DOI: https://doi.org/10.36910/6775-2524-0560-2021-42-18
УДК 510.635:004.891(045)
Вавіленкова Анастасія Ігорівна, д.т.н., професор,
https://orcid.org/0000-0002-9630-4951
Національний авіаційний університет, м. Київ, Україна.

ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ЕТАПУ ПРОЄКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ SCRUM-КОМАНДОЮ У ПРОГРАМНОМУ СЕРЕДОВИЩІ VISUAL STUDIO

Вавіленкова А. І. Особливості реалізації етапу проєктування програмного продукту Scrum-командою у програмному середовищі Visual Studio. Проаналізовано та описано роль членів командного проєкту за методологією Scrum. Зосереджено увагу на важливості проєктування, як одного з основних етапів життєвого циклу розробки програмного забезпечення. Висвітлено основні моменти процесу реалізації етапу проєктування в програмному середовищі Visual Studio на базі Team Foundation Server з використанням шаблону Scrum. Це дає змогу відтворити всі характерні для Scrum-команди артефакти та візуалізувати процеси вирішення поставлених задач, а за допомогою UML-діаграм отримати необхідні моделі для проєктування роботи команди.
Ключові слова: командний проєкт, проєктування, Product Backlog, інтерфейс, UML-діаграма, Visual Studio.

Вавиленкова А. И. Особенности реализации этапа проектирования программного продукта Scrum-командой в программной среде Visual Studio. Проанализирована и описана роль членов командного проекта согласно методологии Scrum. Внимание сосредоточено на важности проектирования, как одного из основных этапов жизненного цикла разработки программного обеспечения. Высветлены основные моменты процесса реализации этапа проектирования в программной среде Visual Studio на основании Team Foundation Server с использованием шаблона Scrum. Это дает возможность воссоздать все характерные для Scrum-команды артефакты и визуализировать процессы решения поставленных задач, а с помощью UML-диаграмм получить необходимые модели для проектирования работы команды.
Ключевые слова: командный проект, проектирование, Product Backlog, интерфейс UML-диаграмма, Visual Studio.

Vavilenkova A. Features of realization of a designing stage of a software product by Scrum-command in the Visual Studio software environment. It was analyzed and described the role of team project members according to the Scrum methodology. The focus is on the importance of design as one of the main stages of the software development life cycle. The article highlights the main points of the process of implementation of the design stage in the Visual Studio software environment based on Team Foundation Server using the Scrum template. This makes it possible to reproduce all the artifacts characteristic of the Scrum-team and visualize the processes of solving problems, and with the help of UML-diagrams to obtain the necessary models for designing the work of the team.
Keywords: team project, design, Product Backlog, interface, UML-diagram, Visual Studio.

Постановка проблеми.

Широке використання у повсякденному житті людини різноманітних видів інформаційних технологій внесло в область розробки програмного забезпечення кардинальні зміни щодо процесів, концепцій, алгоритмів, та методологій розробки програмних продуктів. Не виключенням стала сфера командної організації роботи на підприємствах. Тому на сьогоднішній день у сфері ІТ актуальною є задача ефективного планування та проєктування роботи команди з метою створення якісних програмних продуктів [1].

Вирішення цієї задачі передбачає не лише безпосередню розробку, але й ретельну роботу над плануванням всіх робочих процесів. Саме для таких потреб в останні роки використовується гнучка методологія Scrum [2], що допомагає командам проводити спільну роботу, представляє собою структуру, у якій основний акцент переноситься на якість процесів, а створення продукту поділяється на декілька частин, що дає можливість робити висновки з досвіду попередніх етапів, засвоювати принципи самоорганізації та постійно удосконалюватися [3].

При всій активності просування на ІТ-ринку України нових концепцій розробки програмного забезпечення, вся практична сторона реалізації, як правило, розкривається тільки при роботі над реальними проєктами у компаніях або завдяки досвіду розробників на різноманітних чатах або у блогах.

Аналіз останніх досліджень і публікацій.

Будь-який командний проєкт представляє собою колекцію робочих елементів, коду, текстів та додатків, які охоплюють всі артефакти, які використовуються у життєвому циклі програмного забезпечення [4]. Програмно командний проєкт можна побудувати на основі шаблону, що складається із набору XML-файлів [5]. Таким шаблоном і слугує обрана методологія, за якою функціонуватиме командний проєкт.

Методологія Scrum передбачає те, що програмний продукт розробляється за декілька ітерацій, спринтів, з фіксованою тривалістю, розбиваючи таким чином великі проєкти на невеликі задачі [6].

© Вавіленкова А. І.

Scrum – це також кістяк процесу, який включає не тільки набір методів, але і попередньо визначених ролей [7]. А саме, при формуванні Scrum-команди виділяють три основні ролі: власник продукту – Product Owner, курівник проєкту – Scrum-master та командна розробників (рис. 1).

