УДК 519.688

О.С. Волковський, І.О. Яровий

ВЗАЄМОДІЯ З ВІДДАЛЕНИМИ РЕПОЗИТОРІЯМИ В СИСТЕМІ МОНІТОРИНГА

Анотація. У роботі запропоновані алгоритми для моніторингу віддалених репозиторіїв. Для вирішення задачі своєчасного інформування членів команди розробників, які працюють над одним проектом, про зміни у віддаленому репозиторії, було створено три алгоритми. Дані алгоритми є універсальними і можуть працювати з будь-якою системою контроля версій. Розроблені та програмно реалізовані в десктопному додатку на мові C++ наступні алгоритми: підключення до віддалених репозиторіїв; відновлення підключення до віддаленого репозиторію при запуску програми; перевірки наявності нової інформації у віддаленому репозитірії з заданим інтервалом.

Ключові слова: система контролю версій, віддалений репозиторій, моніторинг.

Термінологія. Перш ніж почати описувати постановку проблеми, визначимо ключові терміни, які будуть використовуватись протягом усієї статті.

Віддалений репозиторій (ВР) - це модифікація проекту, яка зберігається в інтернеті або ще десь в мережі. ВР може існувати в декількох варіантах, кожний із них, як правило, доступний для вас або тільки для перегляду, або для перегляду і запису [1].

Система контролю версій (СКВ) - програмне забезпечення для полегшення роботи з інформацією, що змінюється. Дозволяє зберігати декілька версій одного й того ж документа, при необхідності повертатись до більш ранньої версії [2].

Постановка проблеми. Весь програмний та вихідний код кожного модуля, бібліотеки або ж проекту взагалі, як правило, знаходиться у ВР, а програмісти працюють з локальною копією цього коду. Коли ж декілька програмістів вносять зміни в однаковий участок коду (наприклад з 300-ого по 400-ий рядок) й один з програмістів відправляє свої зміни у ВР, то інші вже можуть зіткнутись з проблемою, яка називається “merge conflict”. Така ситуація виникає то-

 ISSN 1562-9945

му, що СКВ не знає, який код слід зберегти і додати до ВР: той що “прийшов” від програміста чи той, що вже є у ВР. Щоб вирішити дану проблему, програмісту, який зіткнувся з проблемою “merge conflict”, потрібно “стягнути” (завантажити) код з ВР на свій комп’ютер, зробити локальне злиття даних, що були завантажені з ВР з локальними даними, що знаходяться на локальній машині програміста, вирішити всі конфлікти, впевнитись, що отриманий код дієздатний, а всі тести мають статус “passed” (тобто очікуваний результат збігається з отриманим) і після цього робити повторну спробу відправлення свого коду до ВР. Все це ніби-то дрібниці, але весь цей процес забирає час, який є дуже цінним в командах розробників, особливо перед релізом.

Також необхідність у своєчасному інформуванні виникає тоді, коли вже досвідченому розробнику необхідно стажувати одного або декількох інтернів. Проблема набуває особливої гостроти, коли в компанії не купується ліцензія у сервісів, які надають можливість комфортно проводити код-рев’ю (перегляд коду досвідченими розробниками, які вже дають згоду на злиття коду, що оглядають, з основним кодом). В такому випадку досвідчений розробник повинен ретельно стежити за тим, що і коли додав його інтерн до ВР. Своєчасне інформування може запобігти внесенню помилок в проект або ж взагалі поломці всього проекту.

У роботі пропонується підхід, який може запобігти виникненню вищеописаної проблеми.

Нині існує багато СКВ, які пропонують свої підходи для роботи з інформацією, що змінюється. Але кожна СКВ має ВР і спрямована на децентралізований доступ користувачів до інформації.

Отже, необхідно розробити десктопну програму для операційної системи Linux та Windows, яка зможе працювати з різними СКВ і сповіщати користувача при появі нової інформації у ВР. Десктопна версія програми надає більшу продуктивність, у порівнянні з веб-додатками та сервісами.

Аналіз останніх досліджень. Не зважаючи на високу популярність СКВ, на сьогоднішні день не існує програми (як і в англомовній, так і в україномовній спільноті), яка може працювати з різними СКВ і показувати користувачу повідомлення про зміни у ВР. Існує лише можливість налаштувати поштову розсилку на сервісах ВР у відповідь на зміни.

Формування цілей статті. Виходячи з вищезгаданої інформації були сформульовані наступні завдання:

– розробити алгоритм для підключення до ВР;

– розробити алгоритм для відновлення підключення до ВР при запуску програми;

– розробити алгоритм перевірки наявності нової інформації у ВР із заданим інтервалом.

