Висновки

Проведено аналіз шаблонів розробки програмного забезпечення, розглянуто їх класифікацію за призначенням . В роботі визначено переваги та недоліки використання шаблонів в процесі проектування. Приведено опис основних шаблонів розробки програмного забезпечення, а саме: Singleton, Factory Method, Facade, Strategy.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Інсталяція програмного забезпечення) – URL: http://surl.li/uqewzu (дата звернення 30.10.2024)
2. Одинак (шаблон проєктування) – URL: http://surl.li/tctslf (дата звернення 30.10.2024)
3. Factory method pattern – URL: http://surl.li/yqhfxq (дата звернення 30.10.2024)
4. Facade – https://en.wikipedia.org/wiki/Facade_pattern (дата звернення 30.10.2024)
5. Strategy pattern – URL: http://surl.li/zksgnn (дата звернення 30.10.2024)

УДК 004.42

Рейда М. О.,
Черній А. О.,
Рейда О. М.,
Вінницький національний технічний університет

СИСТЕМИ КОНТРОЛЮ ВЕРСІЙ ПРОГРАМНОГО КОДУ

Анотація. Проведено аналіз систем контролю версій програмного коду.

Ключові слова: Mercurial, Git, Subversion, гілка, розподілений контроль вихідного коду.

Abstract. The analysis of version control systems for source code was conducted.

Keywords: Mercurial, Git, Subversion, branch, distributed source control.

Вступ. Питання групової розробки програмних засобів і систем набуло значної актуальності, що вимагає наявність ефективних та простих у використанні систем контролю версій продуктів. Розробка групових проектів потребує наявності однакових версії коду у всіх розробників, а також можливості зміни модулів незалежно від дій інших, що в свою чергу надає можливість одночасної зміни коду з урахуванням модифікацій інших користувачів. В такому контексті дослідження наявних способів контролю версій відіграє важливу роль у груповій розробці програмних засобів і систем.

Результати дослідження

Mercurial - це безкоштовний інструмент для розподіленого управління контролем вихідного коду (рис. 1.1). За допомогою інтуїтивно зрозумілого інтерфейсу він дозволяє ефективно керувати проєктами будь-якого розміру. Простий у використанні і стійкий до неправильних дій користувачів.

Рисунок 1.1 – Логотип системи Mercurial

Основні відмінності:

- Розподілена архітектура. Традиційні системи контролю версій, такі як Subversion (система керування версіями з відкритим вихідним кодом), є типовими клієнт-серверними архітектурами з центральним сервером для зберігання версій проекту. На відміну від них, Mercurial є розподіленою, надаючи кожному розробнику локальну копію всієї історії розробки. Таким чином, він працює незалежно від доступу до мережі або центрального сервера.

- Швидкість. Реалізація та структури даних Mercurial розроблені для швидкої роботи. Ви можете генерувати відмінності між версіями або повертатися до попередніх. Тому Mercurial ідеально підходить для великих проєктів, таких як nginx або NetBeans. Більша частина Mercurial написана на Python, з невеликою частиною Portable C з міркувань продуктивності. Як результат, бінарні релізи доступні на всіх основних платформах.
- Розширюваність. Функціональність Mercurial можна розширити за допомогою розширень, активувавши офіційні, які публікуються Mercurial, чи написавши власні. Розширення написані на Python можуть змінювати роботу основних команд, додавати нові команди та отримувати доступ до всіх основних функцій Mercurial.
- Простота у використанні. Базовий інтерфейс простий у використанні, навчання основним командам займає декілька хвилин.

Git - це безкоштовна система розподіленого контролю версій з відкритим вихідним кодом, призначена для швидкої та ефективної роботи з будь-якими проектами, від невеликих до дуже великих (рис. 1.2). Git простий у вивченні, займає мінімум місця та має блискавичну продуктивність. Він перевершує такі SCM-інструменти, як Subversion, CVS, Perforce та ClearCase, завдяки таким функціям, як локальне розгалуження з низькими витратами.

Рисунок 1.2 – Логотип системи Git

Основні відмінності:

- Розгалуження та злиття. Особливістю Git, яка відрізняє його від майже всіх інших SCM, є його модель розгалуження. Git дозволяє і заохочує створення декількох локальних гілок, які можуть бути повністю незалежними одна від одної. Створення, об'єднання та видалення цих гілок займає лічені секунди. Створіть гілку, щоб випробувати ідею, перевірте працездатність, застосуйте оновлення та об'єднайте їх.
- Рольові кодові лінії. Майте гілку, яка завжди містить тільки те, що йде у виробництво, іншу, в яку ви об'єднуєте роботу для тестування, і кілька менших гілок для повсякденної роботи.
- Робочий процес на основі функцій. Створюйте нові гілки для кожної нової функції, над якою ви працюєте, щоб ви могли легко переключатися між ними, а потім видаляйте кожну гілку, коли ця функція буде об'єднана з основною лінією.
- Швидкодія. Git працює швидко. У Git майже всі операції виконуються локально, що дає йому величезну перевагу у швидкості роботи над централізованими системами, яким постійно доводиться десь взаємодіяти з сервером.

