МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования «Саратовский государственный

технический университет имени Гагарина Ю.А.»

Реферат

По дисциплине «Вычислительные системы»

На тему «Система контроля версий SVN»

Выполнил  
 Студент группы \_\_мИВЧТ-21\_\_\_\_

№ зачетной книжки \_\_111102\_\_\_\_

Есенгалиев Бауыржан Гибатович

Проверил

кандидат физико-математических наук,

доцент кафедры ИСТ

Кинцель Дмитрий Александрович

Саратов 2013

Содержание

Введение………………………………………………………………………………………3

1 Понятие системы контроля версий………………………………………………………..4

1.1 Распределённые системы управления версиями………………………………..5

2 Виды систем контроля версий……………………………………………………………..6

2.1 CVS - система управления параллельными версиями (www.nongnu.org/cvs)...6

2.2 Система управления версиями Aegis (www.aegis.sourceforge.net).....................7

2.3 Система управления версиями Bazaar (bazaar.canonical.com)…………………8

2.4 Система управления версиями Arch (https://www.archlinux.org/svn/)................9

2.5 Система управления версиями Perforce (www.perforce.com)..............................9

2.6 Система управления версиями Subversion (www.subversion.tigris.org)...........10

2.7 TortoiseSVN………………………………………………………………………11

2.7.1 Установка Tortoise SVN………………………………...……………..12

2.7.2 Основные Возможности…………………………………………....…15

2.7.3 Скачивание репозитория SVN (Checkout)…………………………...15

2.7.4 Обновление скачанного репозитория (Update)………………………20

2.7.5 Сохранение своих изменений в репозиторий (Commit)………….…21

2.7.6 Работа с репозиторием напрямую на сервере………………………..25

2.7.7 Создание веток или меток внутри репозитория……………………..26

2.7.8 Откат изменений в рабочей копии……………………………………26

2.7.9 Работа с утилитой Log Viewer………………………………………...27

2.7.10 Разрешение конфликтов…………………………………..…………28

Заключение………………………………………………………………………………......30

Список использованной литературы………………………………………………………31

Введение

Системы контроля версий стали неотъемлемой частью жизни не только разработчиков программного обеспечения, но и всех людей, столкнувшихся с проблемой управления интенсивно изменяющейся информацией, и желающих облегчить себе жизнь. На сегодняшний день, работая с огромным количеством электронных документов, бывает нужным видоизменять документы на различных стадиях их разработки. За время своего существования электронный документ может быть подвержен большому количеству изменений. Однако часто так бывает, что для дальнейшей работы необходима не только последняя версия документа, но и различные предыдущие варианты. Конечно же, можно хранить несколько различных вариантов необходимого документа, но данный способ неэффективен. Нам приходится тратить большое количество времени и сил, также велика вероятность ошибки. Вследствие этого, появилось большое число различных продуктов, предлагающих широкие возможности и предоставляющих обширные инструменты для управления версиями. Данные средства именуются системами контроля версий. Существует несколько такого рода систем, каждая из которых актуальна при определенных условиях их использования.

Целью данной работы является рассмотрение различного рода систем контроля версий.  
С поставленной целью появляются следующие задачи, которые нужно также решить:

* определить понятие системы контроля версий;
* проанализировать существующие системы контроля версий.

1 Понятие системы контроля версий

Система управления версиями (Version Control System или Revision Control System) представляют собой программное обеспечение для облегчения деятельности с быстро меняющейся информацией. Система контроля версий предоставляет возможность хранить несколько вариантов одного и того же документа. При необходимости можно вернуться к старым версиям, можно узнать, кем были сделаны те или иные изменения и т.д.  
Такого рода системы в большинстве своем используются при разработке программного обеспечения, чтобы можно было хранить исходные коды разрабатываемых программ. Система контроля версий позволяет разработчикам хранить прошлые версии файлов из разработки и доставать их оттуда. Она хранит информацию о версии каждого файла (и полную структуру проекта) в коллекции, обычно называемой репозиторием. Но тем не менее данные системы могут использоваться и в других областях знаний, которые включают в себя огромное количество часто изменяющихся электронных документов.   
Внутри репозитория могут быть несколько параллельных линий разработки, обычно называемых ветвями. Это может быть полезно для хранения стабильной или выпущенной версии ветви, одновременно продолжая работу над рабочей версией. Другой вариант – это открыть выделенную ветвь для работы над экпериментальной возможностью.   
Система контроля версий также позволяет пользователям дать ярлык снимку ветви (часто называемых как тэги) для легкого доставания. Это полезно для обозначения индивидуальных релизов или самых свежих рабочих версий, предназначенных для использования.

Использование системы контроля версий безусловно обязательно для разработчика, если проект больше нескольких сот строк или если для проекта совместно работают несколько разработчиков. Использование хорошей системы контроля версий определенно лучше, нежели использование узко-направленных специальных методов, которые используют некоторые разработчики для управления различными ревизиями своего кода. Большинство систем управления версиями используют централизованную модель, когда имеется единое хранилище документов, управляемое специальным сервером, который и выполняет большую часть функций по управлению версиями. Пользователь, работающий с документами, должен сначала получить нужную ему версию документа из хранилища; обычно создаётся локальная копия документа, по другому «рабочая копия». Может быть получена последняя версия или любая из предыдущих, которая может быть выбрана по номеру версии или дате создания, иногда и по другим признакам. После того, как в документ внесены нужные изменения, новая версия помещается в хранилище. В отличие от простого сохранения файла, предыдущая версия не стирается, а тоже остаётся в хранилище и может быть оттуда получена в любое время. Сервер может использовать т. н. дельта-компрессию - такой способ хранения документов, при котором сохраняются только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Очень часто над одним и тем же проектом трудится несколько человек. Если один из них будет изменять исходный файл, и одновременно с этим другой человек будет выполнять аналогичную операцию, то возможна такая ситуация, что какие-то изменения могут не сохраниться. Системы контроля версий работают с такого рода проблемами и имеют определенный перечень их решения. В большинстве своем эти системы могут автоматически объединять такого рода изменения, которые делают разные члены команды разработчиков. Но стоит отметить, что такого рода объединения чаще всего выполняется для текстовых файлов и с определенным условием: изменения происходили в разных частях файла. Данное ограничение имеет место, поскольку в большинстве своем системы контроля версий направлены на поддержку процесса разработки программных продуктов, а изначальные коды находятся в текстовых файлах. В случае если автоматически выполнить объединение не получилось, то система предлагает исправить ситуацию вручную. Зачастую осуществить объединение не возможно ни с помощью системы, ни вручную. Ярким примером является ситуация когда формат файла очень сложный или неизвестен. Отдельные системы контроля версий предоставляют возможность пользователю заблокировать файл в хранилище. Данная операция не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла и обеспечивает, таким образом, исключительный доступ только тому пользователю, который работает с документом.

* 1. Распределённые системы управления версиями

Существуют системы управления версиями, которые, вместо традиционной клиент-серверной, используют распределённую модель. Такие системы, в общем случае, не нуждаются в централизованном хранилище: вся история изменения документов хранится на каждом компьютере. Фактически, каждый компьютер, помимо рабочей копии, хранит локальную копию всего хранилища. В некоторых системах рабочая копия сама является хранилищем. Когда пользователь такой системы выполняет обычные действия, такие как извлечение определённой версии документа, создание новой версии и тому подобное, он работает со своей локальной копией хранилища. По мере внесения изменений, копии, принадлежащие разным разработчикам, начинают различаться и возникает необходимость в их синхронизации. Такая синхронизация может осуществляться с помощью обмена патчами или так называемыми наборами изменений между пользователями. Описанная модель аналогична созданию отдельной ветки для каждого разработчика в классической системе управления версиями. Пока разработчик изменяет только свою ветвь, его работа не влияет на других участников проекта и наоборот. Однако при необходимости выполнить слияние ветвей (или синхронизацию локальных хранилищ в распределённой модели) могут возникнуть конфликты.

Основное преимущество распределённых систем заключается в их гибкости. Каждый разработчик может вести работу независимо, как ему удобно, сохраняя промежуточные варианты документов и передавая результаты другим участникам, когда это нужно. При этом обмен наборами изменений может осуществляться по различным схемам. В небольших коллективах участники работы могут обмениваться изменениями по принципу «каждый с каждым», за счет чего отпадает необходимость в создании выделенного сервера. Крупное сообщество, наоборот, может использовать централизованный сервер, с которым синхронизируются копии всех его участников. Возможны и более сложные варианты - например, с созданием групп для работы по отдельным направлениям внутри более крупного проекта. Распределенная система контроля версий позволяет клонировать удаленный репозиторий, производя точную копию. Она также позволяет распростанять изменения из одного репозитория на другой. В нераспределенных VCS разработчику нужен репозиторий для того, чтобы зафиксировать изменения в нем. С распределенной VCS каждый разработчик может склонировать главный репозиторий, поработать над ним и потом распространить свои изменения на главный репозиторий.

