Реализация механизма слияния карт памяти (Mind Maps) в продукте Comapping*

Е. В. Ларчик
eugene.larchick@gmail.com

Д. В. Кознов
dkoznov@yandex.ru

М. М. Плискин
Michael.Pliskin@lanit-tercom.com

Данная работа посвящена решению задачи слияния карт памяти (Mind Maps) в контексте коллективной работы с ними в Интернете. Эта задача актуальна, например, в ситуации, когда один из разработчиков теряет Интернет-соединение, но продолжает изменять карту памяти локально, и его товарищи также меняют ее. Таким образом, при восстановлении Интернет-соединения возникает необходимость объединить локальную и серверную копии. В работе представлена реализация алгоритма слияния карт памяти, который основывается на известном алгоритме слияния XML-файлов 3DM. Представленная реализация интегрирована в продукт Comapping и поддерживает два режима работы — слияние по умолчанию, когда все конфликты разрешаются автоматически, и «ручное» слияние для пользователей, которые хотят управлять тем, какие данные из какой версии попадают в итоговую. В работе представлено описание пользовательского интерфейса, реализующего два этих режима.

Ключевые слова: mind maps, коллективная разработка, слияние XML-файлов.

* Работа выполнена при финансовой поддержке РФФИ гранта № 11-01-00622-а.

Введение

Задача слияния (merge) двух разных версий одного и того же файлового актива является классической задачей конфигурационного управления (configuration management): два или более человек меняют один и тот же информационный актив независимо друг от друга и после этого необходимо объединить сделанные изменения в рамках единой версии актива. Слияние текстовых файлов является решенной задачей (см., например, работу [7]), соответствующие алгоритмы входят в многочисленные промышленные средства версионного контроля, например в CVS/Subversion. Однако для более сложных структур данных — XML-файлов, UML-моделей и т. д. — эта задача требует более сложных решений, так как простое текстовое слияние часто нарушает структуру актива — дерева, графа и пр.[1]

Задача слияния XML-файлов стала актуальна в силу широкого использования данного формата для представления различных данных, совместно используемых в Интернете. Сюда относится и коллективная разработка офисных документов (существуют многочисленные XML-форматы документов — OpenXML, DocBook и др.), и синхронизация XML-данных в условиях ограниченной пропускной способности сети (например, для мобильных устройств) и пр. В настоящее время разработано несколько алгоритмов решения задачи слияния XML-файлов [11, 15, 16, 21, 22]. Существуют также практические реализации этой задачи: средство с открытым кодом 3DM [27], XML Diff and Merge — свободно распространяемое Java-средство от компании IBM [28], а также коммерческий продукт DeltaXML [10]. Все они позволяют получить разницу между двумя XML-файлами, а также слить их в один файл. Кроме того, существует система контроля изменений документов So6 [24], которая позволяет нескольким пользователям редактировать XML-файлы одновременно. Таким образом, можно сказать, что в общем виде данная проблема решена. Однако конкретные практические задачи формулируют особые требования к слиянию XML-файлов, что влечет необходимость модификации существующих алгорит-

[1] Подробный обзор различных практических задач, где появляется необходимость сливать XML-файлы, приводится в работе [15].

мов. Одной из таких задач является слияние карт памяти (Mind Maps), представленных в виде XML, в процессе их коллективной разработки Интернет-средствами.

Карты памяти (Mind Maps) — это графическая нотация и метод, предназначенные для работы со знаниями в самых разных областях: в обучении, бизнесе, при написании книг, статей, в научной деятельности, при планировании личных и семейных мероприятий, при психологическом тестировании и т. д. Данный подход был предложен Тони Бьюзеном в конце 70-х годов прошлого века [8] и доказал свою состоятельность. В связи с этим появилось много программных продуктов, поддерживающих карты памяти, — Free Mind [12], Mind Manager [17] и др.[2] В последнее время стали активно развиваться Интернет-средства, поддерживающие коллективную работу с картами памяти, — Comapping [9], Mindomo [20] и Mind Meister [19]. Эти средства хранят карты памяти на сервере в виде XML-файлов, предоставляя пользователям возможность раздельного редактирования в online-режиме. Однако до настоящего момента в них отсутствовала возможность слияния двух версий одной и той же карты памяти. Это необходимо, например, когда группа пользователей работает с одной картой памяти в online-режиме и один из них теряет подключение к Интернету (например, он находится в дороге и продолжает работать с картой памяти локально на своем ноутбуке). В это же время его товарищи, также работая с этой картой, изменяют серверную версию. После восстановления Интернет-подключения необходимо слить вместе две версии.

