Основы систем управления версиями файлов
на примере Subversion (SVN)

Представленная статья посвящена системе управления версиями под названием Subversion (SVN). Почему SVN? Потому что это достаточно простая, эффективная и бесплатная система. Для многих разработчиков ее возможностей более чем достаточно. Цель этой статьи — показать преимущества систем управления версиями файлов и с помощью простых примеров показать основы работы с SVN. Для демонстрации используется бесплатный клиент TortoiseSVN, который можно взять по адресу http://tortoisesvn.net. Данная статья адресована тем, кто ничего не знает о системах управления версиями файлов или знает, но по каким-либо причинам не пользуется ими. Данную статью можно использовать как пособие для быстрого освоения SVN до уровня, достаточного для большинства видов работ с этой системой. Для тех, кто давно и профессионально освоил подобные системы, эта статья будет не интересна.

Денис ШЕХАЛЕВ
shdv@micran.ru

Системы управления версиями файлов. Зачем они нужны?

Каждый программный/программно-аппаратный проект проходит в своей жизни несколько последовательных стадий: постановка задачи, поиск вариантов решения, написание кода и его отладка, поддержка проекта.

При этом многие разработчики и даже команды разработчиков вручную фиксируют историю изменений проекта (при его разработке и поддержке), осуществляют формирование готовых релизов (временных слепков состояния проекта), ветвление проекта (если нужна его адаптация под конкретные условия ТЗ или нужно проверить пару идей).

Естественно, такая работа с изменяющейся информацией чревата большой путаницей и частыми ошибками. Для автоматизации этой работы и были разработаны системы управления версиями файлов. Существует достаточно большое количество систем управления версиями файлов: CVS, SVN, Bazzar, Git и т. д. Выбор той или иной системы определяется личными предпочтениями разработчика и его требованиями. Но с точки зрения пользователя все эти системы похожи и решают одну задачу: контроль и управление версиями файлов.

Для того чтобы погрузиться в азы систем управления версиями файлов, ответим на простой, но в то же время важный вопрос: зачем они нужны? Этот вопрос является философским, особенно для разработчиков, полагающихся на свою память и предубеждение, что они никогда не ошибаются и лучше, чем машина, знают, что и где хранить. У таких разработчиков на жестком диске в разделе проектов часто можно найти папки, похожие на следующие:

```
Project
Project_1
Project_2
Project_test
Project_01022008
Project_for_demo
```

Но при развитии проекта неплохо делать ревизии проекта (сохранять состояние всего проекта), и тогда у таких разработчиков появляется что-то вроде вот этого:

```
Project_arh
        01022008
        01022007
        01042007
```

Причем, так как файлов много, то проекты начинают архивировать, и в таких папках лежат архивы. Автор прекрасно разбирается, что лежит в каждой из папок и зачем они нужны, естественно, полагая, что он всегда будет это помнить. Так происходит в краткосрочной перспективе, пока проект только один и автор в одиночестве работает с ним постоянно без перерывов.

Но как только автор вынужден оторваться от проекта или проектов становится несколько, то память начинает сдавать. Автор забывает, какая папка к чему относится и каково состояние проекта в конкретной папке.

При откате назад по ревизии автору нужно сначала выяснить, какую же ревизию использовать (вряд ли он делал полное описание всех изменений ревизии), для этого надо пройтись вручную по ревизиям и найти требуемую. Если же нужно сравнить ревизии, например, для выяснения, почему перестал работать проект, то надо вручную взять эти ревизии и искать, чем они отличаются.

Ситуация усугубляется, когда автор начинает заимствовать исходные коды одного проекта в другом. Надо помнить, откуда был взят файл, и если в нем будет найдена ошибка, то методом ручного копирования его нужно будет обновить в изначальном проекте.

Но вот автор заболел, уволился или просто не хочет больше работать над этим проектом, и нужно передать проект другому разработчику. Что в этом случае делает автор? Обычно люди такого типа говорят: «Вот тебе все исходные коды, разберешься» — и отдают ворох ревизий проекта, чаще всего без сопроводительной документации и разработчику, который не в курсе темы.

Поставьте себя на место того разработчика, которому поручили вести проект, и этот проект нужно было сдать вчера. В результате банальное наращивание функциональности проекта или поиск неисправностей и нерабочих участков проекта превращается в сплошную головную боль. Но разработчика-автора это мало волнует, он свой проект отдал и это его не касается.

