**Обзор систем контроля версий**

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. Вследствие этого, появилось большое число различных продуктов, предлагающих широкие возможности и предоставляющих обширные инструменты для управления версиями. В этой статье будут кратко рассмотрены наиболее популярные из них, приведены их достоинства и недостатки.

Для сравнения были выбраны наиболее распространенные системы контроля версий: RCS, CVS, Subversion, Aegis, Monoton, Git, Bazaar, Arch, Perforce, Mercurial, TFS.

**RCS - система управления пересмотрами версий.**[**(www.gnu.org/software/rcs/rcs.html)**](http://www.gnu.org/software/rcs/rcs.html)

Начнем наш обзор с одной из первых систем контроля версий – RCS (Revision Control System – система управления пересмотрами версий), разработанной в 1985 году. Она пришла на смену популярной в то время системы контроля версий SCCS (Source Code Control System – система управления исходным кодом).

На данный момент RCS активно вытесняется более мощной системой контроля версий CVS, но все еще - достаточно популярна, и является частью проекта GNU.

RCS позволяет работать только с отдельными файлами, создавая для каждого историю изменений. Для текстовых файлов сохраняются не все версии файла, а только последняя версия и все изменение, внесенные в нее. RCS также может отслеживать изменения в бинарных файлах, но при этом каждое изменение хранится в виде отдельной версии файла.

Когда изменения в файл вносит один из пользователей, для всех остальных этот файл остается заблокированным. Они не могут запросить его из репозитория для редактирования, пока первый пользователь не закончит работу и не зафиксирует изменения.

Рассмотрим основные преимущества и недостатки системы контроля версий RCS.

**Преимущества:**

1. RCS - проста в использовании и хорошо подходит для ознакомления с принципами работы систем контроля версий.

2. Хорошо подходит для резервного копирования отдельных файлов, не требующих частого изменения группой пользователей.

3. Широко распространена и предустановленна в большинстве свободно распространяемых операционных системах.

**Недостатки:**

1. Отслеживает изменения только отдельных файлов, что не позволяет использовать ее для управления версиями больших проектов.

2. Не позволяет одновременно вносить изменения в один и тот же файл несколькими пользователями.

3. Низкая функциональность, по сравнению с современными системами контроля версий.

**Выводы:**

Рейтинг продукта 1 балл из 5Система контроля версий RCS предоставляет слишком слабый набор инструментов для управления разрабатываемыми проектами и подходит разве что для ознакомления с технологией контроля версий или ведения небольшой истории откатов отдельных файлов.

**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:**

Рейтинг продукта: 3 балла из 5Несмотря 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.

**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:**

Рейтинг продукта 4.5 баллов из 5Subversion – 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.

**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:**

Рейтинг продукта: 2 балла из 5Сложность 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.

**Система управления версиями Monotone.**[**(monotone.ca)**](http://monotone.ca)

Monotone – еще одна децентрализованная система управления версиями, разработанная Грейдоном Хоэм. В ней каждый клиент сам отвечает за синхронизацию версий разрабатываемого продукта с другими клиентами.

Работа с этой системой контроля версий - достаточно проста, а многие команды - схожи с командами, используемыми в Subversion и CVS. Отличия, в основном, заключаются в организации слияния ветвей проектов различных разработчиков.

Работа с Monotone строится следующим образом. В первую очередь, создается база данных проекта SQLite, и генерируются ключи с использованием алгоритма хеширования SHA1 (Secure Hash Algorithm 1).

Затем, по ходу корректировки проекта пользователем, все изменения сохраняются в этой базе данных, аналогично сохранению изменений в репозитории других систем контроля версий.

Для синхронизации проекта с другими пользователями необходимо:

- Экспортировать ключ (хэш - код последней версии проекта) и получить аналогичные ключи от других клиентов.

- Сохранить всем клиентам полученные ключи в связке ключей своих локальных проектах Monotone.

- Теперь каждый, зарегистрированный таким образом пользователь, может синхронизировать разработку со своими коллегами, используя простой набор команд.

