. Суть изменения в том, что можно ввести в граф и какую-либо другую вершину P_j (тоже с ребром в A_i), а P_i убрать; или «поменять местами» P_i и P_j, – и при этом будет видно, что сохраняется достижимость A_i из других A_k или Cm_k.

Пусть R[G] обозначает разновидность редукции графа G, выполняющейся в два этапа:

1) для каждого пути $v_1,\ldots,v_n$ такого, что $(v_k \in \{A_i\} \cup \{Cm_j\}) \Leftrightarrow (k \in \{1,n\})$, в граф добавляется ребро v_1v_n;

2) из графа удаляются все вершины, кроме $\{A_i\} \cup \{Cm_j\}$, вместе со всеми инцидентными ребрами (то есть удаляются все неразличимые вершины, а также, возможно, какие-то вспомогательные).

С точки зрения пользователя содержимое документа описывается состояниями (атрибутов), приписанными неразличимым примитивам P_i, и связями между P_i, а в редуцированном графе всегда можно воспринимать A_i как P_i (сохранив при этом пометки), – перейдя обратно от представления в правой части рисунка 2 к представлению в левой части, – значит, содержимое документа описывается графом R[G], а G может дополнительно содержать вспомогательную информацию для представления документа в памяти.

Можно определить отношение эквивалентности графов документа G и G’ через изоморфизм их редуцированных графов: G~G’ iff R[G]=R[G’]. Само удаление P_i тогда определяется как отображение π, переводящее G в G':

1) $V[R[G']] = (V[R[G]] \setminus P_i) \cup Cm_k$, где Cm_k – операция удаления.

2) $E[R[G']] \setminus \{v_1v_2: v_1=Cm_k$ или $v_2=Cm_k) = E[R[G]] \setminus \{v_1v_2: v_1=P_i$ или $v_2=P_i)$, то есть из G удаляются ребра, инцидентные P_i, но могут добавляться какие-либо ребра, инцидентные Cm_k.

Существенно, что эти ограничения накладываются на уже редуцированный граф, а значит, сами G и G' могут отличаться какими-либо вспомогательными вершинами и ребрами.

Требование к обратимости операции удаления означает, что для π необходимо предложить обратное отображение π^{-1}, переводящее G' в G'' так, что $R[G'']\setminus Cm_k=R[G]$. Удаление вершины Cm_k включено из-за того, что для повторного выполнения (Redo) операции Cm_k она должна остаться в графе после своей отмены (в множестве Redo[G'']). А если переход от G к G' был первым выполнением Cm_k, то в G вершины Cm_k еще не было, и в этом простое равенство R[G'']=R[G] не выполнилось бы.

Задача исследования: предложить типовое проектное решение для проектирования операций π и π^{-1}, обладающих перечисленными свойствами.

Проведенное автором исследование составляющих операции удаления, промежуточные результаты которого были опубликованы в материалах конференции НИТиС, позволило получить модель, представленную в виде статической структурной диаграммы UML на рисунке 3.

Одна из составляющих операции удаления согласно этой модели – сохранение состояния удаляемого объекта для последующего восстановления. Наиболее сложно обеспечить сохранение одной из частей состояния объекта: ссылок данного объекта на другие или других на данный, при возможности того, что целевой объект был удален и затем восстановлен.

Связи между примитивами в графе 2 могут отображаться в архитектуре приложения различными способами. В простейшем случае (и так обычно и бывает, если имеется уже частично завершенный программный продукт) эти ссылки представлены указателями; возможны и другие варианты – индексы в каких-либо массивах, алгоритмы поиска объекта и др.

В настоящем исследовании предлагается ввести дополнительные объекты-посредники, в которые переместится необходимая прослойка косвенности. Это разновидность идентификации объекта по индексу вместо указателя, в первоначальном варианте предложенная автором на конференции CSIT'07. Ссылающийся объект вместо ссылки для указания на целевой объект должен использовать ссылку на посредника. Для этого каждая ссылка одного примитива (A_i) на другой (A_j) заменяется ссылкой A_i на уникальный объект-посредник T_k, который уже в свою очередь имеет ссылку на P_j. Результат показан на рисунке 4.