В работе [6] мы представили модификацию известного алгоритма слияния XML-файлов 3DM [16] для решения этой задачи слияния карт памяти. Алгоритм был модифицирован так, чтобы были удовлетворены специфические требования работы с картами памяти. В данной статье мы представляем реализацию этого алгоритма для продукта Comapping [9]. Эта реализация поддерживает два режима работы — слияние по умолчанию, когда все конфликты разрешаются автоматически, и «ручное» слияние для пользователей, которые хотят управлять тем, какие данные из какой версии по-

[2] Полный список программных средств поддержки карт памяти можно найти [18].

падают в итоговую. В работе представлено описание пользовательского интерфейса, реализующего два этих режима.

Статья организована следующим образом. В *разделе 1* делается обзор карт памяти и продукта Comapping. В *разделе 2* вводятся основные термины, используемы в работе. В *разделе 3* уточняется постановка задачи. В *разделе 4* дается обзор алгоритма 3DM, выбранного в качестве основы для решения нашей задачи, а также кратко обсуждаются модификации этого алгоритма. В *разделе 5* приводится описание реализации алгоритма слияния карта памяти в Comapping.

1. Карты памяти и Comapping

В начале 70-х годов прошлого века английским психологом Тони Бьюзеном была предложена техника работы с информацией, основанная на использовании карт памяти (Mind Maps) [8]. Карта памяти представляет собой диаграмму с очень простой нотацией. В центре диаграммы находится главный элемент, олицетворяющий собой ключевую идею или концепцию. Этот элемент затем соединяется с другими элементами, поясняющими и детализирующими его, которые располагаются вокруг и т. д. (рис. 1). Карты памяти имеют следующие достоинства: лёгкость восприятия и запоминания информации, экономия времени на поиск в тексте ключевых слов (благодаря тому, что они более заметны и связаны между собой простыми понятиями), развитие у человека системного мышления в процессе создания таких карт и т. д.

Далее мы опишем Web-приложение Comapping — типичный инструмент в этой области. Читателю будет проще познакомиться с общими принципами и проблемами программной разработки карт памяти на основании описания конкретного приложения[3].

Для представления карт памяти Comapping использует древовидную нотацию, в которой уточнение концепций происходит слева направо (так называемый left-to-right mindmapping), и, соответственно, главный элемент карты памяти находится левее остальных

[3]Продукт Comapping является результатом сотрудничества датской компании Area9, петербургской компании ЗАО «ЛАНИТ-ТЕРКОМ» и Санкт-Петербургского государственного университета. Исследования по применению продукта в образовательном процессе поддержаны компанией Hewlett-Packard.

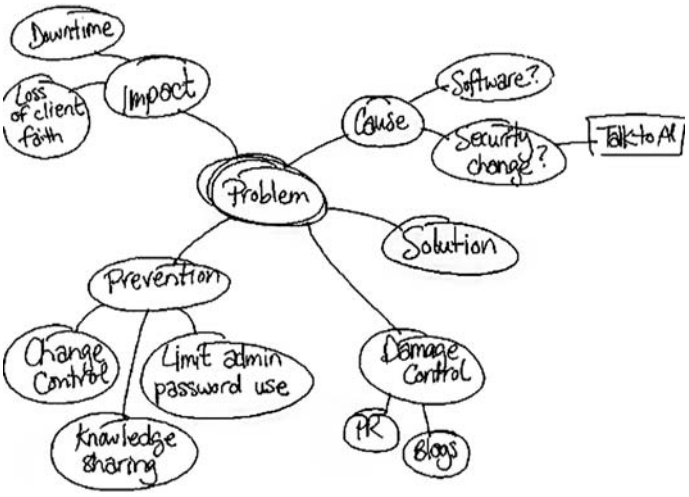

Рис. 1. Пример карты памяти

(рис. 2). Такая нотация в сочетании с алгоритмом автоматического перераспределения элементов на экране облегчает чтение и понимание карты.

Comapping также включает в себя следующие возможности: связывание с элементами карты памяти файлов и заметок, проверка правописания текста в узлах карты памяти; текстовый поиск и фильтрация элементов карты, а также публикация карт в блогах и на сайтах; печать карты любых размеров; экспорт карты памяти в форматы PDF, HTML, RTF, SVG; импорт/экспорт в форматы других средств работы с картами памяти (FreeMind, Mind Manager).

Кроме этого, Comapping фокусируется на предоставлении максимально удобных средств для совместной работы пользователей: несколько пользователей одновременно могут открыть одну и ту же карту для просмотра или редактирования, при этом можно видеть, какие участки карты редактируются в данный момент другими пользователями (рис. 3), можно также общаться во встроенном чате; есть средства для объединения пользователей в группы, а также для изменения прав доступа к карте памяти сразу всем участникам группы; существует механизм email-оповещений об из-