Чем помогают системы контроля версий в данном случае? Как следует из названия, они берут на себя работу по управлению вер-

сиями и ревизиями исходных файлов проекта, то есть:

1) больше не нужно помнить обо всех исправленных ошибках и модификациях, все это есть в системе;
2) нет смысла складировать ревизии на жестком диске, все это уже лежит в репозитории системы;
3) нет смысла плодить много папок проектов с незначительными отличиями и помнить, в чем эти отличия;
4) нет смысла переносить проекты вручную, достаточно получить их из репозитория в новом месте;
5) нет смысла вручную поддерживать когерентность последних библиотек в проектах, которые их используют.

Все это возможно благодаря тому, что эти системы автоматически (то есть без ошибок) ведут историю изменений файла. Эта история, в пределах одного репозитория, может развиваться только в одном направлении. Проще говоря, репозиторий никогда ничего не забывает. Единственное, что требуется от пользователя, это аккуратно заполнять комментарии к изменениям версий файлов для быстрой навигации по репозиторию.

Подводя итог, можно дать простой ответ на вопрос этой главы. Системы управления версиями файлов нужны для того, чтобы автоматизировать рутинную работу по отслеживанию изменений файлов и облегчить жизнь разработчику. Но не надо обольщаться, использование систем управления версиями файлов не заменяет документирование проекта, а всего лишь помогает быстро и просто получить ответы на вопросы, кто, когда, что и зачем изменял в файлах.

Введение в SVN

Subversion (SVN) — бесплатная система управления версиями файлов с открытым исходным кодом. SVN позволяет управлять файлами и каталогами, а также сделанными в них изменениями во времени. SVN предоставляет следующие возможности:

1. Контроль изменений каталогов. SVN использует «виртуальную» файловую систему с возможностями управления версиями, которая способна отслеживать изменения во времени целых структур каталогов.
2. Настоящая история версий. SVN делает возможным добавление, удаление, копирование и переименование как файлов, так и каталогов. При этом каждый вновь добавленный файл начинает жизнь с чистого листа, сохраняя собственную историю изменений.
3. Атомарная фиксация изменений. Каждый набор изменений либо попадает в хранилище целиком, либо не попадает туда вовсе. То есть если при фиксации изменений проекта произошла ошибка при обработке файла, то изменения всего проекта не будут зафиксированы.
4. Метаданные с версиями. Каждый файл и каталог имеет собственный набор свойств, представленных в виде названия и значения. Вы можете создавать и сохранять любые необходимые пары названий свойств и их значений. Свойства файлов точно так же находятся под управлением системы, как и их содержимое.
5. Единый способ работы с данными. SVN обнаруживает различия между файлами с помощью специального бинарного алгоритма, который одинаково работает как с текстовыми, так и с бинарными файлами. Файлы записываются в хранилище в сжатом виде независимо от их типа, а различия между отдельными версиями могут передаваться по сети в обоих направлениях.
6. Эффективные ветки и метки. SVN создает ветки и метки путем простого копирования проекта, используя механизм, похожий на жесткие ссылки в файловых системах. Благодаря этому операции по созданию веток и меток занимают немного времени.

Для погружения в использование SVN перечислим приводимые в статье основные термины и команды SVN:

- Репозиторий *(**repository**)* — централизованное хранилище исходных кодов, рабочих материалов и документации. Любое количество клиентов подключается к хранилищу и читает или записывает эти файлы.
- Рабочая копия (***working copy/WC***) — обычное дерево каталогов на диске компьютера, содержащее набор файлов для работы над проектом. Изменения в рабочей копии недоступны для других пользователей репозитория до тех пор, пока они не будут зафиксированы.
- ***Trunk (ствол)*** — основное направление разработки.
- ***Branch (ветка)*** — направление разработки, которое существует независимо от другого направления, но имеет с ним общую историю. Ветка всегда начинается как копия чего-либо и движется от этой точки, создавая свою собственную историю.
- ***Tag (метка)*** — срез состояния проекта в определенный момент времени. Отделяется от основного проекта явно, через создание отдельной папки.
- ***Revision*** — номер ревизии репозитория, в пределах репозитория номер ревизии является уникальной величиной.
- ***Checkout*** — команда, которая выполняет начальное извлечение файлов проекта из репозитория в ***WC*** и помещает эти файлы под контроль SVN.
- ***Commit*** — команда, которая выполняет фиксацию изменений файлов проекта в ***WC*** в репозиторий.
- ***Update*** — команда, которая выполняет синхронизацию файлов проекта в ***WC*** с файлами проекта из репозитория.
- ***Revert*** — команда, которая выполняет отмену любых изменений в файлах проекта в ***WC***.
- ***Merge*** — команда, которая выполняет слияние файлов из разных веток проекта и помещает результат слияния в ***WC***.
- ***Conflict*** — ситуация, возникающая при фиксации изменений, когда одни и те же файлы изменяли несколько разработчиков.
- ***Resolve*** — набор правил по разрешению возникающих конфликтов.
- ***Import*** — команда для быстрого копирования дерева файлов в репозиторий.
- ***Export*** — команда для быстрого копирования файлов проекта из репозитория.
- ***Switch*** — команда, которая выполняет переключение ***WC*** на другую ветку разработки.
- ***Create***, ***Add***, ***Delete***, ***Copy***, ***Move***, ***Rename*** — команды для управления файлами и папками в репозитории или ***WC***.