Предлагается следующий алгоритм работы посредников. Посредник получает способность реагировать на удаление и восстановление объектов; при удалении объекта, на который он указывает, он автоматически переходит в особое пассивное состояние, при восстановлении возвращается в активное (в котором у него можно опрашивать ссылку), одновременно получая от механизма отмены операций уведомление о новом адресе, по которому

размещается объект после восстановления. Использование посредников такого вида позволяет предложить шаблон обратимой операции удаления, структурная диаграмма которого приведена на рисунке 5.

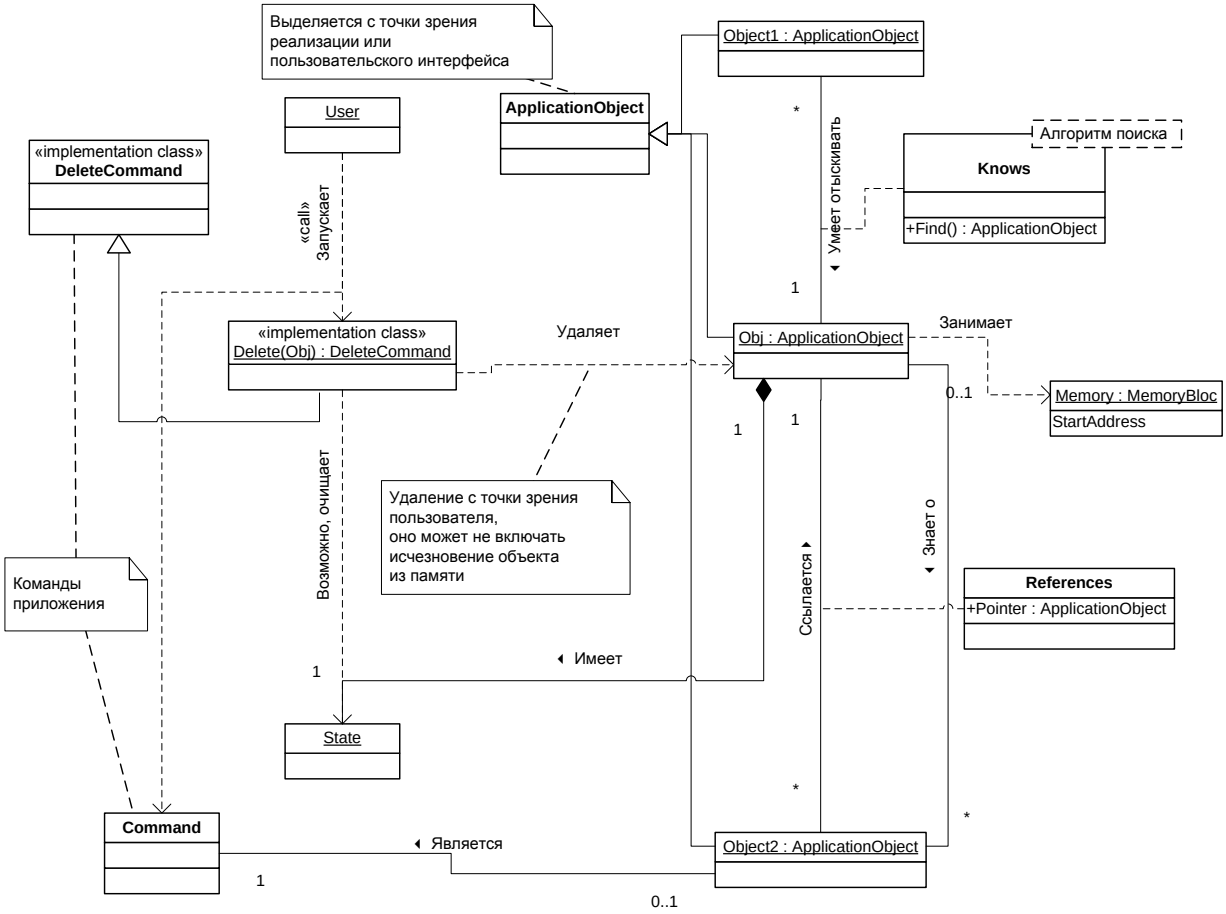

Рисунок 3 – Принципиальная модель операции удаления объекта

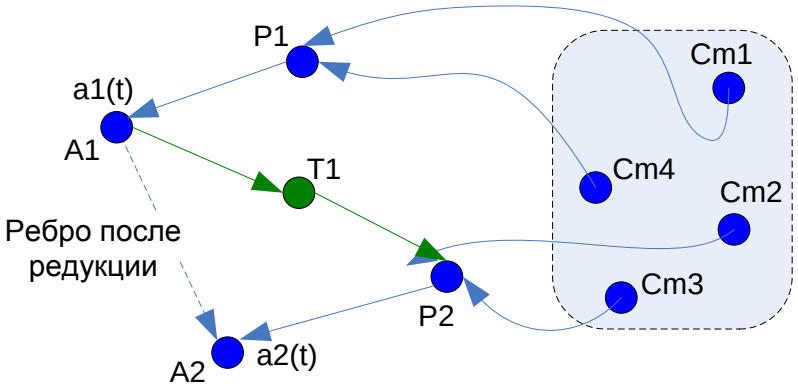

Рисунок 4 – Граф G(t), расширенный посредниками

Алгоритм работы обратимой операции удаления соблюдает инвариант, согласно которому объектам, в настоящее время адресуемым хотя бы одним посредникам, биективно сопоставляются ключи. Сопоставленный объекту ключ не меняется, пока на объект ссылается хотя бы один посредник.

С учетом показанного выше способа обработки посредником сообщений $Del(K_i)$ и $Undel(K_i, P_i)$, возможный алгоритм работы

разработанного шаблона может быть представлен диаграммой последовательности, показанной на рисунке 6.