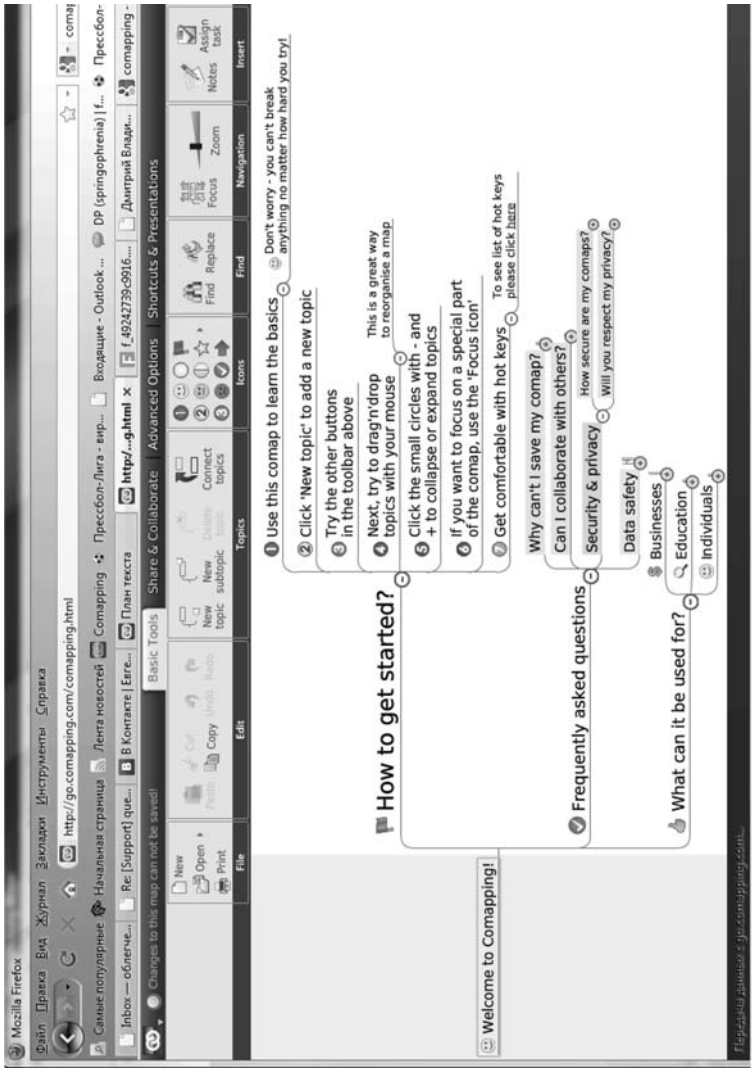

Рис. 2. Пример карты памяти в Comapping

менениях карты памяти другими пользователями; хранится история изменений карты, которая позволяет отследить, кем и когда была изменена та или иная часть карты памяти.

Рис. 3. Совместная работа над картой памяти в Comapping

Более детально с Comapping, а также с его использованием в образовании можно ознакомиться в [4, 14].

2. О задаче слияния XML-файлов

Задача слияния (merge) является синхронизацией двух активов с тем ограничением, что в итоге обе версии данных становятся идентичными: их разные часть объединяются, похожие — сливаются, и одна из версий перестает существовать[4]. Интерес к задаче слияния XML-файлов возник в начале 2000-х годов и был обусловлен тем, что XML-формат стал широко использоваться в различных программных средствах для хранения данных. Задача слияния XML-файлов тесно связана с более ранней задачей нахождения разницы (change detection) между двумя XML файлами или, в более общем виде, между двумя деревьями [25, 26], поскольку любой алгоритм

[4]Эта задача встречается во многих разделах программной инженерии, например в модельно-ориентированном подходе [2, 3, 5].

слияния так или иначе сравнивает сливаемые файлы и находит их разницу. Чаще всего эта разница представляется в виде списка операций, которые переводят один файл в другой. Этот список называют edit-скриптом (edit-script) [25]. Важной частью задачи слияния XML-файлов является идентификация (matching) — процедура нахождения одних и тех же узлов в разных деревьях [25, 26].

Вот список операций над XML-файлами, которые обычно рассматривают в задаче слияния:

- добавить (Insert) — вставка нового узла в дерево;
- удалить (Delete) — удаление поддерева с корнем в заданном узле;
- изменить (Update) — изменение содержимого узла;
- перенести (Move) — «перевешивание» поддерева с корнем в заданном узле к новому родителю.

Еще одним важным аспектом слияния XML-файлов является конфликт (conflict) — ситуация, возникающая при работе алгоритма, когда нельзя однозначно определить, какой из нескольких возможных вариантов предпочесть. Как указывалось в [16], алгоритм не должен быть слишком «умным», самостоятельно разрешая все такие ситуации — окончательные ответ здесь должен дать пользователь, проанализировав ситуацию и выбрав наилучший вариант. Конфликты бывают следующих типов:

- Update/Update — происходит тогда, когда один и тот же узел был изменён в разных версиях различными способами;
- Move/Move — происходит тогда, когда одно и то же поддерево было перемещено в разных версиях в различные локации;
- Delete/Update — происходит тогда, когда в одной из версий поддерево было удалено, а в другой изменено (т. е. в него был добавлен или перемещён хотя бы один узел, либо был изменён какой-либо из существующих узлов).