Работа с SVN рассмотрена на основе демонстрационного репозитория ***demo_repo*** и тестового проекта ***demo_project***, который изначально содержит один файл ***readme.txt***. Положим, что с проектом работают два программиста: Вася и Петя.

В статье содержатся полезные заметки о практической работе с SVN. Заметки составлены на основе личного опыта использования SVN, выделены в отдельные абзацы и помечены следующим образом:

- важное замечание;
- подсказка;
- примечание;
- совет.

Кому-то они покажутся не важными, но для начинающих использовать SVN рекомендую эти заметки учитывать. Так вы избежите неприятных сюрпризов при первоначальном знакомстве с SVN. Итак, начнем с азов.

Репозиторий SVN

Основными объектами при работе с SVN являются рабочая копия (WC) и репозиторий.

В репозитории SVN хранятся все структуры папок и файлов. Репозиторий хранит все изменения, зафиксированные в нем, с момента создания.

Для отслеживания изменений во времени каждой операции, которая изменяет содержимое репозитория, присваивается уникальный «номер ревизии», запоминаются время и автор изменения. Все ревизии папок и файлов в репозитории доступны любому пользователю.

SVN запоминает все изменения, даже самые небольшие. Для облегчения поиска нужного состояния проекта рекомендуется заполнять строку комментария к любым изменениям в репозитории. Пустая строка комментария к фиксации изменений является признаком дурного тона. Данные виды фиксаций очень сложно отслеживать и искать.

В случае удаления папок или файлов из репозитория они будут удалены только в текущей ревизии. И в случае необходимости папки и файлы могут быть легко восстановлены.

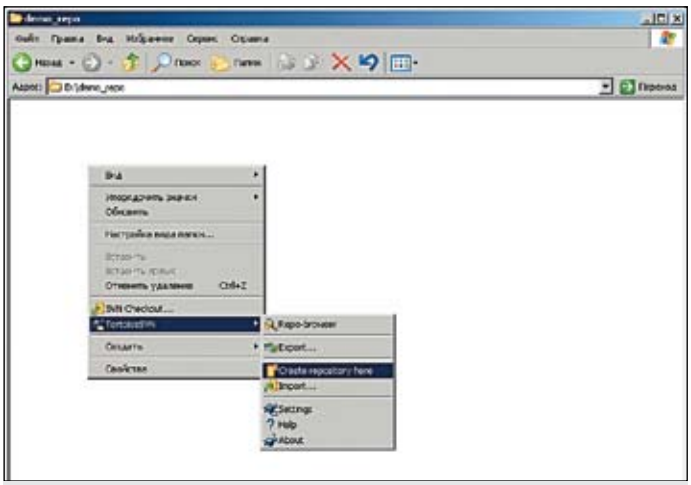

Рис. 1. Создание локального репозитория

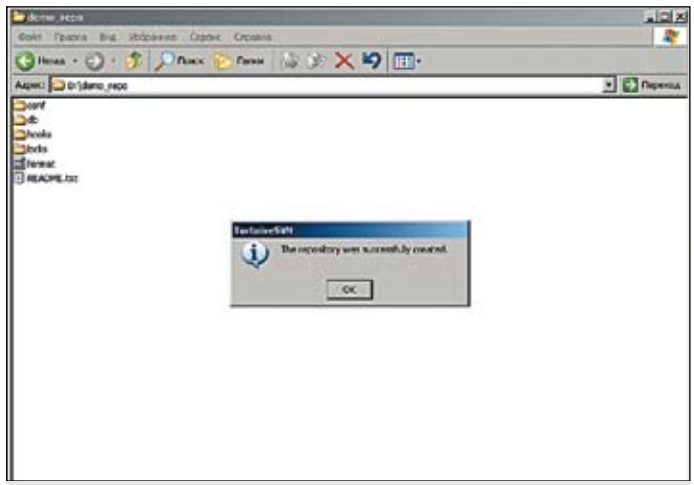

Рис. 2. Локальный репозиторий SVN

Основной областью работы пользователя является рабочая копия. Любые изменения папок, файлов и их содержимого в рабочей копии недоступны для других пользователей до тех пор, пока эти изменения не будут зафиксированы в репозитории.