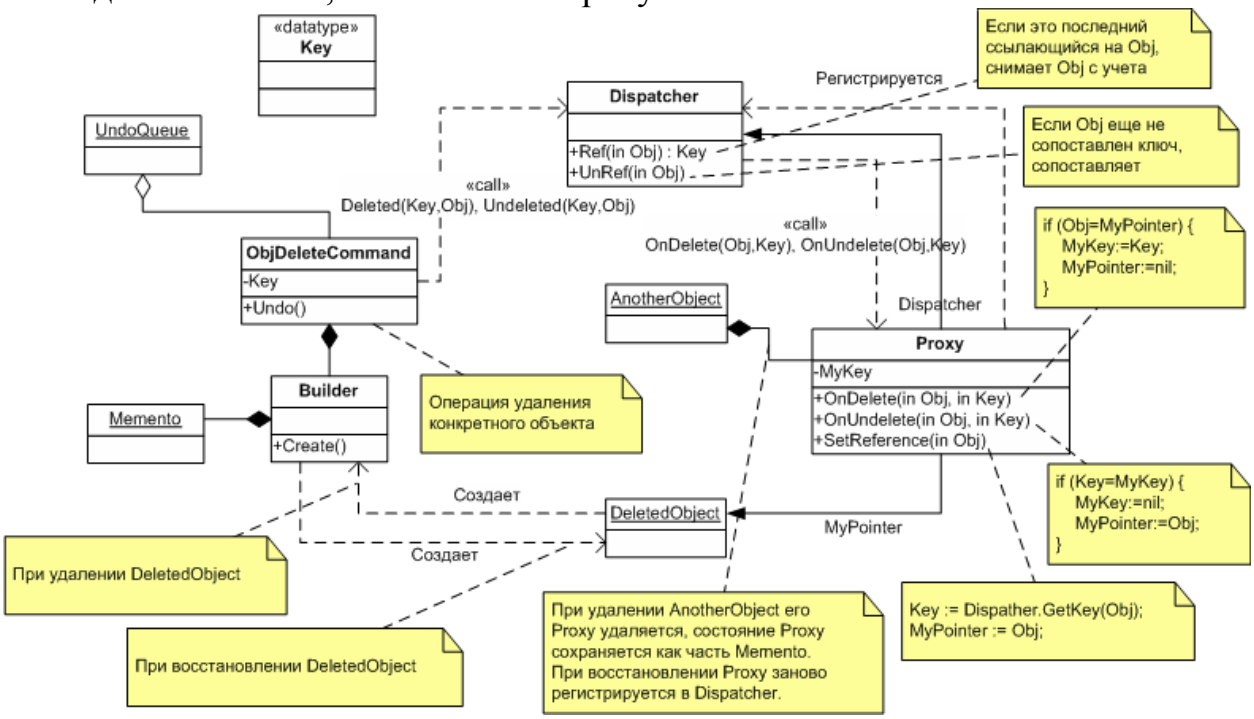

Рисунок 5 – Шаблон обратимой операции удаления

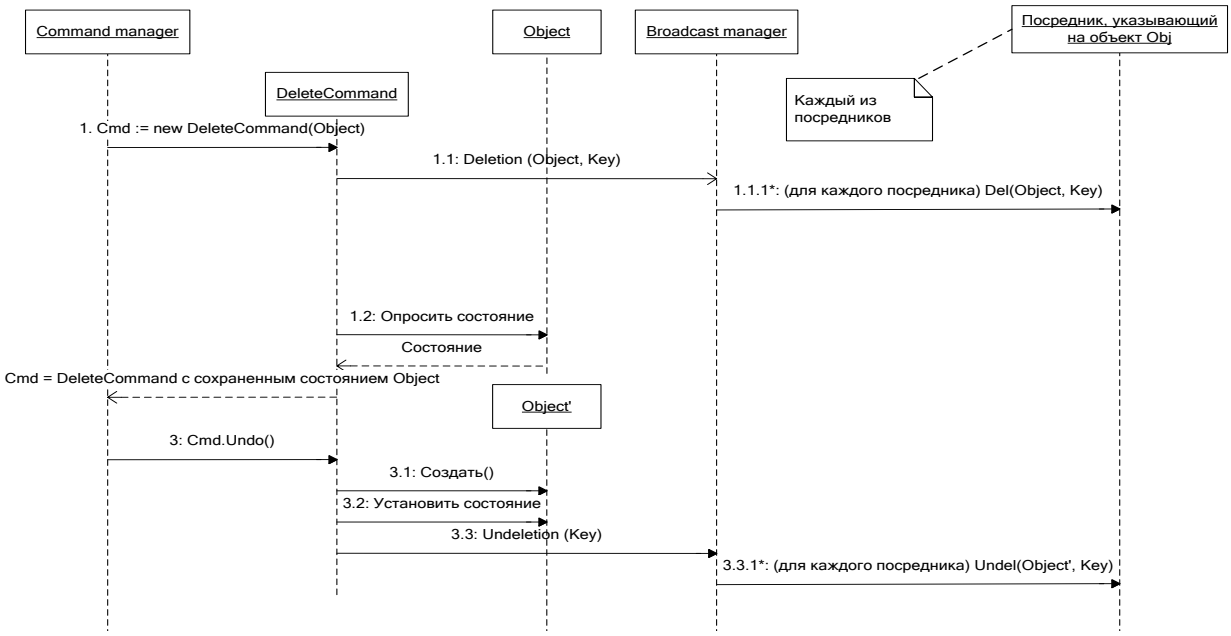

Рисунок 6 – Алгоритм работы обратимой операции удаления

Третья глава посвящена разработке модели качества программного обеспечения. За основу взяты виды качества и подходы к его оценке в распространенной модели качества SQuaRE, проанализированной ранее в первой главе. Также в данной главе предлагается метод проектирования набора тестовых методов, нацеленный на удовлетворение требований к методу тестирования, сформулированных в первой главе.

