Процесс создания программного обеспечения трудоемкий, в который постоянно приходится вносить сложные изменения на каждом этапе разработки. Если проект небольшой, то приходится делать резервные копии для сохранения текущей версии проекта. Как правило, этот архив хранится локально на жестком диске компьютера разработчика. Но практика показывает, что с развитием информационных технологий программные продукты усложняются, и их сопровождение в процессе разработке требует более надежные и гибкие решения хранения различных версий продукта. Поэтому, если речь идет о ведении огромного проекта, состоящего из сотен и даже тысяч файлов, копирование проекта и описание каждой версии вручную превращается в долгий и трудоемкий процесс, что значительно затягивает и усложняет разработку. Во-первых, требуется много места на диске для хранения всех версий проекта. Кроме того, существует большая вероятность путаницы в архиве. Все это значительно усложняется, когда проект ведут несколько человек, иногда территориально удаленных друг от друга на сотни и тысячи километров. В этом случае у каждого будут образовываться свои архивы версий, и сопровождение разрабатываемого программного обеспечения сведется к длительному и сложному процессу. Таким образом, возникла потребность в специализированном программном обеспечении, позволяющим просто и надежно сопровождать большие программные проекты. Решением проблемы является внедрение системы контроля версий в процесс разработки.

Система управления версиями (от англ. Version Control System, VCS или Revision Control System) — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое. Системы контроля версий позволяют:

1. **Сохранять все этапы разработки**.

В процессе работы над проектом программист вносит изменения в репозиторий – хранилище всех версий и изменений проекта. Следует отметить, что в репозиторий добавляются только файлы, претерпевшие изменения, а не весь проект целиком. Такая система позволяет экономить место и время сохранения изменений (сервер с репозиторием может быть удаленным, и, если проект - очень большой, то требуется достаточно большое время для передачи всех его файлов по сети).

1. **Обновляться до последней версии разработанного программного обеспечения**.

Одной из основных задач системы контроля версий является возможность отслеживать все изменения проекта и быстро обновлять программное обеспечение клиентов до актуальной версии. Поэтому, члены команды разработки без труда обмениваются своими наработками посредством системы контроля версий.

1. **Объединять изменения**.

Часто несколько программистов одновременно работают над совместными файлами. Если изменения не пересекаются, то системы контроля версий позволяют легко и просто объединить эти изменения.

1. **Решать конфликты**.

Если несколько человек одновременно изменили один и тот же участок кода, то такие изменения не поддаются объединению в автоматическом режиме. Как правило, системы контроля версий предоставляют возможность вручную внести необходимые правки в тест программ, чтобы объединить конфликтующие части кода.

1. **Откатываться к предыдущим версиям**.

Системы контроля версий позволяют вернуться к одному из последних рабочих состояний проекта, скопировав из репозитория нужную версию программного обеспечение, либо отдельные файлы.

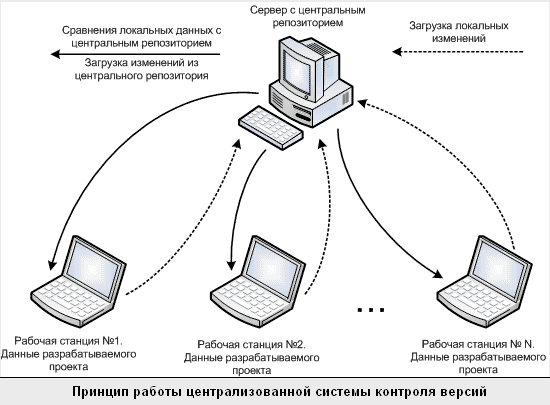
1. **Сопровождение нескольких направлений развития программного обеспечения**.

Бывают ситуации, когда работа над частью проекта занимает продолжительное время и включает в себя несколько этапов. Объединять с основным программным обеспечением необходимо по завершении всех этапов в рамках задачи, но возникает необходимость в сохранении промежуточных результатов. В этом случае многие системы контроля версий позволяют организовывать параллельные ветки по контролю нескольких направлений развития программного обеспечения, быстро переключаться между ними, а затем объединять их в единое целое.

Кроме этого, системы контроля версий обладают удобным интерфейсом для быстрого и простого выполнения перечисленных выше действий и надежного контроля версий разрабатываемого программного обеспечения.

