Уфа : УГАТУ, 2008 Вестник УГАТУ • Управление, ВТ и И Т. 11, № 1 (28). С. 157–163

УДК 004.415.5

М. М. ГУМЕРОВ, А. М. ФРИДЛЯНД

МОДЕЛИ ОПЕРАЦИИ УДАЛЕНИЯ ПРИМИТИВОВ РЕДАКТИРОВАНИЯ В ПРИКЛАДНЫХ ПРОГРАММНЫХ ПРОДУКТАХ

Рассматривается задача проектирования отменяемой операции удаления, нередко возникающая при разработке программного обеспечения. Предложена модель операции удаления в достаточно общем случае, без привязки к специфике конкретных приложений. Предложена модель функционирования обратимой операции удаления. Оценены трудозатраты, необходимые для внедрения описанного подхода. *Отмена операций; команда удаления*

Одним из необходимых условий комфортной работы пользователя с приложением, осуществляющим редактирование каких-либо документов (электронная таблица, текстовый документ или картографическое приложение), является возможность отмены ошибочно выполненных действий. При этом желательно, чтобы и сама операция отмены была обратима — ведь и она могла быть выполнена по ошибке.

Собственно концепция отмены выполненных операций на сегодняшний день существует в следующем виде. Информация о выполняемых пользователем действиях заносится в очередь, пока пользователь не воспользуется операцией отмены последнего выполненного действия или не выполнит действия, делающего невозможным отмену уже совершенных действий. История работы приложения представляется в виде последовательности проходимых им состояний, переход от более раннего состояния к непосредственно следующему за ним, более позднему осуществляется путем выполнения совершаемых пользователем операций, обратный переход — путем выполнения операций, обратных к ним. Состояние приложения в любой момент времени является результатом выполнения некоторого числа первых действий, записанных в очереди. Отмена ошибочно выполненной пользователем операции состоит, по сути, в переходе от сложившегося после ее выполнения состояния к предыдущему состоянию. Для отмены самой операции отмены пользователю предоставляется обратная к ней операция воспроизведения (Redo) последнего отмененного действия. Разумеется, часть действий не может быть отменена — например, печать документа на принтере.

Вопросы проектирования и реализации системы поддержки отмены операций нашли отражение в целом ряде публикаций [1, 3, 4, 5]; в то же время предметом рассмотрения повсеместно является принципиальная организация такой системы, частные же задачи отмены специфических операций относятся к деталям реализации и на этом основании обходятся вниманием. Такое положение вещей отчасти оправдано тем, что специфика операций определяется конкретным приложением — однако вполне можно выделить некоторые классы операций, присутствующих в том или ином виде во множестве различных приложений. Хотя конкретное приложение определяет специфику реализации каждой такой операции, имеет смысл выделить некоторые из них, для которых возможно сформулировать типовые проектные решения, чтобы избавить каждый новый коллектив разработчиков от необходимости заново проектировать обращение этих операций (для некоторых операций обеспечение обратимости оказывается нетривиальной задачей).

Одной из таких операций, по мнению автора, является операция удаления какого-либо объекта взаимодействия с пользователем (графического примитива, фрагмента текста и т. п.). Помимо того, что ее обращение нетривиально само по себе даже при рассмотрении в отрыве от других операций приложения, оно еще и, как следует из показанной далее модели, предъявляет дополнительные требо-

Контактная информация: (347)273-79-67

вания к последним. Предметом рассмотрения в данной работе является именно операция удаления и ее обращение.

Удаляемые объекты зависят от специфики приложения и могут быть любыми: от закладок в текстовом редакторе до обозначений элементов на электрической схеме или записей в базе данных. С точки зрения данного исследования интерес представляет выделение объектов в соответствии с тем принципом, что объекты могут являться операндами каких-либо обратимых операций, а также могут иметь ссылки на другие объекты.