Обобщим достоинства и недостатки системы контроля версий Monotone.

**Достоинства:**

1. Простой и понятный набор команд, схожий с командами Subversion и CVS.

2. Поддерживает переименование и перемещение файлов и директорий.

3. Качественная документация, значительно облегчающая использование системы контроля версий.

**Недостатки:**

1. Низкая скорость работы.

2. Отсутствие мощных графических оболочек.

3. Возможные (но чрезвычайно низкие) совпадения хэш - кода отличных по содержанию ревизий.

**Выводы:**

Рейтинг продукта 2.5 балла из 5Monotone - это мощный и удобный инструмент для управления версиями разрабатываемого проекта. Набор команд - продуман и интуитивно понятен, особенно, он будет удобен для пользователей, привыкших к работе c Subversion и CVS. Прекрасно оформленная и полная документация позволит быстро освоиться и использовать все возможности Subversion на полную мощность.

Однако, относительно низкая скорость работы и отсутствие мощных графических оболочек, возможно, сделает работу с большими проектами несколько затруднительной. Поэтому, если вам требуется система контроля версий для поддержки сложных и объемных продуктов, стоит обратить внимание на Git или Mercurial.

**Система управления версиями Git.**[**(www.git-scm.com)**](http://www.git-scm.com)

С февраля 2002 года для разработки ядра Linux’а большинством программистов стала использоваться система контроля версий BitKeeper. Довольно долгое время с ней не возникало проблем, но в 2005 году Лари МакВоем (разработчик BitKeeper’а) отозвал бесплатную версию программы.

Разрабатывать проект масштаба Linux без мощной и надежной системы контроля версий – невозможно. Одним из кандидатов и наиболее подходящим проектом оказалась система контроля версий Monotine, но Торвальдса Линуса не устроила ее скорость работы. Так как особенности организации Monatone не позволяли значительно увеличить скорость обработки данных, то 3 апреля 2005 года Линус приступил к разработке собственной системы контроля версий – Git.

Практически одновременно с Линусом (на три дня позже), к разработке новой системы контроля версий приступил и Мэтт Макал. Свой проект Мэтт назвал Mercurial, но об этом позже, а сейчас вернемся к распределенной системе контроля версий Git.

Git – это гибкая, распределенная (без единого сервера) система контроля версий, дающая массу возможностей не только разработчикам программных продуктов, но и писателям для изменения, дополнения и отслеживания изменения «рукописей» и сюжетных линий, и учителям для корректировки и развития курса лекций, и администраторам для ведения документации, и для многих других направлений, требующих управления историей изменений.

У каждого разработчика, использующего Git, есть свой локальный репозиторий, позволяющий локально управлять версиями. Затем, сохраненными в локальный репозиторий данными, можно обмениваться с другими пользователями.

Часто при работе с Git создают центральный репозиторий, с которым остальные разработчики синхронизируются. Пример организации системы с центральным репозиторием – это проект разработки ядра Linux’a (http://www.kernel.org).

В этом случае все участники проекта ведут свои локальны разработки и беспрепятственно скачивают обновления из центрального репозитория. Когда необходимые работы отдельными участниками проекта выполнены и отлажены, они, после удостоверения владельцем центрального репозитория в корректности и актуальности проделанной работы, загружают свои изменения в центральный репозиторий.

Наличие локальных репозиторием также значительно повышает надежность хранения данных, так как, если один из репозиториев выйдет из строя, данные могут быть легко восстановлены из других репозиториев.

Работа над версиями проекта в Git может вестись в нескольких ветках, которые затем могут с легкостью полностью или частично объединяться, уничтожаться, откатываться и разрастаться во все новые и новые ветки проекта.

Можно долго обсуждать возможности Git’а, но для краткости и более простого восприятия приведем основные достоинства и недостатки этой системы управления версиями

**Достоинства:**

1. Надежная система сравнения ревизий и проверки корректности данных, основанные на алгоритме хеширования SHA1 (Secure Hash Algorithm 1).

2. Гибкая система ветвления проектов и слияния веток между собой.