Добавлять, перемещать и удалять папки и файлы проекта лучше в рабочей копии. Использовать для этих целей репозиторий не рекомендуется. Использовать возможности репозитория для целей управления папками и файлами можно только в случае, если нужное действие сложно сделать в рабочей копии. К таким действиям относится копирование и перемещение папок и файлов вне пределов рабочей копии.

При работе с репозиторием помните, что у папок и файлов есть история изменений. Если вы удалите файл и создадите новый файл с таким же именем, это будут два разных файла, с непересекающейся историей.

Подсказка. *Не стоит в комментариях писать дифирамбы результатам вашей работы. Комментарий должен кратко описывать вашу работу и ее результаты. Если комментарий получается слишком длинный, то стоит рассмотреть вопрос фиксации проекта по частям. Дифирамбы можете писать в сопроводительной документации на проект.*

Создание репозитория

Репозиторий SVN может быть как сетевым, так и локальным. Сетевые репозитории нужны для использования командой разработчиков или в случае, когда необходим доступ к репозиторию с другого компьютера. Локальные репозитории удобно использовать тогда, когда разработчик один. В данной статье, несмотря на то что разработчиков двое (Вася и Петя), используется локальный репозиторий ***demo_repo***. Это сделано потому, что цель статьи — показать основы работы с SVN, а для этого локального репозитория более чем достаточно. Локальный репозиторий можно сделать в любом месте на любом жестком диске. Для этого нужно создать папку репозитория, в нашем случае это ***D:\demo_repo***, и выполнить в ней команду ***Create repository here*** (рис. 1). Все, репозиторий создан (рис. 2). Можно приступать к работе над проектом.

Важное замечание. *Операции над папками репозитория, в обход команд клиента SVN, могут привести к повреждению репозитория и возможной потере информации в нем. Репозиторий использует специальный алгоритм хранения файлов. При обновлении версии файла он не создает новый файл, а создает файл разницы между этими двумя файлами. Эта разница и сохраняется в репозитории. Это сделано для экономии места и экономии трафика (если используется сетевой репозиторий).*