Предлагаемые изменения относятся к критериям качества, свойственным шести характеристикам качества, обозначенным в SQuaRE.

Среди критериев переносимости, с некоторыми оговорками, для оценки архитетуры подходят почти все описанные в SQuaRE критерии и метрики. Две метрики из SQuaRE включены в предлагаемую модель с изменениями трактовки. Это Continued Use Of Data и Functional inclusiveness.

Существенные изменения касаются критериев эффективности по ресурсам и времени, и критериев трудоемкости разработки и сопровождения.

Метрики поведения во времени, включенные в SQuaRE, оказываются пригодны для анализа обобщенных архитектурных решений, если позволить выражать результат в виде оценки порядка сложности (а не конкретных численных значений). В предлагаемой модели принята общеизвестная асимптотически точная оценка сложности Θ: g(n) является асимптотически точной оценкой f(n), если при g>0 при n>0 существуют положительные c_1, c_2, n_0 такие, что при $n>n_0$ выполняется двойное неравенство $c_1 g(n) \le f(n) \le c_2 g(n)$. Записывается Θ-оценка так: f(n) = Θ(g(n)). Метрики объема используемых ресурсов можно также позаимствовать из SQuaRE без каких-либо доработок, разрешив указывать значения объема ресурсов как Θ-оценку.

В качестве метрик трудоемкости разработки и сопровождения предлагаемая модель предлагает использовать метрики качества объектно-ориентированной архитектуры, отобранные из различных источников по степени их применимости в условиях обобщенных решений.

Помимо оценки качества обобщенного решения, необходимо проводить тестирование разрабатываемых конкретных программных продуктов, и это тоже часть процесса разработки, ускорению которого посвящена данная работа. Существует множество способов тестирования качества ПО, которым посвящена не одна книга – так, К. Бек описал разработанную им парадигму разработки через тестирование (TDD), и С. Канер, Дж. Фолк и Е. К. Нгуен в своей книге затронули множество подходов.

В рамках данного исследования разработана методика, применяющая к регрессионному тестированию идеи, изложенные в книге M. Feathers, посвященной тестированию уже существующего программного проекта; автор книги относил эти идеи в первую очередь не к интеграционному, а к модульному тестированию. Поскольку интеграционное тестирование предполагает тестирование целого без уверенности в работе частей, в качестве основного типа тестов предлагается использовать не эталонное тестирование, а регрессионное характеризующее (сравнение результатов запуска функции и предыдущего ее запуска при тех же исходных данных).

Еще одна рационализаторская идея состоит в широком использовании косвенного тестирования. Тестируемая функция может не иметь обособленного программного интерфейса для своего вызова – например, может быть активирована только через интерфейс пользователя, но не программно. Рекомендуемый выход состоит в косвенной активации данной функции посредством запуска другой, включающей данную.

Четвертая глава описывает опыт внедрения разработанных методов в коммерческих программных продуктах. Также в ней приведен анализ эффективности предложенного шаблона проектирования при помощи

разработанной модели качества, а также описан опыт внедрения метода тестирования ПО.

Разработанный шаблон был внедрен в программных комплексах «NGT Smart» и «BNView». Эти программы содержат, среди прочих составляющих, редактор карт нефтяного месторождения.

На момент начала внедрения операция удаления в редакторе уже была отменяемой, но отмена была неполной: удаленные примитивы (например, каналы коммуникаций, узлы соединения или слои изображения) восстанавливались, но информация об их связи при удалении терялась. Например, после удаления примитива «узел соединения» и последующей отмены удаления каналы коммуникаций, ранее подсоединенных к примитиву, оказывались лежащими в прежних координатах, но редактор карт считал их несвязанными объектами. При просмотре информации об узле больше не выводилась информация об этих каналах, а при перемещении пользователем узла на карте (речь идет о проектировании обустройства месторождения) каналы не тянулись вслед за узлом, как должно было быть.

Эта ситуация следовала из существовавшей архитектуры операции отмены, показанной на рисунке 7. Такая структура соответствует принятым представлениям о роли шаблонов Command и Memento в обеспечении обратимости операций вообще, хотя и не учитывает специфику операции удаления примитивов при наличии связей между последними.

