МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ БІЗНЕС-КОЛЕДЖ

***Циклова комісія програмування***

ІНДЗ

Реферат

на тему системи контролю версій

**Робочий план**

з курсу «Основи інформатики, програмування та алгоритмічні мови»

*Приходька Вадима Ігоровича*

студента групи **2П-19**

Викладач Марченко С. В.

Черкаси-2020

зміст

Вступ

1. Поняття системи контролю версій

1.1 Розподілені системи управління версіями

2. Види систем контролю версій

2.1 CVS

2.2 Subversion

2.3 Arch

2.4 OpenCM

2.5 Aegis

2.6 Monotone

2.7 BitKeeper

2.8 Perforce

2.9 Darcs

висновок

Список літератури.

Додаток 1.

Додаток 2.

Вступ

Сьогодні в світі, де існує величезна кількість складних систем, існує необхідність видозміни електронних документів на різних стадіях їх розробки. За час свого існування електронний документ може бути підданий великій кількості змін. Однак часто так буває, що для подальшої роботи необхідна не тільки остання версія документа, але і різні попередні варіанти.

Безумовно, можна зберігати кілька різних варіантів необхідного документа, але даний спосіб неефективний. Нам доводиться витрачати купу часу і сил, необхідно особливу увагу і велика ймовірність помилки. Крім того нам доводиться зберігати величезну кількість практично ідентичних документів.

Внаслідок цього були розроблені програмні засоби, які спрощують даний механізм. Дані кошти іменуються системами контролю версій. Існує кілька такого роду систем, кожна з яких є актуальною при певних умовах їх використання.

Метою даного реферату є розгляд різного роду систем контролю версій.

Відповідно до поставленої мети необхідно вирішити такі завдання:

o визначити поняття системи контролю версій;

o проаналізувати існуючі системи контролю версій;

o розглянути основні види даного роду систем з зарубіжної і російської практики.

Завдання роботи визначили її структуру. Реферат складається з трьох розділів, вступу, висновків, списку використаних джерел та додатки.

Теоретичною базою роботи є наукова, російська і зарубіжна періодика і ресурси мережі Інтернет.

1. Поняття системи контролю версій

Система управління версіями (Version Control System або Revision Control System) представляють собою програмне забезпечення для полегшення діяльності до мінливої ​​інформацією. Система контролю версій надає можливість зберігати кілька варіантів одного і того ж документа. При необхідності можна повернутися до старих версій, можна дізнатися, ким були зроблені ті чи інші зміни і т.д.

Такого роду системи в більшості своїй використовуються при розробці програмного забезпечення, щоб можна було зберігати вихідні коди програм, що розробляються. Система контролю версій дозволяє розробникам зберігати минулі версії файлів з розробки і діставати їх звідти. Вона зберігає інформацію про версії кожного файлу (і повну структуру проекту) в колекції, яку часто називають репозиторієм. Але тим не менше дані системи можуть використовуватися і в інших областях знань, які включають в себе величезну кількість часто змінюються електронних документів. Наприклад, вони все частіше застосовуються в САПР, зазвичай, у складі систем управління даними про виріб (PDM). Управління версіями використовується в інструментах конфігураційного управління.

Теки сховища можуть бути кілька паралельних ліній розробки, зазвичай званих гілками. Це може бути корисно для зберігання стабільної або випущеної версії гілки, одночасно продовжуючи роботу над робочою версією. Інший варіант - це відкрити виділену гілка для роботи над експериментальне можливістю.

Система контролю версій також дозволяє користувачам дати ярлик знімку гілки (часто званих як теги) для легкого діставання. Це корисно для позначення індивідуальних релізів або найсвіжіших робочих версій, призначених для використання.

Використання системи контролю версій безумовно обов'язково для розробника, якщо проект більше кількох сотень рядків або якщо для проекту спільно працюють кілька розробників. Використання оптимальної системи контролю версій безперечно краще, ніж використання вузько-спрямованих спеціальних методів, які використовують деякі розробники для управління різними ревізіями свого коду.

