УДК 004.658.5.012.1

А. В. Марущак
В. А. Шмалюх
наук. Керівник О. О. Коваленко

ПРАКТИЧНИЙ КЕЙС «ФОРМУВАННЯ КОМАНДИ ТА ЗАСТОСУВАННЯ МЕТОДОЛОГІЇ SCRUM»

Вінницький національний технічний університет

Анотація

У статті наведено вимоги розробки програмного продукту до сучасного розробника. Зокрема проаналізовано актуальність розробки у команді. Зображено переваги використання методології на практиці та на власному досвіді.

Ключові слова: розробник, команда розробників, методологія, Scrum, проект, ефективність роботи в ІТ.

Abstract

This article presents the requirements for software development to a modern developer. In particular, the relevance of team development was analyzed. The advantages of using the methodology in practice and in one's own experience are presented.

Keywords: developer, development team, methodology, Scrum, project, efficiency in IT.

Вступ

Сьогодні світ інформаційних технологій постійно розвивається. Ринок праці для розробників програмного продукту стає все більш вимогливим. Складність проектів стає все важчою. Таким чином виконання проекту середньої складності неможливо для окремого розробника. Команда розробників – це найбільш оптимальне рішення для виконання поставленої задачі. Зокрема, виконання складних комплексних курсових робіт командою дає можливість не тільки реалізувати цікавий проект, а і набути навички ефективної командної роботи, використання сучасних методологій розробки.

Цілі дослідження

Довести думку, що сучасний світ ІТ вимагає програмістів мати навички співпраці у команді. Дослідити переваги та недоліки використання методології у навчанні в процесі виконання дослідження та програмної реалізації за дисципліною «Алгоритми та структури даних» на власному досвіді. Шляхом аналізу визначити основні вимоги, яких потрібно притримуватися командам. Також звернути увагу на вимоги розробки програмних продуктів, особливості реалізації розробки.

Виклад основних результатів дослідження

Необхідно розуміти, що уявити світ без ІТ команд вже неможливо. Складні ІТ-продукти, їх розробка та підтримка здійснюється потужними командами розробників. Ніхто не доручить розробку навіть не дуже складної онлайн платформи всього одній людині. І зрозуміти замовників дуже легко, адже ризики зриву проекту високі, а шукати людей, які візьмуться доводити чужий код, архітектуру та алгоритми зазвичай важко й у результаті потрібно буде заплатити двічі, а можливо й більше.

Для того, щоб сформувати поняття, що ж таке команда і взагалі чи потрібна вона, необхідно розуміти наступне. Не кожна людина вміє чи може виконувати усе, що сьогодні необхідно для команди розробників. Можливо припустити, що навіть існують такі індивідууми, але час виконання дій не зменшується, а проект простоює. Голова одна і рук всього дві. Сучасний світ ІТ є швидкозмінним й ідея, що була придумана учора, сьогодні вважається застарілою, тому кожен зайвий день, що йде на створення програмного продукту призводить до величезних втрат прибутку.

Опираючись на особистий досвід у розробці програмних продуктів, можна стверджувати, що практично будь-який проект необхідно розбивати на модулі, частини для виконання, кожен із яких має відрізнятися складністю, підходом і навіть сферою програмування. Тобто забезпечити кожного члена команди розробки своєю посильною роботою та внести вклад у проект [1,2]. Більш того у команді мають бути люди з різними навичками роботи у сфері ІТ. Для кодувальників, основної робочої потужності проекту, також існує поділ. Існує загальноприйнятий поділ на три категорії або рівні програмістів: джуніор, мідл і сеньйор [3]. Звісно існує обширний поділ і умовні кар'єрні східці можуть відрізнятися, або сприйматися по іншому. Для прикладу, є компанії, які сприймають категорії програмістів, як посади й відповідно визначають заробітну плату працівника. Тобто сеньйор за рівнем знань, як дивно це не звучало, може виконувати роботу джуніора й отримувати відповідну зарплату.

Тому компаніям або командам розробників необхідно поділяти правильно ролі для кожного учасника. Дизайнер не може бути кодувальником, якщо він не всебічно розвинена людина. І це не означає, що усі в команді мають бути рівня сеньйор. Можливо це й зменшує ризики помилок, але прийдеться платити чималі кошти за дії, які з легкістю міг би виконати джуніор або мідл. Окрім цього необхідно розуміти, що команда розробників має мати в своєму складі не лише програмістів. Така команда не зможе організувати увесь процес, щоб завдання було виконано вчасно, або виконання займе більше часу, адже програміст буде займатися роботою архітектора, планувальника чи менеджера.

Часто для того, щоб не винаходити велосипед у галузі ІТ, керівник проекту обирає уже готову, продуману та оптимізовану методологію для постанови робочого процесу. Гарним прикладом є Scrum методологія [4]. Для такої команди передбачається відносно небагато членів команди від 5 до 12. Зрозуміло, що така команда із замовленням середнього, інколи складного, рівня складності справиться з високим рівнем якості виконання. У Scrum команді такі ролі учасників: власник продукту, scrum-майстер, команда розробників. Принцип роботи простий: власник знаходить замовлення чи ідею, оплачує роботу команди та контролює її; Scrum-майстер виконує функції менеджера і слідкує за настроєм та працездатністю команди; команда розробників може містити різного рівня кодувальників, дизайнерів чи ще когось, якщо є потреба. Але головне, що в такій команді немає одного лідера, що тисне на інших і кожен почуває себе повноцінним членом у родині розробників програмного продукту.

Загальні переваги використання Scrum [5]:

- Швидше випуск корисного продукту для користувачів та клієнтів.
- Висока якість.
- Висока продуктивність.
- Менші витрати.
- Більша можливість включати зміни в міру їх виникнення.
- Кращий моральний дух співробітників, адже команда не має начальника.
- Краще задоволення користувачів.
- Вміння виконувати складні проекти, які раніше не можна було виконати.

Загальні недоліки використання Scrum:

- Сутички у команді часто призводить до відтягування роботи через відсутність точної дати завершення.
- Шанси на провал проекту високі, якщо люди не дуже віддані або погано співпрацюють.
- Прийняти структуру Scrum у великих командах є дуже складним завданням.
- Фреймворк може бути успішним лише з досвідченими членами команди.
- Щоденні зустрічі часом можуть бути неефективними, особливо без чітких обмежень в часі, за змістом та організацією. .
- Якщо якийсь член команди покине проект посеред процесів реалізації проекту, це може мати величезний негативний вплив на проект.
- Якість важко підтримувати, поки команда не пройде агресивний процес тестування.

Команда розробників є найбільш привабливим вибором і для розробників, і для замовників. У команді кожен обирає або ж йому призначають посаду, що максимально підходить конкретній людині. Тобто кожен займає посаду і виконує те, що виходить найкраще.

Команда повинна мати планувальника, архітектора, генератора ідей, відповідального за графічну частину та код, програміста-кодувальника, тестувальника. Ці ролі повинні поєднуватись з ролями в команді Scrum.

Для програмістів існує індивідуальний підхід для визначення рівня компетентності. Це не тільки рівень досвіду, а і рівень м'яких навичок управління часом, спілкування із замовником, рівень навичок співпраці в команді тощо. Для формування команди зазвичай краще використовувати людей, обізнаних у різних сферах. Це гарантує появу цікавих ідей для проекту та забезпечує його затребуваність чи привабливість для більшої аудиторії.

Питання вибору учасників для формування команди є гострим і практично основним для власника та скрам-майстра на стадії зародження проекту. Однак, для полегшення цього існують загальні принципи та підходи (методологія) для формування команд різного напряму. Дотримання методології забезпечує високу працездатність команди як в теорії на папері, так і на практиці.

Під час виконання власної курсової роботи було використано Scrum методологію для вирішення поставленої задачі. Це забезпечило гнучку роботи команди. Зокрема перевагами використання такого підходу є наступне.

У Scrum команда розробників є ключовою. Кількість членів команди, як правило, від п’яти до дев’яти людей, що відповідає стандарту максимізації роботи в команді, учасники якої тісно взаємодіють і працюють із девізом «Один за всіх і всі за одного». Їх основною метою є якомога швидше виконати корисні сегменти пріоритетних робіт, які мають ділову цінність. Допомагаючи виконувати будь-яку роботу там, де це потрібно, команда в цілому відповідає за те, що потрібно й виконую усі поставлені завдання. На практиці роботи Scrum програмування заохочується ідея «думати двома головами (буває і більше) краще, ніж однією». Це підвищує продуктивність і зменшує час до завершення проекту з кращою якістю.

Оскільки у команді Scrum усі учасники є рівними, це допомагає підвищити моральний дух команди. Можливість працювати із Scrum Master, який проводить настанови та захищає команду від негативного тиску ззовні, зменшує ризики впливу негативного настрою та допомагає контролювати зайнятість усіх працівників у команді є одною з пріоритетних особливостей цієї методології. Крім того, завдяки такому підходу, як парне програмування, яке також активно використовується, рівень потужності знань програмістів посилюється швидше та сильніше, аніж зазвичай. Завдяки цьому моральний дух і задоволеність роботою також вищі. Процес розробки формується таким чином як зображено на рис. 1.

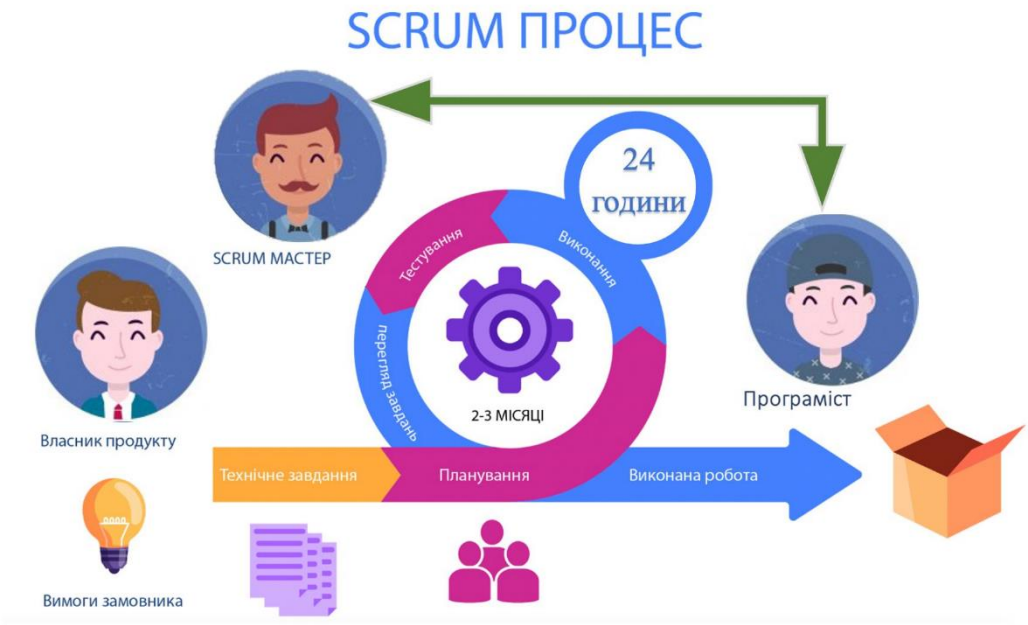

Рис. 1. Процес Scrum роботи у команді

Однак даний підхід до роботи передбачає, що усе є гнучким та мінливим протягом усього життя проекту. Те саме стосується тестування коду, яке виконується постійно, а не лише в кінці після завершення всього проекту.

Зокрема перевагою використання даної методології для малочисельної команди є те, що команда Scrum не передбачає великої кількості ролей для членів команди [6]:

Серед неприємних моментів використання Scrum є лише те, що згідно методології робота виконується командою розробників одночасно, а не послідовно. Тобто програмісти кодують "на льоту" і не чекають, поки не отримають відповіді на всі запитання та стане достовірно зрозумілим, перш ніж вони почнуть програмувати. Це обумовлено рівнем невизначеності вимог до програмного продукту. На прикладі комплексної курсової роботи разом з викладачем, була реалізована методологія Scrum з можливістю проведення спрінтів, підтримки Канбан дошки, аналізу виконаних етапів, процесів запровадження швидких змін з врахуванням вимог до курсової роботи та уявного замовника. Парне програмування дало можливість контролювати роботу кожного з програмістів, а завдання з планування, контролю часу та основних завдань за допомогою канбан-дошки – зрозуміти процеси розробки та моніторингу результатів на кожному її етапі.

Практичний кейс був реалізований командою під керівництвом викладача за таким планом:

Визначення ролей команди: власник та скрам-майстер – викладач; команда – студенти.

Визначення платформи та інструментарію – середовище програмування, канбан-дошка; аналіз результатів; парне програмування.

Звіти за результатами у вигляді спринтів на практичних заняттях; приклади щоденних скрам-спринтів для набуття досвіду.

Отримано якісний програмний продукт та опис документації у вигляді пояснювальної записки з досвідом розробки алгоритмів, реалізації програмного коду та набуття навичок роботи в команді й за методологією Scrum.

Висновки

Отже, команда є невід'ємною частиною для роботи у світі ІТ. Розподіл ролей є досить важливим процесом, до якого потрібно підходити обгрунтовано. Використання методології оптимізує роботу та забезпечує оптимальне використання ресурсва розробників. Команду потрібно формувати відповідно до визначеної методології. Цікавим є формування гнучкої гібрідное методології – наприклад методологія Scrum як система гнучких методів та інструментарію управління проектом та парне програмування як один з потужних методів колективного програмування.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Why is Teamwork Important? [Електронний ресурс] – Режим доступу до ресурсу: https://the-happy-manager.com/articles/why-is-teamwork-important/.
2. Setapp Team. Why Teamwork Is Important [Електронний ресурс] / Setapp Team. – 2020. – Режим доступу до ресурсу: https://setapp.com/business/why-teamwork-is-important.
3. Software Engineer Qualification Levels: Junior, Middle, and Senior [Електронний ресурс] // altexsoft. – 2018. – Режим доступу до ресурсу: https://www.altexsoft.com/blog/business/software-engineer-qualification-levels-junior-middle-and-senior/.
4. CLAIRE DRUMOND. Что такое Scrum? [Електронний ресурс] / CLAIRE DRUMOND // atlassian – Режим доступу до ресурсу: https://www.atlassian.com/ru/agile/scrum.
5. Advantages and Disadvantages of Scrum [Електронний ресурс] // Chandana. – 2020. – Режим доступу до ресурсу: https://www.simplilearn.com/scrum-project-management-article.
6. Granko O. Scrum чи не-Scrum - який підхід обрати? [Електронний ресурс] / Olga Granko // Блог "PM Рішення". – 2017. – Режим доступу до ресурсу: https://worksection.com/ua/blog/scrum.html.

Марущак Артем Володимирович — студент другого курсу групи 3ПІ-19б, ФІТКІ, Вінницький національний технічний університет, Вінниця, e-mail: maryskhak@gmail.com

Шмалюх Владислав Анатолійович — студент другого курсу групи 3ПІ-19б, ФІТКІ, Вінницький національний технічний університет, Вінниця, e-mail: zskat02@gmail.com

Науковий керівник: ***Коваленко Олена Олексіївна*** — к.т.н, доцент кафедри програмної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: ok@vntu.edu.ua

Marushchak Artem Volodymyrovych - second-year student of group 3PI-19b, FITKI, Vinnytsia National Technical University, Vinnytsia, e-mail: maryskhak@gmail.com

Shmaliukh Vladyslav Anatoliyovych - second-year student of group 3PI-19b, FITKI, Vinnytsia National Technical University, Vinnytsia, e-mail: zskat02@gmail.com

Scientific adviser: ***Kovalenko Olena Oleksiivna*** - Ph.D., Associate Professor of Software Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: ok@vntu.edu.ua