В силу такого соответствия, она хорошо вписывается в предлагаемый шаблон проектирования, также базирующийся на принятом подходе к проектированию обратимых операций, что очевидно из сравнения рисунков 7 и 5. Для внедрения шаблона в такой ситуации необходимо было лишь заменить прямые ссылки AnotherObject на DeletedObject на использование посредников, а также реализовать класс посредника и систему регистрации посредников и их уведомления – согласно шаблону 5. В результате архитектура команд была приведена к такому виду, как на рисунке , а для отмены операции удаления было обеспечено восстановление связей между примитивами, а также их связей с уже выполненными командами.

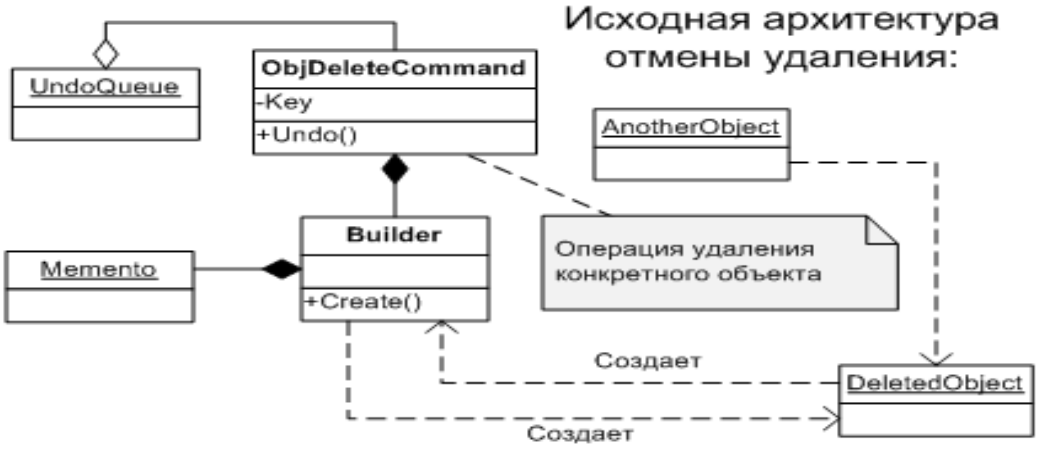

Рисунок 7 – Унаследованная архитектура операции удаления примитива в ПК «NGT Smart»