Більшість систем управління версіями використовують централізовану модель, коли є єдине сховище документів, кероване спеціальним сервером, який і виконує велику частину функцій з управління версіями. Користувач, який працює з документами, повинен спочатку отримати потрібну йому версію документа зі сховища; зазвичай створюється локальна копія документа, т. н. «Робоча копія». Може бути отримана остання версія або будь-яка з попередніх, яка може бути обрана за номером версії або датою створення, іноді і за іншими ознаками. Після того, як в документ внесені потрібні зміни, нова версія поміщається в сховище. На відміну від простого збереження файлу, попередня версія не стирається, а також залишається в сховище і може бути звідти отримана в будь-який час. Сервер може використовувати т. Н. дельта-компресію - такий спосіб зберігання документів, при якому зберігаються тільки зміни між послідовними версіями, що дозволяє зменшити обсяг збережених даних. [1]

Дуже часто над одним і тим же проектом працює кілька людей. Якщо один з них буде змінювати вихідний файл, і одночасно з цим інша людина буде виконувати аналогічну операцію, то можлива така ситуація, що якісь зміни можуть не зберегтися. Системи контролю версій працюють з такого роду проблемами і мають певний перелік їх вирішення. Здебільшого ці системи можуть автоматично об'єднувати такого роду зміни, які роблять різні члени команди розробників. Але варто зазначити, що такого роду об'єднання найчастіше виконується для текстових файлів і з певною умовою: зміни відбувалися в різних частинах файлу. Дане обмеження має місце, оскільки в більшості своїй системи контролю версій спрямовані на підтримку процесу розробки програмних продуктів, а початкові коди знаходяться в текстових файлах. У разі якщо автоматично виконати об'єднання не вийшло, то система пропонує виправити ситуацію вручну.

Найчастіше здійснити об'єднання не можливо ні з допомогою системи, ні вручну. Яскравим прикладом є ситуація коли формат файлу дуже складний або невідомий. Окремі системи контролю версій надають можливість користувачу заблокувати файл в сховище. Дана операція не дозволяє іншим користувачам отримати робочу копію або перешкоджає зміні робочої копії файлу і забезпечує, таким чином, винятковий доступ тільки тому користувачеві, який працює з документом. [1]

Багато системи управління версіями надають ряд інших можливостей:

 Дозволяють створювати різні варіанти одного документа, т. Н. гілки, із загальною історією змін до точки розгалуження і з різними - після неї.

 Дають можливість дізнатися, хто і коли додав або змінив конкретний набір рядків у файлі.

 Ведуть журнал змін, в який користувачі можуть записувати пояснення

1.1 Розподілені системи управління версіями

Існують системи управління версіями, які, замість традиційної клієнт-серверної, використовують розподілену модель. Такі системи, в загальному випадку, не потребують централізованому сховищі: вся історія зміни документів зберігається на кожному комп'ютері. Фактично, кожен комп'ютер, крім робочої копії, зберігає локальну копію всього сховища. У деяких системах робоча копія сама є сховищем.

Коли користувач такої системи виконує звичайні дії, такі як витяг певної версії документа, створення нової версії тощо, він працює зі своєю локальною копією сховища. У міру внесення змін, копії, що належать різним розробникам, починають різнитися і виникає необхідність в їх синхронізації. Така синхронізація може здійснюватися за допомогою обміну патчами або так званими наборами змін між користувачами.

Описана модель аналогічна створенню окремої гілки для кожного розробника в класичній системі управління версіями (в деяких розподілених системах перед початком роботи з локальним сховищем потрібно створити нову гілку). Поки розробник змінює лише свою гілку, його робота не впливає на інших учасників проекту та навпаки. Однак при необхідності виконати злиття гілок (або синхронізацію локальних сховищ в розподіленої моделі) можуть виникнути конфлікти.

Основна перевага розподілених систем полягає в їх гнучкості. Кожен розробник може вести роботу незалежно, так, як йому зручно, зберігаючи проміжні варіанти документів і передаючи результати іншим учасникам, коли вважатиме за потрібне. При цьому обмін наборами змін може здійснюватися за різними схемами. У невеликих колективах учасники роботи можуть обмінюватися змінами за принципом «кожен з кожним», за рахунок чого відпадає необхідність у створенні виділеного сервера. Велике співтовариство, навпаки, може використовувати централізований сервер, з яким синхронізуються копії всіх його учасників. Можливі й більш складні варіанти - наприклад, зі створенням груп для роботи за окремими напрямами всередині більшого проекту.

Розподілена система контролю версій дозволяє клонувати віддалений репозиторій, виробляючи точну копію. Вона також дозволяє розповсюджується зміни з одного сховища на інший. У нерозподілених VCS розробнику потрібен репозиторій для того, щоб зафіксувати зміни в ньому. Це робить розробника без доступу до сховища людиною другого сорту. З розподіленої VCS кожен розробник може склоніровать головний репозиторій, попрацювати над ним і потім поширити свої зміни на головний репозиторій.

