Інформаційні технології

УДК 004.42

Дідух Юлія Володимирівна
студентка
Національного університету «Львівська політехніка»
Дидух Юлия Владимировна
студентка
Национального университета «Львовская политехника»
Didukh Yuliia
Student of the
Lviv Politechnic National University

АНАЛІЗ СИСТЕМ КОНТРОЛЮ ВЕРСІЙ ТА ЇХ ІНТЕГРАЦІЯ У MS SQL MANAGEMENT STUDIO
АНАЛИЗ СИСТЕМ КОНТРОЛЯ ВЕРСИЙ И ИХ ИНТЕГРАЦИЯ В MS SQL MANAGEMENT STUDIO
ANALYSIS OF VERSION CONTROL SYSTEMS AND THEIR INTEGRATION IN MS SQL MANAGEMENT STUDIO

Анотація. *У дослідженні здійснено аналіз взаємодії систем контролю версій з MS SQL Management Studio. Розглянуто ключові фактори взаємодії. Здійснена порівняльна характеристика існуючих систем контролю версій.*

Ключові слова: *система контролю версій, GitHub, TFS, MS SQL Management Studio.*

Аннотация. *В исследовании проведен анализ взаимодействия систем контроля версий с MS SQL Management Studio. Рассмотрены ключевые факторы взаимодействия. Осуществлена сравнительная характеристика существующих систем контроля версий.*

Ключевые слова: *система контроля версий, GitHub, TFS, MS SQL Management Studio.*

Summary. *The study analyzes the interaction of version control systems with MS SQL Management Studio. Key factors of interaction are considered. Comparative characteristics of existing version control systems have been made.*

Key words: *version control system, GitHub, TFS, MS SQL Management Studio.*

Поставка проблеми. Програмою MS SQL Management Studio користуються досить велика кількість програмуючих фірм та проектних груп. На даний момент в програмі відсутня система контролю версій і розробники витрачають багато свого часу на перевірку змін в програмному коді, таких як ким був змінений код, коли і по якій причині.

Аналіз останніх досліджень і публікацій. Серед актуальних статей на дану тему можна виділити статтю авторства Albrecht A.J. та Gaffney J.R. [1]. У даній статті автори пояснюють, як переконатися, що ваша система управління версіями повністю підтримує всі етапи життєвого циклу бази даних: управління, розробки, операцій.

Метою даної роботи є плагіну до MS SQL Server Management Studio для роботи з системами контролю версіями для розробників та користувачів, які будуть переглядати зміни в програмному коді, коли вони відбулися, ким і по якій причині.

Виклад основного матеріалу. Системи, які команди розробників використовують для відстеження змін та різних версій коду, називаються системами управління версіями (VCS). Як і в усьому іншому, кожен VCS має свої унікальні особливості та має свій набір переваг та недоліків. Однак є деякі основи, які роблять VCS таким, яким він є.

Перш за все, вони повинні зберігати багаторічну історію змін, внесених у проект, включаючи створення, видалення, редагування тощо. Сюди також слід включити автора, дату та будь-які примітки до змін.

Крім того, вони повинні мати рішення для розгалуження та об'єднання нових змін коду до основного проекту, щоб дозволити одночасну роботу з декількома членами команди.

Існують добре встановлені протоколи та процедури для перевірки об'єктів у програмному забезпеченні для керування версією у компанії. DBA та розробники ретельно перевіряють об'єкти, які вони будуть модифікувати в Visual Source Safe або іншому програмному забезпеченні для керування версіями.

Для великих баз даних це може бути тривалим процесом. Часто розробник SQL працює в середовищі розробки, який не підтримується системою контролю версій, і відновлення цих об'єктів неможливе.

У таких середовищах здійснюється обробка пакетів, яка підключається до всіх серверів SQL в середовищі, скриптує всі об'єкти бази даних, а потім виштовхує модифіковані об'єкти у Visual Source Safe [2].

У Visual Source Safe легко можна отримати попередню версію об'єкта. Можна позначити групи об'єктів, щоб можна було генерувати сценарій установки для набору об'єктів для конкретної версії.

Для початку необхідно розглянути існуючі системи контролю версій, які можна інтегрувати в MS SQL Management Studio. До порівняння взято дві системи GitHub [3] I Team Foundation Server [4].

TFS - це централізована версія. Git розподілена система, оскільки кожен має повну копію всього репозиторію та його історії. TFS має свою власну мову: реєстрація / вихід. Користувачі TFS "реєстрація" викликають блокування файлів, тоді як користувачі Git роблять коміти на основі розподілених повних версій з перевіркою різниці. TFS надає "коміт", щоб тимчасово проводити локальні зміни. Сховище Git знаходиться подалі від

файлів, які закомітилися. Набори полиць у TFS зберігаються на центральному сервері. Заховані елементи в Git залишаються локальною машиною. Групи TFS змінюються в наборах послідовно пронумерованих змін. Git призначає 32-байтовий хеш для кожного коміту. Відділення TFS створює нову папку.

