Семенюк Р. А.
студент фізико-математичного факультету
Науковий керівник: *О.Ю.Усата*
канд. пед. наук, доцент,
доцент кафедри прикладної математики та інформатики
Житомирський державний університет імені Івана Франка

Анотація*. У статті розглянуто основні поняття та головну ідею рефакторигу.*

Ключові слова*: рефакторинг .*

Аннотация*. В статье рассмотрены основные понятия и принципы рефакторинга.*

Ключевые слова: *рефакторинг.*

Abstract*. The article describes the main concepts and principles of refactoring*

Keywords*: refactoring.*

ІДЕЯ РЕФАКТОРИНГУ ТА ЙОГО КОРИСТЬ ДЛЯ ПРОГРАМІСТА

В наш час, коли розробка програмних проектів набирає все більших обсягів, програми стають громіздкі через свою поширеність. Для їх підтримки потребуються фахівці, які постійно вносять зміни в програмний код. Кожен фахівець змінює його за власним стилем, від чого прочитати код стає складніше, що збільшує об’єм роботи над ним для наступного. Методи ідеї рефакторингу дозволяють поетапно модифікувати програмний код, вносячи щоразу невеликі зміни, завдяки чому зменшується ризик, пов’язаний з розвитком проекту.

Мета даної статті полягає в розгляді поняття ідеї рефакторінгу, як важливої частини розробки громіздких програмних ресурсів.

Рефакторинг являє собою процес такої зміни програмної системи, при якій не змінюється поведінка програмного коду, але покращується його внутрішня структура. Це спосіб систематичного приведення коду в порядок, при якому шанс появи нових помилок зводиться до мінімуму. Проведення рефакторингу коду покращує його дизайн уже після того як він написаний.[1]

Рефакторинг – інструмент, який можна використати для кількох цілей. А саме: полегшення читання коду як для себе, так і для сторонніх програмістів, спрощення і пришвидшення модифікації коду в подальшому шляхом створення зручної композиції розроблюваного проекту.

Без рефакторингу код втрачає свою композицію. Розібратися в проекті, читаючи код, стає все складніше. Рефакторинг нагадує наведення порядку в коді. Прибираються компоненти, що опинилися не на своєму місці. Чим складніше розібратися у внутрішній будові коду, тим важче його зберегти і відбувається розпад. Регулярне проведення рефакторингу допомагає зберегти форму коду.

Погано спроектований код, зазвичай, займає багато місця, тому що виконує в кількох місцях одне і теж. Тому важливу частину композиції складає видалення дублюючого коду. Скорочення об’єму коду не зробить систему більш швидкою, тому що зміна розмірів програми в пам’яті не завжди має значення. Об’єм коду має велике значення, коли доводиться його модифікувати. Чим більше коду, тим важче в ньому розібратися. При зміні коду в певному місці система веде себе не відповідно розрахункам, оскільки немодифікований інший участок, котрий робить теж саме, але дещо в іншому контексті. Усуваючи дублювання, ми гарантуємо, що в коді є все, що потрібно і при цьому в одному місці, чим визначається ідея хорошої постановки композиції проектування.

Працюючи над тим, щоб примусити програму працювати, зовсім не потрібно замислюватись про розробника, який буде займатися нею в майбутньому. Рефакторинг допомагає зробити код більш легким для читання. При проведенні рефакторингу ми беремо код, який працює, проте не виділяється ідеальною структурою. Завдяки витраті часу на рефакторинг, програмний код краще інформує про ціль.

Також, при проведенні рефакторингу, програміст заглиблюється в код, намагається зрозуміти, що він робить, і, досягши розуміння, повертає його в код, поліпшуючи його. Отже рефакторинг допомагає відшукати помилки.

Беручи до уваги вище описане, можна дійти висновку, що рефакторинг сприяє пришвидшенню розробки. Хороший дизайн системи важливий для швидкої розробки програмного забезпечення. Вся ідея хорошого дизайну коду в тому, щоб зробити можливою швидку розробку. Без нього деякий час можна швидко просуватися, проте невдовзі поганий дизайн починає гальмувати весь процес. Час витрачатиметься не на додавання нових функцій, а на пошук і виправлення помилок. Внесення модифікацій займатиме більше часу, оскільки доведеться розбиратися в системі і шукати дубльований код. Хороший дизайн важливий для збереження швидкості розробки програмного продукту.

Список використаних джерел та літератури:

1. Рефакторинг: улучшение существующего кода. - Пер. с англ. - СПб: Символ-Плюс, 2003. - 432 с.
2. Код с душком [Електронний ресурс] – Режим доступу: https://habrahabr.ru/post/169139/
3. Рефакторинг [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3