Отметим, что конфликты могут быть вложенными, когда узел или поддерево, участвующие в конфликте, являются частью бóльшего поддерева, участвующего в другом конфликте.

Существуют две основных парадигмы слияния XML-файлов: 3 way merge и 2 way merge. Алгоритмы первого вида используют при слиянии двух «разошедшихся» версий одного файла, используя исходный файл (в дальнейшем мы будем называть его *базовая*

версия). Алгоритмы второго вида базовую версию не используют. Результат слияния (в дальнейшем — *целевая версия*) в первом случае получается более качественным, однако 2 way merge алгоритмы также необходимы в том случае, когда базовая версия недоступна или ее вообще не может быть (например, при слиянии онтологий [23][5]).

3. Уточнение постановки задачи

Наше решение должно позволить мобильным партнерам разрабатывать карты памяти, при этом партнеры могут терять Интернет-соединение во время работы. При восстановлении Интернет-соединения как раз и возникает задача слияния двух версий: одной, которая образовалась на локальном компьютере, второй — серверной. Мы сформулировали следующие требования к решению задачи слияния двух версий одной карты памяти.

1. Процедура слияния должна иметь полностью автоматический режим для пользователей, которые не могут или не хотят разбираться в ее деталях. Это значит, что должен существовать способ разрешать все конфликты, обнаруженные при слиянии, автоматически, т. е. по умолчанию.
2. Процедура слияния должна иметь «продвинутый» режим использования, т. е. информация о конфликтах должна сохраняться и показываться по требованию пользователя, чтобы у него была возможность разрешить конфликты способом, отличным от принятого по умолчанию.
3. Чтобы предотвратить потерю информации в процессе слияния, конфликтные ситуации, возникающие при удалении какой-либо части карты памяти в одной из её версий и при изменении этой части в другой, должны решаться в пользу сохранения изменённой части.
4. Локальная и серверная копии не являются в нашем случае, в отлчие от 3DM-алгоритма симметричными — при разрешении конфликтов предпочтение отдается серверной копии. Мы считаем, что пользователи, у которых Интернет-соединение не прерывалось, должны получить минимум сюрпризов после

[5] Слияние онтологий оказывается одним из возможных этапов в их жизненном цикле [1, 30].

выполнения процедуры слияния. Далее мы будем использовать термин *локальная версия* для обозначения версии карты памяти, с которой пользователь работает локально, после потери соединения с Интернетом, а термин *серверная версия* — версию карты памяти, с которой работают пользователи, сохранившие Интернет-соединение.

5. Поскольку мы предполагаем встроить наше решение в Comapping, то необходимо учесть тот факт, что последний заботится о сохранении истории изменений карты памяти (она позволяет проследить, как, кем и когда изменялась карта памяти, что важно для любого совместно редактируемого документа), и поэтому нельзя сохранить результат слияния, просто удалив предыдущую серверную версию, т. к. иначе история всех предыдущих изменений станет бесполезной. В связи с этим необходимо получить edit-скрипт для перевода серверной версии в целевую.
6. Реализуется 3 way merge алгоритм, поскольку в нашем случае базовая версия карты памяти доступна, и 3 way merge гарантирует лучшее качество результата.

Первые два требования относятся к разработке пользовательского интерфейса целевого программного сервиса, а остальные — к модификации алгоритма слияния XML-файлов.

4. 3DM-алгоритм и его модификации

Итогом интереса к задаче слияния XML-файлов стало появление нескольких различных законченных подходов [11, 15, 16, 21, 22], а также ряда программных средств [10, 24, 27, 28], решающих данную задачу. Подробно эти подходы рассматриваются в [6], там же обосновывается выбор 3DM-алгормта [15, 16, 27] для решения нашей задачи. В этом разделе мы сделаем обзор 3D-алгоритма, а также алгоритма слияния карт памяти, предложенного нами в работе [6].

4.1. 3DM-алгоритм

В основе алгоритма 3DM лежит следующая идея. Для каждого узла двух сливаемых деревьев рассматриваются изменения, произошедшие с ним относительно базового дерева, и целевое дерево строится применением к базовому всех этих изменений.

3DM-алгоритм производит слияние XML-файлов в два шага. На первом шаге происходит сопоставление (идентификация, matching) узлов сливаемых деревьев с узлами в базовой версии. При анализе этой информации становится понятно, каким образом изменились сливаемые деревья относительно базовой версии: если узлу в сливаемом дереве не сопоставлен узел в базовой версии, то он был вставлен, и т. д. (алгоритм идентифицирует все возможные операции: вставка / изменение поддерева, удаление / перемещение / копирование поддерева). На втором шаге на основе информации об этих изменениях производится слияние деревьев. Обход сливаемых деревьев производится так, что сопоставленные узлы посещаются одновременно. Построение целевой версии происходит «в ширину»: на основе списков детей сопоставленных узлов в базовом и двух сливаемых деревьях с помощью специальных эвристик строится список детей узла в целевой версии.