1. Виды систем контроля версий

Система контроля версий может быть любой формы и размеров, но есть основные положения об их архитектуре. Некоторые системы поддерживают Атомарные Фиксации, которые значат, что состояние всего репозитория меняется полностью. Без Атомарных Фиксаций, каждый файл или часть меняется отдельно и поэтому состояние всего репозитория в любой точке не может быть зафиксировано.  Большинство обычных VCS (Version Control System ) систем позволяют объединять изменения между ветвями. Это значит, что изменения, зафиксированные в одной ветви, будут зафиксированы в главной линии или в другой ветви с помощью одной автоматической или, по крайней мере, полуавтоматической операцией.

Рассмотрим несколько вариантов современных систем контроля версии. После на примере TortoiseSVN рассмотрим инструкцию по установке инструмента, инструкцию по использованию основных возможностей инструмента и т.д.

2.1 CVS - система управления параллельными версиями [(www.nongnu.org/cvs)](http://www.nongnu.org/cvs)

Система управления параллельными версиями (Concurrent Versions System) – логическое развитие системы управления пересмотрами версий (RCS), использующая ее стандарты и алгоритмы по управлению версиями, но значительно более функциональная, и позволяющая работать не только с отдельными файлами, но и с целыми проектами.

CVS основана на технологии клиент-сервер, взаимодействующих по сети. Клиент и сервер также могут располагаться на одной машине, если над проектом работает только один человек, или требуется вести локальный контроль версий.

Работа CVS организована следующим образом. Последняя версия и все сделанные изменения хранятся в репозитории сервера. Клиенты, подключаясь к серверу, проверяют отличия локальной версии от последней версии, сохраненной в репозитории, и, если есть отличия, загружают их в свой локальный проект. При необходимости решают конфликты и вносят требуемые изменения в разрабатываемый продукт. После этого все изменения загружаются в репозиторий сервера. CVS, при необходимости, позволяет откатываться на нужную версию разрабатываемого проекта и вести управление несколькими проектами одновременно.

Приведем основные достоинства и недостатки системы управления параллельными версиями.

**Достоинства:**

1. Несколько клиентов могут одновременно работать над одним и тем же проектом.

2. Позволяет управлять не одним файлом, а целыми проектами.

3. Обладает огромным количеством удобных графических интерфейсов, способных удовлетворить практически любой, даже самый требовательный вкус.

4. Широко распространена и поставляется по умолчанию с большинством операционных систем Linux.

5. При загрузке тестовых файлов из репозитория передаются только изменения, а не весь файл целиком.

**Недостатки:**

1. При перемещении или переименовании файла или директории теряются все, привязанные к этому файлу или директории, изменения.

2. Сложности при ведении нескольких параллельных веток одного и того же проекта.

3. Ограниченная поддержка шрифтов.

4. Для каждого изменения бинарного файла сохраняется вся версия файла, а не только внесенное изменение.

5. С клиента на сервер измененный файл всегда передается полностью.

6. Ресурсоемкие операции, так как требуют частого обращения к репозиторию, и сохраняемые копии имеют некоторую избыточность.

**Выводы:**

Несмотря на то, что CVS устарела и обладает серьезными недостатками, она все еще является одной из самых популярных систем контроля версий и отлично подходит для управления небольшими проектами, не требующих создания нескольких параллельных версий, которые надо периодически объединять. CVS можно порекомендовать, как промежуточный шаг в освоении работы систем контроля версий, ведущий к более мощным и современным видам таких программ.

2.2. Система управления версиями Aegis [(www.aegis.sourceforge.net)](http://www.aegis.sourceforge.net/)

Aegis, созданная Питером Миллером в 1991 году, является первой альтернативой централизованным системам управления версиями. Все операции в ней производятся через файловую систему Unix. К сожалению, в Aegis нет встроенной поддержки работы по сети, но взаимодействия можно осуществлять, используюя такие протоколы, как NFS, HTTP, FTP.

Основная особенность Aegis – это способ контроля вносимых в репозиторий изменений.

Во-первых, перед занесением каких-либо изменений, они должны обязательно пройти ряд тестов. И если нововведения в исходный код программы не проходят тесты, то требуется либо добавлять новые тесты, либо исправлять возможные ошибки в исходном коде.

Во-вторых, перед внесением изменений в основную ветку разрабатываемого проекта, они должны быть одобрены обозревателем.

В третьих, предусмотрена иерархия доступа к репозиторию, основанная на системе прав доступа Unix-подобных операционных систем к файлам.

Все это делает использование системы контроля версий Aegis надежным, но крайне сложным, и даже хорошо проработана документация не сильно это облегчает.

Выделим основные достоинства и недостатки системы контроля версий Aegis.

**Достоинства:**

1. Надежный контроль корректности загружаемых изменений.

2. Возможность предоставлять различные уровни доступа к фалам репозитория, что дает приличный уровень безопасности.

3. Качественная документация.

4. Возможность переименовывать файлы, сохраненные в репозитории, без потери истории изменений.

5. Возможность работы с локальным репозиторием, если отсутствует сетевой доступ к главному репозиторию.

**Недостатки:**

1. Отсутствие встроенной поддержки сетевого взаимодействия.

2. Сложность настройки и работы с репозиторием.

3. Слабые графические интерфейсы.

**Выводы:**

Сложность работы Aegis может оттолкнуть пользователей от использования систем контроля версий, поэтому ее нельзя рекомендовать для ознакомления или ведения небольших программных проектов. Однако, она имеет ряд преимуществ, которые могут быть полезны в некоторых специфических ситуациях, особенно, когда требуется жесткий контроль за качеством разрабатываемого программного обеспечения.

2.3. Система управления версиями Bazaar [(bazaar.canonical.com)](http://bazaar.canonical.com/)

Bazaar – распределенная, свободно распространяемая система контроля версий, разрабатываемая при поддержке компании Canonical Ltd. Написана на языке Python и работает под управлением операционных систем Linux, Mac OS X и Windows.

В отличие от Git и Mercurial, создаваемых для контроля версий ядра операционной системы Linux, а поэтому ориентированных на максимальное быстродействие при работе с огромным числом файлов, Bazaar ориентировался на удобный и дружественный интерфейс пользователя. Оптимизация скорости работы производилось уже на втором этапе, когда первые версии программы уже появились.

Как и во многих других системах контроля версий, система команд Bazaar’a - очень похожа на команды CVS или Subversion, что, впрочем, неудивительно, так как обеспечивает удобный, простой и интуитивно понятный интерфейс взаимодействия с программой.

Приятно, что огромное внимание уделяется работе с ветками проектов (создание, объединение веток и т.д.), что очень важно при разработке серьезных проектов и позволяет проводить доработки и эксперименты без угрозы потери основной версии программного обеспечения.

Большой плюс этой системе контроля версий дает возможность работы с репозиториями других систем контроля версий, таких как Subversion или Git.

Кратко приведем наиболее значительные достоинства и недостатки этой системы контроля версий.

**Достоинства:**

1. Кросплатформенная поддержка.

2. Удобный и интуитивно понятный интерфейс.

3. Простая работа с ветками проекта.

4. Возможность работы с репозиториями других систем контроля версий.

5. Великолепная документация.

6. Удобный графический интерфейс.

7. Чрезвычайная гибкость, позволяющая подстроится под нужды конкретного пользователя.

**Недостатки:**

1. Более низкая скорость работы, по сравнению с git и mercurial, но эта ситуация постепенно исправляется.

2. Для полноценного функционирования необходимо устанавливать достаточно большое количество плагинов, позволяющих полностью раскрыть все возможности системы контроля версий.

**Выводы:**

Bazaar – удобная система контроля версий с приятным интерфейсом. Хорошо подходит для пользователей, которых отталкивает перспектива работы с командной строкой. Множество дополнительных опций и расширений позволит настроить программу под свои нужды. Схожесть системы команд с Git и Subversion, и возможность работы напрямую с их репозиториями, - сделает переход на Bazaar быстрым и безболезненным. Об успешности базара говорит и тот факт, что ей пользуются разработчики Ubuntu Linux.

2.4. Система управления версиями Arch (https://www.archlinux.org/svn/)

Arch – распределенная система контроля версий, созданная Томом Лордом. Изначально она создавалась для решения проблем CVS, что им вполне удалось.

