МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ БІЗНЕС-КОЛЕДЖ

***Циклова комісія програмування***

**РЕФЕРАТ**

з навчальної дисципліни: „Основи алгоритмізації та програмування ”

на тему: «Системи контролю версій»

**Виконали:**

Студенти 2 курсу

спеціальності: «ІПЗ»

групи: 2П-20

Дворяківська Єлизавета Георгіївна

Чухало Ілля Віталійович

**Перевірив викладач:**

Марченко С.В.

Черкаси 2021

**ЗМІСТ:**

Вступ

1. Поняття системи контролю версій

1.1 Розподілені системи керування 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.

**ВСТУП**

Сьогодні у світі, де існує безліч складних систем, існує необхідність видозміни електронних документів на різних стадіях їх розробки. За час свого існування електронний документ може бути схильний до великої кількості змін. Однак часто так буває, що для подальшої роботи потрібна не тільки остання версія документа, але й різні попередні варіанти.

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

Внаслідок цього були розроблені програмні засоби, які спрощують цей механізм. Ці засоби називаються системами контролю версій. Існує кілька таких систем, кожна з яких актуальна за певних умов їх використання.

Метою даного реферату є розгляд різноманітних систем контролю версій.

Відповідно до поставленої метою необхідно вирішити такі завдання:

* визначити поняття системи контролю версій;
* проаналізувати існуючі системи контролю версій;
* озглянути основні види такого роду систем із зарубіжної та російської практики.

**1. Поняття системи контролю версій**

Системи управління версіями (Version Control System або Revision Control System) являють собою програмне забезпечення для полегшення діяльності з інформацією, що швидко змінюється. Система контролю версій надає можливість зберігати кілька варіантів одного й того самого документа. За необхідності можна повернутися до старих версій, можна дізнатися, ким було зроблено ті чи інші зміни тощо.

Такі системи в більшості своїй використовуються при розробці програмного забезпечення, щоб можна було зберігати вихідні коди програм, що розробляються. Система контролю версій дозволяє розробникам зберігати попередні версії файлів із розробки та діставати їх звідти. Вона зберігає інформацію про версію кожного файлу (і повну структуру проекту) у колекції, яка зазвичай називається репозиторієм. Проте дані системи можуть використовуватися і в інших галузях знань, які включають величезну кількість електронних документів, що часто змінюються. Наприклад, вони все частіше застосовуються в САПР, як правило, у складі систем управління даними про продукт (PDM). Керування версіями використовується у інструментах конфігураційного керування.

Усередині репозиторію може бути кілька паралельних ліній розробки, зазвичай званих гілками. Це може бути корисним для зберігання стабільної або випущеної версії гілки, одночасно продовжуючи роботу над робочою версією. Інший варіант – це відкрити виділену гілку для роботи над екпериментальною можливістю.

Система контролю версій також дозволяє користувачам дати ярлик знімку гілки (часто званих як тэги) 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 хорошої системи контролю версій безумовно краще, ніж використання вузькоспрямованих 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 виконує більшу частину функцій з управління версіями. Користувач, який працює з документами, повинен спочатку отримати потрібну версію документа зі сховища; зазвичай створюється локальна копія документа. Може бути отримана остання версія або будь-яка з попередніх, яка може бути обрана за номером версії або датою створення, іноді за іншими ознаками. Після того, як в документ внесено потрібні зміни, нова версія міститься у сховищі. На відміну від простого збереження файлу, попередня версія не стирається, а також залишається в сховищі і може бути отримана звідти в будь-який час. Сервер може використовувати т.з. дельта-компресію - такий спосіб зберігання документів, у якому зберігаються лише зміни між послідовними версіями, що дозволяє зменшити обсяг збережених даних.

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

Найчастіше здійснити об'єднання неможливо ні з допомогою системи, ні вручну. Яскравим прикладом є ситуація, коли формат файлу дуже складний або невідомий. Окремі системи контролю версій дозволяють користувачеві заблокувати файл у сховищі. Ця операція не дозволяє іншим користувачам отримати робочу копію або перешкоджає зміні робочої копії файлу і забезпечує, таким чином, винятковий доступ лише користувачеві, який працює з документом.

Багато систем управління версіями надають ряд інших можливостей:

* Дозволяють створювати різні варіанти одного документа, т.з. гілки, із загальною історією змін до точки розгалуження та з різними – після неї.
* Дають можливість дізнатися, хто і коли додав або змінив конкретний набір рядків у файлі.
* Ведуть журнал змін, до якого користувачі можуть записувати пояснення про те, що і чому вони змінили цю версію.
* Контролює права доступу користувачів, дозволяючи або забороняючи читання або зміну даних, залежно від того, хто запитує цю дію.