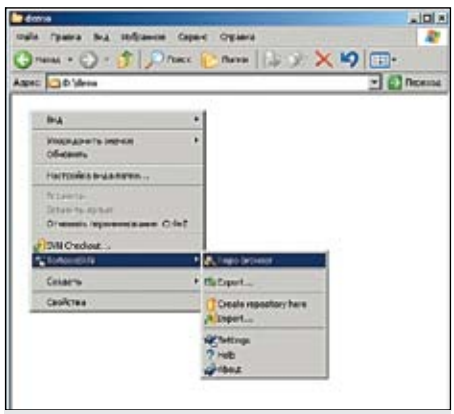

Рис. 3. Вызов браузера репозитория

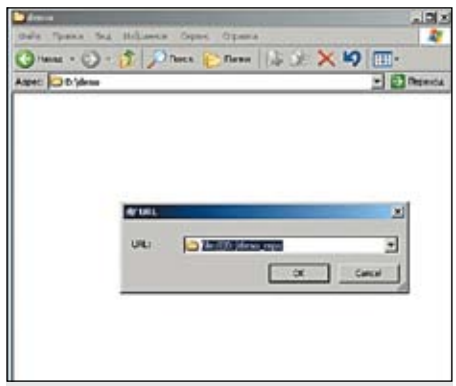

Рис. 4. Диалоговое окно задания используемого репозитория

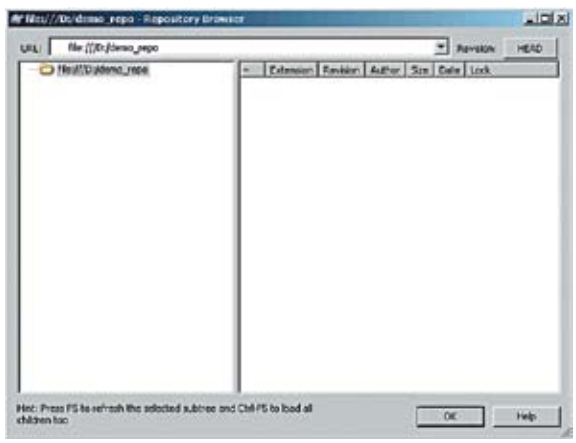

Рис. 5. Начальное состояние нового локального репозитория

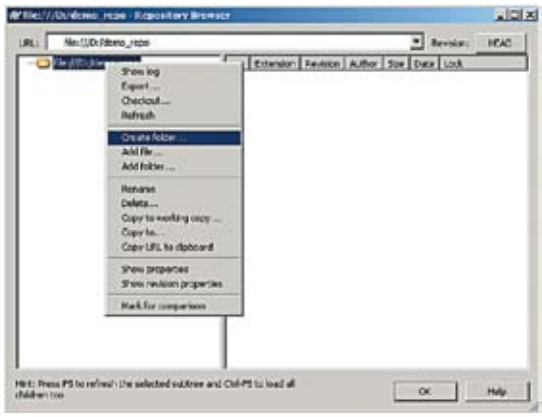

Рис. 6. Создание папки непосредственно в репозитории

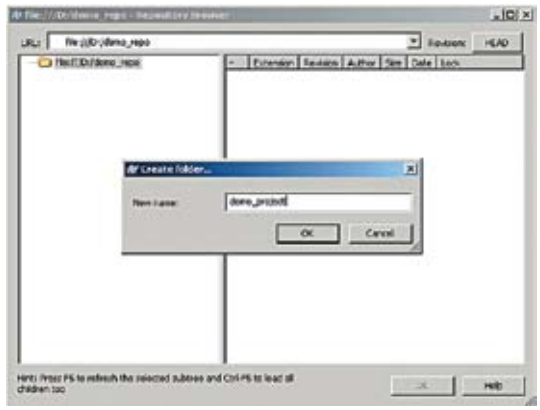

Рис. 7. Диалоговое окно задания имени создаваемой папки

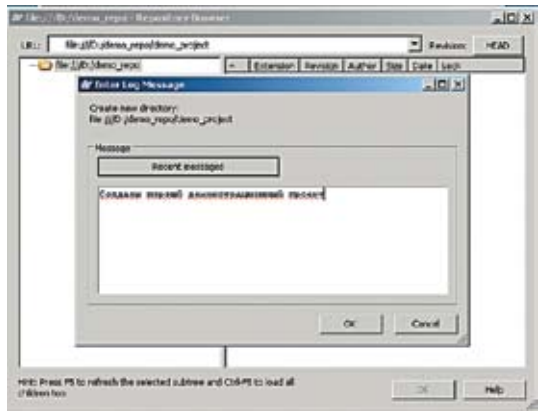

Рис. 8. Диалоговое окно для задания комментария при создании папки

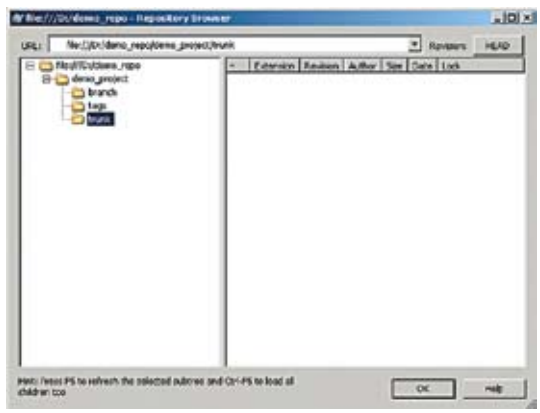

Рис. 9. Состояние репозитория после создания проекта непосредственно в репозитории

Работа с репозиторием

Работа с репозиторием является обязательной составляющей работы с проектами, находящимися под контролем SVN. Для работы с репозиторием используется браузер репозитория (repo-browser). Чтобы им воспользоваться, заходим в корневую папку любого жесткого диска и путем нажатия правой кнопки мыши запускаем браузер (рис. 3).

Для просмотра репозитория нужно указать, к какому именно репозиторию будет обращение (рис. 4). Префикс ***file:///*** указывает на то, что репозиторий ***D:\demo_repo*** является локальным. Наш репозиторий пока пуст (рис. 5).

