**Комп’ютерні науки**

**УДК** [**004.415.53**](http://teacode.com/online/udc/00/004.415.53.html)

**РОЗРОБКА ПРОГРАММНОГО ЗАБЕЗПЕЧЕННЯ ЧЕРЕЗ ТЕСТУВАННЯ – TEST-DRIVEN DEVELOPMENT**

*Рогачов О.В., студент групи 401-ТН, Дмитренко Т.А., кандидат технічних наук, Деркач Т.М., кандидат технічних наук*

*Полтавський національний технічний університет імені Юрія Кондратюка, Україна, Полтава*

**Анотація**

*У статті розглянуті головні елементи циклу розробки ПЗ та* [*технологія розробки програмного забезпечення*](http://uk.wikipedia.org/w/index.php?title=%D0%A2%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D1%96%D1%8F_%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F&action=edit&redlink=1) *Test-driven development (TDD). В роботі представлен перелік переваг розробки ПЗ через тестування.*

***Ключові слова:*** *розробка програмного забезпечення, тестування, цикл розробки ПЗ, Test-Driven Development.*

**Вступ.** На сучасному етапі становлення інформаційного суспільства виникає потреба в підвищенні якості та надійності програмного забезпечення (ПЗ). Враховуючи основні тенденції розвитку інформаційних технологій формуються нові концепції щодо процесів розробки програмного забезпечення та застосування новітніх технологій створення, таких як:

* *Rapid Application Development* (RAD) – концепція створення засобів розробки програмних продуктів, що приділяє особливу увагу швидкості й зручності програмування, створенню технологічного процесу, що дозволяє програмістові максимально швидко створювати комп'ютерні програми [4];
* *Extreme Programming* (XP) – використовується для скорочення вартості змін, завдяки представленню простих значень, принципів і методів. При використанні екстремального програмування, проект повинен стати гнучкішим щодо змін [5];
* *Agile Software Development* (ASD) – клас методологій розробки програмного забезпечення, що базується на ітеративній розробці, в якій вимоги та розв'язки еволюціонують через співпрацю між самоорганізовуваними багатофункціональними командами[6].

В сучасних умовах актуальності набуває завдання підвищення якості програмного продукту шляхом модифікації існуючих методів, технологій та моделей якості.

Одним із головних елементів циклу розробки ПЗ є тестування. Тестування – це техніка контролю якості, що перевіряє відповідність між реальною і очікуваною поведінкою програми завдяки кінцевому набору тестів, які обираються певним чином. Якість не є абсолютною, це суб'єктивне поняття. Тому тестування не може повністю забезпечити коректність програмного забезпечення. Воно тільки порівнює стан і поведінку продукту зі специфікацією.

Існує багато підходів до тестування програмного забезпечення, але ефективне тестування складних продуктів – це по суті дослідницький процес, а не тільки створення і виконання рутинної процедури.

Аналіз результатів

Набір тестів

Результати тестування - звіти

Розробка тестів

Виконання тестів

Тестоване ПЗ

Вимоги, що перевіряються

Критерії повноти тестування

Критерії повноти тестування

Вимоги, що перевіряються

Тестоване ПЗ

Критерії повноти тестування

Вимоги, що перевіряються

Тестоване ПЗ

Виконання тестів

Розробка тестів

Рис.1. Схема процесу тестування

**Основний матеріал та результати.** Однією з популярних методологій є розробка через тестування, або TDD (Test-Driven Development).

Test-driven development (TDD) − [технологія розробки програмного забезпечення](http://uk.wikipedia.org/w/index.php?title=%D0%A2%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D1%96%D1%8F_%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F&action=edit&redlink=1), яка використовує короткі ітерації розробки, що починаються з попереднього написання тестів, які визначають необхідні покращення або нові функції. Кожна ітерація має на меті розробити код, який пройде ці тести. Нарешті, програміст або група вдосконалюють код для погодження змін. Один із ключових моментів TDD полягає у тому, що підготовка тестів перед написанням самого коду пришвидшує процес внесення змін. Варто зауважити, що розробка через тестування є методологією розробки програмного забезпечення, а не його [тестування](http://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F).

Test-Driven Development відноситься до концепції [екстремального програмування](http://uk.wikipedia.org/wiki/%D0%95%D0%BA%D1%81%D1%82%D1%80%D0%B5%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), яка стверджує, що в першу чергу потрібно писати тести, а вже потім код.