**1.1 Розподілені системи керування версіями**

Існують системи управління версіями, які замість традиційної клієнт-серверної використовують розподілену модель. Такі системи, загалом, не потребують централізованого сховища: вся історія зміни документів зберігається 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 видалення певної версії документа, створення нової версії тощо, він працює зі своєю локальною копією сховища. У міру внесення змін копії, що належать різним розробникам, починають розрізнятися і виникає необхідність їх синхронізації. Така синхронізація може здійснюватися за допомогою обміну патчами або про набори змін між користувачами.

Описана модель аналогічна до створення окремої гілки для кожного розробника в класичній системі управління версіями (у 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 злиття гілок (або синхронізацію локальних сховищ у розподіленій моделі) можуть виникнути конфлікти.

Основна перевага розподілених систем полягає в їхній гнучкості. Кожен розробник може вести роботу незалежно, оскільки йому зручно, зберігаючи проміжні варіанти документів і передаючи результати іншим учасникам, коли вважає за потрібне. При цьому обмін наборами змін може здійснюватись за різними схемами. У невеликих колективах учасники можуть обмінюватися змінами за принципом «кожний з кожним», за рахунок чого відпадає необхідність у створенні виділеного сервера. Велика спільнота, навпаки, може використовувати централізований сервер, з яким синхронізуються копії всіх учасників. Можливі і складніші варіанти - наприклад, зі створенням груп до роботи з окремих напрямів усередині більшого проекту.

Розподілена система контролю версій дозволяє клонувати віддалений репозиторій, роблячи точну копію. Вона також дозволяє розповсюджувати зміни з одного репозиторію на інший. У нерозподілених VCS розробнику потрібен репозиторій у тому, щоб зафіксувати зміни у ньому. Це робить розробника без доступу до репозиторію людиною другого гатунку. З розподіленою VCS кожен розробник може схиляти головний репозиторій, попрацювати з нього і потім поширити свої зміни на головний репозиторій.

**2. ВИДИ СИСТЕМ КОНТРОЛЮ ВЕРСІЙ**

Система контролю версій може бути будь-якої форми та розмірів, але є основні положення про їхню архітектуру. Деякі системи підтримують Атомарні Фіксації, які означають, що стан всього репозиторію змінюється повністю. Без Атомарних Фіксацій кожен файл або частина змінюється окремо і тому стан всього репозиторію в будь-якій точці не може бути зафіксовано.

Більшість звичайних VCS (Version Control System) систем дозволяють поєднувати зміни між гілками. Це означає, що зміни, зафіксовані в одній галузі, будуть зафіксовані в головній лінії або в іншій галузі за допомогою однієї автоматичної або, принаймні, напівавтоматичної операції.

**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 лише об'єднання змін.

Це зріла та відносно надійна система контролю версій. Багато проектів з відкритими вихідними кодами, включаючи KDE, GNOME та 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 та клієнт з'єднуються через локальну мережу або через Інтернет, але можуть працювати і на одній машині, якщо необхідно вести історію версій локального проекту. Серверне ПЗ зазвичай працює під управлінням Unix, тоді як CVS клієнти доступні у всіх популярних операційних системах.

Сервер зберігає в репозиторії поточну версію (версії) проекту та історію змін, а клієнт з'єднується з ним, щоб отримати потрібну версію або записати нову. Отримавши з сервера необхідну версію, клієнт створює локальну копію проекту (або його частини) - так звану робочу копію. Коли файли, що у робочої копії, внесено необхідні зміни, вони пересилаються на сервер.

Користувачі мають можливість порівнювати порівняти різні версії файлів, переглянути історію змін або навіть отримати історичний образ проекту на певну дату. Багато проектів формату «Відкритих ресурсів» дозволяють анонімний доступ до читання. Ця 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, в яку вносяться лише виправлення помилок, тоді як активна розробка може вестися в паралельній гілці, що включає значні покращення або зміни з моменту виходу стабільної версії.

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

Незважаючи на свою популярність, CVS має обмеження. Наприклад, вона не підтримує перейменування файлів та директорій. Крім того, бінарні файли не обробляються досить добре. CVS – нерозподілена система та атомарні фіксації змін не підтримуються. Оскільки вже є найкращі альтернативи, які містять ширший набір функцій, ви, можливо, вважаєте за краще почати новий проект, використовуючи щось інше.

