МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ БІЗНЕС-КОЛЕДЖ

***Циклова комісія програмування***

**Реферат**

На тему «Професії в геймдеві»

Феофілова Ярослава Володимировича і Петрова Владислава Олеговича

*ПІБ студента*

студента групи **2П-19**

Викладач Марченко С. В.

Черкаси-2020

**Зміст**

1. **Розподілені системи управління версіями.**
2. **Види систем контролю версій.**
   1. **CVS**
   2. **Arch**
   3. **OpenCM**
   4. **Monotone**
   5. **BitKeeper**
   6. **Darcs**

**Вступ**

Сьогодні у 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 схильний до великої 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, необхідна особлива увага і велика ймовірність помилки. Крім того, нам доводитися зберігати величезну кількість практично ідентичних документів.

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

Метою даного реферату є розгляд різного роду систем контролю версій.

Відповідно до поставленої мети необхідно вирішити наступні завдання:

\* визначити поняття системи контролю версій;

\* проаналізувати існуючі системи контролю версій;

\* розглянути основні види даного роду систем із зарубіжної та російської практики.

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) являють собою програмне забезпечення для полегшення діяльності з швидко мінливою 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 розроблювальних програм. Система контролю версій дозволяє розробникам зберігати минулі версії файлів з розробки і діставати їх звідти. Вона зберігає інформацію про версію шкірного файлу (і повну структуру проекту) в колекції, зазвичай званої репозиторієм. Але тим не менш дані системи можуть використовуватися і в інших областях знань, які включають в собі величзну 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.

Всередині сховища можуть бути кілька паралельних ліній розробки, які зазвичай називаються гілками. Це може бути корисно 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 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 декількох сот рядків або якщо для проекту спільно працюють кілька розробників. Використання хорошої системи контролю версій безумовно 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 більшу частину функцій з управління версіями. Користувач, що працює з документами, повинен спочатку отримати потрібну йому версію документа зі сховища; зазвичай створюється локальна копія документа, т.зв. «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 теж залишається в сховищі і може бути звідти отримана в будь-який час. Сервер може використовувати Т. зв. дельта-компресію-такий спосіб зберігання документів, при якому зберігаються тільки зміни між послідовними версіями, що дозволяє зменшити обсяг збережених даних.

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 відзначити, 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 неможливо ні за допомогою системи, ні вручну. Яскравим прикладом є ситуація коли формат файлу дуже складний або невідомий. Окремі системи контролю версій надають можливість користувачеві 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. Багато систем управління версіями надають ряд інших можливостей:

\* Дозволяють створювати різні варіанти одного документа, т. зв. 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 про те, що і чому вони змінили в даній версії.

\* Контролюють права доступу користувачів, дозволяючи або забороняючи читання або зміна даних, в залежності від того, хто запитує цю дію.[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 централізованого сховища: 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 відрізнятися 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 тільки свою гілку, його робота не впливає на інших учасників проекту і навпаки. Однак при необхідності виконати злиття гілок (або синхронізацію локальних сховищ в розподіленій 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. Велика спільнота, навпаки, може використовувати централізований сервер, з яким синхронізуються копії всіх його учасників. Можливі і більш складні варіанти-наприклад, зі створенням груп для роботи за окремими напрямками всередині більшого проекту.

Розподілена система контролю версій дозволяє клонувати віддалений репозиторій, виробляючи точну копію. Вона також дозволяє распростанять зміни з одного репозиторію на інший. У нерозподілених VCS розробнику потрібен репозиторій для того, щоб зафіксувати зміни в ньому. Це робить розробника без доступу до сховища людиною другого

сорта. З розподіленою VCS кожен розробник може схиляти 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 фіксації, які означають, що стан всього сховища 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 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, принаймні, напівавтоматичної 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 лише об'єднання змін.