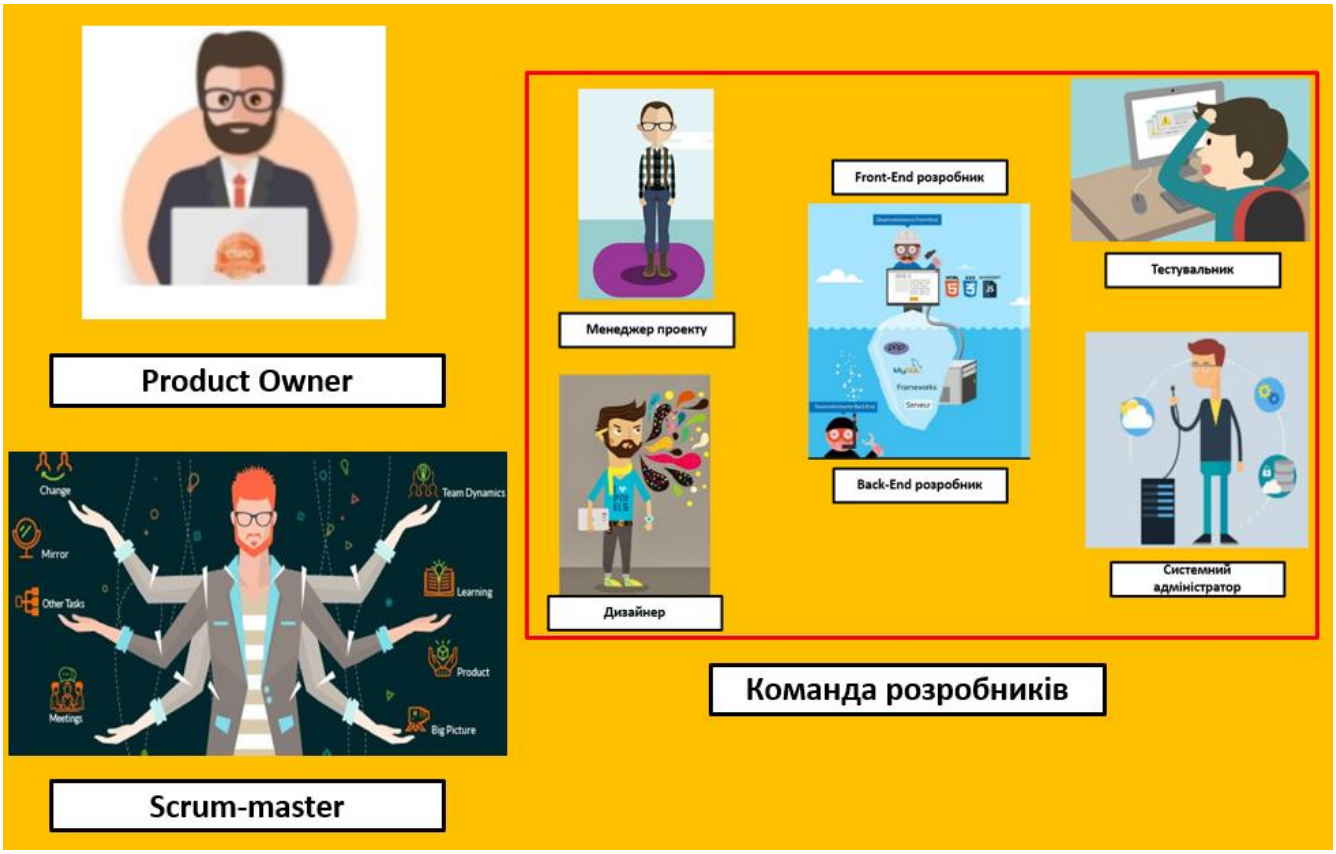


Рис. 1 – Структура Scrum-команди

Product Owner – це ланка команди, яка пов'язує замовника, формуючи список задач Product Backlog, з командою розробників. Product Owner повинен розуміти, як буде виглядати кінцевий продукт, забезпечує зрозумілість Product Backlog та вимог до продукту, над якими потрібно буде працювати команді. Задача Product Owner оптимізувати процес розробки так, щоб робота команди мала максимальну цінність, про що можна знайти масу літератури, проте без наочних прикладів програмної реалізації [8-9].

Scrum-master або керівник проєкту – забезпечує роботу команди в рамках методології Scrum, тобто забезпечує правильну комунікацію в команді, слідкує за процесом розробки та зміно статусів задач у спринті, організовує планування та ретроспективу [10-11]. Scrum-master виявляє методи ефективного управління Product Backlog, транслює бачення та цілі проєкту команді розробників, застосовує та практикує гнучкі методи розробки та управління.