1. МОДЕЛЬ ОПЕРАЦИИ УДАЛЕНИЯ ОБЪЕКТА

Рассмотрим в качестве иллюстрации команду удаления объекта, представляющего собой геологический маркер, из содержащей его коллекции. Пусть она параметризуется двумя значениями (возможно, указателями), каким-то образом идентифицирующими коллекцию и маркер (вопрос о способе идентификации сущностей сам по себе нетривиален и также рассматривается в рамках этой главы). Для удаления объекта из коллекции этой информации — двух идентификаторов — вполне достаточно, если при проектировании объекта и коллекции учитывалась возможность такого удаления. Итак, команда удаляет объект — но она предварительно должна каким-то образом сохранить состояние объекта для возможности последующего восстановления. Это первая задача, требующая решения.

Вторая задача заключается в том, что, помимо состояния удаляемого объекта, команда должна сохранить состояние и других объектов, если оно зависит от удаляемого. Типичный пример такой зависимости — ссылки из других объектов на удаляемый. Конечно, не все такие ссылки непременно нужно сохранять. Простой пример: пусть в какой-то момент пользователь отобразил в каком-то окне свойства некоей вершины графа, а затем ее удалил — что удалило ведущие к ней ребра и очистило окно свойств. Тогда при отмене удаления следует восстановить ведущие к вершине и от нее ребра; в то же время, может быть необязательно восстанавливать состояние окна свойств.

В силу указанных выше причин на сегодняшний день, насколько известно автору, не построено имеющих научную ценность формальных моделей каких-либо операций в приложении, отражающих место операции удаления объекта в системе связей приложения достаточно полно, чтобы служить основой для типового проектного решения. Достаточно полные описания встречаются лишь для специфических видов операции удаления. Так, в [1] приводится ряд моделей и методов с формальными доказательствами их работоспособности, актуальных, однако, в первую очередь применительно к операциям над записями в базах данных. Таким образом, отправной точкой для разработки типового проектного решения (паттерна проектирования) должно стать предложение адекватной модели операции удаления объекта в ее общем виде.

Проведенное автором исследование составляющих операции удаления позволило получить следующую модель, представленную в виде статической структурной диаграммы UML на рис. 1.

Модель разработана с учетом трудностей обращения операции удаления, обнаруженных автором при исследовании принципиальной возможности обращения удаления в [2] и анализе ряда алгоритмов удаления из нескольких программных продуктов, в разработке которых автор принимал участие в процессе своей профессиональной деятельности. Анализируемые продукты относились к различным сферам: система кеширования web-запросов в Internet [8, 9], редактор справочников базы данных из состава ПК «МегаТек» (ООО НПФ «Интек»), ПК «Геология и Добыча» (ОАО НК «Роснефть»); в рамках последнего впоследствии проводилась разработка обратимой операции удаления по материалам настоящего исследования. Кроме того, во всех этих случаях использовались различные программные платформы — Win32, .Net и виртуальная машина Java (программная платформа в контексте данного исследования в первую очередь определяет размещение объектов в оперативной памяти и реализацию ссылок на них, а также время жизни объектов).

В соответствии с целью создания этой модели, показанные в ней ассоциации сущностей соответствуют трудно обратимым составляющим эффекта удаления. В процессе выполнения операции удаления участвуют: объект-операция (команда), осуществляющая удаление, удаляемый объект, имеющий некое состояние, и другие объекты, осведомленные о данном, часть из которых может быть экземплярами других команд, находящихся в очереди выполненных операций. Удаление, возможно, освобождает занимаемый объектом объем оперативной памяти, а также отменяет сопоставление объекту опре-

