

УДК 004.384

Луценко Н.В.¹, Кошарна К.О.², Роєнко О.І.²

¹ старш. викл. ЗНТУ

² студ. гр. КНТ-525 ЗНТУ

ВИКОРИСТАННЯ СИСТЕМИ КОНТРОЛЮ ВЕРСІЙ GIT ПРИ РОЗРОБЦІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Процес розробки програмного продукту складний і дуже важливий, один з основних напрямків у світі. Залежно від проекту, кількість рядків коду може варіюватися від декількох сотень до кількох тисяч, а іноді і мільйонів. При розробці програмного забезпечення (ПЗ) використовують різні системи контролю версій, які значно поліпшують та прискорюють процес розробки.

Система контролю версій (СКВ) – це набір консольних утиліт, які відстежують і фіксують зміни в файлах проекту, що розробляється, (найчастіше мова йде саме про вихідний код програм, але можна використовувати його для будь-яких файлів). З допомогою СКВ можна повернутися на попередню версію проекту, бачити зміни, порівнювати, аналізувати і багато іншого.

На сьогоднішній день при розробці ПЗ використовуються централізовані СКВ (CVS, Subversion, Perforce), децентралізовані СКВ (Git, Mercurial, Bazaar, Darcs, BitKeeper) [1]. Найбільш популярною серед розробників ПЗ є система контролю версій Git [2].

Система Git є розподіленою, тобто не залежить від одного центрального сервера, на якому зберігаються файли. Замість цього система працює повністю локально, зберігаючи дані в папках на жорсткому диску, які називаються репозиторієм. Потім даними, що збережені в локальному репозиторії, можна обмінюватися з іншими користувачами. Часто при роботі

з Git створюють центральний репозиторій, з яким інші розробники синхронізуються.

Одна з ключових особливостей Git – гнучкість. Ця система підтримує різні види нелінійних технологічних процесів розробки, ефективна як в невеликих, так і в великих проєктах та сумісна з багатьма існуючими системами і протоколами. У Git розгалуження і операції над гілками (наприклад, злиття або повернення) зберігаються як частина історії змін. Не всі системи контролю версій використовують цей рівень відстеження. Це дозволяє в процесі роботи бачити, який учасник проєкту вносив конкретні зміни, мати можливість швидко повернути зміни файлу в початковий стан, мати зручні інструменти для аналізу внесених змін в кожен файл, кожен рядок вихідного коду.

В порівнянні з іншими системами контролю версій система Git має свої переваги та недоліки [3].

Переваги при використанні системи Git:

- незалежна робота над проєктом кожного учасника;
- просте вирішення конфліктів – при виконанні операції "merge" можна порівняти зміни в двох файлах і прийняти рішення по об'єднанню внесених змін;
- швидко виправлення помилок, виявлених під час тестування;
- можливість швидкого оновлення release-версії проєкту – внесення змін не вимагає зупинки робочого рішення, що сприяє безперебійній роботі;
- простий консольний інтерфейс для вибору вхідних в commit змін;
- двостороння синхронізація з CVS і Subversion (але commit в CVS експортуються наполовину вручну);
- Git може бути в якості CVS-сервер.

Недоліки системи Git:

- не відстежується зміна окремих файлів, а тільки всього проєкту в цілому, що може бути незручно при роботі з великими проєктами, які містять безліч незв'язаних файлів. Але є рішення – треба розбити проєкт на частини, кожна з яких складається з взаємозв'язаних файлів;
- при початковому (першому) створенні сховища та синхронізації його з іншими розробниками, потрібно досить тривалий час для скачування даних, особливо, якщо проєкт великий, так як необхідно скопіювати на локальний комп'ютер весь репозиторій;
- високий поріг входження (навчання) для тих, хто раніше використовував інші СКВ;
- обмежена підтримка Windows (в порівнянні з Linux).

На підставі вище сказаного, можна зробити такі висновки:

- система контролю версій Git відповідає всім вимогам сучасного процесу розробки програмного продукту;

- використання даної системи полегшує командну роботу розробників, прискорюючи пошук помилок та їх виправлення;
- в порівнянні з іншими системами – Git більш гнучка, сучасна та більш підходить для розробки великих програмних проектів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Обзор систем контроля версий [Електронний ресурс]. – Режим доступу: http://all-ht.ru/inf/prog/p_0_1.html
2. Pro Git [Електронний ресурс]. – Режим доступу: <https://git-scm.com/book/ru/v2/>
3. Девять причин перейти на Git [Електронний ресурс]. – Режим доступу: <http://tarantsov.com/2007/12/nine-reasons-to-use-git.html>