УДК 004.4`2

А.О. Лавренюк
Н.П. Бабюк

СУЧАСНІ МЕТОДОЛОГІЇ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Вінницький національний технічний університет

Анотація
У роботі розглянуто сучасні методології розробки програмного забезпечення. Також окреслено сфери застосування кожної із методологій та визначено їх переваги і недоліки.
Ключові слова: методологія, програмне забезпечення (ПЗ), розробка, проект, життєвий цикл.

Abstract
The article considers modern software development methodologies. It also outlines the scope of application of each methodology and identifies their advantages and disadvantages.
Keywords: methodology, software, development, project, life cycle.

Вступ

Розробка якісного продукту починається з визначення його життєвого циклу. Це чіткий план дій, що дозволяє зрозуміти, що має вийти в розробників, як досягти результату та які методи для цього слід використати. Методологія розробки програмного забезпечення – це перевірені способи та практики, що дозволяють створити диджитал продукт правильно та якісно.

Програмне забезпечення, що розробляється сучасними IT-компаніями потребує своєчасного здійснення контролю та якості розробки протягом усього життєвого циклу. Від результатів виконання кожного етапу залежить загальний успіх програмного проекту. Ефективна побудова процесу розробки проекту дозволить знизити ризики до мінімуму, а також максимально врахувати вимоги замовника.

Методології розробки програмного забезпечення сьогодення

Сучасні методології розробки програмного забезпечення не є універсальними і мають застосування лише в проектах певного типу. Кожна модель розробки програмного забезпечення має свої унікальні особливості, переваги та недоліки. Визначити, яка з них краща – не можна, оскільки під різні завдання, продукти та ідеї обирається свій принцип розробки. Розлянемо методології, які є найпопулярнішими та найбільш використовуваними у наш час.

Waterfall Model (каскадна модель або “водоспад”) – найпростіша і найстаріша методологія розробки ПЗ. Її принцип роботи досить простий: кожний наступний етап виконується лише тоді, коли повністю завершено попередній. Це чітка та структурована методологія, що зрозуміла та проста для використання. Кожний етап супроводжується документацією, що забезпечує хороше розуміння вимог і рішень проєкту. Завдяки послідовності етапів Waterfall надає передбачуваність у плануванні та виконанні проєкту. Проте методолією не передбачено зміни вимог після початку проєкту, що робить метод неефективним у разі, якщо вимоги часто змінюються. Також через значну кількість послідовних етапів може пройти багато часу, перш ніж замовник отримає робочий продукт. Тестування наприкінці процесу є ще одним недоліком цієї методології, оскільки воно відбувається на завершальних стадіях, то може призвести до виявлення серйозних проблем лише на пізніх етапах, що збільшує витрати на їх виправлення. Можна використовувати такий підхід у тому випадку, якщо у проекті є докладний прототип або вже існуюча програма.

Incremental Model (інкрементна модель) – методологія розробки ПЗ, що передбачає поділ проєкту на декілька незалежних частин або інкрементів, кожен з яких розробляється і тестується окремо. Кожен інкремент додає нову функціональність до ПЗ, і в результаті отримується повноцінний продукт, що складається з усіх інкрементів. Кожна частина роботи в такому проекті є готовим елементом. Іноді його можна використовувати окремо. Перевагами моделі насамперед є те, що мінімізуються ризики, забезпечується швидкий реліз та запуск продукту. Крім того, базовий функціонал вже працюватиме й приноситиме користь для бізнесу, а за необхідності завжди можна впровадити нові сформовані

інструменти. Ця модель дозволяє знизити ризики і витрати, пов’язані з розробкою ПЗ, саме тому ідеально підходить для проєктів, у яких вимоги до ПЗ можуть змінюватися протягом розробки, або де потрібно швидко випустити прототип або мінімально працездатний продукт.

Iterative Model (ітеративна або ітераційна модель) передбачає повторне виконання кожного етапу проєкту з урахуванням отриманого зворотного зв’язку від користувачів або замовника. Кожна ітерація включає аналіз вимог, проєктування, реалізацію, тестування і впровадження ПЗ. А кожна наступна ітерація покращує якість і функціональність ПЗ, додаючи нові можливості або виправляючи помилки. Ітеративна модель підходить для складних і динамічних проєктів, де вимоги до ПЗ не можуть бути повністю визначені на початку розробки, або де потрібно забезпечити особливо високу якість. Поміж іншим ця модель дозволяє доволі швидко адаптуватися до змін на ринку або у технологіях. За таким методом розробляються, наприклад, соціальні мережі та великі корпоративні платформи.