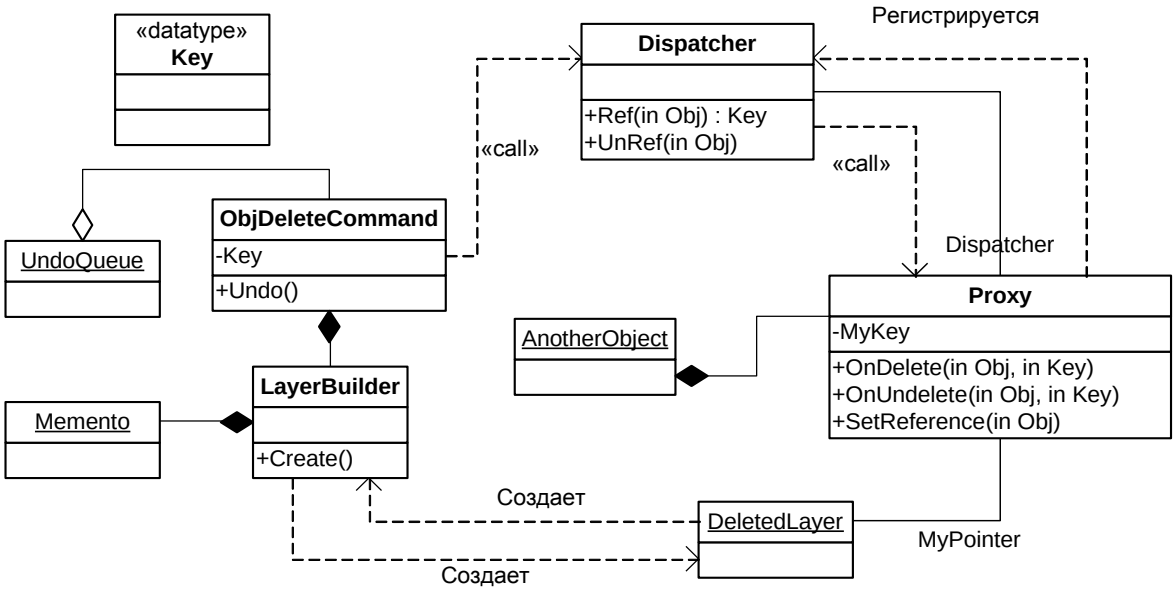

Рисунок 8 – Полученная архитектура для операции удаления слоя

Метод тестирования внедрялся в тех же двух проектах. Поводов к внедрению было несколько.

Прежде всего, полностью отсутствовала автоматизация тестирования. Ранее в истории проекта предпринималась попытка внедрить систему тестирования DUnit из классического семейства xUnit, основанную на модульном тестировании, но разработка модульных тестов зашла в тупик из-за специфики модульного тестирования в сочетании со следующим пунктом.

Далее, ко моменту принятия решения объем исходного кода в обоих проектах превышал 1.5 миллиона строк в 1500 исходных файлах, содержащих около 7500 различных классов, в каждом из которых в среднем было около 7 подлежащих тестированию методов. Поскольку модульное тестирование для получения ощутимой отдачи должно покрыть значительную долю классов в приложении, объем работы был слишком велик для того, чтобы сделать ее побочной для кого-то из разработчиков.

Первым тестом было создание нового документа, одним из следующих стало построение графиков по группе выбранных в документе объектов (нефтяных скважин). Спустя неделю регрессионный анализ по этим тестам выявил только что внесенную в программу ошибку. В дальнейшем покрытие программы тестами постоянно росло, на создание теста для каждой функции программы уходило не более одного дня работы. Кроме того, дальнейшая доработка архитектуры проекта NGT Smart позволила формулировать тесты не в виде программного кода, а в виде текстовых сценариев, запускаемых в программе и имитирующих действия пользователя (т.е. для чисто количественного изменения набора тестов больше не нужен программист).

Проведен анализ эффективности разработанного шаблона проектирования с помощью :предложенной модели качества. Наиболее существенные показатели таковы:

– Шаблон соответствует парадигме Undo/Redo для перемещения вдоль истории изменений документа.

– Ни один из показателей качества архитектуры не свидетельствует о наличии архитектурных изъянов (см. табл. 1).

Таблица 1. Значения некоторых важных метрик сложности архитектуры.

Среднее количество закрытых атрибутов	Среднее количество закрытых атрибутов (полей) в классе.	Рекомендации: x<6 Факт 0.625
Взвешенное количество методов	Суммарная цикломатическая сложность методов класса.	Рекомендации: x≤100 Факт 3.5
Взаимосвязи классов	Количество классов, с которыми связан данный.	Рекомендации: x<6 Факт 1.8
Отклик класса	Кол-во методов класса, которые могут быть вызваны в ответ на событие, и вызываемых из этих.	Рекомендации: x≤100, x≤5*N, N= кол-во методов Факт 3.0, N=2.5 (в среднем)

– Сложность алгоритма работы операции удаления в предложенном шаблоне линейная от числа ссылок на удаляемый примитив.

Нет специальных мер по обеспечению безопасности и надежности работы приложения – но такая задача и не ставилась в настоящей работе.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ И ВЫВОДЫ

1. Построена формальная модель операции удаления примитива редактирования. Модель описывает отношения между состоянием различных сущностей, участвующих в выполнении прямой и обратной операций удалений. Модель *позволяет* разбить задачу проектирования обратимой операции удаления примитива на отдельные подзадачи.