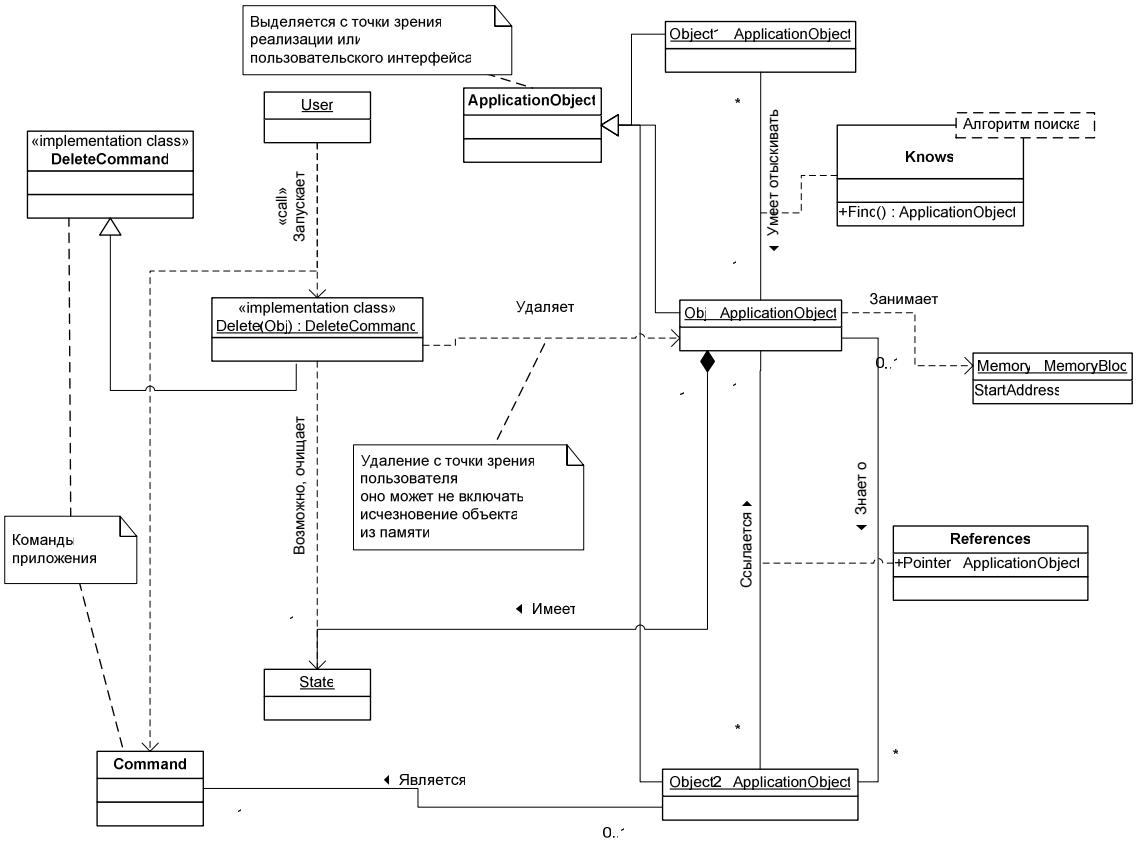

Рис. 1. Принципиальная модель операции удаления объекта

деленного адреса в ней (после чего невозможно без написания собственной системы управления распределением памяти добиться, чтобы при отмене удаления объект снова начал занимать ту же область памяти). Часть осведомленных объектов использует для идентификации данного объекта какие-либо ключи, не связанные с его положением в оперативной памяти (адресом) — индексы в каких-либо массивах и т. п., другие объекты ссылаются на него указанием его адреса. Наиболее существенные подзадачи, возникающие в связи с обозначенными ассоциациями, рассмотрены в следующем разделе.

Предложенная модель полезна в первую очередь тем, что идентифицирует ряд подзадач, решение которых позволит обеспечить обратимость операции удаления. В разделе 3 эта модель, объединенная с решениями подзадач, переходит в модель обратимой операции удаления, которая может применяться для реализации отменяемой операции удаления примитивов редактирования в различных программных продуктах.

2. ПОДХОД К РЕШЕНИЮ ПОДЗАДАЧ

В первую очередь рассмотрим, как вообще можно сохранить для последующего восстановления состояние удаляемого объекта. Многообразие возможных реализаций сохранения состояния объекта можно свести к трем основным вариантам.

В первом варианте команда имеет доступ к интерфейсам объекта, предоставляющим достаточно информации для сохранения его состояния. Другая возможность: объект способен сам сохранять свое состояние для последующего восстановления. Наконец, команда может сохранить в качестве состояния объекта сам этот объект. Последний, однако, может иметь ряд ссылок на другие объекты. Часть из этих ссылок указывает на фрагменты его состояния, часть же становится более не нужна и даже вредна после удаления из коллекции: так, объект более не должен получать события, на которые был подписан; напротив, при восстановлении объекта в коллекции эти ссылки тоже надо восстановить. Получается, объект должен иметь интерфейс активации/пассивации.