Spiral model (спіральна модель) – модель, за якою життєвий шлях продукту, що розробляється, зображується у вигляді спіралі, яка, розпочавшись на етапі планування, розкручується з проходженням кожного наступного кроку. Таким чином, на виході з чергового витка отримуємо готовий протестований прототип, який доповнює існуючу збірку. Прототип, що задовольняє всі вимоги, готовий до випуску. У такій моделі приділяється особлива увага управлінню ризиками, додаткові функції можуть бути додані на пізніх етапах, а також є можливість гнучкого проєктування. Недоліками є те, що оцінка ризиків на кожному етапі є досить витратною, а постійні відгуки і реакція замовника може провокувати все нові і нові ітерації, які можуть призводити до тимчасового затягування розробки продукту. Spiral model підходить для великих і складних проєктів, де ризики розробки ПЗ високі, а вимоги до ПЗ нестабільні або не надто ясні. Ця модель дозволяє контролювати якість і вартість ПЗ, а також забезпечити гнучкість і адаптацію до змін.

V-Model (V-подібна модель) – модель, особливість підходу якої полягає в наголосі на тестування та перевірку працездатності систем під час розробки. Тести виконуються одночасно із самим процесом створення продукту. Сам принцип успадковує базовий підхід при каскадній розробці. Процес іде покроково, існує чіткий план дій, складається суворе технічне завдання. Але паралельно виконуються тести, у разі виявлення помилок вони одразу виправляються, незалежно від етапу розробки. Особливість цієї методології розробки ПЗ полягає у тому, що на ранніх стадіях створення проводяться тести, а до нової стадії можна перейти лише тоді, коли усуваються всі помилки. При цьому на новій стадії тести аналізують не лише новий етап, а й усі попередні. Це дозволяє контролювати взаємозв'язок компонентів та їхню працездатність. Модель підходить для проєктів, де вимоги до ПЗ чітко визначені і не змінюються в процесі розробки, а також, де потрібно забезпечити високу якість і надійність ПЗ. Однак вона вимагає добре спланованого графіку робіт і ресурсів.

Feature Driven Development (FDD) – це метод розробки програмного забезпечення, який фокусується на створенні конкретних функціональних можливостей у продукті. Проєкт розбивається на набір конкретних функціональних можливостей, кожна з яких є окремим проєктом усередині загального контексту. Кожну функціональність розглядають як окремий проєкт із визначеним часом виконання та ресурсами. Це дає змогу команді зосереджуватися на конкретних завданнях і досягати кращих результатів. FDD акцентує увагу на високій якості кожної функціональності. Кожна функціональна можливість ретельно тестується і перевіряється на відповідність вимогам. Команда розробників бере активну участь у розробці кожної функціональності, що сприяє колективному володінню проєктом. FDD забезпечує чітке фокусування на конкретних функціональностях, що робить розробку більш керованою і сприяє створенню високоякісних продуктів. Також ітеративний характер FDD дає змогу швидко адаптуватися до мінливих вимог і умов ринку. Проте проект потребує чіткого визначення функціональностей для успішної реалізації методології, а також недостатньо досвідчена або не згуртована команда може зіткнутися з труднощами в реалізації методології. FDD є підходом, що підходить для проєктів, які потребують ясного визначення функціональностей і готових до активної участі команди в розробці.

Lean Software Development (Ощадлива розробка ПЗ) – це методологія, що розроблена для оптимізації виробничих процесів та управління ресурсами. Цей підхід прагне підвищити ефективність, усуваючи втрати в процесі розробки програмного забезпечення. Це досягається оптимізацією робочих процесів, зменшенням часу очікування між завданнями та покращенням якості продукту. Метод також приділяє увагу мінімізації надлишкових рухів і передавання інформації, що сприяє ефективнішій роботі команди. Перевагами методології є висока ефективність і продуктивність, зумовлені оптимізацією процесів. Також до переваг можна віднести високу якість продукту, оскільки завдяки

прагненню усувати помилки з самого початку, продукти створюються більш високої якості, що задовольняє потреби клієнтів, і гнучкість та адаптивність, адже Lean дає змогу швидко реагувати на зміни у вимогах і ринкових умовах завдяки своїй гнучкості та фокусу на постійному потоці роботи. Саме акцент на усуненні втрат і оптимізації процесів цієї методики робить її цінним методом у швидко мінливому світі розробки ПЗ.

Rapid Application Development (RAD) – методологія швидкої розробки додатків, що передбачає застосування інструментальних засобів візуального моделювання і розробки. Дана методологія спирається на вимоги, але також існує можливість їхніх змін під час розробки системи. Такий підхід дозволяє скоротити витрати і звести час розробки до мінімуму. Метод дозволяє швидко розробляти навіть складні продукти, але має низку особливостей: клієнт має брати активну участь в розробці та постійно спостерігати за результатами, вартість використання такого методу висока, оскільки доведеться наймати великий штат розробників, необхідно чітке розуміння, що має вийти у результаті, щоб кожна команда знала, яку задачу вирішує її модуль. Такий підхід дає можливість швидко протестувати ідею, вивести новий продукт на ринок протягом короткого часу та створювати потужні додатки з великим функціоналом.