Команда розробників – створює програмний продукт та надає певні версії програмного забезпечення в кінці кожного спринту. Як правило, включає програмістів, дизайнерів та тестувальників, які слідкують за власною результативністю, бере на себе рішення щодо розробки та дизайну, оцінює елементи Product Backlog, несе відповідальність за результат перед Product Owner [12].

Під час пошуку інформаційних джерел для реалізації командних проєктів можна побачити багато літератури на тему контролю версій, наприклад, у роботах [13-15], а також посилання на програмне забезпечення для здійснення управління проєктами [16-17]. Питання ж програмного планування та реалізації командних проєктів для створення програмного забезпечення розкривається не чітко, супроводжуючись або теорією створення команд, або демонстрацією коду для налаштувань уже існуючих версій програмного продукту.

Мета дослідження.

© Вавіленкова А. І.

Під час аналізу літературних джерел на тему реалізації командних проєктів для розробки програмного забезпечення та постановки курсу дисципліни «Командна розробка програмних систем» було виявлено недостатньо інформації для формулювання чіткого розуміння алгоритму дій щодо програмної реалізації різноманітних типів команд та методологій, на основі яких вони формулюються. Тому метою даного дослідження є структуризація матеріалу щодо основних етапів створення програмного забезпечення на основі обраного шаблону роботи команди проєкту та виявлення особливостей реалізації етапу проєктування, як одного з найважливіших для отримання якісного програмного продукту.

Виклад основного матеріалу й обґрунтування отриманих результатів дослідження.

Концепція програмної реалізації командної роботи Microsoft Solutions Framework дає змогу організовувати такі етапи розробки програмного забезпечення [18]:

- формування вимог до програмного продукту (створення Product Backlog) – виконує Product Owner;
- аналіз вимог та розбиття їх на окремі підзадачі (Task), що будуть реалізовуватися в конкретних спринтах – здійснюється усіма членами Scrum-команди;
- архітектурне проєктування – візуалізація основних об'єктів та класів, що будуть задіяні у процесі розробки програмного забезпечення, формулюється специфікація рішення та розробляється його архітектура (UML-діаграми та інтерфейси в Power Point) – для більшої ефективності в реалізації цього етапу також приймає участь вся команда;
- розробка – етап, на якому визначаються деталі фізичного дизайну, оцінюється необхідних час та ресурси для реалізації кожного елементу дизайну, отримані з попереднього етапу розробки, розробляється код та контролюється розробка елементів рішення – здійснює командна розробників під керівництвом Scrum-master.
- тестування – процес перевірки відповідності вимог до продукту та реально реалізованої функціональності, що здійснюється шляхом спостереження за роботою продукту у штучно створених ситуаціях та обмеженому наборі тестів, внаслідок чого виявляються всі дефекти програмного забезпечення – здійснюють тестувальники або вся команда розробників;
- впровадження – заключний етап, на якому здійснюється впровадження продукту на території замовника, організовується супровід продукту, інструкції щодо використання та подальшого функціонування – здійснює Product Owner разом з командою розробників.

Як видно з перерахованих вище функціональних особливостей кожного з етапів, неможливо перейти до безпосередньої розробки програмного продукту без попереднього його проєктування.

Для здійснення архітектурного проєктування за методологією MSF на базі шаблону Microsoft Visual Studio Scrum 2.2 використовуються інструменти візуального проєктування на основі мови UML, що дають змогу візуалізувати архітектурні аспекти програмного продукту, створити моделі структури та поведінки, розробити шаблони для проєктування та шаблони майбутнього інтерфейсу програми.

Так, для моделювання функціональних вимог до програмного продукту використовують схему варіантів використання, яка представляє вимоги користувача до системи (тобто є візуальним відображенням створеного на попередньому етапі Backlog Product). Для цього до командного проєкту *Programming_SP* додається рішення нове рішення *ModelingProjectSolution_SP* та обирається шаблон діаграми варіантів *UML Use Case Diagram*.

Функціональність системи, що проектується, задана вимогами користувача, які визначаються та формулюються на попередньому етапі розробки програмного забезпечення, наприклад, для створення програмного продукту розрахунку мінімальної вартості обладнання виділено такі основні задачі, задані у робочому елементі «*Work Items*» Product Backlog:

- авторизація користувача;
- ідентифікація вхідних даних;
- розрахунок мінімальної вартості;
- створення звітів;
- розробка інтерфейсів.

Сформульовані задачі командного проєкту на діаграмі варіантів задаються елементами «*Use Case*» з можливістю задання назви, опису та властивостей, виконавці задач відображаються за допомогою елементів «*Actor*», а зв'язки між задачами та акторами позначаються через елементи «*Association*» (рис. 2).