2. Види систем контролю версій

Система контролю версій може бути будь-якої форми і розмірів, але є основні положення про їх архітектурі. Деякі системи підтримують Атомарні пакунків, які означають, що стан всього сховища змінюється повністю. Без атомарний фіксації, кожен файл або частину змінюється окремо і тому стан всього сховища в будь-якій точці не може бути зафіксовано.

Більшість звичайних VCS (Version Control System) систем дозволяють об'єднувати зміни між гілками. Це означає, що зміни, зафіксовані в одній гілці, будуть зафіксовані в головній лінії або в іншій гілці за допомогою однієї автоматичної або, по крайней мере, напівавтоматичного операцією.

2.1 CVS

Традиційно, де-факто системою контролю версій стала CVS (Система Паралельних Версій), але потім з'явилося багато інших, які допомагають краще. З усіх цих можливостей CVS підтримує тільки об'єднання змін.

Це зріла і щодо надійна система контролю версій. Багато проектів з відкритими вихідними кодами, включаючи KDE, GNOME і Mozilla використовують CVS. Більшість центрів відкритих вихідних кодів, такі як SourceForge, пропонують CVS як сервіс, тому її використовують у багатьох інших проектах.

Система CVS має архітектуру клієнт-сервер. Зазвичай сервер і клієнт з'єднуються через локальну мережу або через Інтернет, але можуть працювати і на одній машині, якщо необхідно вести історію версій локального проекту. Серверне програмне забезпечення зазвичай працює під управлінням Unix, тоді як CVS клієнти доступні у всіх популярних операційних системах. (4)

Сервер зберігає в репозиторії поточну версію (версії) проекту і історію змін, а клієнт з'єднується з ним, щоб отримати потрібну йому версію або записати нову. Отримавши з сервера потрібну версію, клієнт створює локальну копію проекту (або його частини) - так звану робочу копію. Після того як в файли, що знаходяться в робочій копії, внесені необхідні зміни, вони пересилаються на сервер. (4)

Користувачі мають можливість порівнювати порівняти всілякі версії файлів, переглянути історію змін або навіть отримати історичний образ проекту на певну дату. Багато проектів формату «Відкритих ресурсів» дозволяють анонімний доступ на читання. Дана можливість передбачає, що користувачі, можуть отримати доступ до версій файлів без пароля.

CVS також може містити різні гілки проекту. Наприклад, стабільна версія проекту може становити одну гілку, до якої вносяться тільки виправлення помилок, тоді як активна розробка може вестися в паралельній гілці, яка включає значні поліпшення або зміни з моменту виходу стабільної версії. [4]

Крім того, варто також зазначити, що дана система використовує механізм дельта-компресії, щоб більш ефективно зберігати різноманітні версії одного і того ж файлу.

Незважаючи на свою популярність, CVS має обмеження. Наприклад, вона не підтримує перейменування файлів і директорій. Крім того, бінарні файли не обробляються досить добре. CVS - нерозподілений система і атомарні фіксації змін не підтримуються. Так як вже є кращі альтернативи, які містять більш широкий набір функцій, ви, можливо, вважатимете за краще почати новий проект використовуючи щось інше. (2)

Як плюс, CVS дуже добре документована в своїй книзі і в багатьох онлайн посібниках. Також існують безліч графічних клієнтів і доповнень.

2.2 Subversion

Subversion прагне бути кращою альтернативою CVS (див. Додаток 1). Вона підтримує більшість угод CVS, включаючи велику частину набору команд, тому користувачі CVS швидко відчувають себе як вдома. Subversion пропонує багато корисних поліпшень в порівнянні з CVS: копіювання і перейменування файлів і директорій, справжні атомарні фіксації, ефективна обробка бінарних файлів, здатність мережевої роботи по HTTP (і HTTPS). Subversion також має Win32 клієнт і сервер.