Алгоритм 3DM был выбран нами в качестве базового в силу следующих причин. Во-первых, он использует подход 3 way merge, во-вторых, он рассматривает все операции над деревьями, которые могут быть произведены в Comapping, и, в-третьих, он является открытым — в [15, 16] представлено его подробное описание, а в [27] — открытая реализация на языке Java. Так как наша задача обладает рядом специфических требований, то оказалось необходимым изменить 3DM-алгоритм. Ниже мы сделаем краткий обзор модификаций 3DM-алгортма.

4.2. Изменение процедуры идентификации

Поскольку каждый узел карты памяти в Comapping имеет уникальный идентификатор (id), то процедура идентификации по сравнению с 3DM-алгоритмом упрощается — у нас при объединении версий сопоставляются друг другу только узлы с одинаковыми id. С другой стороны, id узлов карты памяти не видны пользователям карты, поэтому в процессе редактирования локальной и серверной копий пользователи могут перестать связывать старое и новое содержимое узла, хотя эта связь остаётся в виде неизменного id. В таких случаях сопоставлять две версии узла друг другу не имеет смысла, несмотря на то, что у них одинаковые id. Поэтому в процедуре идентификации мы дополнительно проверяем

«похожесть» сопоставленных по id узлов с помощью определённой метрики. Идея идентификации с метрикой при слиянии различных структур данных не является новой и подробно обсуждается, например, в контексте слиянии онтологий и баз данных [23]. Наша метрика основывается на нахождении Q-расстояния между строками [29].

4.3. Неравнозначность локальной и серверной копий

3DM-алгоритм не делает различий между двумя сливаемыми файлами, поэтому он часто в процессе своей работы меняет их местами, когда ему это удобно. В нашем случае этот неприемлемо. Поэтому, в частности, симметричные конфликты (Move/Move, Update/Update), а также ситуацию спорного порядка узлов мы разрешаем всегда в пользу серверной версии. Таким образом, в случае конфликта Update/Update узлу присваивается его содержимое из серверной копии, а в случаях, когда для двух узлов невозможно определить, в каком порядке они должны следовать в списке детей, то они сохранят такой же порядок следования, как в серверной версии.

4.4. Разрешение конфликта Move/Move

3DM-алгоритм разрешает конфликт Move/Move слиянием поддеревьев из двух версий и копированием результата в обе локации при порождении целевой версии. Однако при вложенных конфликтах такой подход ведёт, во-первых, к множественному копированию одних и тех же узлов, а во-вторых, к дублированию конфликтов (дублируется как минимум каждый конфликт в скопированном поддереве). Поэтому мы разрешаем этот конфликт по умолчанию помещением «слитого» поддерева только в одну локацию — ту, в которой он оказался в серверной версии. Однако у пользователя остаётся возможность разрешить его другим способом.

4.5. Обработка конфликта Update/Delete

Как уже говорилось выше, наш механизм синхронизации двух версий карты памяти должен избегать потерь изменений, поэтому при разрешении конфликта Update/Delete мы, в отличие от 3DM-

алгоритма, по умолчанию оставляем поддерево в целевой версии. Пользователь опять-таки, может выбрать и вторую альтернативу разрешения данного конфликта.

4.6. Изменения в edit-скрипте

3DM-алгоритм порождает edit-скрипт, переводящий исходную версию карты в целевую. Нам такой подход не годится, как было указано выше. Достичь нужного нам решения можно двумя способами. Во-первых, можно не записывать в edit-скрпит те изменения, которые соответствуют изменениям узлов из исходной версии в серверную. Во-вторых, можно запустить алгоритм повторно, используя в качестве исходной версии серверную, а в качестве локальной версии — целевую. Результат работы такого повторного запуска станет целевое дерево, а в edit-скрипт попадут операции, переводящие серверную версию в целевую. Но поскольку средства Comapping легко позволяют в момент добавления операций в edit-скрипт определять, изменениями в какой из версий оно вызвано (локальной или серверной), то был выбран первый вариант.

5. Реализация алгоритма и его интеграция с Comapping

Для реализации алгоритма мы использовали открытую Java-реализацию 3DM-алгоритма, перенеся ее на язык haXe[6], поскольку именно на этом языке написаны клиентская и серверная части продукта Comapping. Далее, мы внесли изменения в реализацию в соответствии со сделанными нами изменениями в алгоритме 3DM и выполнили интеграцию с Comapping. С работой нашего механизма слияния карт памяти можно ознакомиться, запустив Comapping [9].

[6] Язык haXe — это многоплатформенный язык программирования, предназначенный для разработки Web-приложений. Он может быть использован для создания как серверной, так и клиентской частей приложения. Подробнее см. в [32].