Це зріла і відносно 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 ПЗ зазвичай працює під управлінням Unix, тоді як CVS Клієнти доступні у всіх популярних операційних системах. (4)

Сервер зберігає в репозиторії поточну версію (версії) проекту та 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 у 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, в яку 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 відзначити, 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-нерозподілена система і атомарні фіксації змін не підтримуються. Так як вже є кращі альтернативи, які містять більш широкий набір функцій, ви, можливо, віддасте перевагу 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.

### Subversion прагне бути кращою альтернативою CVS(див. додаток 1). Вона підтримує більшість угод CVS, включаючи більшу 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;

### \* Можливість організації доступу до сховища Subversion через Apache по протоколу WebDAV / DeltaV;

### \* Можливість установки автономного сервера Subversion з доступом за власним протоколом;

### \* "Дешеві" операції створення гілок і міток (потрібна невелика фіксована 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 розміру даних, які порушені змінами;

### \* Два можливих внутрішніх формату репозиторію: база даних або простий файл;

### \* Версіоновані символьні посилання (тільки в робочих копіях під UNIX-системами);

### \* Однаково ефективна робота і з текстовими, і з двійковими файлами;

### \* Висновок клієнта командного рядка однаково зручний і для читання, і для розбору програмами;

### \* Часткова локалізація повідомлень (використовуються Налаштування локалі);

### \* Бібліотеки для мов PHP, Python, Perl, Java;

### \* Можливість віддзеркалення репозиторію. [5]

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

### Сервіс Subversion на основі HTTP (або HTTPS) важкий для розгортання, в порівнянні з іншими системами, так як він вимагає встановленої служби Apache2 зі своїм власним спеціальним модулем. Також є "svnserve" сервер, який менш здатний, але більш простий в установці і використовує спеціальний протокол. Крім того, підтримка Subversion об'єднання змін обмежена і схожа в цьому на CVS (тобто, об'єднання гілок, де файли були переміщені не буде виконано коректно). Вона також відносно вимоглива до ресурсів, особливо на великих операціях. [2]

### Subversion пропонує два варіанти організації репозиторіїв. Репозиторії першого типу використовують для зберігання базу даних на основі Berkeley DB, репозиторії другого типу - в звичайних файлах спеціального формату (доступ до даних організовується 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.(5)

### Subversion добре документована у своїй безкоштовній онлайн книзі, «Контроль версій з Subversion". Невелика онлайн система допомоги, що поставляється з Subversion клієнтом може також бути корисною в якості довідника. Subversion має багато доповнень, але вони все ще менш Зрілі, ніж їх CVS конкуренти

### 2.2. 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 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 репозиторію і 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 пропущені 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 та керівництва.

### 2.3. 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 криптографічної аутентифікації.

### Розглянемо список головних особливостей OpenCM:

### \* Дана система призначена для реальної конфігурації. Це те дивно, що не знає CVS;

### \* Здатність перейменовувати файли, не втрачаючи їх історії;

### \* Управління доступом до історії по гілках;

### \* Шифрувальне встановлення автентичності. Це забезпечує здатність зробити звіти розробників на репозиторії OpenCM, не даючи їм звіт на основній машині (OS), і робить співпрацю мульти-організацій можливим.

### \* Безперервні засоби управління за цілісністю. Якщо у сервера є 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. 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-якою UNIX-системою і на Windows під емуляцією Cygwin . Вона має CVS-подібні команди і добре документована.

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 для виконання якої операції. Незважаючи на той факт, що Aegis не Мережева, вона все ж розподілена в тому сенсі, що репозиторії можуть бути клоновані і зміни можуть бути поширені з одного репозиторію 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 важко переглянути документацію, щоб знайти саме те, що вам потрібно. Однак документація високої якості.

**2.4. Monotone**

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

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 та ревізії контрольними сумами SHA 1.

Monotone підтримує перейменування та копії файлів і 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 був виконаний.

Monotone все ще в розробці і може все ще мати деякі проблеми в поведінці. Розробники Monotone очікують, що проблеми вирішаться, так як робота триває.