Таким чином, можна виділити наступні можливості Subversion:

 Реалізовано більшість можливостей CVS;

 Відстежується історія файлів, директорій і метаданих файлів, в тому числі при перейменуванні і копіюванні;

 Публікації змін атомарний;

 Можливість організації доступу до сховища Subversion через Apache по протоколу WebDAV / DeltaV;

 Можливість установки автономного сервера Subversion з доступом за власним протоколу;

 «Дешеві» операції створення гілок і міток (потрібне невелике фіксовану кількість тимчасових і дискових ресурсів);

 Багаторівнева архітектура бібліотек, спочатку розрахована на клієнт-серверну модель;

 Клієнт-серверний протокол пересилає по мережі тільки різницю між об'єктами, коли це можливо;

 Витрати ресурсів пропорційні розміру змін, а не розміром даних, які порушені змінами;

 Два можливих внутрішніх формату сховища: база даних або простий файл;

 Версіонірованние символьні посилання (тільки в робочих копіях під UNIX-системами);

 Однаково ефективна робота і з текстовими, і з двійковими файлами;

 Висновок клиен

2.3 Arch

система контроль версія документ

GNU Arch - це VCS, спочатку створена Томом Лордом (Tom Lord) для своїх потреб. Спочатку Arch була колекцією shell-скриптів, але зараз її основний клієнт це tla, який написаний на C і повинен переноситися на будь-яку UNIX-систему. Він не був портований на Win32; хоча це і можливо зробити, це не пріоритет для проекту. (2)

Arch - розподілена система контролю версій. Вона не вимагає спеціального сервісу для установки мережевого сховища та підходить будь-який віддалений файловий сервіс (такі як FTP, SFTP або WebDAV). Це робить установку сервісу неймовірно легкою.

Arch підтримує версійні перейменування файлів і директорій, а також інтелектуальне об'єднання, яке може визначити, чи був файл перейменований і потім чисто застосовує зміни. Arch прагне бути краще CVS, але все ще є деякі припущення можливості. Версія Arch переступила 1.0 поріг і тому оголошена зрілим і надійним продуктом для будь-якого використання.

Arch документована дуже простий онлайн системою допомоги і керівництва.

2.4 OpenCM

OpenCM - система контролю версій, створена для проекту EROS. OpenCM не прагне бути багатою можливостями як CVS, але вона має кілька переваг. OpenCM має версійні перейменування файлів і директорій, атомарні фіксації, автоматичне поширення змін від гілки в головну гілку і деяку підтримку криптографічного аутентифікації.

Розглянемо перелік основних особливостей OpenCM:

• Дана система призначена для реальної конфігурації. Це смішно, що не знає CVS;

• Здатність перейменовувати файли, не втрачаючи їх історії;

• Управління доступом до історії по гілках;

• шифрувальні встановлення автентичності. Це забезпечує здатність зробити звіти розробників на репозиторії OpenCM, не даючи їм звіт на основний машині (OS), і робить співпрацю мульти-організацій можливим.

• Безперервні засоби управління за цілісністю. Якщо у сервера є поганий файл, або сервер мультиплицирования активно намагається замінити належне утримання, кінцевий користувач може виявити помилку або заміну. (6)

OpenCM використовує свій власний протокол для зв'язку мужду клієнтом і сервером. Система не є розподіленою. Так як OpenCM не відрізняється багатством можливостей, можливо, що інші системи вам підійдуть більше. Однак ви можете віддати перевагу OpenCM, якщо якась відмінна можливість цієї системи вам сподобається.

OpenCM працює під будь-який UNIX -системою і на Windows під емуляцією Cygwin. Вона має CVS-подібні команди і добре документована.

2.5 Aegis

Aegis - система управління конфігурацією коду створена Пітером Міллером (Peter Miller). Вона не мережева і всі операції робляться через файлову систему UNIX. По суті, вона використовує систему дозволів UNIX для визначення, хто має доступ до виконання будь операції. Незважаючи на той факт, що Aegis НЕ мережева, вона все ж розподілена в тому сенсі, що сховища можуть бути клоновані і зміни можуть бути поширені з одного сховища на інший. Використання по мережі вимагає мережевої операційної системи, такої як NFS.

Будучи SCM системою, Aegis намагається забезпечити коректність коду, який був внесений. Тобто, вона:

• Управляє автоматизованими тестами, запобігає внесення, які не проходять попередні тести і вимагає розробників додати нові тести.

• Управляє оглядами коду. Внесення повинні пройти огляд оглядача, щоб потрапити в головну лінію розробки.

• Має різні інші можливості, які допомагають гарантувати якість коду.

• Її набір команд відображає цю філософію і є дуже нудним, якщо вам потрібна просто система контролю версій.

Aegis документована в декількох troff документах, які були перетворені в PostScript. Іноді буває важко переглянути документацію, щоб знайти точно те, що вам потрібно. Однак документація високої якості.