© Вавіленкова А. І.

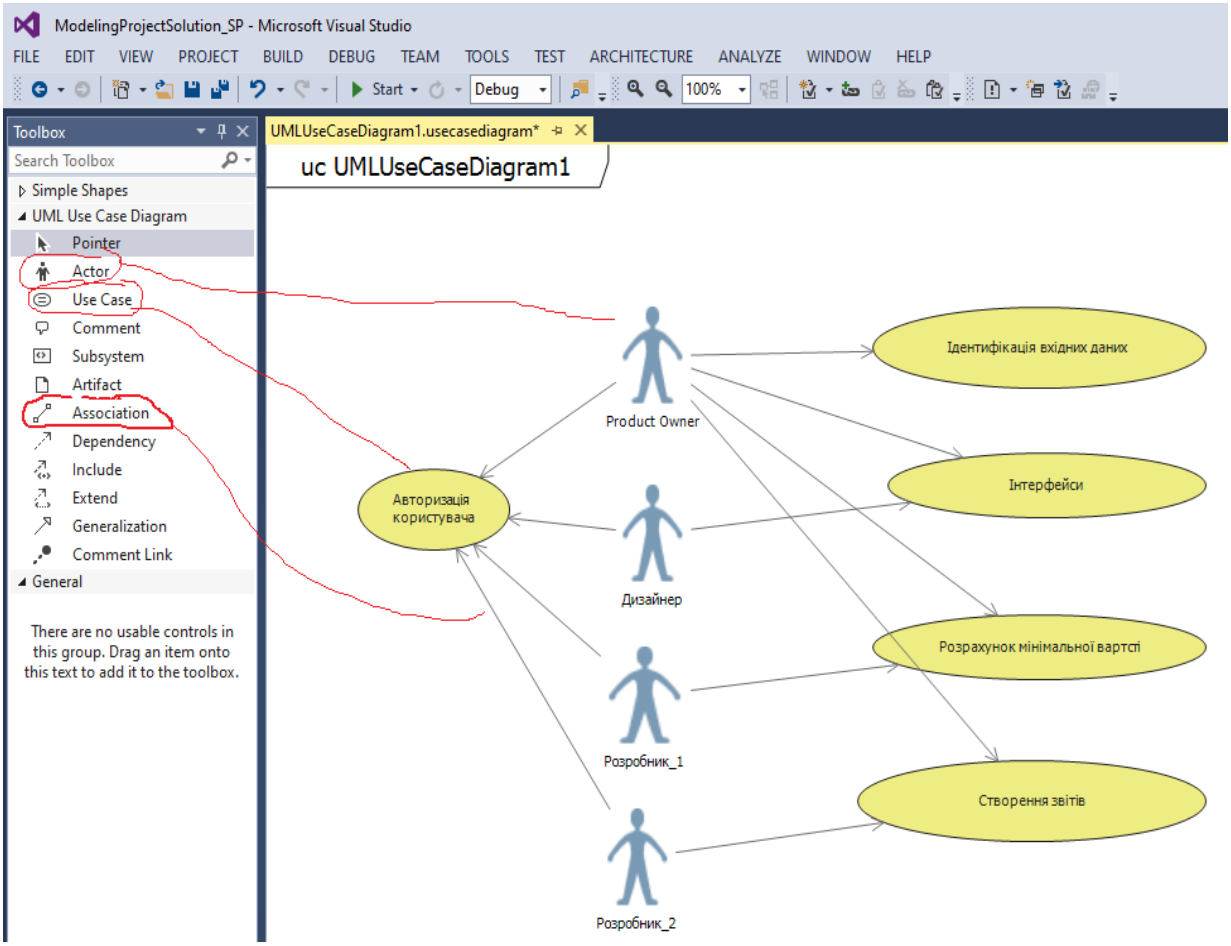
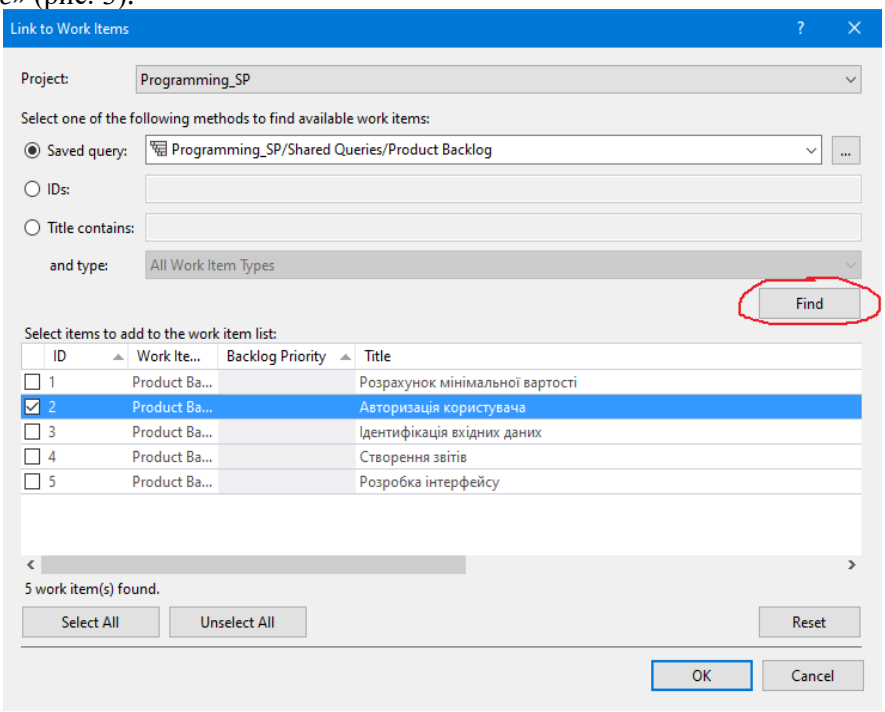