Основна частина. Для того, щоб розробити перші два алгоритми, нам необхідно мати адресу ВР, логін та пароль користувача. Ми повинні перевіряти ці дані на коректність. Тільки коректні дані зможуть забезпечити роботу алгоритмів.

Отже, для того, щоб наша програма працювала і виконувала поставлені задачі необхідно, щоб була можливість додавати нові ВР за якими користувач хоче слідкувати, виставляти таймер перевірки нових даних у ВР та можливість підключення до ВР (наприклад, при запуску нової програми), які вже користувач додав у програму.

Під таймером будемо розуміти інтервал часу після закінчення якого буде відбуватись перевірка: запит до ВР і перевірка нових даних у ньому.

В залежності від розміру проекту і кількості людей, які працюють над ним, рекомендовано застосовувати різний інтервал. Наприклад, якщо проект дуже великий (сотні тисяч строк коду), а розробників 10-15 чоловік, то ймовірність того, що основна гілка коду буде оновлюватись досить часто невелика. В такому разі можна застосовувати інтервал часу 3-4 години. Якщо ж проект новий й інтенсивно розвивається, то краще застосовувати інтервал від 30 хвилин до 1 години. У випадку, коли потрібно стажувати інтерна - від 5 до 15 хвилин.

Інформація, необхідна для роботи програми буде зберігатись в прихованій папці “.configs”. В цій папці для кожного доданого репозиторія буде створена папка з іменем репозиторія. В папці репозиторія зберігається логін і пароль користувача в зашифрованому вигляді алгоритмом шифрування AES-256, а також адреса віддаленого репозиторію. Крім цього, в даній директорії зберігається також ще прихована папка СКВ, яка зберігає всю необхідну інформацію для того, щоб підключитись через API СКВ до ВР.

Головна ідея полягає в тому, що програма повинна працювати у фоні і не вимагати від користувача нічого після того, як він додав новий ВР. Після проходження певного інтервалу часу, відбувається перевірка: чи є нові дані у ВР, яких ще немає у локальному?

Розглянемо базові алгоритми. *Алгоритм підключення до ВР* - підключення до ВР та збереження цього репозиторію в програмі (Рис. 1).