2.6 Monotone

Система контролю версій Monotone була створена Грейдон Хоем (Graydon Hoare) і показує різну філософію, ніж всі системи, наведені вище. Вона розподілена, з наборами змін, що розповсюджуються через певний сховище, яке може бути CGI скрипт, NNTP (новини Usenet) одержувач або SMTP (email). Звідти, кожен розробник поміщає бажані зміни в свою власну копію сховища.

Це може мати сумний ефект втрати синхронізації історії змін або поточного стану індивідуальних репозиторіїв один з одним, так як індивідуальні репозиторії не отримують відповідні зміни або отримують невідповідні.

Monotone дуже залежить від сильного шифрування. Вона ідентифікує файли, директорії і ревізії контрольними сумами SHA 1.

Monotone підтримує перейменування і копії файлів і директорій. Вона має набір команд, який прагне бути CVS -Сумісність, з деякими необхідними відступами, через свою різної філософії. Вона повинна бути переносимо на Win32, але офіційно перенесення ще не був випо

## 2.7 BitKeeper

BitKeeper- не является системой контроля версий с открытыми кодами, но рассмотрена здесь для полноты картины, потому что некоторые проекты с открытыми кодами используют ее. BitKeeper очень надежна и богата возможностями, поддерживая распределенные репозитории; работая по HTTP ; поддерживая переименование и копирование файлов и директорий; управление патчами; распространяя изменения от ветви в главную линию и много других возможностей.

BitKeeper распространяется с двумя лицензиями. Коммерческая лицензия стоит несколько сот долларов на место (лизинг или покупка). Бесплатная лицензия доступна для разработки ПО с открытыми кодами, но имеет некоторые ограничения, среди которых – условие о неконкуренции (создаваемый продукт не может быть конкурентным по отношению к данному) и требование об обновлении системы при появлении более свежей версии, даже если она имеет другую лицензию. Кроме того, исходный код не доступен публично и бинарники существуют только для самых распространенных систем, включая Win 32.

Преимущества BitKeeper:

* Высокая производительность: BitKeeper была разработана, чтобы упростить исходные задачи управления и обеспечить превосходную инфраструктуру для отладки и рассмотрения кода.
* Снижена вероятность человеческой ошибки: BitKeeper осуществляет проверки целостности репозитория, которые немедленно улавливают проблемы.
* Воспроизводимость: Комплексы проектов программного обеспечения с множеством разработчиков требуют инструментов управления конфигурации программного обеспечения, которые учитывают точную воспроизводимость прошлой и настоящей информации. Поскольку BitKeeper поддерживает понятие логической единицы работы, где каждая единица является неизменной - не может измениться, но может быть добавлено - BitKeeper производит полностью восстанавливаемый репозиторий в любой момент времени. BitKeeper управляет процессом развития так, чтобы каждая фаза проекта может быть обновлена в будущем.
* Ответственность: Поскольку репозитории полностью восстанавливаемы в любой момент времени, легко узнать, кто сделал, какие изменения, и что другие файлы были изменены в то же самое время. Отладка становится намного более эффективным с BitKeeper.
* Разъединенные/Распределенные операции: Рабочее пространство каждого пользователя содержит историю пересмотра файлов таким образом, чтобы вся работа могла возобновить без любого взаимодействия с главным репозиторием, таким образом это не потребность, чтобы иметь связь TCP между всеми системами все время.
* Масштабируемость: Архитектура BitKeeper's неотъемлемо масштабируема, так, что одинаково хорошо работает и для пяти разработчиков, и для 1 000 или 10 000. (8)

Горстка проектов используют BitKeeper, включая некоторых из разработчиков Linux ядра и основная команда разработчиков MySQL . Из-за ее лицензии BitKeeper не удобен для разработки ПО с открытым кодом, так как оно заставит отвернуться многих «идеалистических» разработчиков и создаст различные проблемы для пользователей, кто выберет ее для использования. Если вы работаете над непубличным проектом и можете позволить себе заплатить за BitKeeper , это естественный вариант.

2.8 Perforce

Дана комерційна система управління версіями розроблена компанією Perforce Software. В основі неї лежить клієнт-серверна архітектура. Сервер даної системи може одночасно мати кілька репозиторіїв. Сервер Perforce може бути встановлений на операційні системи Unix, Mac OS X, Microsoft Windows.

