КОМП'ЮТЕРНІ НАУКИ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

УДК 05.8.316 **DOI: 10.30977/BUL.2219-5548.2023.101.1.7**

ОПТИМІЗАЦІЯ СКЛАДУ ТА НАВАНТАЖЕННЯ КОМАНДИ В МЕЖАХ ВИКОРИСТАННЯ ФРЕЙМВОРКУ SCRUM

Знахур С. В., Знахур Л. В.
Харківський національний економічний університет імені Семена Кузнеця

Анотація. *У роботі були вирішені такі проблеми: наведено аналіз рішень щодо оптимізації розподілу завдань проєкту в межах використання SCRUM; наведено математичне визначення цієї проблеми та алгоритм її вирішення, наведено варіанти використання алгоритму формування оптимального складу команд та розподілу завдань між виконавцями.*
Ключові слова: *завдання, команда, оптимізація, проєкт, backlog, SCRUM.*

.

Вступ

Оптимізація розподілу завдань є актуальною проблемою сучасного управління проєктами щодо розроблення програмного забезпечення (ПЗ), оскільки вирішення дозволяє максимізувати продуктивність команди та зменшити проєктні витрати. Також вирішення цієї проблеми дозволяє визначити різні типи завдань та вибрати виконавців для їх реалізації у межах компетенцій команди. Варто зазначити, що в процесі проєктування можливі зміни складу команди, тому оптимізація розподілу завдань дозволяє знайти рішення для завершення спринту з мінімальними відхиленнями від плану на основі перерозподілу завдань між учасниками.

Аналіз публікацій

Аналіз публікацій щодо оптимізації та математичного моделювання розподілу завдань у межах використання фреймворку SCRUM демонструє, що ця тема є актуальною для дослідників галузі управління проєктами та розроблення ПЗ. У роботах [3–12] аналізують проблеми щодо оптимізації процесу розподілу завдань, зокрема управління портфелем проєктів, розподіл завдань між учасниками команди. У дослідженнях розглядають такі аспекти оптимізації розподілу завдань [1–18]:

- управління розподілом завдань;
- аналіз складності завдань та аналіз навантаження учасників команди;
- мінімізація часу, витрат на здійснення завдань;
- аналіз впливу ризиків на час здійснення завдань.

Одним із найпоширеніших методів оптимізації розподілу завдань є використання методів математичного програмування. Наприклад, у [3] автори наводять модель математичного програмування оптимізації процесу планування спринту в SCRUM. Також у [4] автори пропонують використовувати багатокритеріальну оптимізацію для вибору найбільш оптимального складу команди та розподілу завдань між її учасниками. У [10–13] аналізують застосування методів статистичного моделювання та машинного навчання для покращення процесу розподілу завдань. У роботі [12] автори пропонують використовувати алгоритми машинного навчання для автоматичного розподілу завдань між учасниками команди.

Варто зазначити, що вирішення проблеми оптимізації розподілу завдань у SCRUM може бути досить складним процесом через велику кількість невизначених факторів, зокрема через взаємодію учасників команди, специфіку проєкту тощо. Тому для вирішення цієї проблеми часто використовують методи статистичного моделювання та емпіричні підходи [14, 18]. У більшості публікацій щодо цієї теми аналізують теоретичні моделі та методи, а їхня практична реалізація не досліджена.

Мета та постановка завдання

Метою статті є визначення алгоритму процесу вирішення проблеми розподілу завдань (user story) Product backlog щодо їх оптимізації у SCRUM.

Для досягнення мети необхідно розробити математичне визначення завдання, алгоритм його вирішення та навести практичні рекомендації щодо використання алгоритму на основі його програмної реалізації.

Виклад основного матеріалу

Згідно з результатами аналізу наявних рішень щодо оптимізації використання Scrum було запропоновано вирішити проблему розподілу завдань між членами команди за допомогою алгоритму пошуку максимуму цільової функції з відповідною системою обмежень. Вирішення ґрунтується на визначенні цільової функції та її системи обмежень. Система обмежень у алгоритмі має декілька проєктних обмежень, які містять ліміти вартості та кількості учасників. Головним результатом вирішення є розрахування оптимального складу команди в межах врахування обмежень проєкту.