На Рис. 1 представлена порівняльна характеристика систем. Основними критеріями для порівняння є:

1. Набір можливостей;
2. Простота у використанні;
3. Підтримка клієнтів;
4. Коефіцієнт випуску;
5. Цінова політика;
6. Розширюваність;
7. Інтеграція третьої сторони;
8. Компанії, які використовують продукт;
9. Навчання.

Як бачимо лідером є система GitHub, яка має перевагу в більшості пунктах.

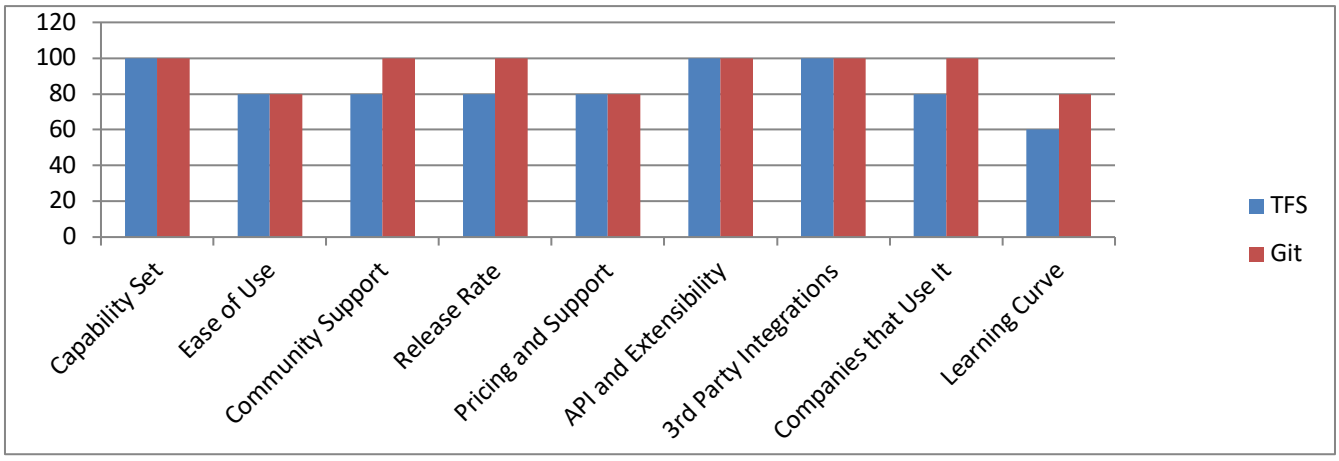

Рис. 1. Порівняльна діаграма Git і TFS

До інтеграції системи контролю версій можна віднести наступні бібліотеки, яких є достатньо для правильної роботи плагіну.

- EnvDTE. Представляє собою збірки, які містять в собі код, який дозволяє розширювати середовище розробки системи контролю версій. Кожна з таких збірок реалізує функціональні можливості, які належать конкретній версії.
- Octokit. Представляє собою пакет, який дозволяє через API отримувати доступ до інтерфейсу GitHub. З його допомогою можна зайти в кабінет необхідного користувача, отримати необхідний репозиторій чи вітку.
- System.Threading.Tasks. Представляє собою пакет, який дозволяє за допомогою пакету Octokit створювати нові функції, які дозволять нам вбудувати функціонал GitHub в інтерфейс MS SQL Management Studio.

Висновки. Сучасні розробники досить часто використовують існуючі системи контролю версій, які не є інтегровані в середовище MS SQL Management Studio. При цьому перевірка змін в програмному коді є невід'ємною частиною їхньої роботи. Оскільки ключовим для розробників є здатність швидко реагувати на зміни і витрачати на це мінімум часу, слід розробити систему контролю версій інтегровану у середовище розробки.

Література

1. Albrecht A.J. Software Function Source Lines of Code and Development Effort Prediction: A Software Science Validation / Gaffney J.R. − IEEE Trans. Software Eng. Vol. 9. №6. 1983. PP. 638−648.
2. Atkins D. Using Version Control Data to Evaluate the Effectiveness of Software Tools / Ball T., Graves T., Mockus A. Proc. Int'l Conf. Software Eng. 1999. PP. 324−333.
3. GitHub [Електронний ресурс]. 2019. URL: https://github.com
4. Team Foundation Server [Електронний ресурс] // Microsoft. 2005. URL: https://visualstudio.microsoft.com/ru/tfs/