Arch осуществляет атомарные операции по сохранению изменений в репозиторий, т.е. исключает ситуацию скачивания репозитория, когда часть изменений загружена, а часть еще не успела загрузиться.

Поддерживаются возможности ветвления версий проекта и объединение отдельных веток, переименование и перемещение файлов и каталогов с сохранением истории изменений, и многие другие приятные возможности.

Не требует специального сервиса для сетевого репозитория и может использовать такие протоколы, как FTP, SFTP или WebDAV и так далее.

Но, к сожалению, поддерживается только UNIX – системами, однако, перевод Arch под другие операционные системы не должен составлять труда.

Трудно отметить какие то принципиально лучшие качества, по сравнению с другими распределенным системами контроля версий, такими как git, mercurial, bazaar, так что если есть выбор, то лучше использовать что-то более мощное и распространенное.

2.5. Система управления версиями Perforce (http://perforce.com)

Одним из коммерческих программ с централизованной системы контроля версий – Perforce, разработанной компанией Perforce Software.

Система Perforce имеет клиент-серверную организацию и позволяет одновременно управлять несколькими проектами, создавая для каждого проекта свой репозиторий.

Perforce – кроссплатформенная система. Существуют версии, способные работать под управлением операционных систем Unix, Mac OS X, Microsoft Windows.

Для работы с системой контроля версий можно использовать, как консоль, так и специально разработанный графический интерфейс.

Серьезное преимущество Perforce’у дает возможность интегрироваться со множеством средств разработки программного обеспечения и такими приложениями, как Autodesk 3D Studio Max, Maya, Adobe Photoshop, Microsoft Office, Eclipse, emacs и многими другими.

Поддержка возможности создания веток версий проекта, гибко управлять ими, объединять, откатываться на предыдущие ревизии, - делает Perforce вполне конкурентно способной системой и способствуют ее широкому распространению. Однако, это продукт - коммерческий, что несколько сужает область его применения и сдерживает распространение. В основном, он используется в больших коммерческих компаниях, для которых важна не только функциональность, но и своевременная техническая поддержка.

2.6. Система управления версиями Subversion [(www.subversion.tigris.org)](http://www.subversion.tigris.org/)

Subversion – эта централизованная система управления версиями, созданная в 2000 году и основанная на технологии клиент-сервер. Она обладает всеми достоинствами CVS и решает основные ее проблемы (переименование и перемещение файлов и каталогов, работа с двоичными файлами и т.д.). Часто ее называют по имени клиентской части – SVN.

Принцип работы с Subversion очень походит на работу с CVS. Клиенты копируют изменения из репозитория и объединяют их с локальным проектом пользователя. Если возникают конфликты локальных изменений и изменений, сохраненных в репозитории, то такие ситуации разрешаются вручную. Затем в локальный проект вносятся изменения, и полученный результат сохраняется в репозитории.

При работе с файлами, не позволяющими объединять изменения, может использоваться следующий принцип:

1. Файл скачивается из репозитория и блокируется (запрещается его скачивание из репозитория).

2. Вносятся необходимые изменения.

3. Загружается файл в репозиторий и разблокируется (разрешается его скачивание из репозитория другим клиентам).

Во многом, из-за простаты и схожести в управлении с CVS, но в основном, из-за своей широкой функциональности, Subversion с успехом конкурирует с CVS и даже успешно ее вытесняет.

Однако, и у Subversion есть недостатки. Давайте рассмотрим ее слабые и сильные стороны для сравнения с другими системами управления версиями.

**Достоинства:**

1. Система команд, схожая с CVS.

2. Поддерживается большинство возможностей CVS.

3. Разнообразные графические интерфейсы и удобная работа из консоли.

4. Отслеживается история изменения файлов и каталогов даже после их переименования и перемещения.

5. Высокая эффективность работы, как с текстовыми, так и с бинарными файлами.

6. Встроенная поддержка во многие интегрированные средства разработки, такие как KDevelop, Zend Studio и многие другие.

7. Возможность создания зеркальных копий репозитория.

8. Два типа репозитория – база данных или набор обычных файлов.

9. Возможность доступа к репозиторию через Apache с использованием протокола WebDAV.

10. Наличие удобного механизма создания меток и ветвей проектов.

11. Можно с каждым файлом и директорией связать определенный набор свойств, облегчающий взаимодействие с системой контроля версии.

12. Широкое распространение позволяет быстро решить большинство возникающих проблем, обратившись к данным, накопленным Интернет-сообществом.

**Недостатки:**

1. Полная копия репозитория хранится на локальном компьютере в скрытых файлах, что требует достаточно большого объема памяти.

2. Существуют проблемы с переименованием файлов, если переименованный локально файл одним клиентом был в это же время изменен другим клиентом и загружен в репозиторий.

3. Слабо поддерживаются операции слияния веток проекта.

4. Сложности с полным удалением информации о файлах попавших в репозиторий, так как в нем всегда остается информация о предыдущих изменениях файла, и непредусмотрено никаких штатных средств для полного удаления данных о файле из репозитория.

**Выводы:**

Subversion – современная система контроля версий, обладающая широким набором инструментов, позволяющих удовлетворить любые нужды для управления версиями проекта с помощью централизованной системы контроля. В Интернете множество ресурсов посвящено особенностям Subversion, что позволяет быстро и качественно решать все возникающие в ходе работы проблемы.

Простота установки, подготовки к работе и широкие возможности позволяют ставить subversion на одну из лидирующих позиций в конкурентной гонке систем контроля версий.

2.7 TortoiseSVN

TortoiseSVN — это бесплатный Windows-клиент с открытыми исходным кодом для системы управления версиями Apache™ Subversion®. То есть TortoiseSVN управляет файлами и директориями во времени. Файлы хранятся в центральном хранилище. Хранилище больше похоже на обычный файловый сервер, кроме того он запоминает каждое изменение когда-либо сделанное в ваших файлах и директориях. Это позволяет вам восстановить старые версии ваших файлов и проверить историю изменений — как, когда и кто изменял ваши данные. Вот почему многие думают о Subversion, и вообще о системах управления версиями, как о своего рода «машине времени».

TortoiseSVN работает под операционной системой Windows XP (c Service Pack 3) или выше, и доступен как в 32-битной, так и в 64-битной версии. Установщик для 64-битной Windows также включает 32-битную часть. Что означает что вы не должны устанавливать 32-битную версию отдельно чтобы контекстное меню и оверлей TortoiseSVN работало в 32-битных приложениях.

Для скачивания Tortoise SVN необходимо перейти на официальный сайт проекта расположенный по адресу <http://tortoisesvn.net/downloads.html>. На главной странице доступны две ссылки для  32-х и 64-х битной версии продукта.

Мы должны установить и настроить сервер или иметь доступ к хранилищу Subversion. Только после этого можем установить клиентскую часть на компьютер.

2.7.1 Установка Tortoise SVN

Запустите двойным щелчком исполняемый файл, который вы только что скачали. Перед вами должно появиться следующее окно (рис 2.7.1.1). Здесь разработчики приветствуют вас и предлагают начать установку нажав на кнопку Next. Чтож так и сделаем, жмем Next.

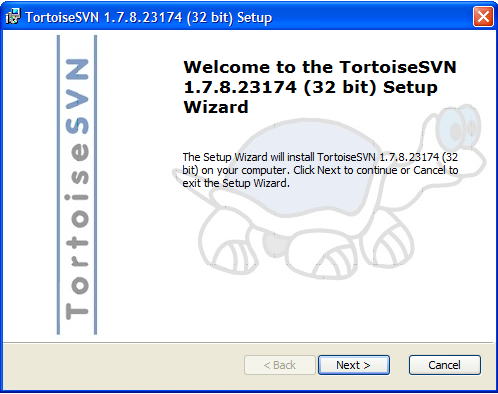


Рисунок 2.7.1.1 Окно приветствия

На следующем шаге приводится текст лицензионного соглашения, его необходимо прочитать и принять если вы желаете пользоваться программой или отклонить если оно вас не устраивает и вы хотите отменить установку (рис 2.7.1.2). Если все вас устраивает ставим переключатель на пункт I accept the terms in the License Agreement и нажимаем Next.

|  |
| --- |
| http://1.bp.blogspot.com/-Tz5D6BznSVs/UDXIwwC1bOI/AAAAAAAAAKI/rQAkUSCd1xE/s1600/05.png  Рисунок 2.7.1.2 Окно приветствия |

На следующем шаге (рис 2.7.1.3) у нас имеется возможность отключить ненужные нам компоненты. Получить описание каждого компонента можно нажав на его название и увидев справа в блоке Feature Description краткую характеристику. Выставляем нужные настройки и нажимаем Next.

