УДК 004.04

В.Д. Шиндирук
Н.П. Бабюк

Аналіз принципів вибору методології розроблення ПЗ

Вінницький національний технічний університет

Анотація

У роботі описано означення та використання програмного забезпечення. Зображено термін «Методологія програмного забезпечення», його причини використання.. Також детально описано процес керування та організацію роботи. Проаналізовано принципи вибору методології розроблення програмного забезпечення за А.Коуберном.

Ключові слова: *програмне забезпечення, проект, розробник, команда, робота, методологія.*

Abstract

The work describes the definition and use of software. The term "Software Methodology" is depicted, the reasons for its use. The management process and organization of work are also described in detail. The principles of software methodology selection according to A. Koburn are analyzed.

Keywords: *software, project, developer, team, work, methodology.*

Вступ

Програмне забезпечення (ПЗ) - це набір програм, що складається з інструкцій, даних та документації, що дозволяє комп'ютеру виконувати різноманітні завдання та функції.

Програмне забезпечення може бути класифіковано за різними критеріями, наприклад:

- за призначенням: системне, прикладне, вбудоване тощо;
- за способом поширення: комерційне, вільне, відкрите тощо;
- за способом розробки: індивідуальне, командне, розподілене тощо;
- за способом взаємодії з користувачем: графічне, текстове тощо.

Програмне забезпечення використовується в різних галузях та сферах, таких як бізнес, наука, медицина, освіта, транспорт, промисловість та інші. Воно може бути розроблений для різних платформ, таких як комп'ютери, мобільні пристрої, сервери тощо.

Методологія програмного забезпечення - це систематичний підхід до розробки програмного забезпечення, який допомагає забезпечити якість, ефективність та надійність розроблюваного програмного продукту.

Основні причини використання методологій програмного забезпечення:

- *Покращення якості програмного забезпечення*: Методології дозволяють забезпечити якість програмного забезпечення завдяки стандартизації процесу розробки, який дозволяє виявити та виправити помилки ще на ранніх стадіях проекту.
- *Ефективність*: Методології розробки програмного забезпечення дозволяють забезпечити оптимальне використання часу та ресурсів при розробці програмного продукту, що сприяє збільшенню продуктивності розроблювачів та зниженню загальної вартості проекту.
- *Надійність*: Методології розробки програмного забезпечення допомагають забезпечити надійність програмного продукту та зменшення ймовірності виникнення помилок в процесі його використання.
- *Спільна робота команди*: Методології розробки програмного забезпечення надають команді стандартизований підхід до розробки, що полегшує спільну роботу розробників та сприяє покращенню комунікації між учасниками проекту.
- *Зручність підтримки та розвитку*: Методології розробки програмного забезпечення надають стандартизований підхід до документації та підтримки програмного продукту, що сприяє зручності розуміння коду та внесенню змін в майбутньому.

Керування та організація роботи

Процес керування вирізняється набором методів та технологій, які відграватимуть ключову роль у розробці програмного забезпечення. Зазвичай, даний вибір втілює керівник проекту, та обрана ним методологія вирізняється досвідом, потім – вимогами замовника.

Алістер Коберн (Alistair Cockburn) – відомий американський спеціаліст з методологій розробки програмного забезпечення та автор книг з даної тематики [1]. Він є одним із розробників Agile-методології та сприяв її поширенню. Крім того, він є автором багатьох інших методологій та моделей, включаючи Crystal, Use Cases та Human Interaction Management.

У роботі Аліестра Коуберна будуються принципи вибору методології розроблення. Всього існує чотири таких принципи [2].

Перший принцип розкриває, що велика за розміром методологія потрібна великим командам розробників. Розмір методології визначається кількістю елементів, потрібних для керування в розробці, а саме: види діяльності, застосовані стандарти, створювані артефати, покказники якості, тощо. На рисунку 1 показано складові методології за А. Коуберном [2]:

