1. Титул
2. Регресійне тестування – дорогий, але необхідний етап тестування програмного забезпечення, спрямований на повторну перевірку коректності зміненої програми.

Звичайно використовувані методи регресійного тестування включають повторні прогони попередніх тестів, а також перевірки, чи не потрапили регресійні помилки в чергову версію в результаті злиття коду. Тому важливим завданням планування регресійного тестування є зменшення кількості виконуваних тестів, що зменшить сукупну вартість і тривалість тестування.

* 1. У даній роботі ставиться за мету: Аналіз методів регресійного тестування;Аналіз систем управління версіями; Розробка методу регресійного тестування на основі систем управління версіями; Розробка програмного продукту для планування регресійного тестування на основі систем управління версіями

1. Нехай - множина з ***N*** тестів, що використовується при первинній розробці програми ***P***, а - підмножина регресійних тестів для тестування нової версії програми ***P'***.

Інформація про покриття коду, яке забезпечується , дозволяє вказати блоки ***P'***, що потребують додаткового тестування, для чого може знадобитися повторний запуск деяких тестів з множини , або навіть створення - набору нових тестів для ***P'*** - і оновлення . Правильне виявлення множини є дуже важкою задачею.

Завдання відбору тестів з набору для заданої програми ***P*** і зміненої версії цієї програми полягає у виборі підмножини для повторного запуску на зміненій програмі , де

1. Реалістичний варіант вирішення завдання вибіркового регресійного тестування полягає в отриманні корисної інформації за результатами виконання ***P*** і об'єднання цієї інформації з даними статичного аналізу для отримання множини у вигляді апроксимації . Цей підхід застосовується у всіх відомих вибіркових методах регресійного тестування, заснованих на аналізі коду. Множина повинно включати всі тести з , що активують змінений код, і не включати ніяких інших тестів, тобто тест входить до тоді і тільки тоді, коли задіє код ***P*** в точці, де в код був видалений або змінений, або де був доданий новий код.

Якщо певний тест задіє в ***P*** той самий код, що і в , вихідні дані ***P*** і для відрізнятися не будуть. З цього випливає, що якщо , повинен задіяти певний код, змінений в по відношенню до , тобто повинно виконуватися відношення . З іншого боку, оскільки не кожне виконання зміненого коду відбивається на вихідних значеннях тесту, можуть існувати деякі такі , що . Таким чином, містить цілком і може використовуватися в якості його альтернативи без шкоди для якості програмного продукту.

1. У галузі розробки програмного забезпечення, контроль версій є єдиною можливістю відстежувати і забезпечувати контроль за змінами у вихідному коді. Команди розробників одночасно проектують, розробляють та впроваджують декілька версій одного і того ж програмного забезпечення, які будуть розгорнуті в різних місцях, і одночасно працюють над оновленнями(patches). Певні помилки або функції програмного забезпечення часто присутні тільки в деяких версіях. Таким чином, з метою пошуку та виправлення помилок, життєво важливо мати можливість контролювати процес регресійного тестування і запускати різні набори тестів на різних версіях.
2. Системи контролю версій мають високу функціональність і дозволяють отримати детальну інформацію в яких саме файлах(модулях програмного забезпечення) відбулися зміни в порівнянні з попередньою версією. Спираючись на цю інформацію, а також на взаємозв'язки між файлами можна побудувати когнітивну карту змін у ПЗ – виявити не тільки змінені модулі, але й ті модулі що могли зазнати випадкового негативного впливу при внесенні змін або при рефактринзі коду.
3. Проведений аналіз методів регресійного тестування та систем управління версіями дав можливість розробити метод планування регресійного тестування заснований на використанні систем управління версіями та розробити програмне забезпечення що реалізує дані функції за допомогою побудови когнітивних карт.