5.1. Базовая функциональность

В момент отключения Интернет-соединения текущее состояние карты памяти сохраняется на компьютере пользователя в двух экземплярах: базовая и локальная версии. Первая остается неизменной, вторая пополняется изменениями, которые делает пользователь, находясь в режиме offline. При следующем запуске Comapping в режиме online (в режиме offline Commaping не запустить!) пользователь может просмотреть изменения карты памяти, сделанные им в режиме offline следующим образом: во-первых, при нажатии в стартовом диалоге Comapping кнопки Open Autosave (рис. 4) откроется список карт памяти, когда-либо сохранённых локально, среди которых можно найти нужную и открыть её; во-вторых, при попытке открыть серверную версию карты пользователь будет оповещён о том, что на его компьютере имеется сохранённая локальная версия, и ему будет предложено просмотреть её — сообщение CONFIRM (рис. 4).

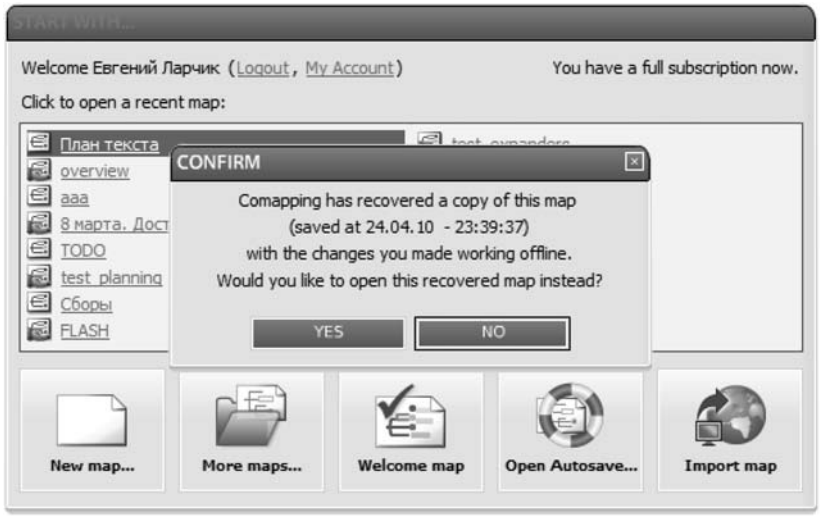

Рис. 4. Оповещение о существовании локальной версии карты памяти

При открытии сохранённой локально версии пользователю будет сразу же предложено слить эту версию с той, которая находится

на сервере. Пользователь может отказаться от слияния и продолжить работу над локальной версией карты памяти только в режиме read-only. Если же он соглашается, то запускается алгоритм слияния. После окончания его работы в браузере пользователя открывается целевая версия, на которой отображается информация о случивших конфликтах. Все дальнейшие изменения карты памяти, которые будет выполнять пользователь, сохраняются на сервере, т. е. работа над картой происходит в обычном режиме.

5.2. Организация работы пользователя с конфликтами

В окне Comahhing, где отображается целевая версия, полученная после слияния двух версий карты памяти, присутствует кнопка Hide Conflicts, с помощью которой можно разрешить все возникшие при слиянии конфликты разом. Рассмотрим подробнее типы конфликтов и возможные варианты их разрешения пользователем.

Конфликт Update/Update по умолчанию разрешается в пользу серверной версии карты памяти. Однако пользователю предоставляется возможность заменить содержимое этого же узла из локальной версии. На рис. 5 приведен пример разрешения конфликта Update/Update. На рис. 5, (1) показано, что у узла «Name patterns» есть конфликт — описание этого конфликта появляется при наведении курсором мыши на знак конфликта. На рис. 5, (2) во всплывающем меню конфликтного узла есть возможность выбрать просмотр его серверной или локальной версии. На рис. 5, (3) показана локальная версия данного узла (видно, что у узла изменилось имя). В том же всплывающем меню есть возможность зафиксировать текущий вариант как окончательный (пункт Mark as resolved). В данном случае разрешение конфликта было выполнено в пользу локальной версии, как показано на рис. 5, (4) — знака конфликта рядом с узлом уже нет.

Работа с конфликтом Update/Delete показана на рис. 6, а с конфликтом Move/Move — на рис. 7.