Рис. 2 – Схема варіантів використання на основі шаблону *UML Use Case Diagram*

Варіанти використання потрібно зв'язати з Product Backlog Items для відслідковування взаємозв'язку робочих елементів проекту та схем архітектурного моделювання, використовуючи «*Link to Work Item*». Внаслідок такої операції кожній задачі з Product Backlog буде поставлено у відповідність елемент «*Use Case*» (рис. 3).



© Вавіленкова А. І.

Рис. 3 – Встановлення зв'язків між вимогами до проєкту та елементами діаграми варіантів

Моделювання сутностей програмної системи відбувається за допомогою UML-діаграми класів, що створюється на початкових етапах проектування програмного продукту та дозволяє здійснювати попередній аналіз. Для цього в рішенні *ModelingProjectSolution_SP* обирається шаблон діаграми *UML* Class Diagram.

Нехай в результаті обговорення з замовником Product Owner виявив, що для проектування програмного продукту потрібні наступні класи: вхідні дані, інтерфейс, вартість, звіт з роботи проекту, звіт з вартістю обладнання, звіт з визначення мінімальної вартості. Для їх моделювання будуть використовуватися елементи «Class» та різні види зв'язків між ними (рис. 4).

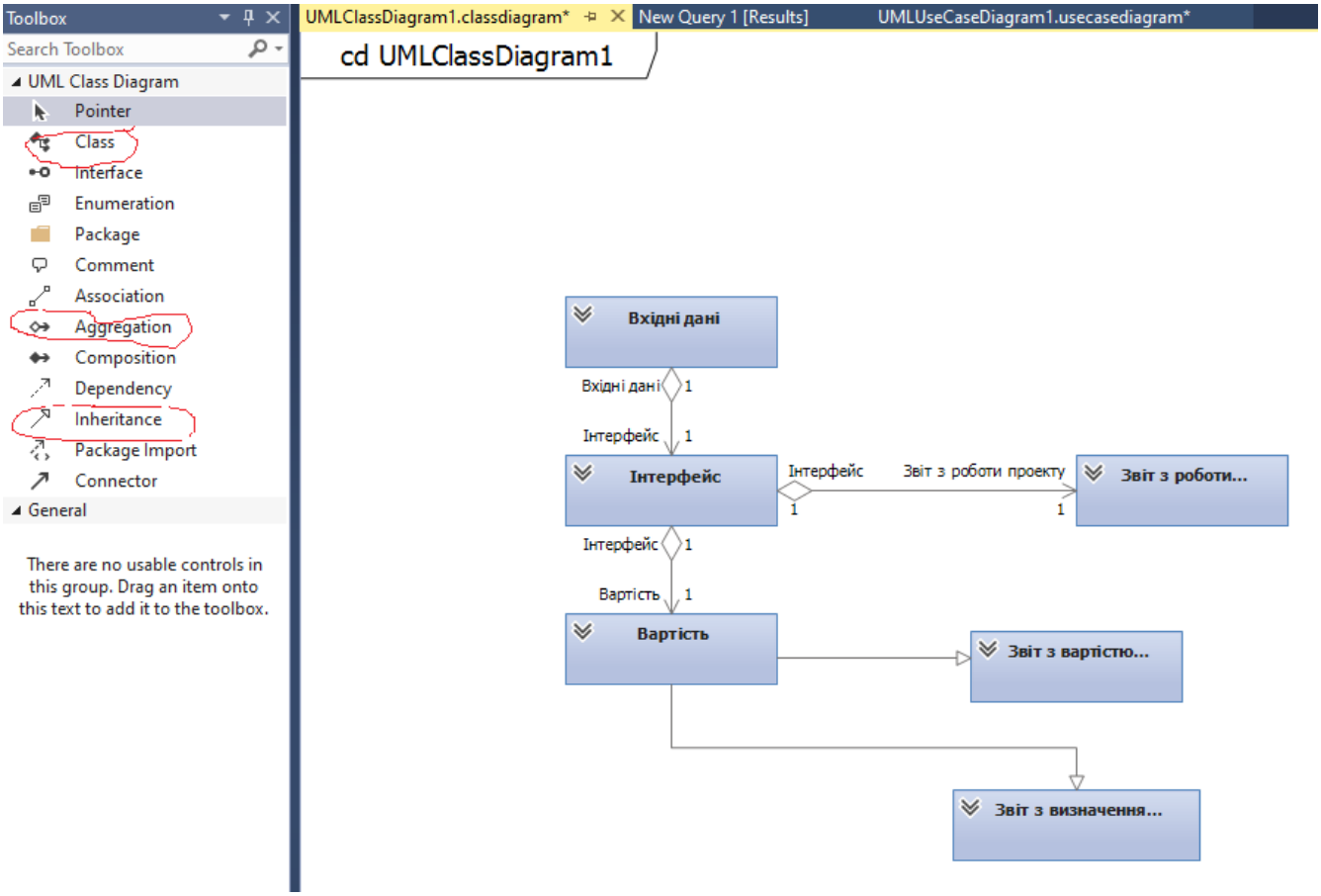


Рис. 4 – Моделювання сутностей програмного продукту за допомогою діаграми класів

Після закінчення моделювання елементи рішень зберігаються в базі даних Team Foundation Server.

Ще одним необхідним моментом проєктування програмного забезпечення являється програмний інтерфейс. Зручним інструментом макетування користувацького інтерфейсу та реалізації вимог користувача є PowerPoint, який інтегровано з Visual Studio. Для задач із Product Backlog здійснюють розкадровку представлення екранних форм або веб-сторінок додатку на вкладці Storyboards.

Для зв'язку створеного макету програмного інтерфейсу із задачами командного проєкту існує декілька особливостей:

- створювати презентацію PowerPoint на вкладці STORYBOARDING за допомогою форм Storyboard Shapes (рис. 5);
- зберегти створену презентацію у мережевій папці для можливості доступу до всіх членів проєкту або зберігати презентацію на сайті SharePoint, опублікувавши слайди, надавши презентації загальний доступ через меню File;
- зв'язувати створену та опубліковану презентацію з командним проєктом в Team Foundation Server через посилання «Storyboard Links», обравши необхідний проєкт та задачу, до якої потрібно прив'язати інтерфейс (рис. 6);
- перевірити наявність розкадровки із вікна перегляду задачі Product Backlog (рис. 7).

© Вавіленкова А. І.

