К. А. Паламарчук

ТЕСТУВАННЯ. МОДЕЛІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Вінницький національний технічний університет;

Анотація
В роботі проведений аналіз моделей та методологій розробки програмного забезпечення, досліджені принципи та види різних моделей, проведена оцінка переваг та недоліків найбільш поширених моделей, визначено підхід до вибору методологій розробки програмного продукту на користь проекту.

Ключові слова: тестування програмного забезпечення, моделі розробки програмного забезпечення, модель водоспаду, V-модель, інкрементна модель, модель швидкої розробки додатків, «гнучка» модель, ітераційна модель, спіральна модель, модель прототипу

Abstract
The paper analyzes the models and methodologies of software development and investigates them principles and types of different models, the advantages and disadvantages of the most common models were evaluated, the approach to the choice of methodologies of software development for the project is defined.

Keyword: software testing, software development models, waterfall model, V-model, incremental model, RAD model, agile model, iterative model, spiral model, prototype model.

Вступ

Програмні системи використовувалися та розроблялися в рамках програм наукових досліджень або програм міністерств оборони. Тестування таких продуктів проводилося із записом всіх тестових процедур, тестових даних, отриманих результатів. Тестування виділялося в окремий процес, який починався після завершення кодування, далі виконувалось тим же персоналом. Тестові сценарії записувалися на папір. З їх допомогою перевірялися потоки управління, обчислення складних алгоритмів і маніпулювання даними. В середині 1990-х років з розвитком Інтернету і розробкою великої кількості веб-додатків особливу популярність отримало «гнучке тестування»

Результати дослідження

Тестування програмного забезпечення - це процес виконання програми з метою виявлення програмних помилок. Тестування програмного забезпечення є важливою частиною розробки програмного забезпечення. Якщо тестування програмного забезпечення не виконується належним чином, додатки можуть мати помилки, що можуть призвести до переробки, відмови або, що ще гірше, втрати життя. Воно необхідне для виявлення помилок в програмному забезпеченні , перевірки відповідності програмного забезпечення вимогам .Це допомагає команді розробників виправляти помилки і надавати якісний продукт.

Бізнес-організації та їх репутація залежать від якості продуктів. Користувачі виберуть якісний конкуруючий продукт, порівняно з продуктом, який має низьку якість. Це може призвести до втрати прибутку організації. У сучасному світі якість є одним з головних пріоритетів для будь-якої організації.

Моделі розробки програмного забезпечення - це різні процеси або методології, які обираються для розробки проекту в залежності від завдань проекту. Існує багато моделей життєвого циклу розробки, які були створені для досягнення необхідної мети. Моделі визначають різні етапи процесу і порядок їх виконання. Вибір моделі впливає на проведене тестування. Обрана модель визначає, де і коли планується тестування, які методи тестування потрібно використовувати. Вибір потрібної моделі для розробки програмного продукту або програми важливий. На основі моделі виконуються процеси розробки та тестування. Існують різні моделі або методології розробки програмного забезпечення:

- Waterfall model (модель водоспаду);
- V model (V-модель);
- Incremental model (Інкрементна модель);
- RAD model (модель швидкої розробки додатків);
- Agile model («гнучка» модель);
- Iterative model (ітераційна модель);
- Spiral model (спіральна модель);
- Prototype model (модель прототипу).

Модель водоспаду (waterfall model) була першою моделлю процесу, яка була представлена. Вона також називається "лінійно-послідовна моделлю життєвого циклу". У моделі водоспаду кожна фаза повинна бути повністю завершена до початку наступної фази. Цей тип моделі розробки програмного забезпечення використовується для невеликого проекту, в якому немає невизначених вимог. В кінці кожного етапу проводиться перевірка, щоб визначити, чи проект відповідає вимогам. У цій моделі тестування програмного забезпечення починається тільки після завершення розробки.
Переваги моделі водоспаду: ця модель проста і легка для розуміння , використання; у цій моделі етапи обробляються і завершуються послідовно по одному; фази не перекриваються; модель водоспаду добре працює для невеликих проектів, де вимоги чітко визначені та зрозумілі.
Недоліки моделі водоспаду: коли додаток знаходиться на стадії тестування, дуже важко повернутися назад і змінити щось, що не було добре продумано на етапі розробки концепції; велика кількість ризику через невизначеності; невдала модель для складних і об'єктно-орієнтованих проектів; не підходить для проектів, де вимоги змінюються від помірного до високого ризику; модель не підходить для тривалих і поточних проектів.