Як плюс, CVS дуже добре документована у своїй книзі та у багатьох онлайн посібниках. Також існує безліч графічних клієнтів та доповнень.

**2.2 Subversion**

Subversion прагне бути найкращою альтернативою CVS (див. додаток 1). Вона підтримує більшість угод CVS, включаючи більшу частину набору команд, тому користувачі CVS швидко почуваються як удома. Subversion пропонує багато корисних покращень порівняно з CVS: копіювання та перейменування файлів та директорій, справжні атомарні фіксації, ефективна обробка бінарних файлів, здатність мережної роботи з HTTP (та HTTPS). Subversion також має Win32 клієнт та сервер.

Таким чином, можна виділити такі можливості Subversion:

* Реалізовано більшість можливостей CVS;
* Відстежується історія файлів, директорій та метаданих файлів, у тому числі при перейменуванні та копіюванні;
* Публікації змін атомарні;
* Можливість організації доступу до сховища Subversion через Apache за протоколом WebDAV/DeltaV;
* Можливість встановлення автономного сервера Subversion з доступом за власним протоколом;
* «Дешеві» операції створення гілок та міток (потрібна невелика фіксована кількість тимчасових та дискових ресурсів);
* Багаторівнева архітектура бібліотек, спочатку розрахована на клієнт-серверну модель;
* Клієнт-серверний протокол пересилає по мережі лише різницю між об'єктами, коли це можливо;
* Витрати ресурсів пропорційні розміру змін, а не розміру даних, які стосуються змін;
* Два можливі внутрішні формати репозиторію: база даних або простий файл;
* Версіоновані символьні посилання (тільки у робочих копіях під UNIX-системами);
* однаково ефективна робота і з текстовими, і з двійковими файлами;
* Виведення клієнта командного рядка однаково зручне і для читання, і для розбору програм;
* Часткова локалізація повідомлень (використовуються налаштування локалі);
* Бібліотеки для мов PHP, Python, Perl, Java;
* Можливість дзеркалювання репозиторію.

Subversion нещодавно почала бета-період, після того, як була довгий час в альфа-періоді. Тому, вона ще може мати невеликі чудасії і її продуктивність у деяких місцях невелика. І все-таки вона дуже корисна для свого бета-періоду і була такою навіть у більшій частині свого альфа-періоду.

Сервіс Subversion на основі HTTP (або HTTPS) важкий для розгортання, в порівнянні з іншими системами, оскільки він потребує встановленої служби Apache2 зі своїм спеціальним модулем. Також є "svnserve" сервер, який менш здатний, але більш простий в установці та використовує спеціальний протокол. Крім того, підтримка Subversion об'єднання змін обмежена і схожа на CVS (тобто, об'єднання гілок, де файли були переміщені не буде виконано коректно). Вона також є відносно вимогливою до ресурсів, особливо на великих операціях.

Subversion пропонує два варіанти організації репозиторіїв. Репозиторії першого типу використовують для зберігання бази даних на основі Berkeley DB, репозиторії другого типу - у звичайних файлах спеціального формату (доступ до даних організується за допомогою власних бібліотек, без використання сторонніх баз даних). Обидва типи репозиторіїв забезпечують достатню надійність при правильній організації, кожна з них має свої переваги і недоліки. Вважається, що другий тип легше правильно налаштувати, вона вимагає меншої уваги від адміністратора.

Subversion добре документована у своїй безкоштовній онлайн книзі, "Контроль Версій з Subversion". Невелика онлайн система допомоги, що поставляється з Subversion клієнтом, може бути корисною як довідник. Subversion має багато доповнень, але вони все ще менш зрілі, ніж їх конкуренти CVS.

**2.3 Arch**

***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 має переноситися на будь-яку UNIX-систему. Він не був портований Win32; Хоча це і можна зробити, це не пріоритет для проекту.

Arch - Розподілена система контролю версій. Вона не потребує спеціального сервісу для встановлення мережевого репозиторію та підходить будь-який віддалений файловий сервіс (такі як FTP, SFTP або WebDAV). Це робить встановлення сервісу неймовірно легким.

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

Arch документована дуже простою онлайн системою допомоги та 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 та директорій, атомарні фіксації, автоматичне розповсюдження змін від гілки до головної галузі та деяку підтримку криптографічної аутентифікації.

Розглянемо список основних особливостей OpenCM:

• Ця система призначена для реальної конфігурації. Це дивно, що не знає CVS;

• Здатність перейменовувати файли, не втрачаючи їхньої історії;

• управління доступом до історії по гілках;

• Шифрування автентичності. Це забезпечує здатність зробити звіти розробників на репозиторії OpenCM, не даючи їм звіту на основній машині (OS), і робить співпрацю мульти-організацій можливою.