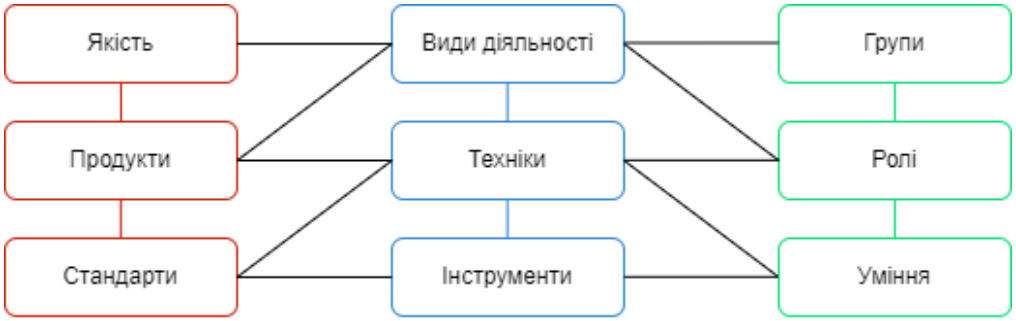

Рисунок 1 – Складові методології за А. Коуберном

1. Ролі – посади, що будуть розміщені в оголошенні про пошук роботи.
2. Уміння – навички, що потрібні для працевлаштування на конкретну посаду.
3. Групи – поділення учасників на групи, відповідно до їх ролей.
4. Інструменти – використовують розробники, відповідно до їх технік та принципів.
5. Техніки – використовують розробники у своїй діяльності.
6. Види діяльності – відгуки, зустрічі, які використовує розробник.
7. Продукти – результат, що передається від розробників іншим групам.
8. Стандарти – це дозволи або заборони на використання в продукті. Існують стандарти позначень, враховуючи мови програмування, стандарти позначень, домовленість проекту.
9. Якість – правки, зауваження, які варто врахувати, для отримання хорошого результату.

Варто врахувати те, що зазвичай проекти з полегшеними методологіями є більш часто успішними, а ніж проекти з формалізованими етапами.

Другий принцип розкриває те, що рівень критичності системи визначає «щільність методології». Рівень критичності системи визначається її важливістю для життєдіяльності людей, компаній, держав тощо. Щільність методології, у свою чергу, визначає рівень деталізації та формалізації процесів, що використовуються при розробці системи.

Для систем з високим рівнем критичності необхідно докладати особливу увагу до деталей та надавати детальну документацію. Це зменшує ризик помилок та підвищує надійність системи. Проте, ці заходи можуть збільшити вартість проекту.

Більш щільні методології є краще систематизованими. А Коуберн виділяє чотири групи, в залежності від рівня втрат [3]:

- дискомфорт – це є найнижча критичність. Приклад: при збоях в системі інформацію можна розрахувати вручну;
- втрата невеликої суми матеріальних ресурсів. Яскравим прикладом може слугувати наступна ситуація: При неправильно нарахованій заробітній платі системою проблему можна вирішити у річному режимі;

- Матеріальні втрати не можна відшкодовувати. Зазвичай, йдеться про збої в національних системах, на які немає можливості повпливати;
- Втрата життя -найвища критичність. Прикадом може слугувати ситуація на атомній станції, а саме збій в системі управління. До такого виду відносять також системи керування польотами, та ситеми, що відносяться до космічної системи.

Отже, розробка систем з високим рівнем критичності вимагає від розробників використання щільної методології та детальної документації. При цьому важливо зберігати баланс між надійністю системи та вартістю її розробки.

Третій принцип – чим більша щільність, тим більшою буде вартість проекту. Для координації робочих груп та надання їм певної інформації потребують часу для якісного сприйняття. А. Коуберн пояснює, що розмір методології залежить від розміру проекту. Для менших команд розробників не потрібна сильно формалізована методологія, оскільки вони можуть легко комунікувати та передавати артефакти. Однак, важливо знати, що на початку проекту неможливо точно визначити його обсяг.

Четвертий принцип стверджує, що найкращим засобом зв'язку між розробниками є безпосереднє спілкування. Це правило не вимагає, щоб всі розробники були в одній кімнаті, але рекомендується, щоб невеликі команди забезпечували максимальну ефективність спілкування. Для зменшення кількості проблем керування важливо ретельно спроектувати проект, врахувати можливі ризики, забезпечити регулярне спілкування між розробниками та виділити незалежні компоненти, які можуть бути передані стороннім розробникам.

Висновки

В результаті, можна зробити такий висновок, що вибір методології розроблення ПЗ повинен бути обґрунтованим та враховувати потреби проекту, його розмір та складність, вимоги до якості ПЗ, процес розробки та можливості комунікації та співпраці між учасниками проекту. А. Коуберн яскраво ілюструє принципи, що значно впливають на результативність майбутнього проекту.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Alistair Cockburn [Електронний ресурс] // Wikipedia. – 2022. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Alistair_Cockburn
2. Cockburn A. Crystal Clear: A Human-Powered Methodology for Small Teams, Alistair Cockburn, October 2004, Addison-Wesley Professional, ISBN 0-201-69947-8.
3. Agile Software Development: The Cooperative Game, Alistair Cockburn, 2nd edition, October 2006, Addison-Wesley Professional, ISBN 0-321-48275-1, ISBN 978-0-321-48275-4

Шиндирук Вікторія Дмитрівна — студентка групи 1ПІ-21б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: svikaa1998@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

Viktoriia Dmytrievna Shyndyruk — student of group 1PI-21b, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: svikaa1998@gmail.com

Academic 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.