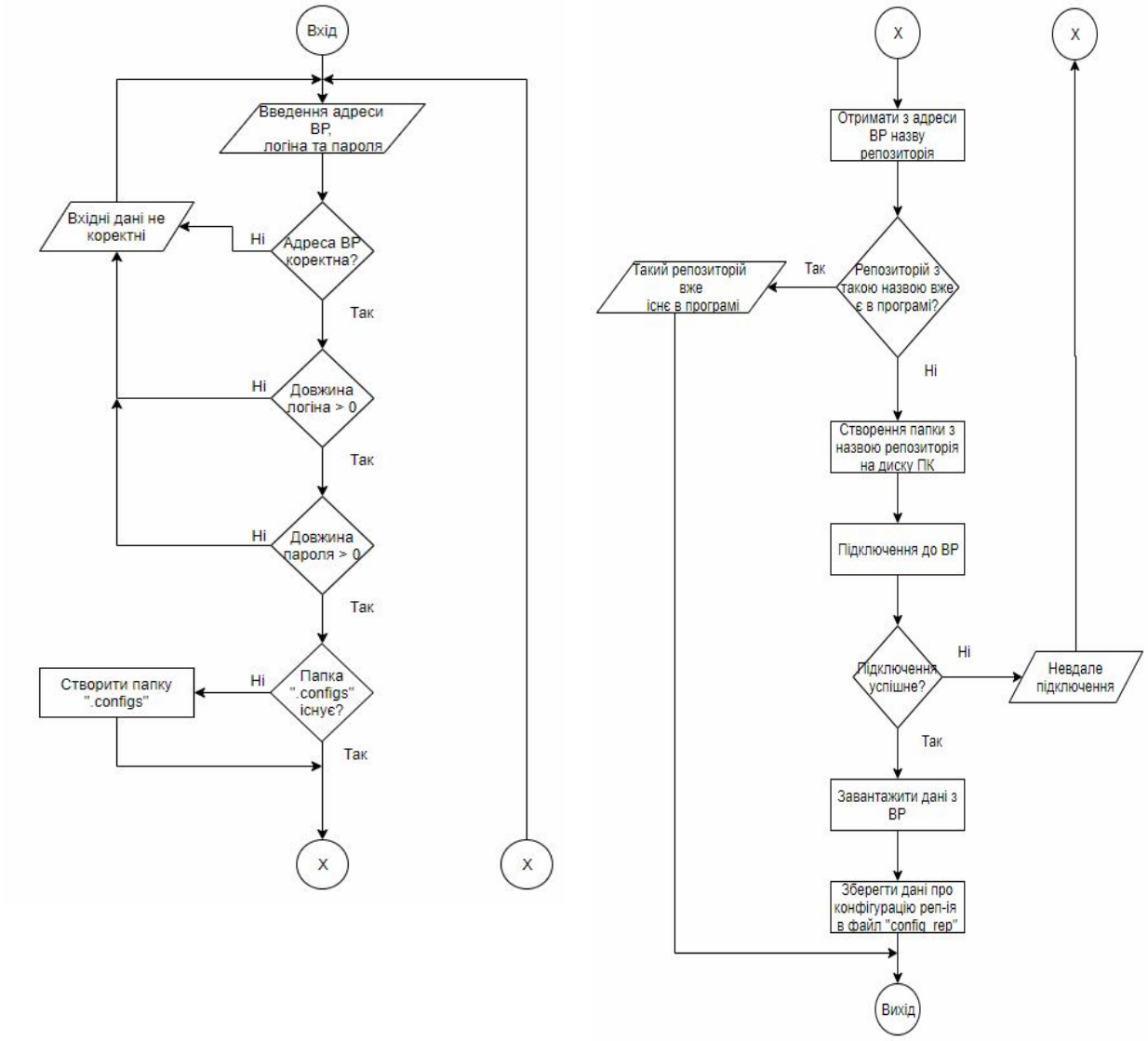

Рисунок 1 - Алгоритм підключення до ВР

Даний алгоритм отримує від користувача адресу ВР, його логін та пароль, щоб підключитися до нього. Після введення даних відбувається перевірка адреси ВР на її коректність, тобто відбувається парсинг http адреси, далі, перевіряється чи поля логіну і пароля взагалі були заповнені чи ні. Якщо ні, то виводиться повідомлення користувачу про те, що він ввів некоректні дані. Наступним етапом є перевірка наявності папки “.configs”, якщо її нема, то створюємо її (це означає, що репозиторій, який ми намагаємось додати є першим в нашій програмі). Далі, отримуємо з адреси ВР його назву і перевіряємо чи є вже такий репозиторій в програмі, якщо є, то показуємо відповідне повідомлення і завершуємо роботу алгоритму, інакше – створюємо папку з іменем репозиторія й підключаємось до ВР. В разі невдалого підключення - виводимо повідомлення

користувачу про невдале підключення і повертаємось до початку алгоритму, очікуючи даних від користувача. Якщо ж підключення успішне, то завантажуємо поточні дані з ВР і зберігаємо логін, пароль та тип репозиторію (оскільки програма повинна працювати з різними типами СКВ, то ми повинні ідентифіковувати до якої СКВ належать дані користувача) на диску ПК користувача у файл під назвою “config_rep” і завершуємо роботу алгоритму. “config_rep” - бінарний файл, який зберігає у зашифрованому вигляді інформацію.

Алгоритм відновлення підключення до ВР - виконує відновлювальну функцію при повторному запуску програми (Рис. 2). Даний алгоритм відповідає за дуже важливу функцію відновлення стану програми після її запуску (за умовою, що попередньо вона була вимкнена користувачем або ПК користувача був вимкнений). Застосування цього алгоритму запобігає повторному введенню даних для того, щоб продовжити спостереження за бажаними репозиторіями користувача. Опорною точкою цього алгоритму є папка “.con-figs”, якщо її не існує, то він завершує свою роботу.