|  |
| --- |
| http://1.bp.blogspot.com/-Tl3d14X4q2Y/UDXIx8YZuEI/AAAAAAAAAKQ/Ts7JZ2idcew/s1600/06.png |
| Рисунок 2.7.1.3 Выбор компонентов |
|  |

На следующем шаге (рис 2.7.1.4) нам сообщают что все готово к установке и если вы вдруг что то забыли настроить или еще раз хотите проверить свои настройки но можете нажать Back. А для установки необходимо нажать Install, что мы и делаем.

|  |
| --- |
| http://1.bp.blogspot.com/-vG8KrefHeZc/UDXIy7VfxMI/AAAAAAAAAKY/acOOiJgPlH0/s1600/07.png |

Рисунок 2.7.1.4 Окно приветствия

Начинается процесс установки (рис 2.7.1.5).

|  |
| --- |
| http://2.bp.blogspot.com/-yOOvYNLo5sc/UDXI0IxqAhI/AAAAAAAAAKg/bgCQBFYOzuU/s1600/08.png |
| Рисунок 2.7.1.5 Процесс установки |

После окончания установки вы увидите финишное окно (рис 2.7.1.6). Здесь разработчики говорят что установка закончена и вы можете начать пользоваться Tortoise SVN нажав на кнопкуFinish. Также вы можете поддержать разработчиков нажав на кнопке Donate!. На этом установка клиента TortoiseSVN закончена.

|  |
| --- |
| http://4.bp.blogspot.com/-l9sc0EmS9Fc/UDXI2D3cn2I/AAAAAAAAAKw/TXrvYsHBC9Q/s1600/10.png |

Рисунок 2.7.1.6 Окончание процесса установки

2.7.2 Основные Возможности

Одной из наиболее заметных особенностей TortoiseSVN являются пометки на значках, которые появляются для файлов в рабочей копии. Они сразу же показывают, какие файлы были изменены. (рис. 2.7.2.1)

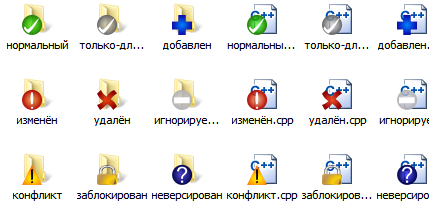


Рисунок 2.7.2.1. Проводник с пометками на значках

2.7.3 Скачивание репозитория SVN (Checkout)

Если Вы хотите скачать репозиторий какого-нибудь проекта, то Вы должны знать его URL, а также логин и пароль для доступа (если не указаны, то для анонимного доступа (только чтение) используется логин и пароль anonsvn). URL’ы могут быть:

* стандартный svn:// — подключение напрямую к Subversion-серверу;
* http:// — подключение к Subversion-серверу через веб (так работает большинство серверов, т.к. позволяет просматривать проект не имея SVN-клиента напрямую через веб-браузер);
* https:// — то же, что и http://, но через SSL;
* svn+ssh:// — svn через SSH (редко используется в настоящее время).

Допустим, что URL нам известен, поэтому чтобы скачать репозиторий, мы должны сделать следующее:

* Создаём папку, в которую будем скачивать репозитории (для удобства лучше хранить все репозитории в одной папке, например, C:\SVN).
* Заходим в неё в проводнике Windows, нажимаем правой кнопкой мыши по свободному месту и выбираем «SVN Checkout…»., как это показано на рисунке. (рис 2.7.3.1)

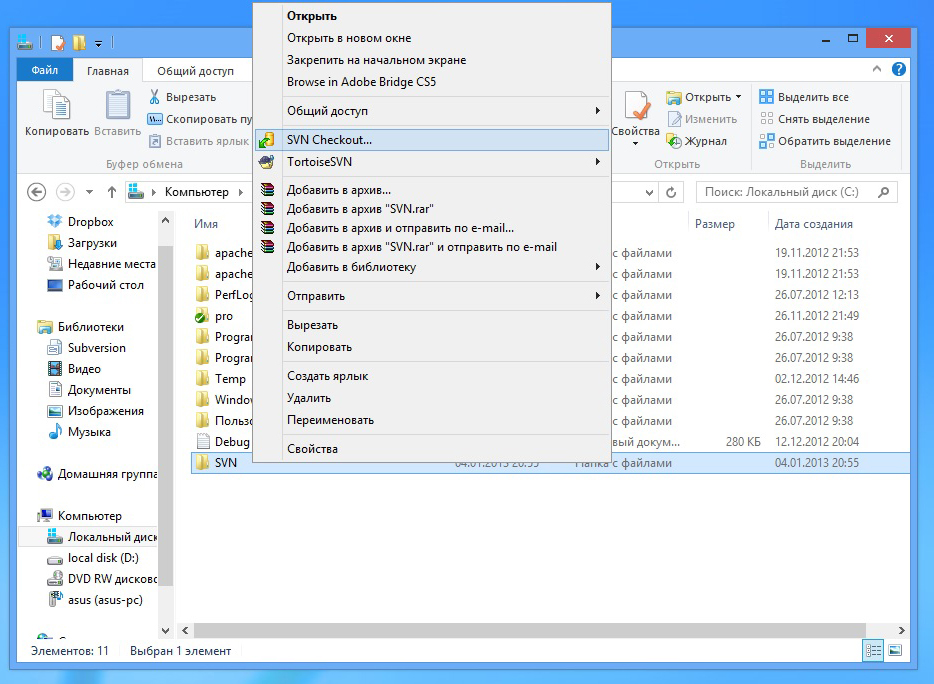


Рисунок 2.7.3.1. SVN Checkout

* В поле «URL of repository» пишем полный URL до репозитория SVN, который нам нужно скачать, а в поле «Checkout directory» корректируем путь, в которой будет помещена локальная копия. Необходимо делать checkout только каталога /trunk/ svn-репозитория, либо корневого, если /trunk/ не существует (очень редко, т.к. в 99% репозиториев trunk существует и содержит всегда самую последнюю версию файлов) (рис 2.7.3.2).

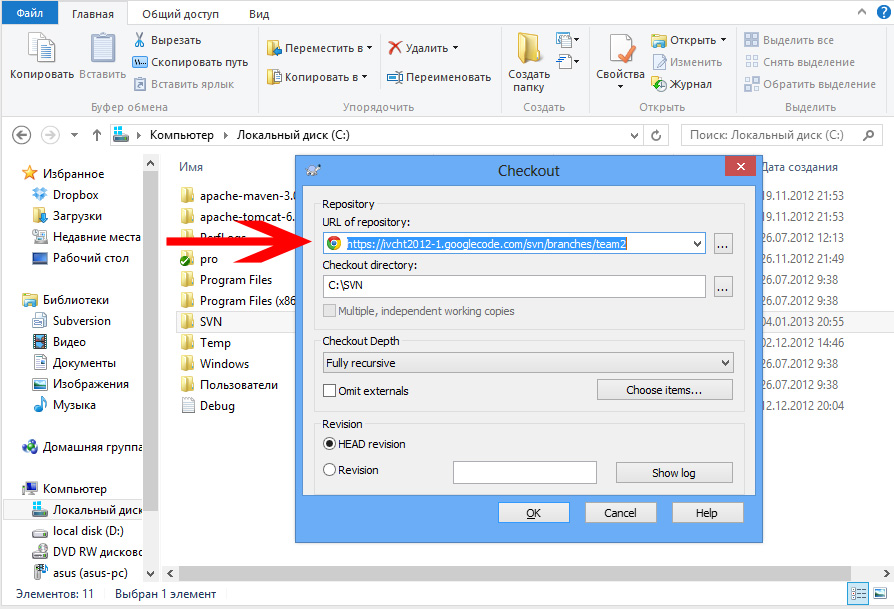


Рисунок 2.7.3.2. URL of repository

* В списке «Checkout Depth» укажите «Fully recursive«, что означает, что будет скачан весь репозиторий от указанного пути. Если установить флажок «Omit externals«, то внешние файлы, на которые есть ссылки в репозитории (например, из других репозиториев, либо репозиториев третьих лиц), скачаны не будут, поэтому ставить флажок здесь не рекомендуется.
* В блоке «Revision» Вы можете запросить выдать Вам последнюю версию репозитория: «HEAD revision» (рекомендуется именно этот вариант), либо указанную Вам в поле ревизию: «Revision ###» (не рекомендуется).
* Кнопка «Show log» покажет Вам список изменений в репозитории с информацией об авторах каждого изменения, описании изменений (если авторы их вводили при коммите изменений), а также списке добавленных, изменённых и удалённых файлов. Здесь же можно просмотреть чем файл одной ревизии (версии) отличается от файла другой, а также запросить показать унифицированный diff-файл изменений либо нескольких файлов, либо нескольких ревизий (отображается только для текстовых файлов). Более подробную информацию о Log Viewer читайте ниже.
* Нажимаем кнопку «OK» и ждём скачивания репозитория с Subversion-сервера (зависит от скорости Вашего Интернет-соединения, а также от загруженности svn-сервера). Некоторые svn-серверы могут запросить пароль. Если Вы не знаете пароля, то вводите anonsvn как логин и пароль. Это стандартный логин/пароль для анонимного доступа. Если Вы хотите в дальнейшем публиковать свои изменения в этот репозиторий и у Вас есть на это право, то введите здесь свой логин и пароль.
* Готово. В каталоге, который Вы указали на третьем шаге в поле «Checkout directory» теперь находится локальная версия репозитория (рабочая копия).