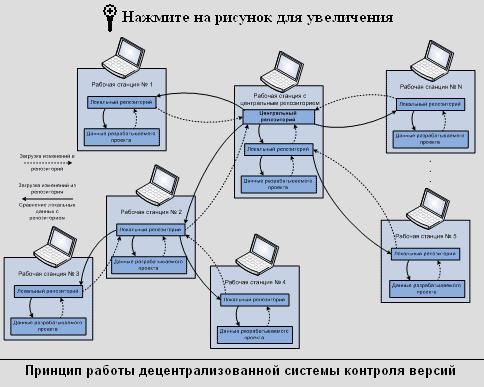
Несмотря на единый принцип работы, системы контроля версий можно разделить на две группы: централизованные и распределенные системы контроля версий.

Традиционные системы управления версиями используют централизованную модель, когда имеется единое хранилище документов, управляемое специальным [сервером](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5)), который и выполняет большую часть функций по управлению версиями. Пользователь, должен сначала получить нужную ему версию документа из хранилища; обычно создаётся локальная копия документа, т. н. «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.



Распределенные системы управления версиями также известны как [англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Distributed Version Control System, DVCS. Такие системы используют распределённую модель вместо традиционной клиент-серверной. Они, в общем случае, не нуждаются в централизованном хранилище: вся история изменения документов хранится на каждом компьютере, в локальном хранилище, и при необходимости отдельные фрагменты истории локального хранилища синхронизируются с аналогичным хранилищем на другом компьютере. В некоторых таких системах локальное хранилище располагается непосредственно в каталогах рабочей копии.

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, хранилища, принадлежащие разным разработчикам, начинают различаться, и возникает необходимость в их синхронизации. Такая синхронизация может осуществляться с помощью обмена [патчами](http://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%82%D1%87" \o "Патч) или так называемыми наборами изменений ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) change sets) между пользователями.

[](http://all-ht.ru/inf/prog/p_0_0_p1.html)

Обзор систем контроля версий

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название системы** | **Средства взаимодействия** | **Ветвление** | **Тип** | **Применение** |
| **CVS** | Обладает огромным количеством удобных графических интерфейсов. | Сложности при ведении нескольких параллельных веток одного и того же проекта. | Основана на технологии клиент-сервер, взаимодействующих по сети. | Подходит для управления небольшими проектами, не требующих создания нескольких параллельных версий, которые надо периодически объединять. |
| **Subversion** | Разнообразные графические интерфейсы и удобная работа из консоли. | Наличие удобного механизма создания меток и ветвей проектов.  Слабо поддерживаются операции слияния веток проекта. | Централизованная система управления версиями, основанная на технологии клиент-сервер. | Обладает широким набором инструментов, позволяющих удовлетворить любые нужды для управления версиями проекта с помощью централизованной системы контроля. |
| **Aegis** | Слабые графические интерфейсы. |  | Централизованная система управления версиями. | Полезна в некоторых специфических ситуациях, особенно, когда требуется жесткий контроль за качеством разрабатываемого программного обеспечения. |
| **Monotone** | Отсутствие мощных графических оболочек. |  | Децентрализованная система управления версиями | Не рекомендуется использовать для поддержки сложных и объемных продуктов. |
| **Git** | Множество графических оболочек, позволяющих быстро и качественно вести работы с Git’ом. | Гибкая система ветвления проектов и слияния веток между собой. | Гибкая, распределенная (без единого сервера) система контроля версий | Подходит для большого проекта, территориально удаленного. Один из лидеров систем контроля версий. |
| **Mercurial** | Ориентирован на работу в консоли. | Возможность работы с несколькими ветками проекта. | Распределенная система контроля версий. | Надежность и скорость работы позволяют использовать его для контроля версий огромных проектов. |
| **Bazaar** | Удобный графический интерфейс. | Простая работа с ветками проекта. | Распределенная система контроля версий. | Хорошо подходит для пользователей, которых отталкивает перспектива работы с командной строкой. Множество дополнительных опций и расширений позволит настроить программу под свои нужды. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Название системы** | **Достоинства** | **Недостатки** | **Документация** |
| **CVS** | Несколько клиентов могут одновременно работать над одним и тем же проектом.  Позволяет управлять не одним файлом, а целыми проектами.   Широко распространена и поставляется по умолчанию с большинством операционных систем Linux.  При загрузке тестовых файлов из репозитория передаются только изменения, а не весь файл целиком. | Ограниченная поддержка шрифтов.   Ресурсоемкие операции, так как требуют частого обращения к репозиторию, и сохраняемые копии имеют некоторую избыточность. |  |
| **Subversion** | Встроенная поддержка во многие интегрированные средства разработки, такие как KDevelop, Zend Studio и многие другие.  Возможность доступа к репозиторию через Apache с использованием протокола WebDAV. | Полная копия репозитория хранится на локальном компьютере в скрытых файлах, что требует достаточно большого объема памяти.  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. | Широкое распространение позволяет быстро решить большинство возникающих проблем, обратившись к данным, накопленным Интернет-сообществом. |
| **Aegis** | Надежный контроль корректности загружаемых изменений.  Возможность предоставлять различные уровни доступа к фалам репозитория, что дает приличный уровень безопасности. | Отсутствие встроенной поддержки сетевого взаимодействия.  Сложность настройки и работы с репозиторием. | Качественная документация. |
| **Monotone** | Простой и понятный набор команд. | Низкая скорость работы.  Возможные (но чрезвычайно низкие) совпадения хэш - кода отличных по содержанию ревизий. | Качественная документация, значительно облегчающая использование системы контроля версий. |
| **Git** | Надежная система сравнения ревизий и проверки корректности данных, основанные на алгоритме хеширования SHA1.  Высокая производительность и скорость работы.  Удобный и интуитивно понятный набор команд.  Универсальный сетевой доступ с использованием протоколов http, ftp, rsync, ssh и др. | Не отслеживается изменение отдельных файлов, а только всего проекта целиком, что может быть неудобно при работе с большими проектами, содержащими множество несвязных файлов.  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. | Качественная документация. |
| **Mercurial** | Быстрая обработка данных.  Кросплатформенная поддержка. | Ориентирован на работу в консоли. |  |
| **Bazaar** | Чрезвычайная гибкость, позволяющая подстроится под нужды конкретного пользователя. | Более низкая скорость работы, по сравнению с git и mercurial.  Для полноценного функционирования необходимо устанавливать достаточно большое количество плагинов, позволяющих полностью раскрыть все возможности системы контроля версий. | Великолепная документация. |