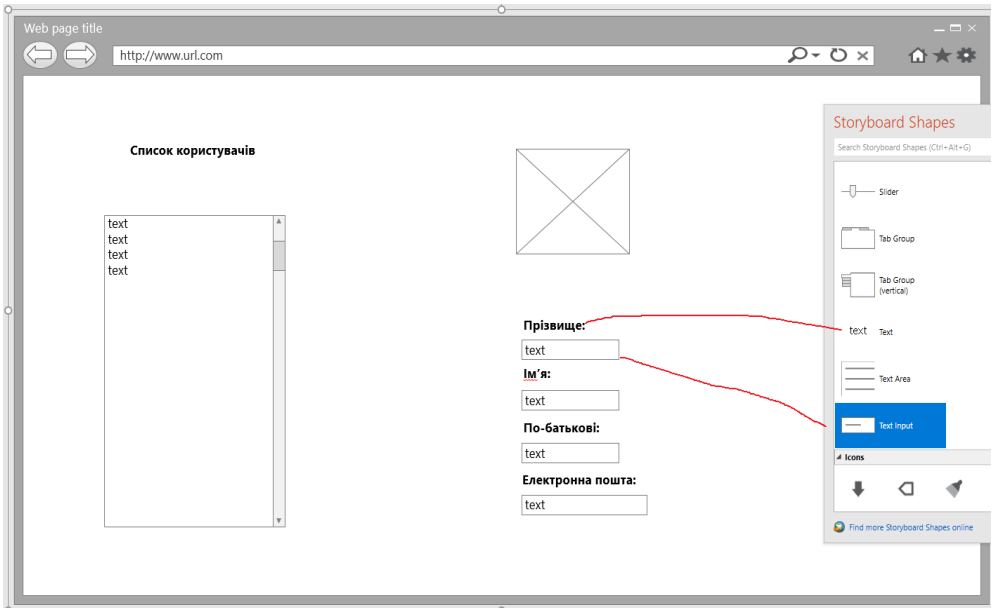


Рис. 5 – Моделювання програмного інтерфейсу за допомогою форм Storyboard Shapes

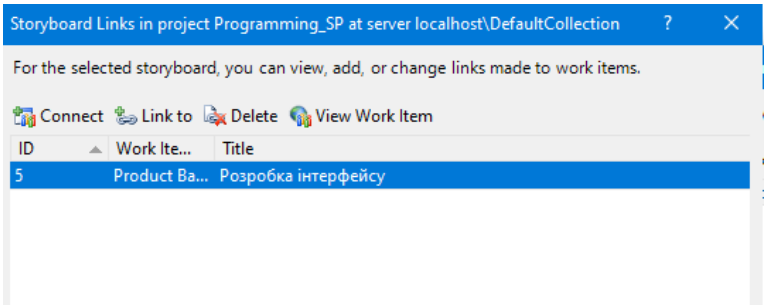


Рис. 6 – Зв'язок моделі програмного інтерфейсу з задачею Product Backlog

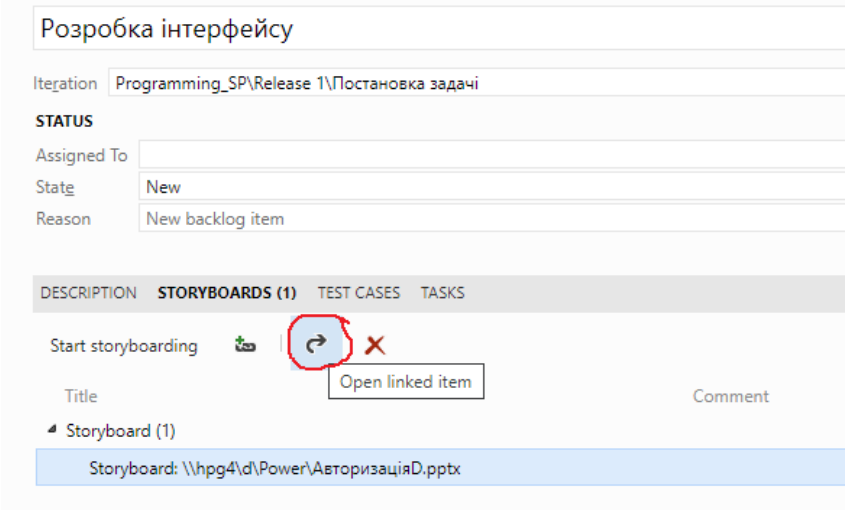


Рис. 7 – Перевірка наявності макету інтерфейсу для задачі першого спринту «Розробка інтерфейсу»

Висновки та перспективи подальшого дослідження.

У матеріалах статті проаналізовано роль членів командного проєкту за методологією Scrum та запропоновано використовувати для реалізації всіх етапів розробки програмного продукту середовище Visual Studio на базі Team Foundation Server. Етап проєктування виділено, як один з основних у життєвому циклі розробки програмного забезпечення, без реалізації якого не можна переходити до безпосередньої розробки. На прикладі програмно створеного командного проєкту для розрахунку

© Вавіленкова А. І.

мінімальної вартості обладнання продемонстровано специфіку створення діаграм варіантів використання та діаграм класів, виявлено особливості макетування програмного інтерфейсу з використанням PowerPoint, який інтегровано з Visual Studio.

Матеріали даного дослідження можна використовувати для програмної реалізації програмних проєктів, як базу для виконання подальших етапів розробки та тестування у життєвому циклі програмного забезпечення.