V-модель (V-model) означає модель верифікації та валідації. Як і модель водоспаду, V-подібний життєвий цикл - послідовний шлях виконання процесів. Кожен етап має бути завершений до початку наступного етапу. В цій моделі до початку розробки створюється план тестування системи. План тестування спрямовано на виконання функціональності, що зазначена в зборі вимог.
Переваги V-моделі: легко і просто використовувати; дії з тестування, такі як планування, розробка тестів відбувається задовго до написання коду, це економить багато часу; запобігає спадний потік дефектів; добре працює для невеликих проектів, де вимоги легко зрозуміти.
Недоліки V-моделі: програмне забезпечення розробляється на етапі реалізації, тому ранні прототипи програмного забезпечення не виготовляються; якщо будь-які зміни відбудуться на півдорозі, тоді необхідно оновити документи випробувань та документи до вимог.

Інкрементна модель (Incremental model). В інкрементній моделі всі вимоги розділені на різні збірки. Тут відбувається безліч циклів розробки. Цикли поділяються на більш дрібні, керовані модулі. В моделі кожен модуль проходить етапи вимог, проектування, впровадження та тестування. Робоча версія програмного забезпечення створюється під час першого модуля. Тому є можливість працювати з програмним забезпеченням на ранніх етапах життєвого циклу програмного забезпечення. Кожен наступний випуск модуля додає функцію до попереднього випуску. Процес триває до тих пір, поки не буде досягнута повна система.

Переваги інкрементної моделі: модель більш гнучка - вона менш затратна для зміни області застосування та вимог; знижує первісну вартість доставки, простіше управляти ризиками, тому що ризиковані частини ідентифікуються та обробляються під час ітерації, генерує працююче програмне забезпечення швидко на початку життєвого циклу.
Недоліки інкрементної моделі: потребує правильного планування дизайну, потрібне чітке та повне визначення всієї системи, перш ніж вона може бути розбита і побудована поступово, загальна вартість .

Модель швидкої розробки додатків (RAD model) - це тип інкрементної моделі. У моделі RAD компоненти або функції розвиваються паралельно. Розробки вчасно упаковані, доставлені і потім зібрані в робочий прототип. Ця модель дає можливість замовнику побачити те, що можна використати, а також здійснити зворотний зв'язок щодо доставки та їх вимог.
Переваги моделі RAD: скорочення часу розробки, збільшує можливість повторного використання компонентів, інтеграція з самого початку вирішує безліч проблем, заохочує відгуки клієнтів.
Недоліки моделі RAD: тільки система, яка може бути модульною, може бути побудована з використанням RAD; потрібні висококваліфіковані розробники / дизайнери; висока залежність від навичок моделювання; не застосовується до дешевших проектів, так як вартість моделювання та автоматичної генерації коду дуже висока.

«Гнучка» модель (agile model) також є різновидом інкрементної моделі. Програмне забезпечення розробляється в послідовних, швидких циклах. Це призводить до невеликих інкрементних випусків з кожним випуском на основі попередньої функціональності. Кожен випуск ретельно перевіряється, щоб гарантувати якість програмного забезпечення. Використовується для додатків, критичних до часу.
Переваги моделі Agile: задоволеність клієнтів від швидкої і безперервної доставки корисного програмного забезпечення; люди і взаємодії підкреслюються, а не процес і інструменти; клієнти, розробники і тестери постійно взаємодіють один з одним; постійна увага до технічної досконалості і правильного дизайну; працююче програмне забезпечення поставляється часто (тижні, а не місяці).
Недоліки моделі Agile: у разі великих програмних продуктів, важко оцінити зусилля, які необхідні на початку життєвого циклу розробки програмного забезпечення; тільки старші програмісти здатні приймати рішення, необхідні в процесі розробки, отже, в ньому немає місця для початківців-програмістів.