Рис. 10. Проект на жестком диске перед импортом в репозиторий

Важное замечание. *Будьте осторожны при работе с браузером репозитория: несмотря на то что в репозитории ничего нельзя удалить навсегда, вы можете случайно нарушить работу ваших коллег.*

Создание проекта

Для того чтобы поместить свои папки и файлы под контроль SVN, необходимо создать первоначальный проект. Сделать это можно двумя способами.

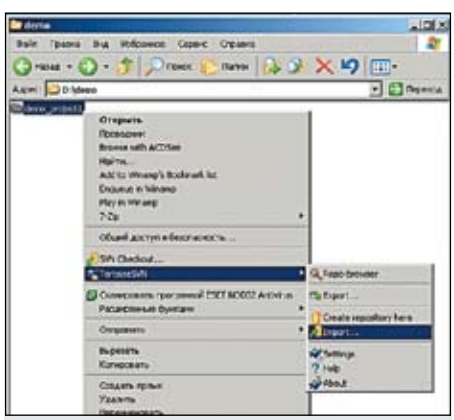

Рис. 11. Импорт проекта в репозиторий

Создание проекта в репозитории

Этот способ подходит для создания изначально пустого проекта. Заходим в репозиторий. В корневой папке репозитория, используя команду ***CreateFolder*** (рис. 6), создаем корневую папку нашего проекта ***demo_project*** (рис. 7). Обязательно оставляем комментарий к действиям по изменению репозитория (рис. 8).

Подобную последовательность шагов проделываем для обязательных папок, назначение которых описано ниже, в итоге получим пустой проект в репозитории (рис. 9).

Импорт проекта

Этот способ больше подходит для случая, когда есть готовый проект и нужно поместить его под контроль SVN. Возьмем тестовый проект ***demo_project1*** (рис. 10). С помощью команды ***Import*** импортируем его в репозиторий (рис. 11). Не забываем про указание имени папки, в которую мы импортируем проект, и о комментарии к изменению репозитория (рис. 12).

После окончания импорта в окне протокола видно, что происходило с папками и файлами проекта (рис. 13). Текущее состояние