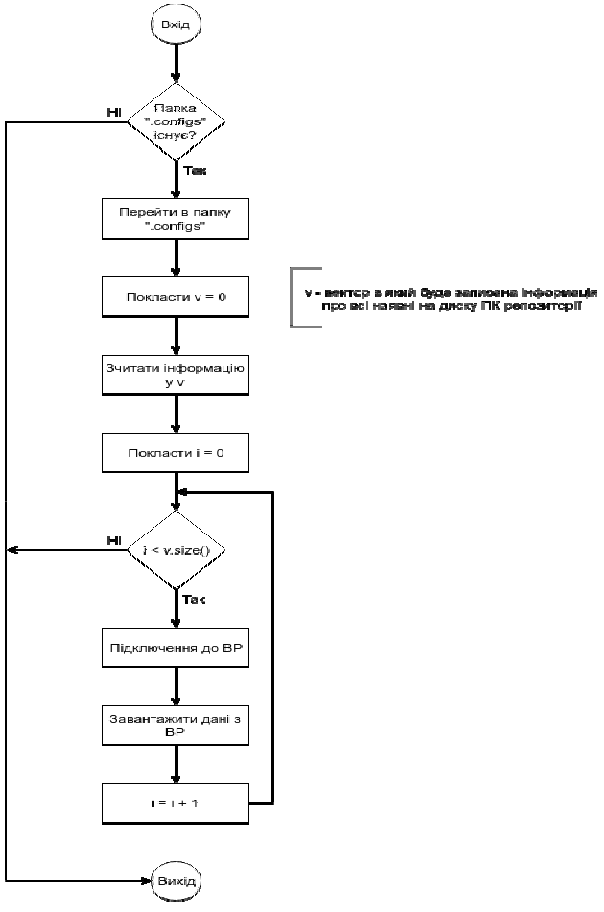

Рисунок 2 - Алгоритм відновлення підключення до ВР

Якщо ж ця папка існує і вона не пуста, відбувається зчитування інформації з усіх наявних репозиторіїв у вектор структур, яка має наступні поля: логін, пароль, тип репозиторія (наприклад Git або Mercurial) та час інтервалу перевірки наявності нових даних у ВР. Після того, як інформація була зчитана, відбувається послідовне підключення кожного елементу вектора (по суті локального репозиторія) до ВР.

Алгоритм перевірки наявності нової інформації у ВР - основний алгоритм навколо якого побудована вся програма (Рис. 3).

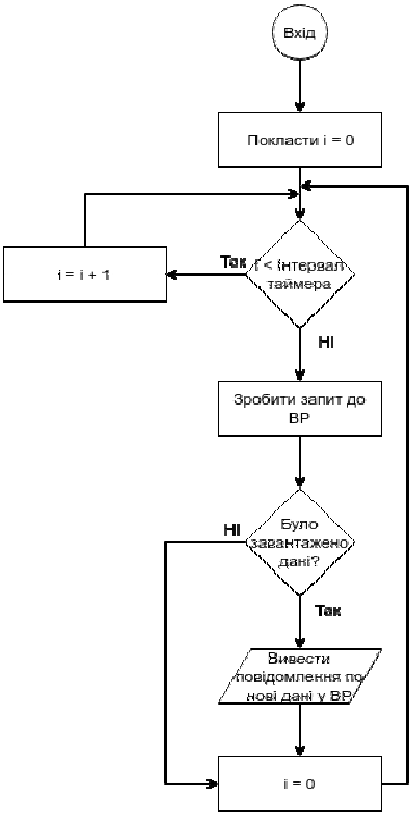

Рисунок 3 - Алгоритм перевірки наявності нової інформації у ВР

Даний алгоритм потребує попереднє виконання одного із двох вищеописаних алгоритмів, тобто в програмі повинен бути хоча б один ВР. Таймер з певним інтервалом часу робить запит до ВР і відбувається спроба зчитати нові дані. Якщо були завантажені якісь нові дані, яких ще немає в локальній копії на ПК, то формується повідомлення для користувача, яке повідомляє про наявність нової інформації у ВР.

У випадку, якщо нічого не було завантажено, то стартує нова ітерація таймера.

Висновки. Оскільки СКВ і ВР дуже тісно пов’язані між собою і весь код проектів великих і малих ІТ-компаній знаходяться у репозиторіях, то постає потреба одному розробникові в завчасному інформуванні про зміни інших розробників.

Було розроблені алгоритми, які дозволять працювати з різними СКВ і стануть основою програми, яка буде вирішувати поставлену задачу.

ЛІТЕРАТУРА

1. Git - Работа с удаленными репозиториями [Електронний ресурс]. Режим доступу: https://git-scm.com/book/ru/v1/Основы-Git-Работа-с-удалёнными-репозиториями

2. Система управления версиями [Електронний ресурс]
Режим доступу: https://ru.wikipedia.org/wiki/Система_управления_версиями

REFERENCES

1. Git - Rabota s udalennyimi repozitoriyami [Electronic resource]. Access mode:https://git-scm.com/book/ru/v1/Основы-Git-Работа-с-удалёнными-репозиториями

2. Sistema upravleniya versiyami [Electronic resource]. Access mode: https://ru.wikipedia.org/wiki/Система_управления_версиями