Scrum – методологія, що ґрунтується на понятті спринту, протягом якого виконується робота над продуктом. Перед початком кожного спринту проводиться планування, на якому проводиться оцінка вмісту списку завдань із розвитку продукту і формування беклога на спринт, у рамках яких і діє команда. Для спринту завжди існують обмеження по часу, зазвичай від тижня до місяця. Життя продукту таким чином розбита на рівні по тривалості спринти. Серед переваг такої методології можна виділити швидкий зворотній зв'язок від фахівців в різних сферах, швидке додавання нового функціоналу і швидкий запуск продукту з мінімальними функціями. Мінусами є те, що деякі люди, які знають продукт, стають незамінними, так як документація не надається в процесі розробки, також неможливо спланувати точну дату завершення, так як все уточнюється за результатами попереднього спринту, а замовники не завжди можуть зрозуміти суть даної методології і необхідно витратити час на їхнє ознайомлення з нею. Загалом Scrum є гнучкою методологією, яка може бути успішно використана в різних сферах та для різних типів проектів, де важлива швидкість, гнучкість та прозорість в розробці продукту.

Agile Model (гнучка методологія розробки) – це сучасна методологія розробки ПЗ, яка базується на принципах гнучкості, співпраці, взаємодії і постійного вдосконалення. Гнучка методологія розробки не є однією конкретною моделлю, а складається з різних підходів. Вона передбачає поділ проєкту на короткі цикли або спринти, кожен з яких має свою мету, план, виконання і результат. Кожен спринт включає постійну комунікацію між учасниками команди, замовниками і користувачами, а також регулярну перевірку і оцінку продукту. Особливість даного методу полягає в тому, що замовник може одразу спостерігати за змінами у розробці та коригувати дії. Метод гнучкої розробки дуже ефективний, але має недоліки. Через те, що неможливо визначити точні результати та зрозуміти, скільки знадобиться часу на реалізацію ідеї, не можна наперед визначити точну вартість. Якщо проект налаштований на тривалий життєвий цикл, повинен мати адаптивність до змін на ринку, то Agile Model відмінно підходить. Вона дозволяє підлаштовуватися під нові вимоги та постійно розвивати продукт.

Висновки

Отже, розробка якісного програмного продукту є складним завданням, що потребує чіткого планування та ефективного контролю на кожному етапі життєвого циклу проекту. Сучасні IT-компанії активно використовують різноманітні методології розробки програмного забезпечення, кожна з яких має свої унікальні особливості, переваги та недоліки. Немає універсальної “найкращої” методології, оскільки вибір моделі залежить від конкретних потреб проекту, його типу та специфіки. Важливо враховувати ці фактори при виборі методології для конкретного проекту. Таким чином вибір методології розробки програмного забезпечення є важливим кроком у процесі розробки проекту, який варто здійснювати на основі аналізу потреб та умов конкретного проекту.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Популярні життєві цикли розробки ПЗ [Електронний ресурс] – Режим доступу до ресурсу: https://training.qatestlab.com/blog/technical-articles/popular-software-development-life-cycles/

2. Моделі життєвого циклу, принципи і методології розробки програмного забезпечення (ПЗ) [Електронний ресурс] – Режим доступу до ресурсу: https://evergreens.com.ua/ua/articles/software-development-metodologies.html

3. Огляд основних підходів до розробки ПЗ [Електронний ресурс] – Режим доступу до ресурсу:

https://foxminded.ua/pidkhody-do-rozrobky-prohramnoho-zabezpechennia/
4. Ключові методології розробки програмного забезпечення: робота команди зсередини [Електронний ресурс] – Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/metodologija-razrabotki-programmnogo-obespechenija

Лавренюк Арсен Олександрович – студент 2 курсу Вінницького національного технічного університету, факультету інформаційних технологій та комп'ютерної інженерії, групи 1ПІ-22б, Вінниця, e-mail: arsenlavreniuk@gmail.com.

Науковий керівник – ***Бабюк Наталя Петрівна***, кандидат технічних наук, доцент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: babiuk@vntu.edu.ua.

Lavreniuk Arsen Oleksandrovich — second year student of Vinnytsia National Technical University, Faculty of Information Technology and Computer Engineering, Group 1PI-22b, Vinnytsia, e-mail: arsenlavreniuk@gmail.com.

Supervisor: ***Babiuk Natalia Petrivna***, Candidate of Engineering Sciences (Ph. D.), associate Professor at the Department of program engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: babiuk@vntu.edu.ua.