Давать рекомендации относительно выбора одного из этих методов бессмысленно, пока не рассмотрена одна из частей состояния объекта, сохранение которой само по себе нетривиальная задача. Речь пойдет о сохранении в каком-то виде ссылок одного объекта на дру-

гой и о последущей идентификации ссылаемого объекта по такому представлению ссылки.

Проблема идентификации, в соответствии с участием удаляемого объекта, имеет две составляющие: 1) идентификация объектов, на которые ссылается удаляемый, для сохранения и восстановления его состояния, 2) идентификация удаляемого объекта — для сохранения и восстановления состояния других объектов, ссылающихся на него. Поскольку для п. 1 целевые объекты также могут быть удалены, то п. 1 сводится к п. 2.

Один, более очевидный и обычно широко применяющийся способ — прямое указание на объект по его адресу в памяти. Идентификация удаляемого объекта по его адресу требует, чтобы при отмене удаления он восстанавливался в прежнем занимаемом им участке памяти. Это реализуемо не во всех языках программирования, и, во всяком случае, потребует особого механизма для создания и удаления таких объектов. Единственный простой способ заключается в том, чтобы вовсе не удалять объект.

Другой способ вносит элемент косвенности: объекту приписывается индекс — например, порядковый номер в некотором упорядоченном списке. Неприятные последствия состоят в том, что код, основанный на индексации (т. е. дополнительной степени косвенности), труднее читается; что более существенно, изначально в приложениях обычно используется первый подход, а тогда для перехода к индексации придется вносить большое количество изменений.

Для преодоления недостатков индексации можно инкапсулировать индекс и способ поиска по нему в класс-поисковик, предоставляющий абстрактный интерфейс «найти объект», как предложено автором в [2]. Эти поисковики соответствуют паттерну проектирования «заместитель» (Proxy) [6], но одновременно и предложенной Microsoft для своей технологии COM концепции моникеров (Monikers) [7], поэтому в дальнейшем изложении они называются моникерами.

Что касается восстановления ссылок на удаляемый объект, предлагается следующий ответ. Объект Referer, ссылающийся на удаляемый объект Referee, держит на него не прямую ссылку, а моникер. При получении от системы отката операций оповещения об удалении объекта X объект Referee проверяет, не имеет ли он моникер, ссылающийся на X, и если да (пусть X = Referee), то в дальнейшей работе объекта Referer этот моникер участвовать перестает (пропускается в циклах и т. п.), оставаясь, однако, в распоряжении Referer. В дальнейшем, при отмене удаления объекта Referee, Referer определяет, что это именно удаленный в прошлом объект, и снова включает моникер в обработку. Если удаляется сам объект Referer, он как часть своего состояния сохраняет и свой моникер объекта Referee, чтобы не потерять ссылку.

3. МОДЕЛЬ ОБРАТИМОЙ ОПЕРАЦИИ УДАЛЕНИЯ

Вместо того, чтобы реализовывать прослушивание сообщений в каждом заинтересованном объекте, вводя разные события для разных видов объектов в приложении, механизм прослушивания сообщений и отключения моникеров можно автоматизировать и унифицировать, возложив эти обязанности на сами моникеры.

Пусть моникеры теперь служат следующим целям. Во-первых, они могут находиться в активном либо пассивном состоянии: в активном состоянии моникер позволяет получить доступ к указуемому объекту, в неактивном — не позволяет. Во-вторых, это состояние у них может быть опрошено; в отличие от предложенной выше простой реализации моникера, который являлся простым proxy, мало чем отличаясь от простого указателя на объект, теперь предполагается, что внешний код будет принимать какие-то решения в зависимости от состояния моникера; это делается для того, чтобы можно было хранить пассивные моникеры среди активных и при обработке просто пропускать пассивные. В-третьих, после удаления объекта его моникеры сохраняют в известном им виде данные, позволяющие его идентифицировать в случае его восстановления, и, наконец, в-четвертых, в момент любого обращения к нему моникер может по этим данным отыскать указуемый объект (если он не был удален или уже был восстановлен), или определить, что объект в данный момент удален.