Если вы пытаетесь подключиться к хранилищу, защищённому паролем, появится диалог аутентификации (рис 2.7.3.3).

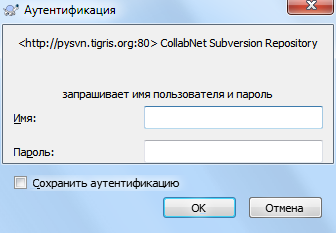


Рисунок 2.7.3.3. Диалог аутентификации

В нашем случае работая с SVN google.com Вводим Login и Password (рис 2.7.3.4). Login - ящик на Gmail, пароль берется здесь: <http://code.google.com/hosting/settings>.

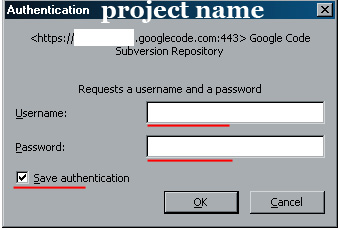


Рисунок 2.7.3.4. Диалог аутентификации

Введите ваше имя пользователя и пароль. При помощи флажка можно сделать так, чтобы эти данные сохранялись TortoiseSVN в папке по умолчанию Subversion: %APPDATA%\Subversion\auth в трёх подпапках:

* svn.simple содержит учётные данные для базовой аутентификации (имя пользователя/пароль). Обратите внимание: эти пароли хранятся при помощи WinCrypt API, а не в виде простого текста.
* svn.ssl.server содержит серверные сертификаты SSL.
* svn.username содержит учётные данные для аутентификации только по имени пользователя (без пароля).

Очистить кэш аутентификации для всех серверов можно со страницы Сохранённые данные диалога настроек TortoiseSVN. Там находится кнопка, очищающая все закэшированные данные аутентификации из папок auth Subversion, так же как и любые аутентификационные данные, сохранённые в реестре более ранними версиями TortoiseSVN (рис 2.7.3.5).

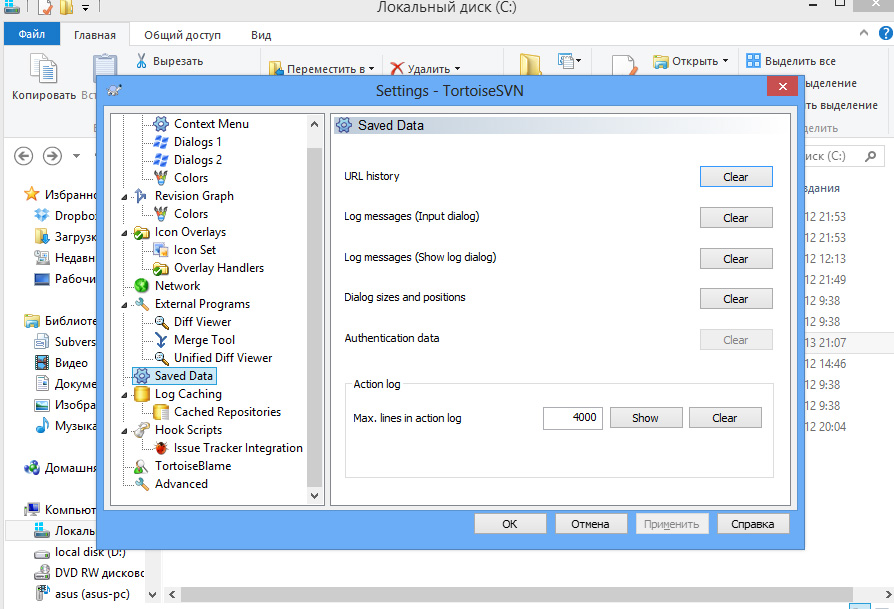


Рисунок 2.7.3.5. Диалоговое окно очистки КЭШа.

Если вы желаете очистить аутентификацию только для одной области, то вам надо зайти в эти папки, найти файл, содержащий информацию, которую вы желаете очистить, и удалить этот файл.

Некоторым людям хочется, чтобы данные об аутентификации удалялись когда они выходят из Windows или при выключении. Это можно сделать с помощью скрипта выключения удалив директорию %APPDATA%\Subversion\auth.

Например:

@echo off

rmdir /s /q "%APPDATA%\Subversion\auth"

Описание того как установить такие скрипты вы можете найти в <http://www.windows-help-central.com/windows-shutdown-script.html>.

Если авторизация прошла успешна, начнется загрузка проекта (3-5 минут). После окончания загрузки в окне будет следующее, как показан на рисунке ниже: (рис 2.7.3.6)

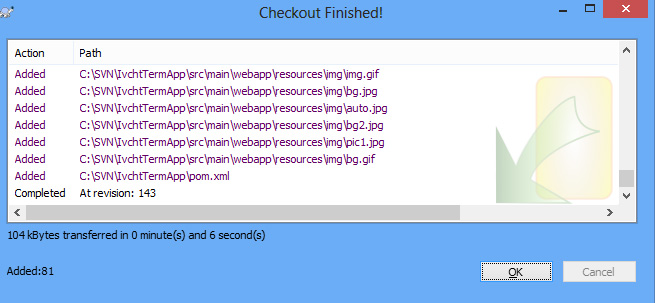


Рисунок 2.7.3.6. Checkout Finished

На изображении каталога появится зеленая галочка, это свидетельствует что содержимое каталога - самое последнее (рис 2.7.3.7)

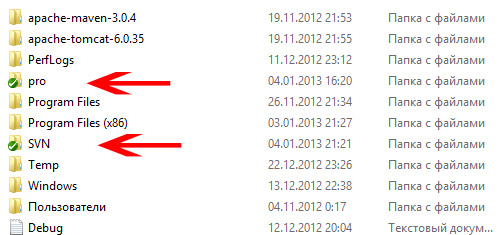


Рисунок 2.7.3.7. Наглядное изображение иконки на папке

2.7.4. Обновление скачанного репозитория (Update)

SVN и создавался для удобного управления проектам, а также поддержания их в актуальном состоянии, поэтому вторая по значимости операция — обновления скачанной рабочей копии. Рекомендуется как можно чаще обновлять свои рабочие копии. При обновлении скачиваются только изменённые файлы, добавляются новые и удаляются удалённые из репозитория файлы. (рис 2.7.4.1) Итак, чтобы обновить рабочую копию, нужно:

1. Заходим в папку, в которой находятся файлы скачанной рабочей копии в Проводнике Windows.
2. Нажимаем правой кнопкой по свободному месту в папке, выбираем «SVN Update».
3. В открывшемся окне наблюдаем за процессом обновления: списком добавленных, изменённых и/или удалённых файлов, текущим номером версии. Здесь же можно нажать кнопку «Show log…» чтобы просмотреть список, прочитать описание изменений (если указано), сравнить текущую версию с предыдущей и т.д.
4. Нажмите «OK» чтобы закрыть это окно. Репозитарий обновлён до последней версии.

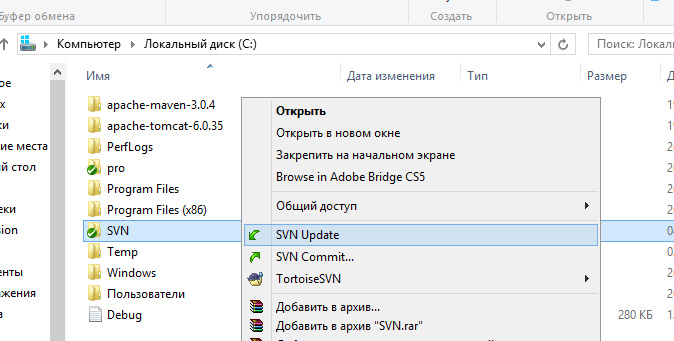


Рисунок 2.7.4.1 Выполнение обновления

После мы также заметим, что у нас самые последние обновления. Это видно по нашему последнему репозиторию 143 (рис 2.7.4.2).