### 2.5. 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 можливостями, підтримуючи розподілені репозиторії; працюючи по HTTP ; підтримуючи перейменування і копіювання файлів і директорій; управління патчами; поширюючи зміни від гілки в головну лінію і багато інших можливостей.

### BitKeeper поширюється 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 даного) і вимога про оновлення системи при появі більш свіжої версії, навіть якщо вона має іншу ліцензію. Крім того, вихідний код не доступний публічно і бінарники існують тільки для найпоширеніших систем, включаючи Win 32.

### Переваги BitKeeper:

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

### \* Знижена ймовірність людської помилки: BitKeeper здійснює перевірки цілісності репозиторію, які негайно вловлюють проблеми.

### \* Відтворюваність: комплекси проектів програмного забезпечення з безліччю розробників вимагають інструментів управління конфігурації програмного забезпечення, які враховують точну відтворюваність минулої і справжньої інформації. Оскільки BitKeeper підтримує поняття логічної одиниці роботи, де кожна одиниця є незмінною-не може змінитися, але може бути додано - BitKeeper виробляє повністю відновлюваний репозиторій в будь-який момент часу. BitKeeper управляє процесом розвитку так, щоб кожна фаза проекту може бути оновлена в майбутньому.

### \* Відповідальність: оскільки репозиторії повністю відновлюються в будь-який момент часу, легко дізнатися, хто зробив, які зміни, і що інші файли були змінені в той же самий час. Налагодження стає набагато більш ефективним 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, щоб мати зв'язок TCP між усіма системами весь час.

### \* Масштабованість: Архітектура Bitkeeper's невід'ємно масштабована, так, що однаково добре працює і для п'яти розробників, і для 1 000 або 10 000. (8)

### Жменька проектів використовують BitKeeper, включаючи деяких з розробників Linux ядра і основна команда розробників MySQL . Через її ліцензії BitKeeperнезручний для розробки ПЗ з відкритим кодом, так як воно змуситьвідвернутисябагатьох «ідеалістичних» розробників і створить різні проблемидля 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 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 та 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 сповіщення, створення та 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)

### 2.6. 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 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 відносяться до багфікса, підтверджуєте зміни і продовжуєте роботу над своєю новою функцією - ваш репозиторій буде гілкою основної розробки.

### \* Режим Лінуса. Наприклад, ви хочете додати нову функцію або виправити помилку в open-source проекті. Ви робите локальну копію репозиторію 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 на запис в репозиторій проекту. Таким чином підтримується сам проект Darcs, подібним же шляхом Лінусом Торвальдсом 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-якому числі 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 з розробки програмного забезпечення, і певною мірою виключити людські помилки. Існуючі системи моніторингу допоможуть у 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 рядом особливостей, які виділяють її.

### На зміну відносно старим системам (CVS) приходять поліпшені альтернативи. Інші системи більш приємні 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 вирішення проблем поширених корпорацій. Безумовно дана область буде розвиватися і надалі можливе збільшення числа систем і істотне доопрацювання існуючих.

### Список літератури

1. Википедия [Электронный ресурс] – Доступ к ст.: /wiki/
2. Новое поколение систем контроля версий [Электронный ресурс] – Доступ к ст.: /docs/Version\_Control\_Systems.html
3. Available CVS Alternatives [Электронный ресурс] – Доступ к ст.: http://better-scm.berlios.de/aegis/
4. Википедия [Электронный ресурс] – Доступ к ст.: /wiki/CVS
5. Википедия [Электронный ресурс] – Доступ к ст.: /wiki/Subversion
6. OpenCM Overview [Электронный ресурс] – Доступ к ст.: /
7. Aegis 4.24 [Электронный ресурс] – Доступ к ст.: /
8. BitKeeper [Электронный ресурс] – Доступ к ст.: /Products.Advantages.html
9. Википедия [Электронный ресурс] – Доступ к ст.: /wiki/Perforce
10. Википедия [Электронный ресурс] – Доступ к ст.: /DarcsWiki/ВведениеВDarcs