Такой усовершенствованный моникер берет на себя слежение за удалением и восстановлением объектов; при удалении объекта, на который он указывает, он автоматически переходит в пассивное состояние, при восстановлении возвращается в активное. В то же время, Referee может подписаться у моникера на уведомления о восстановлении объекта, на который он указывает, чтобы сопроводить удаление и восстановление какими-ли-

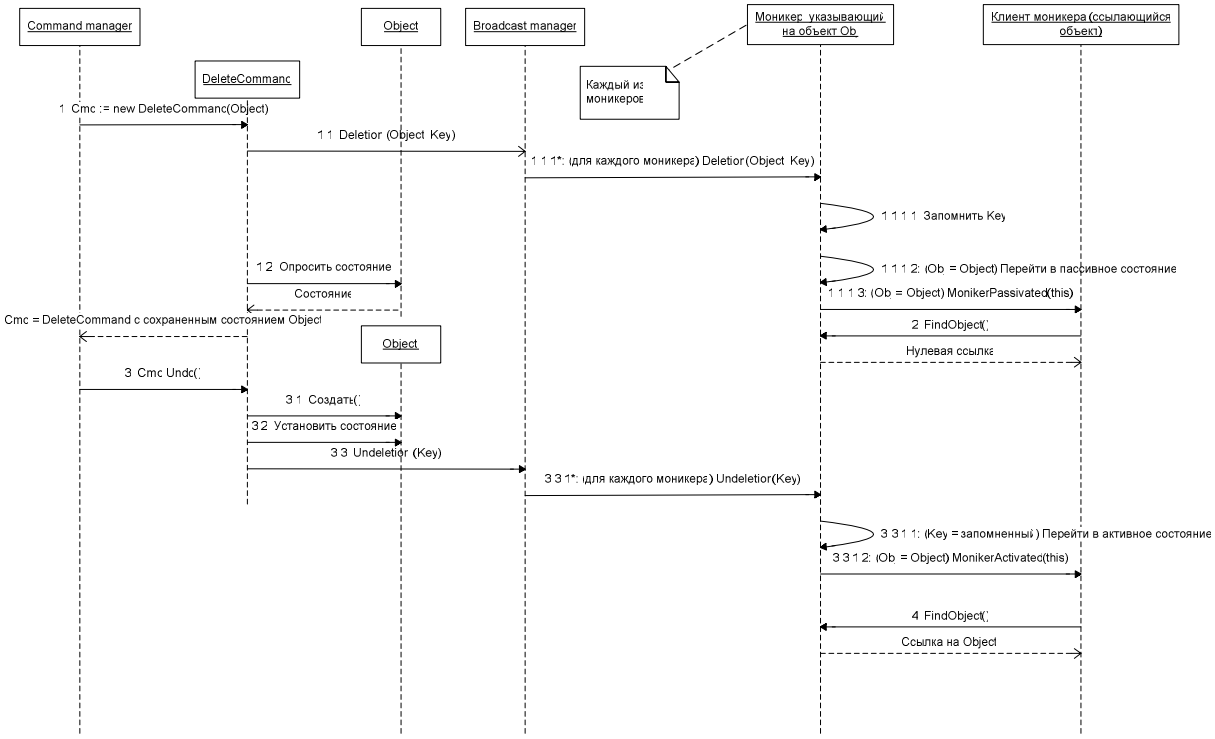

Рис. 2. Модель обратимой операции удаления

бо действиями (например, обновить соответствующим образом свое визуальное представление).

Использование моникеров такого вида позволяет предложить модель обратимой операции удаления, приведенную на рис. 2. Модель представлена в виде диаграммы последовательности UML, так как в большей степени рассматривает операцию как процесс и может быть положена в основу алгоритма для конкретной операции удаления в специфическом приложении. Эта модель интегрирует предложенную схему использования моникеров в принципиальную модель операции удаления для обеспечения сохранения связей объекта с другими, ссылающимися на него по его адресу; тем самым одновременно решается и вопрос о связи между объектом и занимаемой им областью памяти, которую при таком подходе поддерживать уже необязательно.

Сообщение `Deletion(Object, Key)` является упомянутым выше уведомлением об удалении объекта `Object`; команда удаления при помощи какого-либо реестра удаленных объектов сопоставляет объекту ключ `Key`, значение которого для этого объекта должно оставаться постоянным и уникальным среди значений других таких ключей, по меньшей мере до тех пор, пока удаление этого объекта не будет отменено. При отмене удаления объекта, после воссоздания его в ранее сохраненном состоянии, для восстановления ссылок других объектов на него всем моникерам посылается сообщение `Undeletion(Key)`. Моникеры в процессе реакции на это сообщение могут опросить у реестра по назначенному объекту ключу `Key` ссылку на объект `Object`, чтобы заново привязаться к объекту и тем самым перейти в активное состояние.