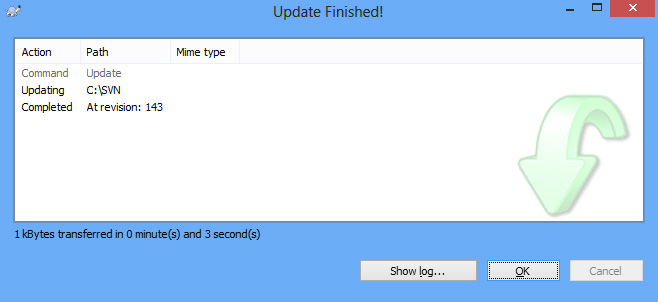


Рисунок 2.7.4.2 Окончание обновления.

2.7.5 Сохранение своих изменений в репозиторий (Commit)

Если Вы хотите внести свои изменения в скачанный репозиторий и у Вас есть на это право (право коммита в репозиторий), то просто начинайте работать с локальной рабочей копией как с обычной папкой: создавайте, редактируйте, удаляйте файлы и/или папки. Теперь чтобы внести изменения в репозиторий, нужно:

1. Заходим в папку, в которой находятся файлы скачанной рабочей копии в Проводнике Windows. Рабочая копия должна быть скачана от логина пользователя, у которого есть право на коммит изменений (коммит-флаг).
2. Нажимаем правой кнопкой по свободному месту в папке, выбираем «SVN Commit…» (рис 2.7.5.3).
3. В открывшемся окне «Commit» в секции «Message» вводим краткое описание своих изменений (это рекомендуется делать всегда, особенно если не Вы один работаете над проектом), но описание можно и оставить пустым (не рекомендуется, т.к. другие разработчики или пользователи не смогут понять что именно было изменено) (рис 2.7.5.4).
4. В секции «Changes made (double-click on file for diff)» Вы увидите список всех изменённых, добавленных и удалённых Вами файлов локальной рабочей копии. У изменённых файлов будут автоматически стоять флажки, у остальных — нет. Установите флажки у тех файлов, изменения которых должны быть загружены на svn-сервер (есть флажок — файл обновляется, добавляется, удаляется; нет флажка — остаётся без изменений). Двойной щелчок по файлу запустит утилиту Tortoise Merge (предназначена для сравнения версий файлов), которая покажет последнюю версию файла из репозитория SVN в левом окне и текущую рабочую копию в правом. Изменения будут выделены: удалённые строки зачёркнуты, добавленные выделены.
5. После выделения нужных файлов или всех сразу («Select / deselect all»), нажмите кнопку «OK» и дождитесь окончания загрузки изменений на сервер. Будет открыто окно со списком добавленных, изменённых и/или удалённых файлов, а также указан текущий номер ревизии.
6. Также можно нужный файл добавить в список, как показано на рис. 2.7.5.1, и иконка файла изменяется при этом, как на рисунке 2.7.5.2. А далее можем выполнить коммит.