Большое многообразие систем контроля версий позволяет выбрать подходящую систему под нужды конкретной команды разработчиков, учитывая индивидуальные особенности разработки. В том случае, если необходимо управлять огромным проектом, который разрабатывается несколькими десятками программистами и состоит из тысячи файлов, то лучше всего подойдет Git или Mercurial. Если основным критерием является удобный интерфейс, а разрабатываемый проект небольших размеров, то система Bazaar будет наилучшим вариантом. Для программистов-одиночек или небольших проектов, не требующих ветвления и создания множества версий, лучше всего подойдет Subversion.

Все эти системы контроля версий ориентированы на профессиональных программистов и предназначены для ведения проектов в команде разработчиков. Но ни одна из вышеупомянутых систем не является обучающей и не рассчитана на ведение проекта под руководством преподавателя. Поэтому, был разработан программный инструмент, который включает в себя все необходимые возможности для организации взаимодействия участников команды разработки, обеспечения контроля за разработкой программного проекта и управления самим процессом разработки согласно техническому заданию:

1) Веб-разработка должна иметь возможность авторизации на сайте и классификацию пользователей в зависимости от назначенной роли в команде.

2) Веб-разработка должна предоставлять каждому пользователю личный кабинет, в котором хранится информация о самом пользователе и его промежуточные работы.

3) Веб-разработка должна иметь общий сервер, где хранятся все файлы проекта от всех участников команды, и который фиксирует дату создания или изменения, имя создавшего и техническую информацию о файле.

4) Веб-разработка должна иметь возможность загрузки, хранения, сортировки файлов в текстовом, графическом форматах и формате кода.

5) Веб-разработка должна вести историю хранилища, которая позволяет не только контролировать активность участников проекта, но и отслеживать правильность выполнения задания.

6) Веб-разработка должна иметь различные средства коммуникаций, такие как общий чат, форум, система сообщений.

7) Веб-разработка должна вести рейтинг студентов, который составляется из активности пользователя, а именно: посещаемость, периодичность выкладывания файлов на сервер и их изменение, участие в обсуждениях.

8) Веб-разработка должна обеспечивать проверку на полноту проекта, то есть контролировать последовательность выполнения всех этапов водопадной модели процесса разработки.

9) Веб-разработка должна обеспечивать проверку на наличие полной документации программного проекта на различных этапах его реализации.