3. Наличие локального репозитория, содержащего полную информацию обо всех изменениях, позволяет вести полноценный локальный контроль версий и заливать в главный репозиторий только полностью прошедшие проверку изменения.

4. Высокая производительность и скорость работы.

5. Удобный и интуитивно понятный набор команд.

6. Множество графических оболочек, позволяющих быстро и качественно вести работы с Git’ом.

7. Возможность делать контрольные точки, в которых данные сохраняются без дельта компрессии, а полностью. Это позволяет уменьшить скорость восстановления данных, так как за основу берется ближайшая контрольная точка, и восстановление идет от нее. Если бы контрольные точки отсутствовали, то восстановление больших проектов могло бы занимать часы.

8. Широкая распространенность, легкая доступность и качественная документация.

9. Гибкость системы позволяет удобно ее настраивать и даже создавать специализированные контроля системы или пользовательские интерфейсы на базе git.

10. Универсальный сетевой доступ с использованием протоколов http, ftp, rsync, ssh и др.

**Недостатки:**

1. Unix – ориентированность. На данный момент отсутствует зрелая реализация Git, совместимая с другими операционными системами.

2. Возможные (но чрезвычайно низкие) совпадения хеш - кода отличных по содержанию ревизий.

3. Не отслеживается изменение отдельных файлов, а только всего проекта целиком, что может быть неудобно при работе с большими проектами, содержащими множество несвязных файлов.

4. При начальном (первом) создании репозитория и синхронизации его с другими разработчиками, потребуется достаточно длительное время для скачивания данных, особенно, если проект большой, так как требуется скопировать на локальный компьютер весь репозиторий.

**Выводы:**

Рейтинг продукта: 4.5 баллов из 5Git – гибкая, удобная и мощная система контроля версий, способная удовлетворить абсолютное большинство пользователей. Существующие недостатки постепенно удаляются и не приносят серьезных проблем пользователям. Если вы ведете большой проект, территориально удаленный, и тем более, если часто приходится разрабатывать программное обеспечение, не имея доступа к другим разработчикам (например, вы не хотите терять время при перелете из страны в страну или во время поездки на работу), можно делать любые изменения и сохранять их в локальном репозитории, откатываться, переключаться между ветками и т.д.). Git – один из лидеров систем контроля версий.

**Система управления версиями Mercurial.**[**(mercurial.selenic.com)**](http://mercurial.selenic.com)

Распределенная система контроля версий Mercurial разрабатывалась Мэттом Макалом параллельно с системой контроля версий Git, созданной Торвальдсом Линусом.

Первоначально, она была создана для эффективного управления большими проектами под Linux’ом, а поэтому была ориентирована на быструю и надежную работу с большими репозиториями. На данный момент mercurial адаптирован для работы под Windows, Mac OS X и большинство Unix систем.

Большая часть системы контроля версий написана на языке Python, и только отдельные участки программы, требующие наибольшего быстродействия, написаны на языке Си.

Идентификация ревизий происходит на основе алгоритма хеширования SHA1 (Secure Hash Algorithm 1), однако, также предусмотрена возможность присвоения ревизиям индивидуальных номеров.

Так же, как и в git’е, поддерживается возможность создания веток проекта с последующим их слиянием.

Для взаимодействия между клиентами используются протоколы HTTP, HTTPS или SSH.

Набор команд - простой и интуитивно понятный, во многом схожий с командами subversion. Так же имеется ряд графических оболочек и доступ к репозиторию через веб-интерфейс. Немаловажным является и наличие утилит, позволяющих импортировать репозитории многих других систем контроля версий.

Рассмотрим основные достоинства и недостатки Mercurial.

**Достоинства:**

1. Быстрая обработка данных.

2. Кросплатформенная поддержка.

3. Возможность работы с несколькими ветками проекта.

4. Простота в обращение.

5. Возможность конвертирования репозиториев других систем поддержки версий, таких как CVS, Subversion, Git, Darcs, GNU Arch, Bazaar и др.