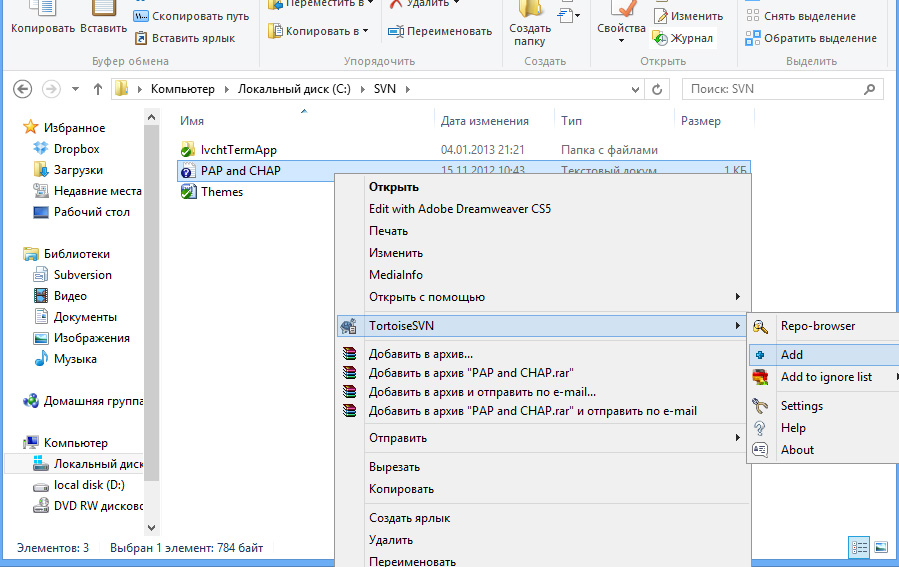


Рисунок 2.7.5.1 Добавление файла в проект

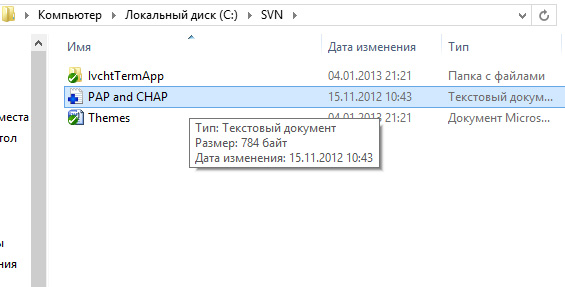


Рисунок 2.7.5.2 Изменение иконки у добавленного файла

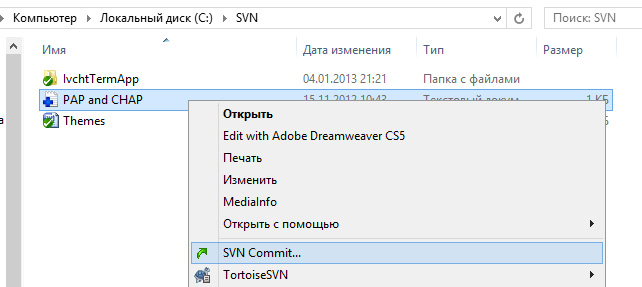


Рисунок 2.7.5.3 SVN Commit…

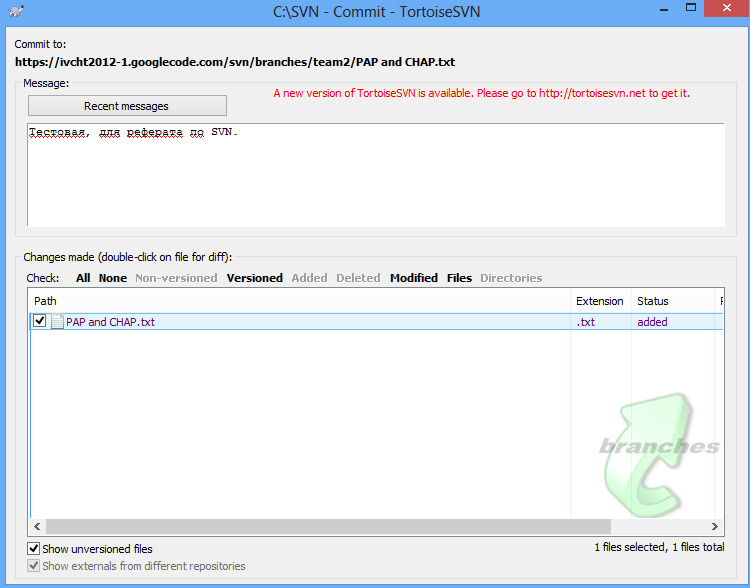


Рисунок 2.7.5.4 Диалоговое окно для ввода примечании и выбора файлов для комита

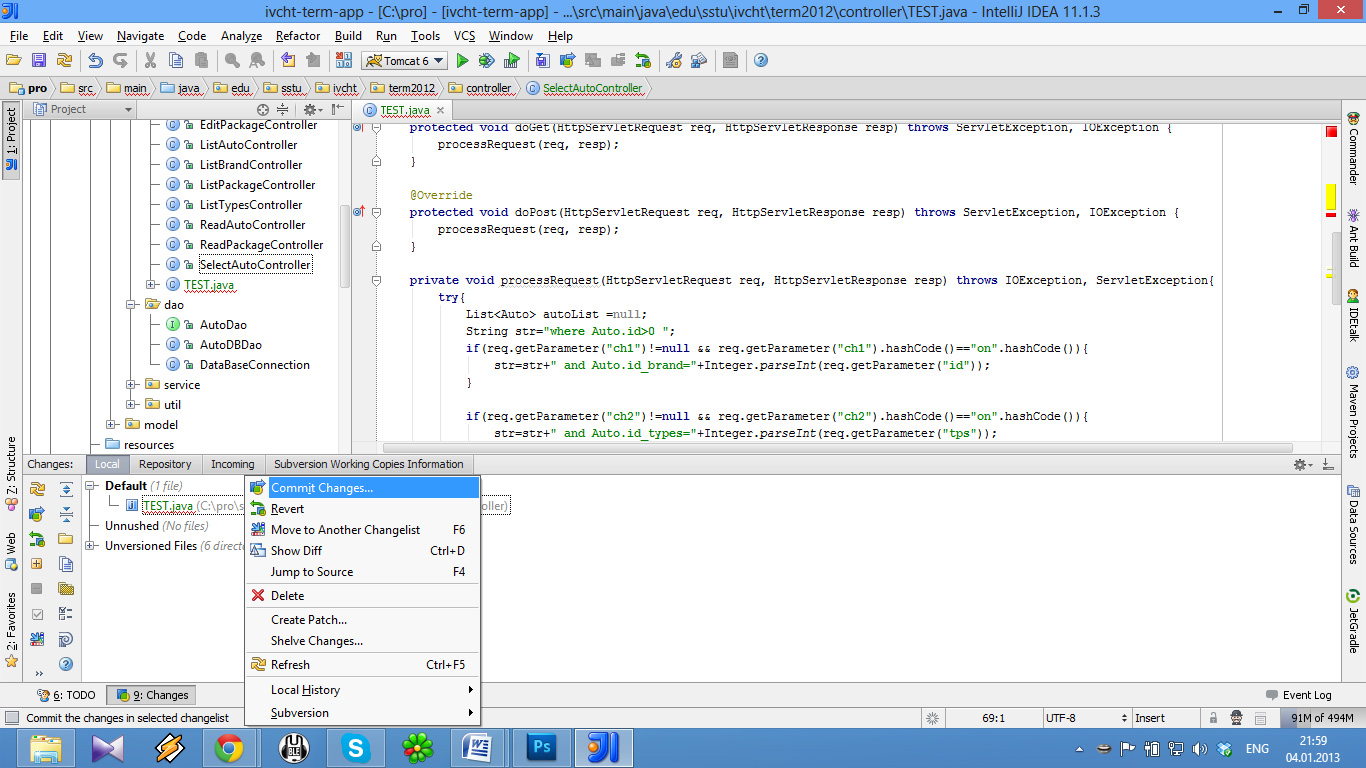


Рисунок 2.7.5.5 Добавление файла и Commit этого файла в среде IntelliJ IDEA

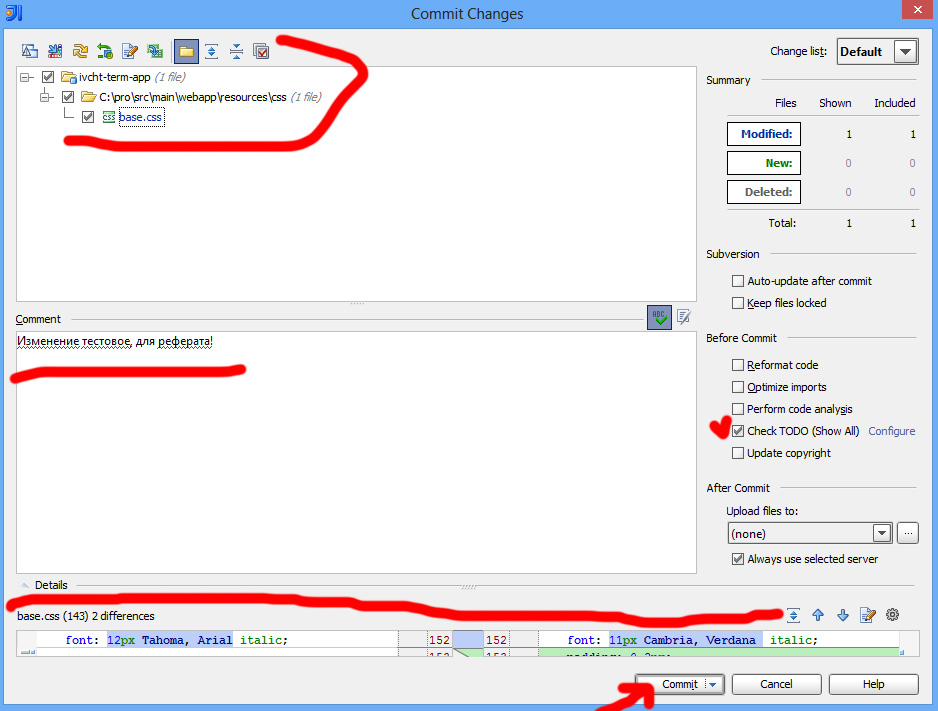


Рисунок 2.7.5.6 Commit Changes

2.7.6 Работа с репозиторием напрямую на сервере

Иногда требуется работать с svn-репозиторием напрямую на сервере не скачивая его на локальный компьютер. Для этих целей существует утилита SVN Repo-browser, также входящая в состав TortoiseSVN (рис 2.7.6.1). Работать с ней очень просто:

1. Запускаем проводник Windows, заходим в любую папку, нажимаем правой кнопкой по свободному месту и выбираем «SVN Repo-browser».
2. В открывшемся окне «URL» вводим полный URL нужного нам svn-репозитория и нажимаем «OK».
3. Будет открыто главное окно утилиты: «%URL% — Repository Browser«, где%URL% — URL репозитория. Работать в этом окне можно так же, как в Проводнике Windows. Разрешено перетаскивание объектов (Drag and Drop). Через контекстное меню файла/папки можно:

* удалять «Delete…», переименовывать «Rename» файлы и каталоги (папки);
* добавить файл («Add file…») или папку («Add folder…») в репозиторий;
* создать пустую папку («Create folder») в репозитории;
* скачать («Checkout…») репозиторий (см. выше);
* извлечь файл или папку из репозитория без служебных каталогов («Export…»). Извлечённая папка отличается от рабочей копии тем, что её нельзя обновлять и она не содержит служебных скрытых каталогов .svn;
* создать ветку или тег («Copy to…»);
* изменять свойства файла или папки («Show properties»).

1. Любое изменение требует ввести текст комментария (разрешается не вводить и оставить поле пустым) и увеличивает номер ревизии на единицу. Изменения применяются немедленно.
2. Нажмите кнопку «OK» для выхода из Repository Browser.

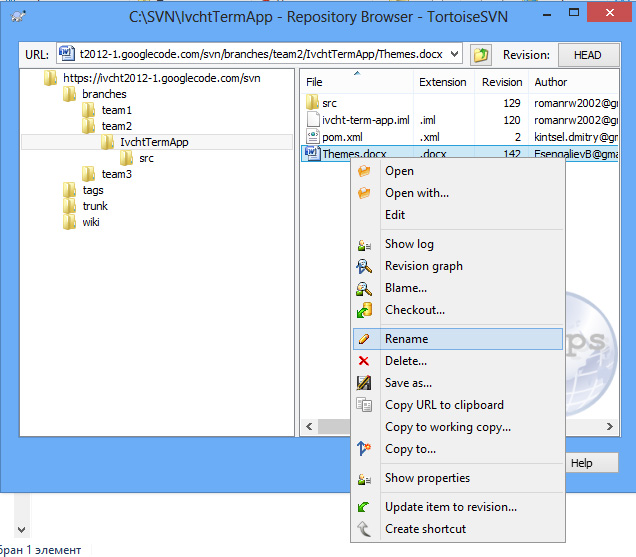


Рисунок 2.7.6.1 Repository Browser – TortoiseSVN

2.7.7 Создание веток или меток внутри репозитория

Если Вы работаете над большим проектом, то желательно для каждой публичной стабильной версии (1.0, 1.1, 2.0 и т.д.) создавать отдельный тег, например,/tags/RELEASE-Mj.Mn, где Mj — Major-версия, а Mn — Minor-версия проекта. Пример: /tags/RELEASE-1.0 (для версии 1.0), /tags/RELEASE-2.63 (для версии 2.63) и т.д.  Теги создаются НЕ в каталоге /trunk/, а выше него на один уровень. Чтобы создать метку или ветвление, нужно (у Вас должно быть право коммита в репозиторий для создания меток или ветвлений):

1. Заходим в папку, в которой находятся файлы скачанной рабочей копии в Проводнике Windows.
2. Нажимаем правой кнопкой по свободному месту и выбираем «TortoiseSVN» — Branch/tag….
3. В открывшемся окне в поле «To URL» введите полный URL с путём до нового тега. Пример: http:// ivcht2012-1.googlecode.com/tags/RELEASE-1.0.
4. В секции «Create copy in the repository from» укажите «HEAD revision in the repository» (последняя ревизия; рекомендуется), либо укажите номер ревизии вручную, выбрав «Specific revision in the repository ###», либо создайте тег из текущей рабочей копии «Working copy».
5. В секции «Log message» укажите комментарий к ревизии (например,%НАЗВАНИЕ\_ПРОЕКТА% %ВЕРСИЯ\_ПРОЕКТА% Released).
6. Нажмите «OK«. Флажок в строке «Switch working copy to new branch/tag» ставить не рекомендуется, т.к. это переключит Вашу текущую рабочую копию на новый путь, что может привести к различным ошибкам и проблемам.