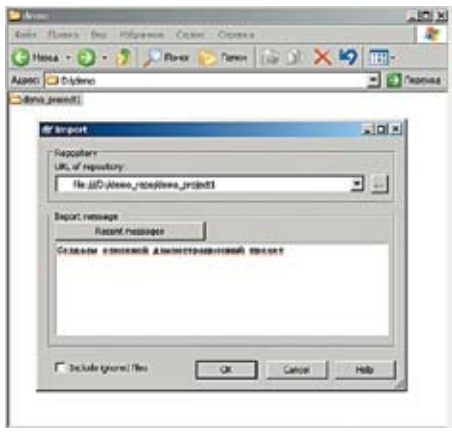

Рис. 12. Диалоговое окно для задания комментария при импорте проекта

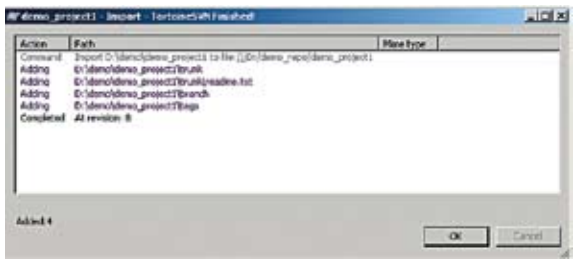

Рис. 13. Окно протокола импорта проекта

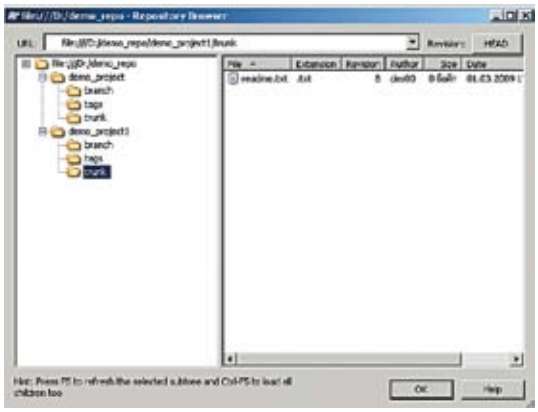

Рис. 14. Состояние репозитория после импорта проекта

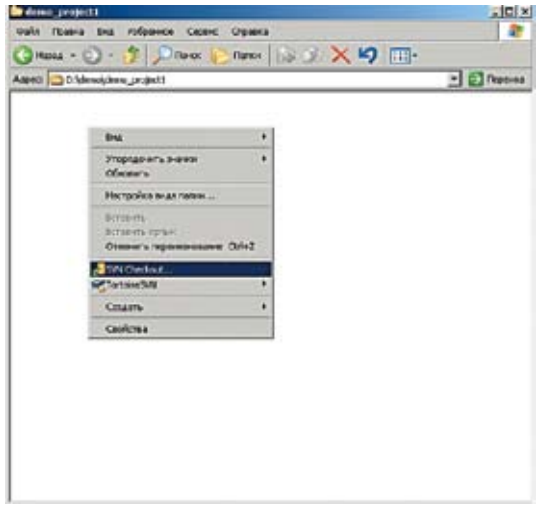

Рис 15. Создание рабочей копии проекта в папке на диске

репозитория показано на рис. 14. Как видите, все папки и файл проекта были добавлены в репозиторий одной командой. Тут нужно уделить внимание тому, что при импорте проекта в репозиторий будут добавлены все файлы, которые находятся в папках проекта.

То, что импорт проекта завершен удачно, не означает, что папка, которую вы использовали для импорта проекта, является теперь рабочей копией. Любые изменения в этой папке не могут быть зафиксированы в репозитории SVN. Работа с рабочими копиями описана в следующей части статьи.

Важное замечание. *При импорте проекта убедитесь в том, что проект с таким именем не существует, и в том, что вы создаете проект в корневой папке репозитория.*

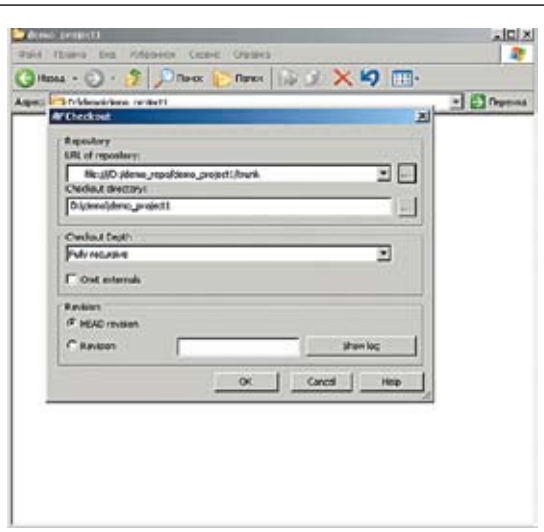

Рис. 16. Диалоговое окно для задания источника папок/файлов для рабочей копии проекта