Следует отметить, что не обязательно буквально каждое обращение к объекту из других, ссылающихся на него, производить через моникер: если известно, что в процессе выполнения некоего метода или группы операторов указуемый объект не может быть удален, можно на время выполнения этой группы операторов опросить у моникера прямую ссылку на объект и пользоваться ею, но только в пределах этой группы.

Ранее уже упоминалось применение предложенной модели в разработке ПК «Геология и Добыча» для НК «Роснефть», в рамках создания модуля поддержки отмены операций для версии 3.0 данного продукта. Разработка шла в условиях существующего проекта, в котором уже было реализовано несколько операций удаления, относящихся к различным классам примитивов, таким как слои карты и ячейки заводнения. На основе модели были созданы алгоритмы, положенные затем в основу перепроектированных вариантов этих операций.

4. ОЦЕНКА ПРЕДЛОЖЕННОГО ПОДХОДА

Рассмотрим количественные оценки ресурсоемкости специфических алгоритмов, которые могут быть получены из приведенной модели в условиях конкретного приложения. Поскольку настоящая работа предлагает не алгоритм, а типовую архитектуру, то и предложенный в ней подход должен характеризоваться не непосредственно ресурсоемкостью (по памяти, процессору и т. п.) частных алгоритмов, а превышением этой ресурсоемкости над ресурсоемкостью аналогичной, но необратимой операции удаления в каждом конкретном приложении. Кроме того, для типового решения особую важность приобретает объем трудозатрат на его реализацию в приложении, и его также необходимо оценить.

Поскольку алгоритм в части, касающейся удаления, не включает существенно затратных по времени операций — как максимум, содержит циклы линейной сложности, а если объекты не удаляются, то лишь добавляет одно косвенное обращение по указателю, — то его введение в приложение не замедляет работу существующего кода, за исключением сценариев, когда удаление или восстановление объектов выполняется в цикле. Последняя ситуация является достаточно нетипичной: ведь если операция запускается с возможностью последущей отмены эффекта, это подразумевает, что запуск происходит по запросу пользователя. С учетом этого, возможным сценарием удаления в цикле может быть удаление большой группы выбранных пользователем объектов. В то же время, удаление и восстановление объектов сопровождается, как правило, перерисовкой пользовательского интерфейса, и если визуальное представление объектов, отображаемых в пользовательском интерфейсе, не совсем тривиально, то процессорное время, необходимое на перерисовку, вероятно, окажется сравнимо или даже существенно превзойдет время работы алгоритма.

При отмене удаления собственно алгоритм также имеет лишь линейную сложность, но необходимо учитывать, что отмена операции удаления включает повторную активацию моникеров, в ходе которой они ищут объект, к которому привязаны. А сложность и ресурсоемкость этого поиска зависит от реализации моникеров конкретных видов объектов — что, впрочем, никак не характеризует эффективность алгоритма, так как в каких-то случаях реализация поиска может быть тривиальна.

Проведенные замеры показали, что действительно накладные расходы на уведомление моникеров некоего объекта о его удалении столь малы, что их можно не принимать во внимание, если, конечно, речь не идет о последовательном удалении очень большого числа объектов, порядка сотен тысяч. Выполнение реальных операций удаления в ПК «Геология и Добыча» на компьютере с процессором Intel Pentium 4 - 2.8 занимало менее одного кванта времени ОС, равного 12 мс, а точнее — около 0,5 мс. Отмена этих операций также выполнялась менее чем за один квант времени, но занимала уже около 5 мс (впрочем, моникеры в ПК «ГиД» сознательно реализованы не оптимально, их быстродействия и в таком варианте вполне достаточно для комфортной работы).

Потребности предложенного подхода в оперативной памяти полностью определяются потребностями используемых моникеров, т. е. также зависят от их реализации.

В качестве отображаемых и удаляемых объектов выступали слои изображения в ГИС-подобной системе ПК «Геология и Добыча».