2.7.8 Откат изменений в рабочей копии

Если работая с локальным репозиторием Вы совершили какую-либо ошибку, например, повредили какой-то файл, неправильно отредактировали и т.д., то это легко исправить:

1. Заходим в папку, в которой находятся файлы скачанной рабочей копии в Проводнике Windows.
2. Нажимаем правой кнопкой по свободному месту и выбираем «TortoiseSVN» — «Revert…».
3. В открывшемся окне Вы увидите список всех изменённых файлов Вашей рабочей копии. Установите флажок у того файла, который Вы хотите откатить до версии из репозитория, либо выберите сразу все файлы («Select / deselect all»). Двойной щелчок по файлу покажет его отличия от оригинального файла (рис 2.7.8.1).
4. Нажмите «OK». Выбранные файлы будут восстановлены, а изменённые будут отправлены в Корзину Windows.

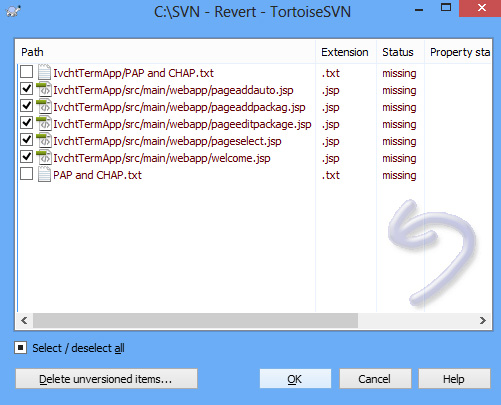
****

Рисунок 2.7.8.1 Откат до версии из репозитория

2.7.9 Работа с утилитой Log Viewer

Утилита Log Viewer, запускаемая после нажатия на кнопку «Show log…», либо при нажатии правой кнопкой по свободному месту в локальной рабочей копии — «TortoiseSVN» — «Show log». По умолчанию утилита показывает последние 50 ревизий и их описаний. Все основные действия выполняются из контекстного меню ревизии (рис 2.7.9.1):

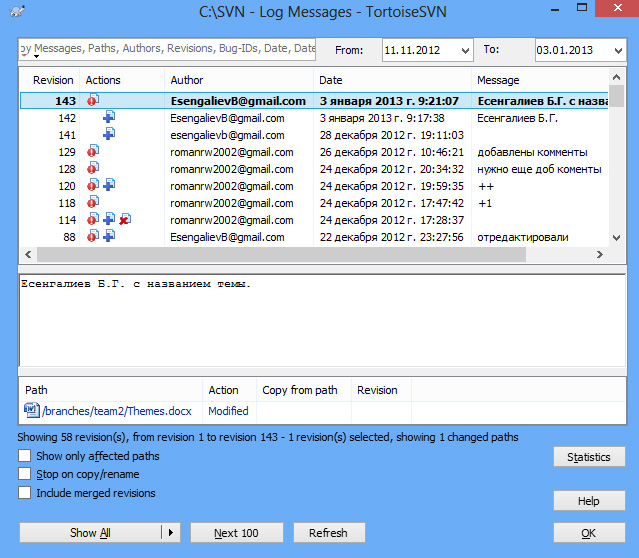
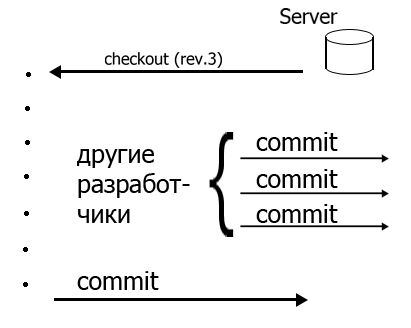


Рисунок 2.7.9.1 Последние ревизии в Log Viewer

* «Compare with woking copy» — сравнить текущую (выделенную) ревизию с локальной рабочей копией. Список изменений будет показан в Tortoise Merge.
* «Show changes as unified diff» — показать изменения выделенных ревизий как унифицированный патч-файл (unified diff). Если выделена только одна ревизия, то она будет сравниваться с предыдущей (например, выделена ревизия 142, она будет сравниваться с 143).
* «Compare with previous revision» — показать изменения выделенной и предыдующей ревизии. Будет отображено окно с различающимися файлами, двойной щелчок по файлу откроет его в Tortoise Merge для просмотра изменений.
* «Compare and blame with previous revision» — то же, что и предыдущее, но здесь отображается кто и когда внёс какое-либо изменение в файл построчно.
* «Browse repository» — открывает описанный выше Repository Browser для прямой работы с репозиторием.
* «Create branch/tag from revision» — позволяет создать ветку или тег из выделенной ревизии.
* «Update item to revision» — обновляет до выделенной ревизии.
* «Revert to this revision» — позволяет откатиться до выделенной ревизии.
* «Revert changes from this revision» — позволяет откатить все изменения, сделанные в выделенной ревизии.
* «Merge revisions to…» — позволяет объединить несколько ревизий в рабочую копию.
* «Checkout…» — скачать выделенную ревизию.
* «Export…» — извлечь выделенную ревизию без служебных каталогов.
* «Edit author» — редактировать имя автора выделенной ревизии (нужны права администратора).
* «Edit log message» — позволяет редактировать текст описания выделенной ревизии (нужны права администратора).
* «Show revision properties» — показать свойства выделенной ревизии.
* «Copy to clipboard» — скопировать номер выделенной ревизии, автора, описание и список изменённых файлов в буфер обмена Windows.
* «Search log messages» — запустить поиск в описаниях ревизий.
* Кнопка «Show all» покажет все ревизии, а «Show range…» позволит задать диапазон ревизий, которые будут отображаться в окне Log Viewer.

Кнопка «Statistics» отображает различную статистическую информацию: наиболее активный и наименее активный автор, количество изменений в репозитории в неделю, общее количество изменений и т.д.

2.7.10 Разрешение конфликтов

При работе над проектом возможны такие ситуации, как конфликты  
Рисунок 2.7.10.1 Конфликты при выполнении Commit.  
Допустим, в какой-то момент времени разработчик обновляется до последней версии проекта и начинает работу над некоторыми файлами.  
Через какое-то время, изменив файлы, он пытается сделать commit. Однако, этот commit не удастся, если другие разработчики уже вносили изменения в те же файлы проекта и сохраняли их в репозитории. Такая ситуация называется конфликтом. У разработчика, чей commit не удался из-за конфликтов, есть два возможных варианта действий:

1. svn revert — отменить все свои изменения
2. svn update — забрать из репозитория новые версии файлов и, разрешив конфликты, снова попытаться сделать commit.

Заключение

В работе были рассмотрены системы контроля версий, проведен краткий обзор по данным системам и описан принцип работы с TortoiseSVN. Такого рода системы позволяют облегчить проекты по разработке программного обеспечения, и в определенной степени исключить человеческие ошибки.  Системы контроля версий являются потрясающим решением проблем распространенных корпораций. Системы контроля версий – это абсолютно необходимое программное обеспечения для каждого разработчика и не только.

Список литературы

1. http://embedders.org/content/zachem-nuzhny-sistemy-kontrolya-versii-tipa-svn
2. http://all-ht.ru/inf/prog/p\_0\_1.html
3. http://ru.wikipedia.org/wiki/
4. http://www.techinfo.net.ru/docs/Version\_Control\_Systems.html
5. http://better-scm.berlios.de/aegis/
6. http://ru.wikipedia.org/wiki/CVS
7. http://ru.wikipedia.org/wiki/Subversion
8. http://www.opencm.org/
9. http://aegis.sourceforge.net/
10. http://www.bitkeeper.com/Products.Advantages.html
11. http://ru.wikipedia.org/wiki/Perforce
12. http://wiki.darcs.net/DarcsWiki/ВведениеВDarcs