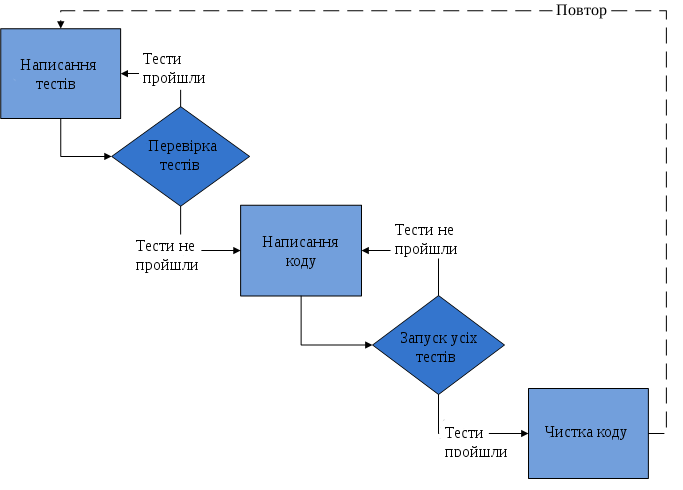


Рис.2. Графічна репрезентація циклу розробки

Три основні закони TDD.

1. Новий робочий код пишеться тільки після того, як буде написаний модульний тест, який не проходить.

2. Розробник пише рівно такий обсяг коду модульного тесту, який необхідно для того, щоб цей тест не проходив (якщо код тесту не компілюється, вважається, що він не проходить).

3. Розробник пише рівно такий обсяг робочого коду, який необхідно для проходження модульного тесту, який в даний момент не проходить.

Розробка в стилі TDD базується на цих трьох законах. Процес розробки складається з коротких циклів (тривалістю від 2 хвилин, залежно від досвідченості і стилю роботи програміста). Кожен цикл складається з наступних кроків:

1. З репозиторія витягається програмна система, що знаходиться в погодженому стані, коли увесь набір модульних тестів виконується успішно.
2. Додається новий тест. Він може полягати в перевірці, чи реалізує система деяку нову поведінку або чи містить деяку помилку, про яку нещодавно стало відомо.
3. Цей тест може полягати в перевірці, чи реалізує система деяку нову поведінку або чи містить деяку помилку, про яку нещодавно стало відомо. Важливо написати тест до внесення змін до коду модуля, тим самим пред'являючи до модуля нову вимогу.
4. Успішно виконується увесь набір тестів, окрім нового тесту, який виконується неуспішно. Цей крок потрібний для перевірки самого тесту – чи включений він в загальну систему тестування і чи правильно відбиває нову вимогу до системи, якій вона, природно, ще не задовольняє.
5. Програма змінюється з тим, щоб якнайскоріше виконувалися усі тести. Треба додати найпростіше рішення, що задовольняє новому тесту, і одночасно з цим не зіпсувати існуючі тести. Велика частина небажаних побічних і віддалених ефектів від змін, що вносяться в програму, відстежується саме на цьому етапі, за допомогою досить повного набору тестів.
6. Увесь набір тестів виконується успішно.
7. Тепер, коли потрібна в цьому циклі функціональність досягнута найпростішим способом, програма перебудовується (рефакторинг) для поліпшення структури і усунення надмірного, дубльованого коду.
8. Увесь набір тестів виконується успішно.
9. Комплект змін, зроблених в цьому циклі в тестах і програмі заноситься в репозиторій (операція commit), після чого програма знову знаходиться в погодженому стані і містить чітко відчутне поліпшення в порівнянні з попереднім станом.

Цей цикл називають ще червоний-зелений-реорганізація (рефакторинг): написання провального тесту (червоний), написання робочого коду для проходження тесту (зелений) та рефакторинг коду (наприклад, для усунення повторень) [1].

Перевагами TDD є:

* Використання розробки через тестування передбачає написання більшої кількості тестів; у свою чергу, програмісти, які пишуть більше тестів, схильні бути більш продуктивними.
* З самого початку сфокусувавшись на тестах, простіше уявити, яка функціональність необхідна користувачеві. Таким чином, розробник продумує деталі інтерфейсу до реалізації. Тести змушують робити свій код більш пристосованим для тестування.
* Незважаючи на те, що при розробці через тестування потрібно написати більшу кількість коду, загальний час, витрачений на розробку, зазвичай виявляється менше. Тести захищають від помилок. Тому час, що витрачається на налагодження, зменшується в рази [3].
* Добре протестований код легко переносить рефакторінг. Впевненість у тому, що зміни не зламають існуючу функціональність, надає впевненість розробникам і збільшує ефективність їх роботи.
* Розробка через тестування сприяє більш модульному і гнучкому коду. Це пов'язано з тим, що при цій методології розробнику необхідно думати про програму, як про безліч невеликих модулів, які написані і протестовані незалежно і лише потім з'єднані разом. Це призводить до менших, більш спеціалізованих класів, зменшенню пов'язаності і чистіших інтерфейсів.
* Оскільки пишеться лише той код, що необхідний для проходження тесту, автоматизовані тести покривають всі шляхи виконання.