Поскольку нет возможности сравнить ресурсоемкость обратимой и необратимой операций удаления в существующих приложениях, — разве что путем исследования и модификации исходного приложений с открытым исходным кодом, что можно отметить как одно из направлений продолжения исследований по рассматриваемой в данной работе теме, — то нет и возможности сравнить ресурсоемкость предложенного подхода с ресурсоемкостью других существующих.

Что касается затрат ресурсов при разработке, в качестве показателей была выбрана доля измененных строк в коде, работающем с неким видом удаляемых объектов, и отношение количества внесенных и выявленных затем ошибок (и человеко-часов на исправление) к объему этого кода. Потребовалось в среднем около 1 измененной строки на 40 строк исходного кода, при этом на каждые 750 строк исходного кода была в процессе внесения изменений внесена одна ошибка. Другой показатель — фактические трудозатраты: на превращение очередного типового фрагмента старого кода в команду и обеспечения ее отмены у разработчиков уходило в среднем 2 человеко-дня.

К сожалению, нет возможности сравнить показатели ресурсоемкости разработки при использовании предложенного подхода с этими же показателями, при реализации других существующих алгоритмов обратимого удаления, поскольку не удается отыскать опубликованные описания методик, используемых в существующем ПО, чтобы попытаться для сравнения применить их в том же ПК «ГиД».

ЗАКЛЮЧЕНИЕ

Построенная в настоящей статье принципиальная модель операции удаления может служить отправной точкой для дальнейших исследований в этой области, позволяя идентифицировать отдельные составляющие операции. В частности, используя определенный набор алгоритмов (в центре которого находится использование особого вида моникеров) для реализации обратимости с учетом выделяемых этой моделью составляющих, в данной работе получена одна из возможных моделей обратимой операции удаления, которая может использоваться для реализации такой операции в приложениях. Проведенный анализ процесса внедрения решения показал его невысокие потребности в трудозатратах, что позволяет рекомендовать его в качестве типового проектного решения (паттерна проектирования) при разработке программного обеспечения.

СПИСОК ЛИТЕРАТУРЫ

1. **Bernstein, P. A.** Concurrency control and recovery in database systems / P. A. Bernstein, V. Hadzilacos, N. Goodman // Addison Wesley, 1987.
2. **Gumerov, M. M.** Reversing deletion of a high-coupled object / M. M. Gumerov // CSIT'2007 Proc. 2007. Vol. 3. PP. 177–179.
3. **Prakash, A.** A Framework for undoing actions in collaborative systems / A. Prakash, M. J. Knister // ACM Trans. on Computer-Human Interaction. 1994. Vol. 1, № 4. PP. 295–330.
4. **Гурин, С.** Расширение возможностей паттерна Command / С. Гурин // RSDN Magazine. 2004. № 5.
5. **Мартынов, А.** Back/Forward и Undo/Redo в .NET-приложениях / А. Мартынов // RSDN Magazine. 2003. №2.
6. **Gamma, E.** Design patterns: elements of reusable object-oriented software / E. Gamma, R. Helm, R. Johnson, J. Vlissides // Addison-Wesley, 1995.
7. **Chappel, D.** Understanding ActiveX and OLE / D. Chappel // Microsoft press, 1996.
8. **Gumerov, M. M.** Applying remote method invocation caching to large-scale distributed Applications / M. M. Gumerov // CSIT'2004 Proc. 2004. Vol. 2. PP. 25–31.
9. **Pfeifer, D.** Method-based caching in multi-tiered server applications / D. Pfeifer, H. Jakschitsch // Proc. of OTM Confererated Intern. Conf. CoopIS, DOA, and ODBASE 2003; Distributed Objects and Applications (DOA) 2003, LNCS 2888. 2003. PP. 1312–1332.

ОБ АВТОРАХ

Гумеров Максим Маратович, Дипл. инж. по програм. обесп. вычисл. техники и автоматизированных систем (УГАТУ, 2004).

Фридлянд Александр Михайлович, доц. той же каф. Дипл. матем.-преп. (БГУ, 1980). Канд. техн. наук по АСУ и системному анализу (УАИ, 1990). Иссл. в обл. моделирования и прогнозирования поведения сложных систем.