2. Разработан (на основе решения этих подзадач) шаблон проектирования обратимой операции удаления, *опирающийся* на шаблон «Undo/Redo» и «команда». Поскольку проектирование редактора обычно включает работу, помимо системы отмены операций, еще и над печатью, представлением документов в файлах и в памяти, отображением документа, командами его редактирования и др., то предложенное решение *позволяет* сократить приблизительно на 20-25% этап проектирования при разработке системы отмены операций для нужд конкретных приложений-редакторов.

3. Разработана модель качества ПО, подходящая для оценивания качества обобщенных повторно используемых фрагментов архитектуры. *Заимствуя* общий подход к анализу качества у модели SQuaRE, опирающейся, в свою очередь, на стандарты ISO/IEC 9126 и ISO/IEC 14598, модель заменяет часть ее метрик на предложения других исследователей, *что обеспечивает* применимость модели для оценки качества обобщенных проектных решений, а не только конкретных программных продуктов.

4. Разработан метод контроля качества внедрения предложенного подхода в программных продуктах. Метод *основывается* на принципе регрессионного тестирования: заключение о работоспособности модулей приложения выносится по совпадению результатов их работы на тестовых данных с замеренными ранее и признанными эталонными. *Особенностью* метода является использование приемов модульного тестирования с целью за короткое время покрыть тестами существенную часть функций приложения.

5. Разработанные шаблон и алгоритмы были внедрены в ПП «BNView» и «NGT Smart» в виде отдельных модулей и в виде изменений обслуживающей инфраструктуры. В результате внедрения получена отменяемая операция удаления примитивов-слоев в документе. К началу внедрения шаблона операция удаления в приложении уже имелась, но не была отменяемой; внедрение позволило реализовать отмену.

Оценка качества шаблона, проведенная в соответствии с разработанной моделью качества, показала высокое качество проектного решения и его соответствие целям его создания.

Эффективность предложенного метода тестирования, согласно экспертным оценкам, существенна, метод справляется с поставленной задачей быстрого обеспечения легко расширяемого покрытия тестами.

ОСНОВНЫЕ ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

В рецензируемых журналах из списка ВАК

1. Быстрое введение модульного тестирования в частично завершенный программный продукт / Гумеров М.М., Фридлянд А.М. // Инфокоммуникационные технологии. Самара: Поволжская государственная академия телекоммуникаций и информатики, том 5, №4, 2007. С. 66-72.

2. Модели операции удаления примитивов редактирования в прикладных программных продуктах / Гумеров М.М., Фридлянд А.М. // Вестник УГАТУ: науч. журнал Уфимск. гос. авиац. техн. ун-та, УГАТУ, том 11, №1, 2008. С. 157-163.

В других изданиях

3. Шаблоны для целенаправленного рефакторинга / М. Гумеров // CSIT’2006: Труды VIII Международной науч.-техн. конференции – Уфа: УГАТУ, 2006, том 2, с. 57-62 (статья на англ. яз.)

4. Некоторые проблемы реализации отмены операций в приложениях / М. Гумеров, И. Костригин // Новые информационные технологии и системы: Труды VII международной науч.-техн. конференции. – Пенза: ПГУ, 2006, том 2, с. 104-109.

5. Обращение удаления объекта с большим количеством связей / М. Гумеров // CSIT’2007: Труды IX Международной науч.-техн. конференции – Уфа: УГАТУ, 2007, том 3, с. 177-179 (статья на англ. яз.)

6. Программный комплекс «NGT Smart» // Свидетельство об официальной регистрации программ ЭВМ. – 2010. − N 2010614274.

7. Программный комплекс «BNView» // Свидетельство об официальной регистрации программ ЭВМ. – 2011. − N 2011613115.

8. Еще один подход к формализации задачи проектирования операции удаления примитивов редактирования в прикладных программных продуктах / М. Гумеров // Наука и современность - 2011: Труды X Международной науч.-практ. конференции. ч.2. - Новосибирск: НГТУ, 2011. - 282 с. - с. 32-38.

Диссертант Гумеров М.М.