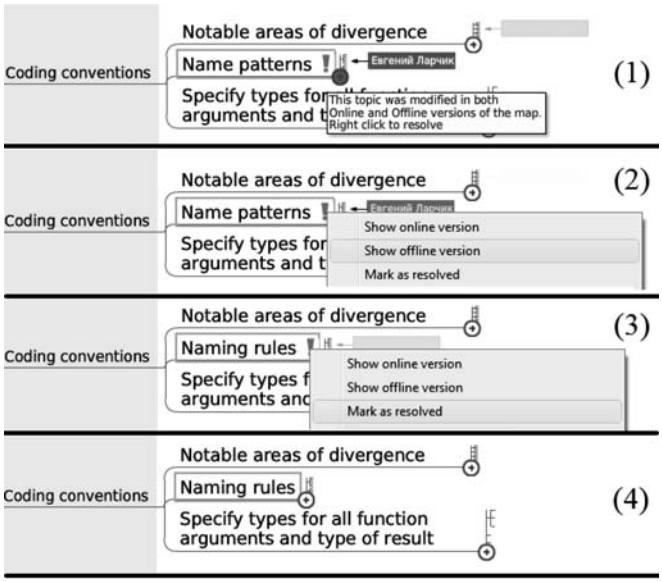

Рис. 5. Пример разрешения конфликта Update/Update: (1) описание конфликта; (2) выбор опции перехода к offline-содержимому узла; (3) выбор опции разрешения конфликта; (4) итог работы с конфликтом

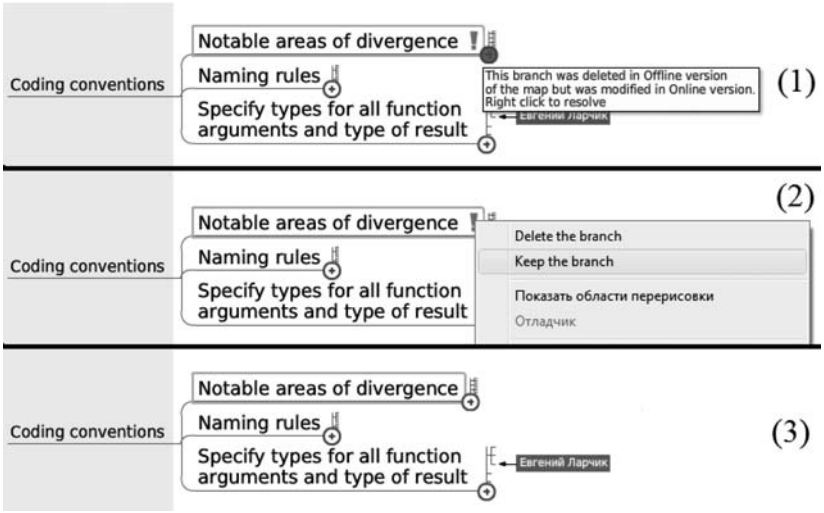

Рис. 6. Пример разрешения конфликта Update/Delete: (1) описание конфликта; (2) выбор опции сохранения конфликтного поддерева; (3) итог работы с конфликтом

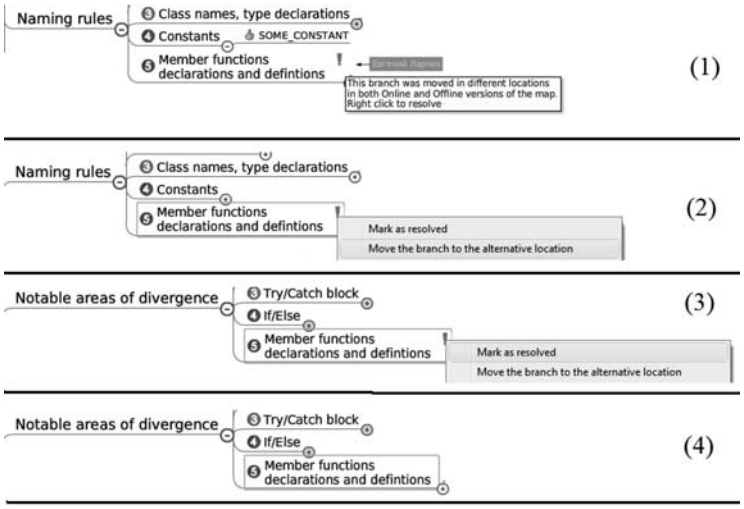

Рис. 7. Пример разрешения конфликта Move/Move: (1) описание конфликта; (2) выбор опции перемещения поддерева к альтернативному родителю; (3) выбор опции разрешения конфликта; (4) итог работы с конфликтом

Заключение

В данной работе представлена реализация задачи слияния карт памяти в процессе групповой разработки через Интернет для продукта Comapping. Алгоритм, представленный в [6], доведен нами до промышленной реализации и интегрирован в продукт Comapping.

Список литературы

[1] *Гаврилова Т. А., Кудрявцев Д. В. Горовой В. А.* Модели и методы формирования онтологий // Научно-технические ведомости СПбГПУ. 2006. № 4. С. 21–28.

[2] *Зверева В. А., Кознов Д. В., Бережной А. С.* Обзор подходов управления согласованностью артефактов разработки ПО при использовании UML // Системное программирование. Вып. 2: Сб. статей / под ред. А. Н. Терехова, Д. Ю. Булычева. СПб.: Изд-во СПбГУ, 2006. С. 240–267.

[3] *Кознов Д. В.* Разработка и сопровождение DSM-решений на основе MSF // Системное программирование. Вып. 3: Сб. статей / под ред.

