

СИСТЕМИ КОНТРОЛЮ ВЕРСІЙ ПРИ РОЗРОБЦІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Сьогодні процес створення програмного забезпечення стає все складнішим. Тому виникає проблема з обміну версій файлів проекту між розробниками. В процесі розробки можуть з'являтися нові ідеї, як слід нові версії програми. Для рішення цих проблем використовують системи контролю версій (СКВ), що забезпечують систематизацію версій, які об'єднуються при редагуванні і спільній роботі. Система управління версіями дозволяє зберігати кілька версій одного і того ж документа (початкового коду програми, скрипту, веб-сторінки, вебсайту, 3D-моделі, текстового документу тощо), при необхідності повертатися до попередніх версій, визначати, хто і коли зробив ту чи іншу зміну, і багато іншого. Цей інструмент дозволяє одночасно, не заважаючи один одному, проводити роботу над груповими проектами.

Сьогодні існує чи мало таких систем з відкритим вихідним кодом, наприклад клієнт- серверні системи (Concurrent Versions System (CVS), OpenCVS, Subversion (SVN)), розподілені системи (Git, Mercurial).

Клієнт- серверна система контролю версій



Рисунок 1- Клієнт- серверна СКВ

Такі системи мають єдиний сервер, який містить всі версії файлів, і деяку кількість клієнтів, які отримують файли з цього централізованого сховища і там же їх зберігають. Але вихід сервера з ладу обернеться втратою всіх даних. Крім того, в таких системах може бути ускладнена одночасна робота декількох розробників над одним файлом.

До систем типу Клієнт- сервер відноситься Subversion («SVN»). Це вільна централізована система управління версіями. SVN має загальний сервер, де зберігаються всі зміни, і розробники оновлюють зміни там. Тому SVN вимагає постійного підключення до інтернету для збереження змін.

Однією з широко розповсюджених є система Git, яка є однією з найефективніших, надійних і високопродуктивних систем керування версіями. Крім того, це повністю безкоштовне програмне забезпечення. Git підтримує багато ОС, таких як Mac, Linux, Windows і Solaris. Завантажити відповідні дистрибутиви можна з офіційного сайту Git.



Рисунок 2- Розподілена СКВ

Система контролю версій Git надає гнучкі засоби розробки, які базуються на відгалуженні та злитті гілок. Git дозволяє зберігати попередні версії файлів та завантажувати їх за потребою, зберігати повну інформацію про версію кожного з файлів, а також повну структуру проекту на всіх стадіях розробки. Це забезпечує декільком розробникам можливість працювати над своїм файлом, не заважаючи іншим і не псуючи основну гілку. Всі файли зберігаються у репозиторії, в середині якого можуть бути

створені паралельні лінії розробки - гілки. Гілка - це напрям розробки, який існує незалежно від іншого напрямку, але має з ним спільну історію. Гілка починається як копія чогось і, рухаючись з цієї точки, створює свою власну історію. Git зберігає дані набором знімків. Кожного разу при фіксації поточної версії проекту Git зберігає знімок того, як виглядають всі файли проекту. Для кожного файлу, який відстежується, Git зберігає розмір, час створення і останньої зміни. Більшість дій можна виконувати на локальній файлової системі без використання інтернет підключення. Вся історія змін зберігається спочатку локально і при необхідності вивантажується у окремий репозиторій.

В своїй базі Git зберігає все по хешам файлів. Як хеш-функція використовується SHA-1. Перед кожним збереженням файлів обчислюється значення хеш-функції SHA-1 для файлу і отриманий хеш стає індексом файлу. Використовуючи хеш Git легко відслідковує зміни в файлах. Існує кілька підходів до використання системи Git, серед яких, найпоширенішою є так звана концепція gitflow — центральний репозитарій працює з двома паралельними між собою гілками: master і develop. Гілка Master містить

версію коду, яка найімовірніше буде запроваджена у виробництво. Паралельно до основної гілки (master) існує ще робоча гілка Develop, яка використовується для розробки нової версії програмного забезпечення та тестування його стабільності.

Система Git дозволяє використовувати віддалений репозиторій. Для цього потрібно створити Git-сервер або скористатися хостингом репозиторіїв. Одним з найпопулярніших є github.com. Віддалені репозиторії дозволяють спростити контроль версій над спільним проектом. Кожен розробник може вести свою гілку і при необхідності додавати свої зміни в основну гілку.

Крім цього багато компаній використовують git як інструмент клонування проекту. Тобто, розробники додають свої зміни на проміжний репозиторій, після чого тестують код і після знаходження багів і їх виправлення перевірений код клонується на робочий сервер.

До переваг Git слід віднести:

1. можливість обміну оновлень файлів між учасниками розробки;
2. не потребує постійного інтернет з'єднання, у кожного своя копія сховища;
3. всі зміни у файлах відбуваються локально (всі програми можуть переглядатись іншими учасниками процесу) і тільки при необхідності завантажуються в загальний репозиторій;
4. висока швидкість роботи (Git не зберігає копії файлів, а зберігає історію змін.)

До недоліків системи Git можна віднести складність вивчення системи.

На основі проведеного аналізу слід зробити висновки, що використання систем контролю версій значно прискорює процес розробки програмного забезпечення, при цьому слід вибирати розподілену систему Git.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Гриценко В. Г. Використання системи управління версіями GIT для організації командної роботи над IT проектом / В. Г. Гриценко, О. М. Подолян // Інформаційні технології і засоби навчання. - 2014. - Т. 39, вип. 1. - С. 250-263.
2. Переклад українською книги «Pro Git» Скотта Чакона (Scott Chacon) та Бена Страуба (Ben Straub) [Електроний ресурс]. – Режим доступу: <http://git-scm.com/book/uk/v2>
3. Apache Subversion Community Guide (aka "HACKING") [Електроний ресурс]. – Режим доступу: <http://subversion.apache.org/docs/community-guide/>
4. Bryan O'Sullivan. Mercurial: The Definitive Guide. — O'Reilly Media, Inc., 2009. — 288 с. — ISBN 9780596800673.
5. Ресурс Git [Електроний ресурс]. – Режим доступу: https://ru.hexlet.io/courses/intro_to_git