• Безперервні засоби керування за цілісністю. Якщо сервер має поганий файл, або сервер мультиплікування активно намагається замінити належний вміст, кінцевий користувач може виявити помилку або заміну. (6)

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

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

**2.5 Aegis**

***Aegis*** - система управління конфігурацією коду створена Пітером Міллером (Peter Miller). Вона не мережна і всі операції робляться через файлову систему UNIX. Фактично, вона використовує систему дозволів UNIX визначення, хто має доступ виконання якої операції. Незважаючи на той факт, що Aegis не мережна, вона все ж таки розподілена в тому сенсі, що репозиторії можуть бути клоновані і зміни можуть бути поширені з одного репозиторію на інший. Використання мережі вимагає мережної операційної системи, такої як NFS .

Будучи SCM системою, Aegis намагається забезпечити коректність коду, який було внесено. Тобто вона:

• Керує автоматизованими тестами, запобігає внесенням, які не проходять попередні тести та потребують розробників додати нові тести.

• Керує оглядами коду. Внесення повинні пройти огляд браузера, щоб потрапити до головної лінії розробки.

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

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

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

**2.6 Monotone**

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

Це може мати сумний ефект втрати синхронізації історії змін або поточного стану індивідуальних репозиторіїв один з одним, оскільки індивідуальні репозиторії не набувають відповідних змін або набувають невідповідних.

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

Monotone підтримує перейменування та копії файлів та директорій. Вона має набір команд, який прагне бути CVS-сумісним, з деякими необхідними відступами через свою різну філософію. Вона повинна бути переносима на Win32, але офіційно перенесення ще не було виконано.

Monotone досі у розробці та може досі мати деякі проблеми у поведінці. Розробники Monotone очікують, що проблеми вирішаться, оскільки робота продовжується.

Загалом, Monotone містить багато обіцяного.

**2.7 BitKeeper**

**BitKeeper**- не є системою контролю версій з відкритими кодами, але розглянута тут для повноти картини, тому що деякі проекти з відкритими кодами використовують її. BitKeeper дуже надійна і багата на можливості, підтримуючи розподілені репозиторії; працюючи по HTTP; підтримуючи перейменування та копіювання файлів та директорій; керування патчами; поширюючи зміни від гілки в головну лінію та багато інших можливостей.

BitKeeper розповсюджується з двома ліцензіями. Комерційна ліцензія коштує кілька сотень доларів на місце (лізинг або покупка). Безкоштовна ліцензія доступна для розробки ПЗ з відкритими кодами, але має деякі обмеження, серед яких – умова про неконкуренцію (створюваний продукт не може бути конкурентним по відношенню до цього) та вимога про оновлення системи при появі свіжішої версії, навіть якщо вона має іншу ліцензію . Крім того, вихідний код не доступний публічно і бінарники існують лише для найпоширеніших систем, включаючи Win 32.

Переваги BitKeeper:

• Висока продуктивність: BitKeeper була розроблена, щоб спростити вихідні завдання управління та забезпечити чудову інфраструктуру для налагодження та розгляду коду.

• Знижено ймовірність людської помилки: BitKeeper здійснює перевірки цілісності репозиторію, які негайно вловлюють проблеми.

• Відтворюваність: Комплекси проектів програмного забезпечення з багатьма розробниками потребують інструментів управління конфігурації програмного забезпечення, які враховують точну відтворюваність минулої та теперішньої інформації. Оскільки BitKeeper підтримує поняття логічної одиниці роботи, де кожна одиниця є незмінною - не може змінитися, але може бути додано - BitKeeper виробляє репозиторій, що повністю відновлюється, в будь-який момент часу. BitKeeper управляє процесом розвитку так, щоб кожна фаза проекту може бути оновлена ​​у майбутньому.

• Відповідальність: Оскільки репозиторії повністю відновлюються в будь-який час, легко дізнатися, хто зробив, які зміни, і що інші файли були змінені в той же час. Налагодження стає набагато ефективнішим з BitKeeper.

• Роз'єднані/Розподілені операції: Робочий простір кожного користувача містить історію перегляду файлів таким чином, щоб вся робота могла відновити без будь-якої взаємодії з головним репозиторієм, таким чином це не потреба мати зв'язок TCP між усіма системами весь час.

• Масштабованість: Архітектура BitKeeper's невід'ємно масштабується так, що однаково добре працює і для п'яти розробників, і для 1 000 або 10 000.