Ітераційна модель (Iterative model) не намагається почати з повної специфікації вимог. Замість цього, розробка починається з визначення та впровадження тільки частини програмного забезпечення, яке потім може бути проаналізовано для визначення подальших вимог. Потім цей процес повторюється, створюючи нову версію програмного забезпечення для кожного циклу моделі. Коли ми працюємо послідовно, ми створюємо необроблений продукт або шматок продукту за одну ітерацію, потім переглядаємо його та покращуємо на наступній ітерації і так далі, аж поки не закінчимо.
Переваги ітераційної моделі: в ітераційній моделі створюється високорівневий дизайн програми лише до того, як створюється продукт і визначається проектне рішення для всього продукту. Пізніше проектується і будується каркасна версія . Потім розвивається дизайн на основі того, що було побудовано; в ітераційній моделі будується і покращується продукт крок за кроком, отже є можливість відслідковувати дефекти на ранніх стадіях; в ітераційній моделі менше часу витрачається на документування, більше часу приділяється проектування.
Недоліки ітераційної моделі: кожна фаза ітерації жорстка, без накладань; можуть виникнути дороговартісні проблеми архітектури або дизайну системи, оскільки не всі вимоги зібрані заздалегідь для всього життєвого циклу.

Спіральна модель (Spiral model) схожа на інкрементну модель, з великим акцентом на аналіз ризиків. Спіральна модель має чотири фази: планування, аналіз ризиків, інжиніринг та оцінка.

Програмний проект багаторазово проходить через фази ітераціями (в цій моделі званими спіралями). Базова спіраль, починаючи з етапу планування, збирає вимоги та оцінює ризик. Кожна наступна спіраль будується на базовій спіралі.

Переваги спіральної моделі: велика кількість аналізу ризиків дозволяє уникнути ризику; підходить для великих і відповідальних проектів; додаткова функціональність може бути додана пізніше; програмне забезпечення створюється на початку життєвого циклу програмного забезпечення.

Недоліки спіральної моделі: може бути дороговартісною моделлю для використання; успіх проекту в значній мірі залежить від етапу аналізу ризиків; не підходить для невеликих проектів.

Модель прототипу (Prototype model). Основна ідея полягає в тому, що замість того, щоб заморозити вимоги до того, як можна приступити до проектування або кодування, для розуміння вимог створюється одноразовий прототип. Цей прототип розроблений на основі відомих на даний час вимог. Використовуючи цей прототип, замовник може отримати «реальне уявлення» про систему, оскільки взаємодія з прототипом може дозволити йому краще зрозуміти вимоги необхідної системи.

Переваги моделі прототипу: користувачі активно беруть участь в розробці; оскільки в цій методології передбачена робоча модель системи, користувачі отримують краще уявлення про те, що розробляється; помилки можуть бути виявлені набагато раніше; відсутня функціональність може бути легко ідентифікована.

Недоліки моделі прототипу: ця методологія може збільшити складність системи, оскільки область дії системи може вийти за рамки первісних планів; неповний або неадекватний аналіз проблеми.

Висновки

Модель життєвого циклу - це структура, що складається із процесів, робіт та задач, які включають в себе розробку, експлуатацію і супровід програмного продукту; охоплює життя системи від визначення вимог до неї до припинення її використання.

Методології програмування розрізняються за загальним витратам на вирішення завдань з різними характеристиками (наукові розрахунки, фінансові завдання, системи реального часу тощо). Масштаб завдань і ефективність створюваного програмного забезпечення також є важливими факторами при виборі методології програмування.

Існує багато методологій розробки програмного забезпечення. Кожна з них має як переваги, так і недоліки. Правильний підхід до вибору методологій буде на користь проекту та сприятиме подальшому розвитку.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. What is Software Testing? Basics, Tutorial, Importance, Interview Questions: http://tryqa.com/what-is-software-testing/
2. Почему тестирование необходимо?:https://qalight.com.ua/baza-znaniy/pochemu-testirovanie-neobhodimo/
3. Техники тест дизайна (Test Design Technics): http://www.protesting.ru/testing/conditions.html

Паламарчук Катерина Арменівна— студентка групи 1АКІТ-17б, факультет комп'ютерних систем і автоматики, Вінницький національний технічний університет, м.Вінниця, e-mail: fkca.1akit.mka@gmail.com

Науковий керівник: ***Васюра Анатолій Степанович***—професор кафедри автоматики і інформаційно-вимірювальної техніки, Вінницький національний технічний університет, м.Вінниця.

Palamarchuk Kateryna A..—Department of Computer System and Automation, Vinnytsia National Technical University,Vinnytsia,, e-mail: fkca.1akit.mka@gmail.com

Supervisor: ***VasyuraAnatoliy S.***— Professor of Automation and Information and Measurement Engineering Department, Vinnytsia National Technical University,Vinnytsia