Список бібліографічного опису

1. Andre B. Bondi Foundations of Software and System Performance Engineering: Process, Performance Modeling, Requirements, Testing, Scalability, and Practice (2015), Addison-Wesley Professional, 412 p.
2. Скрам – це ефективне управління проєктами (2021), https://brainrain.com.ua/uk/scrum-upravlinnya-proektom/ (дата звернення: 09.03.2021).
3. Брауде Э. (2004) Технология разработки программного обеспечения. СпБ.: Питер, 655 с.
4. Резник С., Бьйорк А., де ла Маза М. (2012) Scrum с Team Foundation Server 2010. Профессиональный подход, 416 с.
5. Долженко А. И. (2016) Технологии командной разработки программного обеспечения информационных систем: курс лекций, 3-е изд., М.: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Эр Медиа, 300 с.
6. Putri D., Rozilawati. R., Zulkefli M. (2020) Team Formation for Agile Software Development: A Review. *J Adv Science, Eng Inf Tech*, Vol. 10 no. 2. pp. 555-561. https://doi.org/10.18517/ijaseit.10.2.10191.
7. Satherland D. (2015) Scrum. The Art of Doing Twice the Work in Half the Time. Random House, 256 p.
8. Akif R., Majeed H. Issues and Challenges in Scrum (2012) International Journal of Scientific & Engineering Research, Vo.3 Issue 8
9. Nidagundi P., L.Novickis (2017) Introducing Lean Canvas Model Adaptation in the Scrum Software Testing, Procedia Computer Science, Vo 104, pp. 97 -103. https://doi.org/10.1016/j.procs.2017.01.078.
10. Morampudi N., R. Gaurav (2013) Evaluation Strengths and Weaknesses of Agile Scrum Framework Using Knowledge Management, International Journal of Computer Applications, Vo 65, No 23.
11. Кен Швабер (2019) Скрам. Гибкое управление продуктом и бизнес, Альпіна Паблішер Україна, 236 с.
12. Breno G. Tavares, E, Carlos da Silva, Adler D. de Souza (2017) Risk Management in Scrum Projects: A Bibliometric Study, Journal of Communications Software, Vo. 13, No. 1.
13. Laster B. Professional Git (2016), 480 p.
14. Santacroce F., Olsson A., Voss R., Narebski J. (2016) Git: Mastering Version Control, Packt Publishing, 839 p.
15. Tsitoara M. (2020) Beginning Git and GitHub: A Comprehensive Guide to Version Control, Project Management, and Teamwork for the New Developer, Apress, 289 p. https://doi.org/10.1007/978-1-4842-5313-7.
16. Lewis C., Chatfield C., Johnson T. (2019) Microsoft Project 2019 step by step. Pearson Education: Microsoft Press, 482 p.
17. Вавіленкова А., Литвиненко О., Жолдаков О. (2015) Управління проектами інформатизації: навч. посіб., НАУ, 220 с.
18. Microsoft Solutions Framework. Basic Principles. URL: https://newline.tech/microsoft-solutions-framework-basic-principles/ (дата звернення: 25.01.2021).

References

1. Andre B. Bondi Foundations of Software and System Performance Engineering: Process, Performance Modeling, Requirements, Testing, Scalability, and Practice (2015), Addison-Wesley Professional, 412 p.
2. Scrum is Effective Project Management (2021), https://brainrain.com.ua/uk/scrum-upravlinnya-proektom/ (accessed: 09.03.2021).
3. Braude E. (2004) Technology of Software Development, SpB.: Piter, 655 p.
4. Reznik S., Byork A., Maza M. (2012) Scrum with Team Foundation Server 2010. Professional Approach, 416 p.
5. Dolzhenko A. (2016) Technologies for team development of information systems software, 300 p.
6. Putri D., Rozilawati. R., Zulkefli M. (2020) Team Formation for Agile Software Development: A Review. *J Adv Science, Eng Inf Tech*, Vol. 10 no. 2. pp. 555-561. https://doi.org/10.18517/ijaseit.10.2.10191.
7. Satherland D. (2015) Scrum. The Art of Doing Twice the Work in Half the Time. Random House, 256 p.
8. Akif R., Majeed H. Issues and Challenges in Scrum (2012) International Journal of Scientific & Engineering Research, Vo.3 Issue 8.
9. Nidagundi P., L.Novickis (2017) Introducing Lean Canvas Model Adaptation in the Scrum Software Testing, Procedia Computer Science, Vo 104, pp. 97 -103. https://doi.org/10.1016/j.procs.2017.01.078.
10. Morampudi N., R. Gaurav (2013) Evaluation Strengths and Weaknesses of Agile Scrum Framework Using Knowledge Management, International Journal of Computer Applications, Vo 65, No 23.
11. Ken Shvaber (2019) Scrum. Flexible product management and business, Alpina Publisher Ukraine, 236 p.
12. Breno G. Tavares, E, Carlos da Silva, Adler D. de Souza (2017) Risk Management in Scrum Projects: A Bibliometric Study, Journal of Communications Software, Vo. 13, No. 1.
13. Laster B. Professional Git (2016), 480 p.
14. Santacroce F., Olsson A., Voss R., Narebski J. (2016) Git: Mastering Version Control, Packt Publishing, 839 p.
15. Tsitoara M. (2020) Beginning Git and GitHub: A Comprehensive Guide to Version Control, Project Management and Teamwork for the New Developer, Apress, 289 p. https://doi.org/10.1007/978-1-4842-5313-7.
16. Lewis C., Chatfield C., Johnson T. (2019) Microsoft Project 2019 step by step. Pearson Education: Microsoft Press, 482 p.
17. Vavilenkova A., Litvinenko O., Zholdakov O. (2015) Informatization Project Management, NAU, 220 p.
18. Microsoft Solutions Framework. Basic Principles. URL: https://newline.tech/microsoft-solutions-framework-basic-principles/ (accessed: 25.01.2021).

© Вавіленкова А. І.