Жменька проектів використовують BitKeeper, включаючи деяких із розробників Linux ядра та основна команда розробників MySQL. Через її ліцензію BitKeeper не зручний для розробки програмного забезпечення з відкритим кодом, так як воно змусить відвернутися багатьох «ідеалістичних» розробників та створить різні проблеми для користувачів, хто вибере її для використання. Якщо ви працюєте над непублічним проектом і можете дозволити собі заплатити за BitKeeper, це природний варіант.

**2.8 Perforce**

Ця комерційна система управління версіями розроблена компанією Perforce Software. В основі її лежить клієнт-серверна архітектура. Сервер даної системи може мати кілька репозиторіїв. Сервер Perforce можна встановити на операційні системи Unix, Mac OS X, Microsoft Windows.

Клієнт надає графічний інтерфейс та широкий набір утиліт для роботи з командного рядка. Клієнтська частина реалізована широкого набору операційних систем. Також розроблений великий набір плагінів, що дозволяють інтегруватися з широким колом середовищ розробки програмного забезпечення та програм 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 сповіщення, створення та обслуговування гілок проекту, з потужною системою злиття гілок, точки відкату в базі даних, та взаємодія з системами відстеження дефектів. (9)

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

Основні недоліки даної системи:

• висока ціна ліцензії на сервер;

• поширюється у бінарному вигляді

**2.9 Darcs**

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

Dars має кілька дуже корисних особливостей:

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

• Простота розгалуження. У Dars кожен репозиторій фактично є окремою гілкою. Працюєте над новою функцією, але потрібно терміново виправити помилку? Тоді публікуєте зміни, що стосуються багфіксу, підтверджуєте зміни та продовжуєте роботу над своєю новою функцією – ваш репозиторій буде гілкою основної розробки.

• Режим Лінусу. Наприклад, ви бажаєте додати нову функцію або виправити помилку в open-source проекті. Ви робите локальну копію репозиторію в Darcs, робите потрібні зміни та відправляєте патчі електронною поштою. Майнтейнер проекту вирішує прийняти чи відхилити зміни. Таким чином, вам не потрібно безпосереднього доступу до запису в репозиторій проекту. Таким чином підтримується сам проект Darcs, подібним шляхом Лінусом Торвальдсом ведеться (велася раніше?) розробка ядра Linux.

• Паралельна технологія. Нова гілка розробки ведеться паралельно з основною і при необхідності може бути поєднана з нею у будь-який час. Таким чином, легко розділити розробку, але водночас легко об'єднувати паралельно ведені розробки, інтегруючи напрацювання незалежних розробників - "Форк" перестає бути тим злом, яким його намагаються уявити представники пропрієтарного програмного забезпечення.

• Простота ведення версій конфігураційних файлів. 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 одразу однією командою.

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

**ВИСНОВОК**

У роботі було розглянуто системи контролю версій, проведено короткий порівняльний аналіз систем.

Такі системи дозволяють полегшити проекти з розробки програмного забезпечення, і певною мірою виключити людські помилки. Існуючі системи моніторингу допоможуть у разі виникнення будь-яких нестиковок у роботі.

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

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

Системи контролю версій є приголомшливим для вирішення проблем поширених корпорацій. Безумовно дана область розвиватиметься і надалі можливе збільшення кількості систем та суттєве доопрацювання існуючих.

**СПИСОК ЛІТЕРАТУРИ**

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

2. Нове покоління систем контролю версій [Електронний ресурс] – Доступ до ст.: http://www.techinfo.net.ru/docs/Version\_Control\_Systems.html

3. Available CVS Alternatives [Електронний ресурс] – Доступ до ст.: http://better-scm.berlios.de/aegis/

4. Вікіпедія [Електронний ресурс] – Доступ до ст.: http://ua.wikipedia.org/wiki/CVS

5. Вікіпедія [Електронний ресурс] – Доступ до ст.: http://ua.wikipedia.org/wiki/Subversion

6. OpenCM Overview [Електронний ресурс] – Доступ до ст.: http://www.opencm.org/

7. Aegis 4.24 [Електронний ресурс] – Доступ до ст.: http://aegis.sourceforge.net/

8. BitKeeper [Електронний ресурс] – Доступ до ст.: http://www.bitkeeper.com/Products.Advantages.html

9. Вікіпедія [Електронний ресурс] – Доступ до ст.: http://ua.wikipedia.org/wiki/Perforce

10. Вікіпедія [Електронний ресурс] – Доступ до ст.: http://wiki.darcs.net/DarcsWiki/ВведениеВDarcs