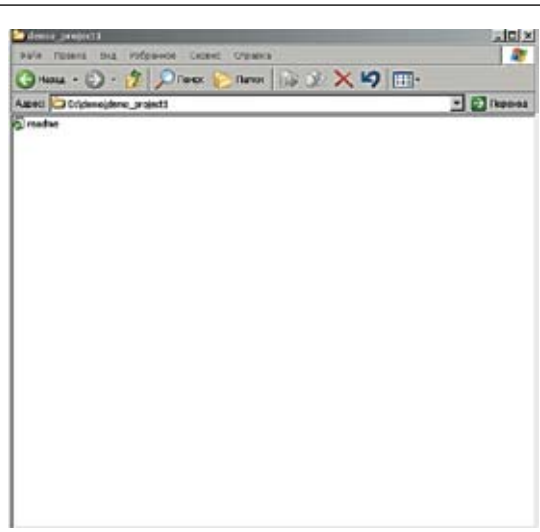

Рис. 17. Созданная рабочая копия проекта, которая находится под контролем SVN

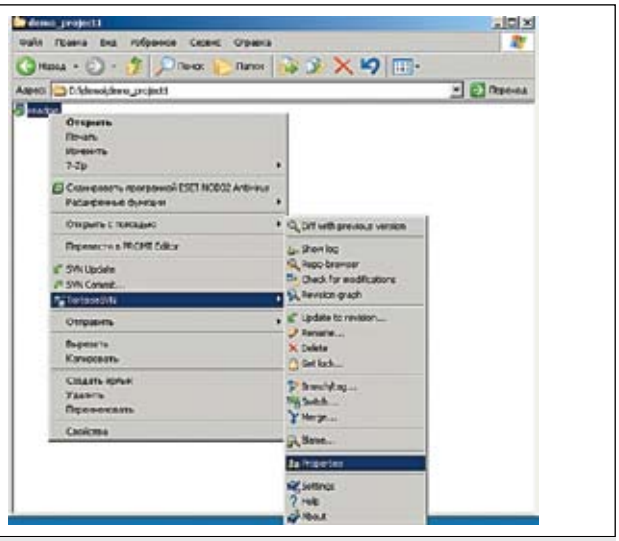

Рис. 18. Доступ к SVN-свойствам файла для их просмотра

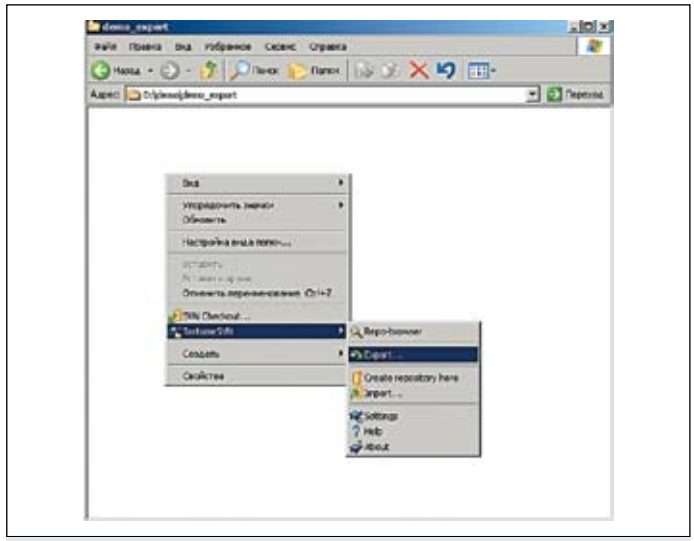

Рис. 19. Экспорт проекта из репозитория в папку на диске

Важное замечание. *При импорте проекта в репозиторий будут добавлены все файлы, которые находятся в папках проекта. Поэтому перед импортом проверьте проект на предмет чистки от ненужных файлов. Какие файлы не имеет смысла хранить в SVN, будет рассмотрено ниже.*

Подсказка. *При создании проектов очень удобно использовать шаблоны проектов. В таком случае, используя шаблон, вы импортируете проект в репозиторий. Это избавит вас от необходимости каждый раз создавать повторяющиеся от проекта к проекту папки и файлы. Например, используемый мной шаблон для проектов на ПЛИС:*

```
ProjectName
        branch
        tag
        trunk
                beh         behaviour models of project units
                core        external not under development IP cores
                doc         project documentation
                include     common header files
                netlist     ready netlist of project modules
                quartus     quartus projects files
                rtl         rtl models of project units
                sim         simulation scripts for verivication
                testbench   verification enviroment for project
                readme.txt
```

Подсказка. *Если вы неправильно создали проект (не те имена папок, не туда поместили по ошибке), то сначала удалите командой* ***Delete*** *все неправильно созданное в репозитории и только потом создавайте проект заново. Или воспользуйтесь командой* ***Move*** *для того, чтобы перенести папки проекта в другое место (в* ***TortoiseSVN*** *в браузере репозитория команда* ***Move*** *выполняется по технологии drag-and-drop).*

Создание рабочей копии

Для того чтобы начать работу с проектом, нужно создать рабочую копию. Для этого создаем корневую папку проекта на диске. Заходим в эту папку и используем команду ***Checkout*** (рис. 15).

С помощью браузера репозитория выбираем интересующий нас проект, папки и файлы в этом проекте. С помощью браузера ревизий ***Show log*** выбираем нужный номер ревизии (рис 16). В данном случае выбрана ***Head*** (головная/последняя) ревизия. Нажимаем на ***OK***. Все, рабочая копия создана и находится под контролем SVN, о чем говорит иконка SVN на файлах и папках (рис. 17).

Теперь можно свободно изменять, удалять, модифицировать папки и файлы проекта, добавлять новые папки и файлы, не опасаясь того, что ваши изменения будут мешать работе ваших коллег.

Важное замечание. *При* ***Checkout*** *проекта из репозитория в проекте создаются служебные папки* ***.svn****. Изменять или удалять их нельзя. В случае повреждения служебных папок информация о сделанных вами изменениях не может быть зафиксирована в репозитории!!!*

Подсказка. *Узнать, к какому репозиторию и проекту относится папка или файл, можно из свойств папки/файла, которая находится под контролем репозитория (рис. 18).*

Подсказка. *Если нужно передать куда-то файлы проекта, воспользуйтесь командой* ***Export*** *(рис. 19). Эта команда не создает служебных папок.*

Продолжение следует