А. Н. Терехова, Д. Ю. Булычева. СПб.: Изд-во СПбГУ, 2008. С. 80–96.

[4] *Кознов Д. В.* Методика обучения программной инженерии на основе карт памяти // Системное программирование. Вып. 3 / под ред. А. Н. Терехова и Д. Ю. Булычева. СПб.: Изд. СПбГУ, 2008. С. 121–140.

[5] *Холтыгина Н. А., Кознов Д. В.* Обзор реализации механизма циклической разработки диаграмм классов и программного кода в современных UML-средствах // Системное программирование. Вып. 5: Сб. статей / под ред. А. Н. Терехова, Д. Ю. Булычева. СПб.: Изд-во СПбГУ, 2010 С. 76–94.

[6] *Кознов Д. В., Ларчик Е. В., Плискин М. М., Артамонов Н. И.* О задаче слияния карт памяти (Mind Maps) при коллективной разработке // Программирование, 2011. № 6. С. 56–66.

[7] *Michael K. Johnson.* Diff, Patch, and Friends // Linux Journal. August. 1996.

[8] *Busen T.* The Mind Map Book. Penguin Books, 1996. 320 p.

[9] Comapping. URL: http://www.comapping.com

[10] DeltaXML. URL: http://www.deltaxml.com/

[11] *R. La Fontaine.* Merging XML Files: a New Approach Providing Intelligent Merge of XML Data sets // Proceedings of XML. Europe. 2002. Barcelona Spain. 21 p.

[12] Free Mind. URL: http://freemind.sourceforge.net

[13] *Thomas Hettel, Michael Lawley,Kerry Raymond.* Model Synchronisation: Definitions for Round-Trip Engineering. ICMT 2008, LNCS 5063. Springer-Verlag: Berlin; Heidelberg, 2008. P. 31–45.

[14] *Koznov D., Pliskin M.* Computer-Supported Collaborative Learning with Mind-Maps. ISoLA 2008. CCIS. Vol. 17. 2008. Springer-Verlag: Berlin; Heidelberg, 2008. P. 478–489.

[15] *Lindholm T.* A 3-Way Merging Algorithm for Synchronizing Ordered Trees — the 3DM Merging and Differencing Tool for XML. Master's Thesis, 2005, Helsinki, University of Technology. 205 p.

[16] *Lindholm T.* A Three-Way Merge for XML Documents. ACM Symposium on Document Engineering. 2004. P. 1–10.

[17] Mind Manager. URL: http://www.mindjet.com/products/mindmanager-9-win/overview

[18] Mind Maps Tools. URL: http://www.mindjet.com/products/overview

[19] Mind Meister. URL: http://www.mindmeister.com

[20] Mindomo. URL: http://www.mindomo.com/

[21] *Oster G., Skaf-Molli H., Molli P., Naja-Jazzar H.* Supporting Collaborative Writing of XML Documents. ICEIS (4). 2007. P. 335–341.

[22] *Ronnau S., Pauli C., Borghoff U. M.* Merging Changes in XML Documents Using Reliable Context Fingerprints. ACM Symposium on Document Engineering. 2008. P. 52–61.

[23] *Shvaiko P., Euzenat J.* A Survey of Schema-based Matching Approaches. Journal on Data Semantics IV. 2005. P. 146–171.

[24] SO6 toolset. URL: http://dev.libresource.org/

[25] *Chawathe Sudarshan S., Rajaraman Anand, Garcia-Molina H., Widom J.* Change detection in hierarchically structured information // ACM SIGMOD International Conference on Management of Data (SIGMOD). 1996. P. 493–504.

[26] *Chawathe Sudarsah S., Garcia-Molina H.* Meaningful Change Detection in Structured Data // Proceedings of the ACM SIGMOD International Conference on Management of Data. ACM Press, 1997. P. 26–37.

[27] 3DMJavaImplementation. URL: http://www.cs.hut.fi/ ctl/3dm/

[28] XML Diff and Merge. URL: http:// www.alphaworks.ibm.com/ tech/ xmldiffmerge

[29] *Ukkonen E.* Approximate String Matching with Q-gram Sand Maximal Matches // Theoretical Computer Science. Vol. 92, No 1. 1992. P. 191–211.

[30] *Yudelson M., Brusilovsky P., Gavrilova T.* Towards User Modeling Meta Ontology // Proceedings of 10th International Conference User Modeling (UM)/ eds. Ardissono L., Brna P. Mitrovic A. LNCS in Artificial Intelligence (LNAI). Vol. 3538. Springer, 2005. P. 448–452.

[31] Blog. URL: http://useless-factor.blogspot.com/2008/01/matching-diffing- and-merging-xml.html, http://stackoverflow.com/ questions/ 2222548/ 3-way-xml-merge-algorithm

[32] haXe. URL: http://haxe.org/