**Недостатки:**

1. Возможные (но чрезвычайно низкие) совпадения хеш - кода отличных по содержанию ревизий.

2. Ориентирован на работу в консоли.

**Выводы:**

Рейтинг продукта 4 балла из 5Простой и отточенный интерфейс, и набор команд, возможность импортировать репозитории с других систем контроля версий, - сделают переход на Mercurial и обучение основным особенностями безболезненным и быстрым. Вряд ли это займет больше нескольких дней.

Надежность и скорость работы позволяют использовать его для контроля версий огромных проектов. Все это делает mercurial достойным конкурентом git’а.

**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:**

Рейтинг продукта: 4 балла из 5Bazaar – 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.

**Система управления версиями Arch.**

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.

Рейтинг продукта 2 балла из 5Трудно 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.

**Система управления версиями Perforce.**[**(www.perforce.com)**](http://www.perforce.com)

Продолжим обзор систем контроля версий и перейдем к коммерческим программам. Начнем 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.

Рейтинг продукта: 3 балла из 5Поддержка 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.

**Система управления версиями Team Foundation Server.**[**(msdn.microsoft.com/en-us/library/ms364061.aspx)**](http://msdn.microsoft.com/en-us/library/ms364061.aspx)

Собственно говоря, нельзя назвать Team Foundation Server (TFC) просто системой контроля версий – это некое комплексное решение, в состав которого входит и система управления версиями, и система сбора данных, и построения отчетов, и другие полезные функции.

Управляемый проект при работе с TFC представляет собой ветки исходного кода проекта, наборы отчетов и пользовательские элементы. При создании проекта заранее выбираются его параметры, которые можно, как выбирать самостоятельно, так и использовать шаблоны. Шаблоны позволяют определить путь развития проекта, сделать его гибким или жестко формализованным, заложить стратегию развития, учесть необходимые заготовки документов и отчетов.

TFC легко интегрируется с Microsoft Excel и Microsoft Project, что значительно облегчает создание и отслеживание элементов контролируемых проектов.

В качестве системы контроля версий, TFC позволяет:

- совместно корректировать файлы проекта;

- разрешать конфликты;

- создавать ветки проектов, а затем объединять их;

- управлять доступом к репозиторию;

- откатываться на предыдущие версии;

- создавать отложенные изменения - изменения, которые, непосредственно, не добавлены в репозиторий, но их могут видеть другие пользователи, причем скачать эти изменения можно, только получив специальное разрешение от владельца изменений;

- помечать отдельные версии файлов в репозитории и группировать их;

Для сохранения данных и репозиториев разрабатываемых проектов используются базы данных SQL Server 2005.

Рейтинг продукта: 3.5 балла из 5TFC - мощный и удобный инструмент, позволяющий не только управлять версиями исходного кода, но и полностью организовывать весь цикл разработки проекта от написания программ до их документации. Однако, эта мощная и сложная система больше подходит для ведения больших проектов, требующих сложного и досконального управления разработкой. Если у вас - небольшая разработка, то имеет смысл использовать менее мощный инструмент, а еще лучше свободно распространяемый, так как это вам сэкономит время, деньги и нервы.

**Обобщение.**

Большой выбор систем контроля версий позволяет удовлетворить любые требования и организовать работу так, как вам необходимо. Однако, среди всего многообразия систем есть явные лидеры. Так, если необходимо управлять огромным проектом, состоящим из десятков тысяч файлов и над которым работу ведут тысячи человек, то лучше всего выбор остановить на Git или Mercurial. Если для вас главное – удобный интерфейс, а разрабатываемый проект - не очень большой, то для вас предпочтительна система Bazaar.

Для программистов-одиночек или небольших проектов, не требующих ветвления и создания множества версий, лучше всего подойдет Subversion.

Но, в конечном итоге, выбор - это дело вкуса, так как сейчас существует множество систем контроля версий, которые дадут вам все необходимое. Так что выбирайте и не пожалеете. Системы контроля версий – это абсолютно необходимое программное обеспечения для каждого разработчика и не только.