Клієнт надає графічний інтерфейс і широкий набір утиліт для роботи з командного рядка. Клієнтська частина реалізована для широкого набору операційних систем. Також розроблений великий набір плагінів, що дозволяють інтегруватися з широким колом середовищ розробки програмного забезпечення і додатків інших розробників: IntelliJ IDEA, XCode, Autodesk 3D Studio Max, Maya, Adobe Photoshop, Microsoft Office, Eclipse, emacs. Крім цього система надає безліч інших можливостей - різного виду повідомлення, створення і обслуговування гілок проекту, з потужною системою злиттів гілок, точки відкату в базі даних, і взаємодія з системами відстеження дефектів. (9)

В даний час такі компанії як Google і Microsoft, широко використовують Perforce в своїх інженерних процесах.

Основні недоліки даної системи:

• висока ціна ліцензії на сервер;

• поширюється в бінарному вигляді. (1)

2.9 Darcs

Дана система контролю версій є розподіленою, подібно Arch і Monotone.

Dars має кілька вельми корисних особливостей:

• Офлайн-режим. Ви можете підтверджувати зміни, навіть якщо ви не маєте доступу до сервера. Це працює, оскільки директорія вашого проекту, керована Darcs є повноцінним репозиторієм, всі зміни зберігаються в ньому. Ви приїжджаєте додому або на роботу, підключаєте ноутбук до мережі і просто переносите все раніше зроблені зміни за допомогою "darcs push" на публічно доступна сервер.

• Простота розгалуження. У Dars кожен репозиторій фактично є окремою гілкою. Чи працюєте над новою функцією, але треба було терміново виправити помилку? Тоді публікуєте зміни стосуються багфіксів, підтверджуєте зміни і продовжуєте роботу над своєю новою функцією - ваше сховище буде гілкою основний розробки.

• Режим Лінуса. Наприклад, ви хочете додати нову функцію або виправити помилку в open-source проект. Ви робите локальну копію сховища в Darcs, робите потрібні зміни і відправляєте патчі по електронній пошті. Майнтейнер проекту вирішує прийняти або відхилити зміни. Таким чином, вам не потрібно безпосереднього доступу запису до головного сховища проекту. Таким чином підтримується сам проект Darcs, подібним же шляхом Лінус Торвальдс ведеться (велася раніше?) Розробка ядра Linux.

• Паралельна розробка. Нова гілка розробки ведеться паралельно з основною і при необхідності може бути об'єднана з нею в будь-який час. Таким чином, легко розділити розробку, але в той же час легко об'єднувати паралельно ведені розробки, інтегруючи напрацювання незалежних розробників - "форк" перестає бути тим злом, яким його намагаються представити представники пропрієтарного програмного забезпечення.

• Простота ведення версій файлів конфігурації. Централізоване ведення загальної конфігурації на будь-якій кількості машин стає простою справою, не тільки тому, що Darcs підтримує перенесення змін, ви можете налаштувати Darcs, щоб він робив це перенесення на всі машини відразу однією командою.

• Cherry-picking ( "Збираємо вершки"). Навіть працюючи в команді, може скластися ситуація, коли хтось має корисні зміни, які вам потрібні, але поки не має можливості їх опублікувати в основну гілку. З Darcs ви можете забрати до себе тільки ті зміни в репозиторії, які вам потрібні в даний момент. [10]

висновок

У роботі були розглянуті системи контролю версій, проведено короткий порівняльний аналіз систем.

Такого роду системи дозволяють полегшити проекти по розробки програмного забезпечення, і в певній мірі виключити людські помилки. Існуючі системи моніторингу допоможуть в разі виникнення будь-яких нестиковок в роботі.

В даний час існує велика кількість систем контролю версій, і найбільш цікаві та інформативні варіанти представлені в роботі. Поки не існує ідеальних систем, кожна з них має певні недоліки, але одночасно з цим і низкою особливостей, які виділяють її.

На зміну щодо старих систем (CVS) приходять поліпшені альтернативи. Інші системи приємніші і надають кращий робочий досвід. У споживача, безумовно, є багато можливостей для вибору. Більшість виробників пропонують демонстраційні версії своїх програмних продуктів.

Системи контролю версій є приголомшливим рішення проблем поширених корпорацій. Безумовно дана область буде розвиватися і надалі можливе збільшення числа систем і істотне доопрацювання існуючих.

Список літератури

1. Вікіпедія [Електронний ресурс] - Доступ до ст .: http://ru.wikipedia.org/wiki/

2. Нове покоління систем контролю версій [Електронний ресурс] - Доступ до ст .: http://www.techinfo.net.