До недоліків відносяться:

* Головним недоліком TDD є те, що до нього тяжко звикнути.
* Існують завдання, які неможливо (принаймні, на поточний момент) вирішити тільки за допомогою тестів. Зокрема, TDD не дозволяє механічно продемонструвати адекватність розробленого коду в області безпеки даних і взаємодії між процесами. Безумовно, безпека заснована на коді, в якому не повинно бути дефектів, проте вона заснована також на участі людини у процедурах захисту даних.
* Модульні тести, створювані при розробці через тестування, звичайно пишуться тими ж, хто пише тестований код. Якщо розробник неправильно сформулював вимоги до розробки, і тест, і тестований модуль будуть містити помилку.
* Велика кількість використовуваних тестів можуть створити помилкове відчуття надійності, що приводить до меншої кількості дій з контролю якості.
* Рівень покриття тестами, що отримується в результаті розробки через тестування, не може бути легко отриманий згодом. Вихідні тести стають все більш цінними з плином часу. Якщо невдалі архітектура, дизайн або стратегія тестування призводять до великої кількості тестів які не проходять, важливо їх всі виправити в індивідуальному порядку. Просте видалення, відключення або поспішна зміна їх може призвести до прогалин у покритті тестами.

**Висновки.** Розробка через тестування (англ. test − driven development, TDD) – є передовою технікою розробки програмного забезпечення, яка має цілий ряд як переваг так і недоліків. Тестування грає ключову роль в забезпеченні належного рівня якості програмного забезпечення і виконанні контрольних завдань. Використання розробки через тестування припускає написання більшої кількості тестів, у свою чергу, програмісти, що пишуть більше тестів, схильні бути продуктивнішими. Програміст повинен вміти не просто генерувати будь-який код, який працює, а і обов’язково володіти якісним стилем програмування, методами документування, застосовувати методи мінімізації коду, проводити ефективний пошук помилок, зокрема не явних, на етапі відладки та вміти якісно тестувати власний програмний продукт.

*Література*

1. *Мартин Р. Идеальный программист. Как стать профессионалом разработки ПО; издательский дом «Питер» – СПб., 2011. – 224 с.*
2. *Erdogmus, Hakan; Morisio, Torchiano. On the Effectiveness of Test-first Approach to Programming. Proceedings of the IEEE Transactions on Software Engineering, January 2005.*
3. *Müller, Matthias M.; Padberg, Frank. About the Return on Investment of Test-Driven Development. Universität Karlsruhe, Germany.*
4. *Швидка розробка програмного забезпечення//Вікіпедія.* [*http://uk.wikipedia.org/wiki/ Швидка*](http://uk.wikipedia.org/wiki/%20Швидка) *розробка програмного забезпечення.*
5. *Екстремальне програмування // Вікіпедія. http://uk.wikipedia.org/wiki/ Екстремальне програмування.*
6. *Гнучка розробка програмного забезпечення // Вікіпедія.* [*http://uk.wikipedia.org/wiki/ Гнучка*](http://uk.wikipedia.org/wiki/%20Гнучка) *розробка програмного забезпечення.*

**РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ЧЕРЕЗ ТЕСТИРОВАНИЕ - TEST - DRIVEN DEVELOPMENT**

*Рогачов О.В., студент групи 401-ТН, Дмитренко Т.А., кандидат технических наук, Деркач Т.Н., кандидат технических наук*

*Полтавский национальный технический университет имени Юрия Кондратюка, Украина, Полтава*

**Аннотация**

*В статьи представлены главные элементы цикла разработки программного обеспечения* *технология* *разработки программного обеспечения Test - driven development (TDD). В работе представлен перечень* *преимуществ разработки программного обеспечения через тестирование.*

***Ключевые слова:*** *разработка программного обеспечения, тестирования, цикл разработки программного обеспечения, Test - Driven Development.*

**SOFTWARE DEVELOPMENT THROUGH TESTING -   
TEST - DRIVEN DEVELOPMENT**

*Rogachov А.В., student, Dimitrenko TA, Ph.D, Derkach TN, Ph.D.*

*Poltava National Technical University named after Yuri Kondratyuk, Ukraine, Poltava*

**Annotation**

*In the articles of is presented staples of cycle of software development technology of the Test software development - driven development (TDD). The list of advantages of software development is in - process presented through testing.*

***Keywords:*** *software development, testing, cycle of software development, Test - Driven Development.*