Таблиця 1.1 – Порівняння Git та Mercurial

	Назва	Git	Mercurial
1.	Архітектура	Базується на концепції об'єктів	Використовує концепцію "репозиторіїв"
2.	Модель гілок	Багатогранна модель, що заохочує багато гілок	Більш формалізовані гілки для етапів розробки
3.	Інтерфейс командного рядка	Складний, але потужний	Простий і інтуїтивно зрозумілий
4.	Продуктивність	Виконує більшість операцій локально	Швидкий, але може бути менш оптимізованим
5.	Розширюваність	Інтеграція розширень більш складна у порівнянні з Mercurial	Легший процес створення та інтеграції розширень
6.	Спільнота та підтримка	Величезна спільнота, популярний у великих проектах	Менша популярність у порівнянні з Git

Висновки

Проведено аналіз систем контролю версій програмного коду таких як: Git, Mercurial: приведено короткий опис розглянутих систем, визначено переваги та недоліки для різних типів користувачів, приведено основні відмінності.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Mercurial source control management – URL: https://www.mercurial-scm.org/about (дата звернення 28.10.2024)
2. About - Branching and Merging – URL: https://git-scm.com/ (дата звернення 28.10.2024)

УДК **004.42**

РЕЙДА М. О.,
ЧЕРНІЙ А. О.,
РЕЙДА О. М.,
Вінницький національний технічний університет

СИСТЕМИ РОЗРОБКИ ІНСТАЛЯЦІЙНИХ ПАКЕТІВ ПРОГРАМ

Анотація. Проведено аналіз систем розробки встановлювальних пакетів програм.

Ключові слова: UNIX, Mac OS, Windows, Inno Setup, dpkg, RPM.

Abstract. Software installation package development systems have been analyzed.

Keywords: UNIX, Mac OS, Windows, Inno Setup, dpkg, RPM.

Вступ. Інсталяційні пакети програм стали невід'ємною частиною програмного забезпечення. Актуальність розробки таких засобів полягає у необхідності швидкого та безпечного встановлення програм на комп'ютер. Сучасні системи надають можливість автоматизації процесів встановлення та підтримку різних платформ. Тому дослідження різних підходів до інсталяції є важливим для процесу оптимізації інсталяційних пакетів програм.

Результати дослідження

Проведено аналіз особливостей використання інсталяційних пакетів програм і таких сімействах операційних систем:

- UNIX. Більшість дистрибутивів операційних систем на базі GNU, Linux і BSD має вбудовані системи керування пакетами, за допомогою яких можна встановлювати як необхідні компоненти операційної системи, так і стороннє програмне забезпечення, найчастіше навіть якщо воно використовує власний установник, яким це не передбачено.
- Mac OS X також використовує систему керування пакетами. Деякі комерційні додатки для Mac OS X використовують окремий установник, наприклад, Installer VISE або Stuffit Installer Maker. Додатка, які не мають потреби в установці додаткових компонентів системи, можуть бути встановлені за допомогою простого копіювання файлів додатка в потрібне місце на жорсткому диску. Mac OS X також включає окремий додаток для відновлення програм Software Update (також відоме як команда оболонки softwareupdate), але воно підтримує тільки програмне забезпечення продуктів Apple.
- Windows, Windows Installer (установник Windows) – підсистема Microsoft Windows, що забезпечує установку програм (інсталятор). Є компонентом Windows, починаючи з Windows 2000; може доустановлюватися й на більш ранні версії Windows. Вся необхідна для установки інформація (іноді й разом із установлюваними файлами) утримується в настановних пакетах (installation packages), що мають розширення .msi.

Деякі архіватори (наприклад WinRAR, 7-Zip і інші) також дозволяють створювати установники. Деякі операційні середовища – наприклад, Windows NT (за замовчуванням) і Xfce 4 (за бажанням користувача), містять функцію автоматичного запуску певної програми, що перебуває на носії, при його вставці в пристрій читання.

Як компроміс між інсталятором Windows і системами керування вмістом UNIX-подібних систем, існують системи керування установкою Windows.

Логічна структура пакета інсталятора Windows: