УДК 510.635:004.891(045) doi:10.20998/2413-4295.2021.01.06

АНАЛІЗ ГНУЧКИХ МЕТОДОЛОГІЙ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РЕАЛІЗАЦІЇ У КОМАНДНИХ ПРОЄКТАХ

А. І. ВАВІЛЕНКОВА

кафедра комп'ютеризованих систем управління ФККПІ, Національний авіаційний університет, Київ, УКРАЇНА
e-mail: vavilenkovaa@gmail.com

АНОТАЦІЯ *У матеріалах статті проаналізовано основні моделі життєвого циклу програмного забезпечення, що лежать в основі гнучких методологій розробки програмного забезпечення для виявлення особливостей їх застосування при реалізації командних проєктів. Виявлено, що використання класичних моделей життєвого циклу, зокрема, каскадної, спіральної, інкрементної,, V-подібної та ітеративної, не являється ефективним при реалізації всіх сучасних аспектів розробки програмного забезпечення на основі використання нових інформаційних технологій. Проведено порівняльний аналіз найбільш популярних гнучких методологій розробки програмного забезпечення Agile, Scrum, Kanban, RUP, DSDM, RAD за такими показниками, як модель життєвого циклу, кількість ітерацій, мета створення проєкту та типи проєктів для реалізації, пріоритети, можливість взаємодії із замовником, адаптація до змін. Виявлені переваги та недоліки гнучких методологій розробки програмного забезпечення дали змогу виокремити методологію MSF, що базується на узгодженні каскадної, спіральної та ітеративної моделей життєвого циклу розробки програмного забезпечення та дає змогу обрати шаблон Scrum, як найбільш вдалий для реалізації та демонстрації роботи у навчальних командних проєктах. MSF використовує у роботі підхід, що передбачає поетапне створення робочого продукту з певною функціональністю, яка відображає вимоги до кінцевого продукту на даному етапі. Запропоновано використовувати рішення компанії Microsoft на основі Visual Studio та Team Foundation Server для централізованого керування елементами командного проєкту, використання інструментів візуального моделювання архітектури, можливості управління якістю коду та отримання всіма учасниками команди актуальної інформації про стан проєкту. Це відбувається завдяки тому, що модель MSF об'єднує у собі п'ять основних моделей: модель команди; модель процесу (послідовність дій, яка необхідна для побудови командного проєкту); дисципліну управління проєктами (передбачає комплексне планування всіх етапів командного проєкту, управління бюджетом, ресурсами, витратами, підготовки графіків); дисципліни управління ризиками та дисципліни управління готовністю (оцінювання знань членів командного проєкту для подальшого розподілу ролей у команді). Продемонстровано приклад створення навчального командного проєкту на основі використання шаблону гнучкої методології Scrum у середовищі Visual Studio на базі Team Foundation Server.*
Ключові слова: *гнучка методологія; командний проєкт; життєвий цикл; програмне забезпечення; аналіз; Scrum-команда; Visual Studio*

ANALYSIS OF FLEXIBLE METHODOLOGIES OF SOFTWARE DEVELOPMENT FOR IMPLEMENTATION IN TEAM PROJECTS

A. VAVILENKOVA

Department of Computerized Control Systems, National Aviation University, Kyiv, UKRAINE

ABSTRACT *The materials of the article analyze the main models of the software life cycle, which underlie the flexible methodologies of software development to identify the features of its application during the implementation of team projects. It was found that the use of classical life cycle models, in particular, cascade, spiral, incremental, V-shaped and iterative, is not effective in implementing all modern aspects of software development based on the use of new information technologies. It was conducted the comparative analysis of the most popular flexible methodologies of software development Agile, Scrum, Kanban, RUP, DSDM, RAD on such indicators as life cycle model, number of iterations, purpose of project creation and types of projects for implementation, priorities, possibility of interaction with the customer, adaptation to change. The advantages and disadvantages of flexible software development methodologies made it possible to single out the MSF methodology, which is based on the harmonization of cascading, spiral and iterative models of the software development lifecycle and allows you to choose the Scrum template as the most successful for implementing and demonstrating work in team development projects. MSF uses an approach that involves the gradual creation of a working product with some functionality that reflects the requirements for the final product at this stage. It is proposed to use Microsoft solutions based on Visual Studio and Team Foundation Server for centralized management of team project elements, use of visual architecture modeling tools, code quality management capabilities and obtaining all project team members up-to-date information on project status. This is because the MSF model combines five following main models: team model; process model (sequence of actions required to build a team project); discipline of project management (provides comprehensive planning of all stages of the team project, budget management, resources, costs, scheduling); risk management disciplines and readiness management disciplines (assessment of team project members' knowledge for further distribution of team roles). The article demonstrates an example of creating a training team project based on the use of the Scrum flexible methodology template in Visual Studio based on Team Foundation Server.*
Keywords*: flexible methodology; team project; life cycle; software; analysis; Scrum-command; Visual Studio*

Вступ

На сьогоднішній день малі чи великі підприємства, які створюються у нашій країні та за її межами, починають свою роботу з поступової реалізації проєкту, розробленого ще до початку їх функціонування. Це передбачає виконання деякої

ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

послідовності дій, тобто роботу згідно з завчасно розробленим алгоритмом для досягнення певної мети, працюючи у команді. Якщо ще декілька років тому сучасні компанії використовували для планування всього життєвого циклу своєї діяльності програмний продукт MS Project [1], то сьогодні автоматизація роботи командних проектів відбувається завдяки різноманітним програмним продуктам, реалізованим на основі гнучких методологій розробки [2]. Гнучкі методології представляють собою системи, що визначають порядок виконання задач, методи оцінювання та контролювання, орієнтовані на використання ітеративного підходу для реалізації певного набору вимог [3,4].

Різноманіття концепцій та велика кількість різних додатків для реалізації командних проєтків, зокрема для розробки програмного забезпечення, викликає сумбур у загальному розумінні концепцій, моделей та алгоритмів функціонування команд розробників. Тому актуальною задачею досі залишається вибір методологій для своєчасного отримання якісних продуктів розробки.

Мета роботи

Аналіз програмних продуктів, що запропоновані сьогодні на ринку інформаційних технологій, показав, що всі програмні середовища для реалізації командних проєктів реалізують типові для розробки програмного забезпечення етапи життєвого циклу, виходячи з спрямованості проєкту, його бюджету та строків реалізації, при цьому по-різному забезпечуючи взаємозв'язки всередині циклу розробки продукту. Стикнувшись з проблемою широкого вибору програмного забезпечення для реалізації командних проєктів та відсутністю чіткої регламентації дій щодо створення навчальних проєктів, виникла необхідність у розробці чіткого алгоритму дій. Тому метою даної статті є аналіз моделей життєвого циклу розробки програмних продуктів, що у свою чергу лежать в основі гнучких методологій розробки програмного забезпечення для виявлення особливостей їх застосування при реалізації командних проєктів.

Виклад основного матеріалу

Перш ніж застосовувати програмну реалізацію командного проєкту, необхідно розібратися з метою його створення, моделлю та алгоритмом функціонування, адже тільки ретельне планування кожного з кроків дасть змогу отримати якісний результат.

Метою реалізації командного проєкту в області інформаційних технологій являється створення якісного програмного продукту в чітко визначені терміни. Для цього важливо правильно обрати модель життєвого циклу розробки програмного забезпечення, тобто структуру, що міститиме процеси дій та завдання, які вирішуватимуться в ході розробки, використання та супроводження програмного продукту.

Виділяють такі основні моделі життєвого циклу розробки програмного забезпечення.

Каскадна модель – передбачає лінійну послідовність виконання етапів, у ній кожен етап розробки, що відповідає певній стадії життєвого циклу програмного забезпечення, продовжує попередній [5-7]. Тобто для того, щоб перейти на наступний етап, потрібно повністю завершити попередній, проєкт реалізується за наступними кроками: визначення вимог, аналіз, проєктування програмного забезпечення, розробка, тестування та впровадження (рис. 1).

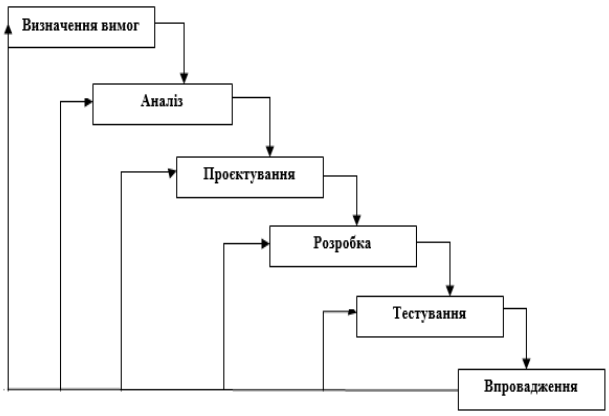

Рис. 1 – Етапи каскадної моделі життєвого циклу розробки програмного забезпечення

Каскадна модель застосовується, коли ще до початку розробки можна досить точно та повно сформулювати всі вимоги до проєкту. Така модель підходить для великих та складних проєктів, які передбачають повний контроль ризиків та які планується переносити з однієї платформи на іншу.

Спіральна модель – розробка програмного забезпечення у вигляді послідовності версій, всі етапи життєвого циклу проходять витками [5-6,8], на кожному з яких відбувається визначення вимог, аналіз, проєктування програмного забезпечення, розробка, тестування та впровадження (рис. 2). В результаті на кожному витку спіралі створюється чергова версія продукту, конкретизуються вимоги проєкту, визначається якість та плануються роботи наступного витка.

Спіральна модель дозволяє швидше показати замовнику робочий продукт, допускає зміну вимог при розробці, забезпечує гнучкість, помилки та слабкі місця виявляються та виправляються на кожній ітерації. Така модель характерна для розробки не типового, принципово нового програмного забезпечення, коли на початку розробки немає однозначного, чіткого бачення кінцевого продукту.

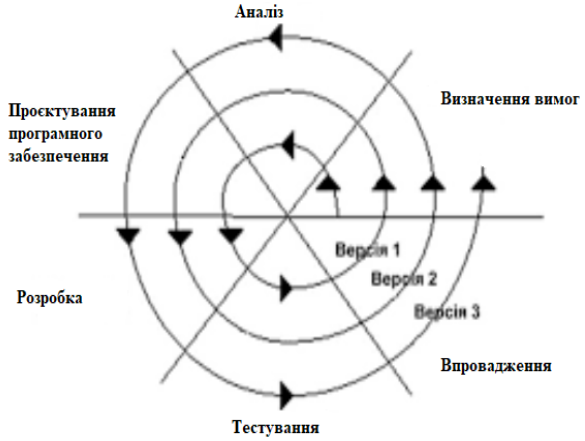

Рис. 2 – Графічна інтерпретація спіральної моделі життєвого циклу розробки програмного забезпечення

Інкрементна модель – передбачає розробку програмного продукту з лінійною послідовністю стадій, проте в декілька інкрементів (версій) із запланованим покращенням продукту протягом всього життєвого циклу, дає змогу розширювати можливості, добудову модулів та реалізацію додаткових функцій [9-10]. Поки одна версія експлуатується, наступна, враховуючи недоліки попередньої, тільки планується або розробляється.

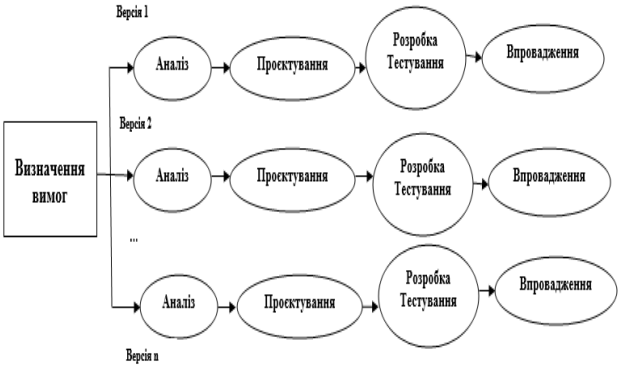

Рис. 3 – Концепція інкрементної моделі життєвого циклу розробки програмного забезпечення

За інкрементною моделлю розробляються проєкти, у яких точне технічне завдання прописане уже на початку, а програмний продукт потрібно створити швидко та регулярно отримувати зворотний зв'язок від користувача.

V-подібна модель – це модель, у якій замовник одночасно з командою програмістів складають вимоги до системи та описують, як будуть її тестувати на етапах аналізу, проєктування програмного забезпечення, розробки та впровадження.

Горизонтальні лінії між завданнями розробки та завданнями тестування показують, як результати кожної з фаз розробки впливають на розвиток системи тестування на кожній з фаз тестування [5-6,11].

V-подібна модель застосовується при розробці проєктів, у яких важлива надійність, а ціна помилки дуже висока, тому кількість помилок в архітектурі зводиться до мінімуму (рис. 4).

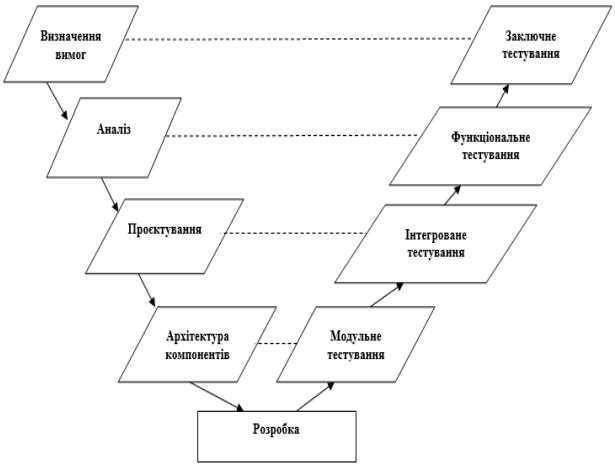

Рис. 4 – V-подібна модель життєвого циклу розробки програмного забезпечення

Ітеративна модель – передбачає розбиття проєкту на частини (ітерації) та проходження етапів життєвого циклу на кожному з них. Кожен етап являється закінченим сам по собі, а сукупність етапів формує кінцевий результат [5-6,12].

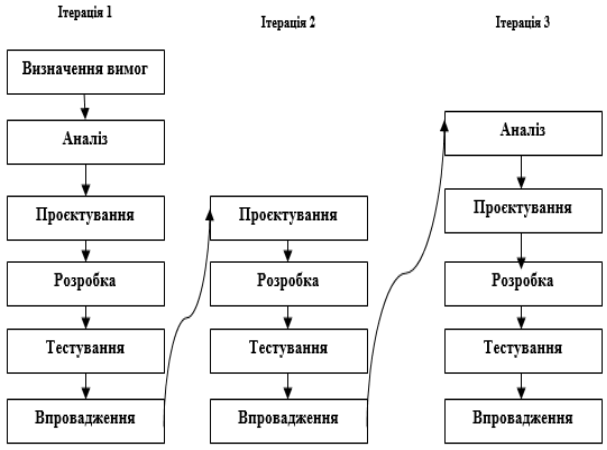

Рис. 5 – Графічна інтерпретація ітеративної моделі життєвого циклу розробки програмного забезпечення

Ітеративна модель застосовується для великих проєктів з невизначеними вимогами або для задач з інноваційним підходом.

Алгоритм функціонування командного проєкту, тобто концепція, за якою буде працювати команда залежить від обраної гнучкої методології, яких у наш час на ринку IT-технологій досить багато.

ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

Проведемо порівняльну характеристику найпопулярніших гнучких методологій, за якими розробляються та функціонують командні проєкти (табл. 1, табл.2) [2,13-16].

Із порівняльної характеристики розглянутих шести гнучких методологій розробки програмного забезпечення видно, що всі вони налаштовані на отримання кінцевого продукту з мінімальними ризиками, високою швидкістю розробки, простотою розуміння для замовника та можливістю взаємодії з ним. Методології відрізняються моделями життєвого циклу розробки програмного забезпечення, а відповідно кількістю ітерацій, на які розбивається процес розробки, та послідовністю їх виконання.

Таблиця 1 – Порівняльна характеристика гнучких методологій Agile, Scrum та RUP

Параметр	**Agile**	**Scrum**	**RUP**
Модель ЖЦ	Ітеративна Спіральна Інкремент.	Ітеративна Спіральна Інкремент.	Ітеративна
Візуалізація ЖЦ	+	+	+
Типи проєктів	Великі з можливістю внесення змін	Навчальні та середні з можливістю внесення змін	Великі
Наявність ітерацій	Спринт	Спринт	Етап
Мета – робочий продукт	+	+	+
Щоденні спринти	+	+	-
Люди та взаємодія на першому місці	+	+	+/-
Процеси та інструменти на першому місці	-	-	+/-
Можливість взаємодії з замовником	+	+	+
Готовність до змін	+	+	+
Простота	+	+	+/-
Адаптація до змін	+	+	+
Швидкість розробки	+	+	+
Мінімізація ризиків	+	+	+
Кількість основних етапів	6	6	4

Таблиця 2 – Порівняльна характеристика гнучких методологій Kanban, DSDM та RAD

Параметр	**Kanban**	**DSDM**	**RAD**
Модель ЖЦ	Ітеративна Спіральна Інкремент.	Ітеративна Інкремент-на	Спіральна
Візуалізація ЖЦ	+	-	-
Наявність ітерацій	Спринт	Стадія	Фаза
Типи проєктів	Великі з можливістю внесення змін	Невеликі	Невеликі
Мета – робочий продукт	+	+	+
Щоденні спринти	+	-	-
Люди та взаємодія на першому місці	+	+/-	+/-
Простота	+	+/-	+/-
Процеси та інструменти на першому місці	-	+/-	+/-
Можливість взаємодії з замовником	+	+	+
Готовність до змін	+	+	+
Простота	+	+	+
Адаптація до змін	+	+	+
Швидкість розробки	+	+	+
Мінімізація ризиків	+	+	+
Кількість основних етапів	6	3 (4)	4

Методологія, розроблена компанією Microsoft, MSF (Microsoft Solutions Framework) базується на узгодженні каскадної, спіральної та ітеративної моделей життєвого циклу розробки програмного забезпечення. MSF використовує у роботі підхід, що передбачає поетапне створення робочого продукту з певною функціональністю, яка відображає вимоги до кінцевого продукту на даному етапі [17]. Реалізація п’яти фаз, тобто послідовного розв’язання задач, та віх, що служать точками переходу від однієї фази до іншої (рис. 6), визначають зміни в поточних задачах рольових кластерів проєктної команди [18].

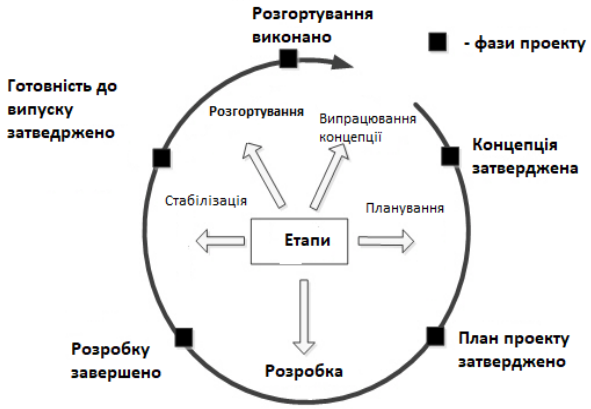

Рис. 6 – Візуалізація процесів методології Microsoft Solutions Framework

Також методологія MSF дозволяє реалізувати концепції роботи Agile, Scrum та CMMI (рис. 7).

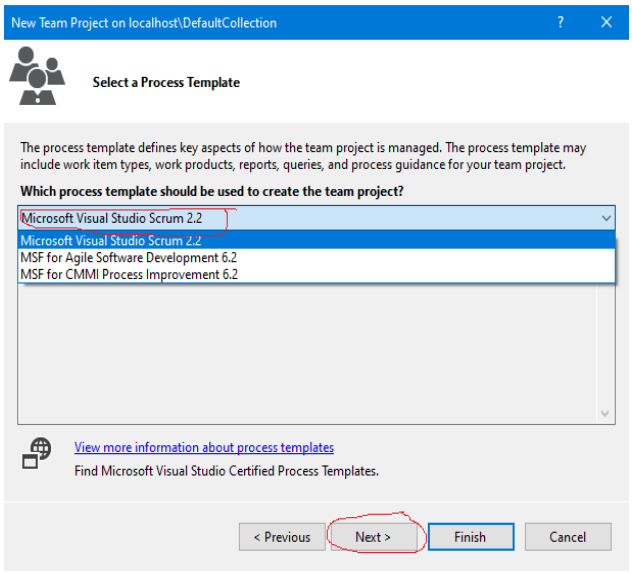

Рис. 7 – Шаблони реалізації у MSF

Модель MSF об'єднує у собі п'ять основних моделей:

- модель команди – передбачає організацію команди за шести основними напрямками: управління продуктом, управління програмою, розробка, тестування, навчання користувачів, супровід; у рамках шаблона Scrum-команди виділяють три основні ролі для реалізації цих шести напрямків – це Product Owner, Scrum Master та Scrum Team;
- модель процесу – тобто послідовність дій, яка необхідна для побудови командного проєкту, представляє собою проходження таких фаз: фаза аналізу, на якій формується представлення про програмний продукт на даному витку спіралі, та, як результат, – віха - затверджене уявлення про проєкт; фаза планування – відбувається прогнозування ризиків, оцінка графіка робіт та необхідних ресурсів, результат – віха – затверджений план проєкту; фаза розробки – визначає проміжні етапи випуску продукту, кожен із яких включає повний цикл тестування, відлагодження та внесення змін, результат – віха – завершення розробки; фаза стабілізації – перевірка функціональності в реальних умовах та повномасштабне тестування, віха – реліз продукту; фаза розгортування – впровадження на місцях завершується, впроваджене рішення стабілізоване, віха – впровадження завершено;
- дисципліна управління проєктами – передбачає комплексне планування всіх етапів командного проєкту, управління бюджетом, ресурсами, витратами, підготовка графіків;
- дисципліна управління ризиками – процес відслідковування та мінімізації ризиків, який обов'язково включає: визначення ризиків, їх аналіз та розстановку пріоритетів, план та графік уникнення, відслідковування ризиків та звітність, їх контроль та знання.
- управління готовністю – передбачає оцінювання знань членів командного проєкту для подальшого розподілу ролей у команді.

Обговорення результатів

Для реалізації концепції управління командним проєктом на всіх етапах його життя компанія Microsoft пропонує рішення на основі Visual Studio та Team Foundation Server (нова версія Azure DevOps) [18], що дає змогу централізовано керувати всіма елементами командного проєкту, використовувати інструменти візуального моделювання архітектури, можливості управління якістю коду, а також надає всім учасникам команди можливість отримувати інформацію про стан проєкту та спрощує комунікацію.

Так, після створення та налаштування параметрів командного проєкту, який можна переглядати як через Team Explorer з Visual Studio, так і через Web-доступ, прописуються основні вимоги замовника, що формуються у список Product Backlog (рис. 8) з можливістю додавання в нього задач шляхом взаємодії з MS Excel та MS Project. Таким чином програмно реалізується стадія визначення вимог до командного проєкту.

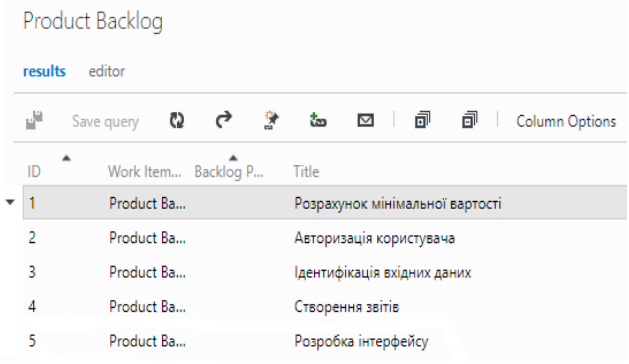

Рис. 8 – Product Backlog командного проекту

ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

Візуальне моделювання архітектури командного проєкту, тобто стадія проєктування, реалізована у Visual Studio за допомогою побудови UML-діаграм, на яких деталізуються задачі проєкту із Product Backlog, визначаються ролі, задаються зв'язки (рис. 9).

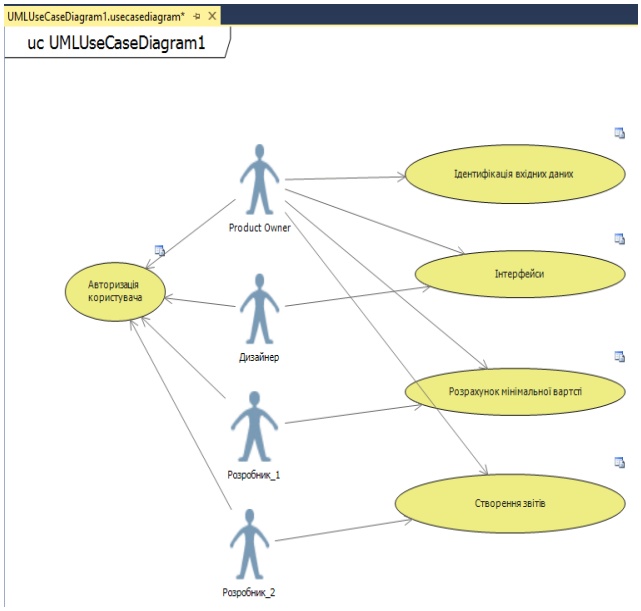

Рис. 9 – Візуальне моделювання структури проекту

Реалізація всього проєкту розбивається на спринти (рис. 10), для кожного з яких задається свій набір задач та підзадач для виконання, що формуються у Sprint Backlog, де учасникам командного проєкту призначаються певні задачі, а план виконання спринту можна простежити на спеціальній дошці Board (рис. 11).

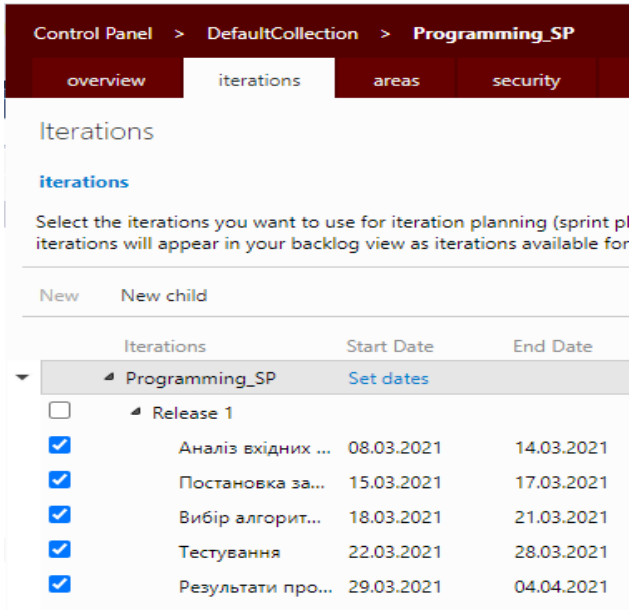

Рис. 10 – Спринти комадного проекту

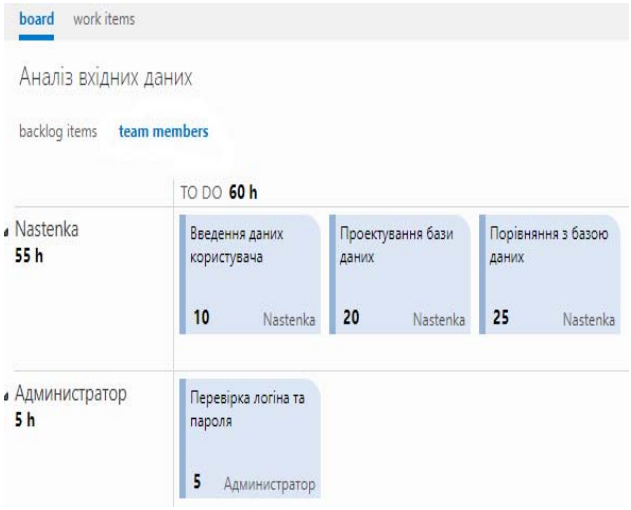

Рис. 11 – Інструмент Board для перегляду Sprint Backlog

В процесі роботи, тобто під час реалізації спринту, поставлені задачі поступово виконуються, тобто згоряють, що можна простежити на діаграмі згоряння, як для конкретного спринту, такі для проєкту в цілому.

З продемонстрованих прикладів видно, що інструменти Visual Studio реалізують концепцію гнучкої методології Scrum, застосовуючи всі характерні їй поняття у роботі та дають змогу візуалізувати як архітектуру, так і стадії реалізації командного проєкту.

Висновки

Основна мета використання гнучких методологій розробки програмного забезпечення – це створення командою прогнозованого процесу розробки програмного продукту необхідної якості [19]. Проведений порівняльний аналіз найбільш популярних гнучких методологій розробки програмного забезпечення Agile, Scrum, Kanban, RUP, DSDM, RAD за такими показниками, як модель життєвого циклу, кількість ітерацій, мета створення проєкту та типи проєктів для реалізації, пріоритети, можливість взаємодії із замовником, адаптація до змін, дав змогу виокремити методологію MSF, що базується на узгодженні каскадної, спіральної та ітеративної моделей життєвого циклу розробки програмного забезпечення та дає змогу обрати шаблон Scrum, як найбільш вдалий для реалізації та демонстрації роботи у навчальних командних проєктах На прикладах продемонстровано поетапне створення робочого продукту в програмному середовищі Visual Studio на базі Team Foundation Server. Це дає змогу оцінити зручність та можливість візуалізації результатів роботи командного проєкту на кожній з його стадій виконання.

Список літератури

1. Lewis C., Chatfield C., Johnson T. *Microsoft Project 2019 step by step*. Pearson Education: Microsoft Press, 2019. 482 p.
2. Вольфсон Б. Л. *Гибкое управление проектами и продуктами*. СпБ.: Питер Пресс, 2016. 600 с.
3. Мартин Р. К., Ньюкирк Дж. В., Косс Р. С. *Быстрая разработка программ. Принципы, примеры, практика*. М.: Вильямс, 2004. 752с.
4. Madachy R. J. *Software Process Dynamics*. Wiley-IEEE Press, 2008. 632 p.
5. Selby R. W. Software Engineering: Barry W. *Boehm's Lifetime Contributions to Software Development, Management, and Research*. Wiley-IEEE Computer Society Pr, 2007. 832 p.
6. Брауде Э. *Технология разработки программного обеспечения*. СпБ.: Питер, 2004. 655 с.
7. Сидоров К. Модели разрабоки программного обеспечения. URL: https://nortex.pro/blog/razrabotka-po-modeli-razrabotki-po (дата звернення: 25.02.2021).
8. Матс И. Модели жизненного цикла программного обеспечения. URL: https://habr.com/ru/post/111674/ (дата звернення: 03.03.2021).
9. Aleem S., Capretz L. F., Ahme F. Game development software engineering process life cycle: a systematic review. *J. Softw. Eng. Res. Dev.* 2016. Vol. 4. 6. doi:10.1186/s40411-016-0032-7.
10. Tahera, K., Wynn, D. C., Earl, C. et al. Testing in the incremental design and development of complex products. *Res. Eng. Design.* 2019. Vol. 30. P. 291–316. doi: 10.1007/s00163-018-0295-6.
11. Моделі життєвого циклу, принципи і методології розробки програмного забезпечення (ПЗ). URL: https://evergreens.com.ua/ua/articles/software-development-metodologies.html (дата звернення: 11.03.2021).
12. SDLS Iterative Model. URL: https://www.tutorialspoint.com/sdlc/sdlc_iterative_model.htm (дата звернення: 11.03.2021).
13. Putri D., Rozilawati. R., Zulkefli M. Team Formation for Agile Software Development: A Review. *J. Adv. Science, Eng. Inf. Tech.* 2020. Vol. 10. 2. P. 555-561. doi: 10.18517/ijaseit.10.2.10191.
14. Alaidaros H., Omar M, Romli R. An improved model of Agile Kanban method: verification process through experts' review. *International Journal of Agile Systems and Management*. 2020. Vol. 13. No. 4. P. 390-416. doi: 10.1504/IJASM.2020.112337.
15. Satherland D. Scrum. *The Art of Doing Twice the Work in Half the Time*. Random House, 2015. 256 p.
16. Kruchten P. *The Rational United Process: An Introduction.* Second Edition. M.: Viliams, 2002. 240 p.
17. Microsoft Solutions Framework. Basic Principles. URL: https://newline.tech/microsoft-solutions-framework-basic-principles/ (дата звернення: 25.01.2021).
18. Giotis T. C. *How to deliver successful IT projects using MSF team model and MSF process model. PMI® Global Congress 2007— EMEA*, Budapest, Hungary. Newtown Square, PA: Project Management Institute. URL: https://www.pmi.org/learning/library/deliver-project-microsoft-solutions-framework-7413 (дата звернення: 11.03.2021).
19. Вавіленкова А. І. *Комп'ютеризовані системи управління*: навч. посіб. К.: НАУ, 2020. 140 с.

References (transliterated)

1. Lewis C., Chatfield C., Johnson T. *Microsoft Project 2019 step by step*. Pearson Education: Microsoft Press, 2019. 482 p.
2. Volfson B. L. *Gibkoe upravlenie proectami i productami*. SpB., Piter Press, 2016. 600 p.
3. Martin R. K., Newkirk J. V., Koss R. S. *Bustraya razrabotka program*. M., Vilyams, 2004. 752 p.
4. Madachy R. J. *Software Process Dynamics*. Wiley-IEEE Press, 2008. 632 p.
5. Selby R. W. Software Engineering: Barry W. *Boehm's Lifetime Contributions to Software Development, Management, and Research*. Wiley-IEEE Computer Society Pr, 2007. 832 p.
6. Braude E. *Technologia razrabotki programmnogo obespecheniya*. SpB., Piter, 2004. 655 p.
7. Sidorov K. *Modeli razrabotki programmnogo obespecheniya*. Available at: https://nortex.pro/blog/razrabotka-po-modeli-razrabotki-po (accessed 25.02.2021).
8. Matzch I. *Modeli zhiznennogo tsikla programmnogo obespecheniya*. Available at: https://habr.com/ru/post/111674/ (accessed 03.03.2021).
9. Aleem S., Capretz L. F., Ahme F. Game development software engineering process life cycle: a systematic review. *J. Softw. Eng. Res. Dev.*, 2016, Vol. 4, 6, doi: 10.1186/s40411-016-0032-7.
10. Tahera, K., Wynn, D. C., Earl, C. et al. Testing in the incremental design and development of complex products. *Res. Eng. Design,* 2019, Vol. 30, pp. 291–316, doi: 10.1007/s00163-018-0295-6.
11. Modeli zhittevogo tsiklu, prinzcipi i metodologii rozrobku programnogo zabezpechennya (PZ). Available at: https://ever greens.com.ua/ua/articles/software-development-metodolo gies.html (accessed: 11.03.2021).
12. SDLS Iterative Model. Available at: https://www.tutorialspoint.com/sdlc/sdlc_iterative_model.htm (accessed: 11.03.2021).
13. Putri D., Rozilawati. R., Zulkefli M. Team Formation for Agile Software Development: A Review. *J. Adv. Science, Eng. Inf. Tech.*, 2020, Vol. 10, 2. pp. 555-561, doi: 10.18517/ijaseit.10.2.10191.
14. Alaidaros H., Omar M, Romli R. An improved model of Agile Kanban method: verification process through experts' review. *International Journal of Agile Systems and Management*, 2020, Vol. 13, 4, pp. 390-416, doi: 10.1504/IJASM.2020.112337.
15. Satherland D. Scrum. *The Art of Doing Twice the Work in Half the Time*. Random House, 2015. 256 p.
16. Kruchten P. *The Rational United Process: An Introduction*. Second Edition. M., Vilyams, 2002. 240 p.
17. Microsoft Solutions Framework. Basic Principles. Available at: https://newline.tech/microsoft-solutions-framework-basic-principles/ (accessed: 25.01.2021).
18. Giotis T. C. *How to deliver successful IT projects using MSF team model and MSF process model. PMI® Global Congress 2007— EMEA*, Budapest, Hungary. Newtown Square, PA, Project Management Institute. Available at: https://www.pmi.org/learning/library/deliver-project-microsoft-solutions-framework-7413 (accessed 11.03.2021).
19. Vavilenkova A. I. *Komputerizovani system upravlinnya*. K. NAU, 2020. 140 p.

ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

Відомості про авторів (About authors)

Вавіленкова Анастасія Ігорівна – доктор технічних наук, доцент, Національний авіаційний університет, професор кафедри комп'ютеризованих систем управління ФККПІ; м. Київ, Україна; ORCID: 0000-0002-9630-4951; e-mail: vavilenkovaa@gmail.com.

Anastasiia Vavilenkova – DSc, Docent, Professor, Department of Computerized Control Systems, Kyiv, Ukraine; ORCID: 0000-0002-9630-4951; e-mail: vavilenkovaa@gmail.com.

Будь ласка, посилайтесь на цю статтю наступним чином:

Вавіленкова А. І. Аналіз гнучких методологій розробки програмного забезпечення для реалізації у командних проєктах. *Вісник Національного технічного університету «ХПІ». Серія: Нові рішення в сучасних технологіях*. – Харків: НТУ «ХПІ». 2021. № 1 (7). С. 39-46. doi:10.20998/2413-4295.2021.01.06.

Please cite this article as:

Vavilenkova A. Analysis of flexible methodologies of software development for implementation in team projects. *Bulletin of the National Technical University "KhPI". Series: New solutions in modern technology*. – Kharkiv: NTU "KhPI", 2021, no. 1 (7), pp. 39-46, doi:10.20998/2413-4295.2021.01.06.

Пожалуйста, ссылайтесь на эту статью следующим образом:

Вавиленкова А. И. Анализ гибких методологий разработки программного обеспечения для реализации в командных проектах. *Вестник Национального технического университета «ХПИ». Серия: Новые решения в современных технологиях*. – Харьков: НТУ «ХПИ». 2021. № 1 (7). С. 39-46. doi:10.20998/2413-4295.2021.01.06.

АННОТАЦИЯ *В материалах статьи проведен анализ основных моделей жизненного цикла программного обеспечения, которые лежат в основе гибких методологий разработки программного обеспечения для обнаружения особенностей их использования при реализации командных проектов. Отмечено, что использование классических моделей жизненного цикла, в частности каскадной, спиральной, инкрементной, V-подобной и итеративной, не является эффективным во время реализации всех современных аспектов разработки программного обеспечения на основании использования новых информационных технологий. Проведен сравнительный анализ наиболее популярных гибких методологий разработки программного обеспечения Agile, Scrum, Kanban, RUP, DSDM, RAD за такими показателями, как модель жизненного цикла, количество итераций, цель создания проекта и виды проектов для реализации, приоритеты, возможность взаимодействия с заказчиком, адаптация к изменениям. Полученные преимущества и недостатки гибких методологий разработки программного обеспечения дали возможность выделить методологию MSF, которая построена на согласовании каскадной, спиральной и итерационной моделях жизненного цикла разработки программного обеспечения и дает возможность выбрать шаблон Scrum, как наиболее удачный для реализации и демонстрации работы в учебных командных проектах. MSF использует в работе подход, который предполагает поэтапное создание рабочего продукта с определенной функциональностью, которая отображает требования к конечному продукту на данном этапе. Предложено использовать решение компании Microsoft на базе Visual Studio и Team Foundation Server для централизованного управления элементами командного проекта, использования инструментов визуального моделирования архитектуры, возможности управления качеством кода и получения всеми участниками команды актуальной информации о состоянии проекта. Это осуществляется благодаря тому, что модель MSF объединяет в себе пять основных моделей: модель команды; модель процесса (последовательность действий, которая необходима для построения командного проекта), дисциплины управления проектами (предполагает комплексное планирование всех этапов командного проекта, управление бюджетом, ресурсами, затратами, подготовкой графиков), дисциплины управления рисками и дисциплины управления готовностью (оценивание знаний членов командного проекта для дальнейшего распределения ролей в команде). Продемонстрирован пример создания программного проекта на базе использования шаблона гибкой методологии Scrum в среде Visual Studio на основании Team Foundation Server.*

Ключевые слова: *гибкая методология; командный проект; жизненный цикл; программное обеспечение; анализ; Scrum-команда; Visual Studio*

Надійшла (received) 26.02.2021