У проекті з розроблення програмного продукту команда розробників працює на основі фреймворку Scrum. Згідно зі Scrum усі завдання визначено Product Owner та згруповано в документі backlog проєкту. Product backlog – це документ, який містить список вимог до функціональності продукту, що впорядковані відповідно до їхньої важливості, тобто Product backlog є переліком функцій продукту, які необхідно реалізувати згідно з пріоритетом. Елементи цього списку називаються «історіями» (user stories) або елементами backlog (backlog items). Product backlog можна редагувати всім учасникам Scrum [1]. Згідно з документом «Оцінка складності задач за методологією Scrum» кожне завдання *user story* (історія користувача) має власну оцінку складності. Якщо складність історій оцінюється за допомогою послідовності Фібоначчі в проміжку значень від ½ до 40, тоді для алгоритму пошуку оптимального складу проєктної команди потрібно перейти від цілих чисел, що мають досить великий вплив на загальне розв'язання рівнянь, до відповідних дробових чисел у проміжку від 1 до 1,5. Визначимо відповідність складності задач на основі чисел Фібоначчі коефіцієнтам їхньої складності (табл. 1).

Таблиця 1 – Відповідність чисел Фібоначчі коефіцієнтам складності задач

Числа Фібоначчі	1/2	1	2	3	5	8	13	20	40
Відносні коефіцієнти	1.004	1.006	1.01	1.016	1.026	1.042	1.068	1.11	1.178

За законом Фібоначчі визначається лише дробова частина числа коефіцієнтів. Час здійснення всього проєкту, відповідно до вибраної методології, розподілено на спринти. Sprint (спринт) – ітерація в Scrum, під час якої створюється функціональне зростання програмного забезпечення. Він жорстко фіксований за часом. Тривалість одного спринту становить від 2 до 4 тижнів. В окремих випадках тривалість спринту має становити не більше ніж 6 тижнів. Чим коротшим є спринт, тим більш гнучким є процес розроблення, частішими є процеси релізів, швидше надходять відгуки від споживача, менше часу витрачається на помилкову роботу. З іншого боку, у разі більш тривалих спринтів команда має більше часу на вирішення проблем, що виникають під час процесу, а власник проєкту зменшує витрати на наради, демонстрації рішення тощо. Різні команди визначають довжину спринту відповідно до вимог замовника, специфіки своєї роботи, складу команд [1–10].

Для розрахування вхідних параметрів під час визначення оптимального складу команди варто здійснити нульовий спринт. Під час нульового спринту буде визначена середня продуктивність членів команди, середній час для здійснення членами команди певного завдання. Ці значення будуть типовими значеннями працездатності команди без огляду на рівень знань виконавця. Оптимальний склад команди буде визначено тоді, коли команда буде здійснювати максимальну кількість історій за спринт. Тоді вирішенням проблеми оптимізації є пошук максимальної кількості історій із Product backlog, які можуть бути реалізовані в Sprint backlog.

Sprint backlog містить функціональність, яка вибрана Product Owner в Product Backlog. Всі функції декомпозовані за задачами, кожна з яких оцінюється командою [4, 9]. Отже, рішенням оптимізації є максимізація цільової функції, тобто необхідно вибрати максимальну кількість історій із Product backlog для Sprint backlog у межах обмежень щодо ресурсів та пріоритетів. Процес проєкту є оптимальним за умови вирішення завдань у найкоротший термін. Обмеженнями цільової функції є система нерівностей, водночас кожна нерівність належить до окремого кандидата проєкту

$$\sum F \to max, \qquad (1)$$

а систему обмежень (2.25) можна записати так:

$$\beta_{11} * x_{11} + \beta_{12} * x_{12} + \ldots + \beta_{1m} * x_{1m} \le C_1,$$
$$\beta_{21} * x_{21} + \beta_{22} * x_{22} + \ldots + \beta_{2m} * x_{2m} \le C_2,$$
$$\ldots$$
$$\beta_{n1} * x_{n1} + \beta_{n2} * x_{n2} + \ldots + \beta_{nm} * x_{nm} \le C_n,$$

де кількість Xm відповідає кількості *user stories* (історій користувача) у спринті проєкту, коефіцієнт β є складним (2) – це добуток частки продуктивності здійснення завдання P та аналізу складності O до рівня професійності Pr та рейту виконавця R:

$$\beta = \frac{P * O}{Pr} * R, \qquad (2)$$

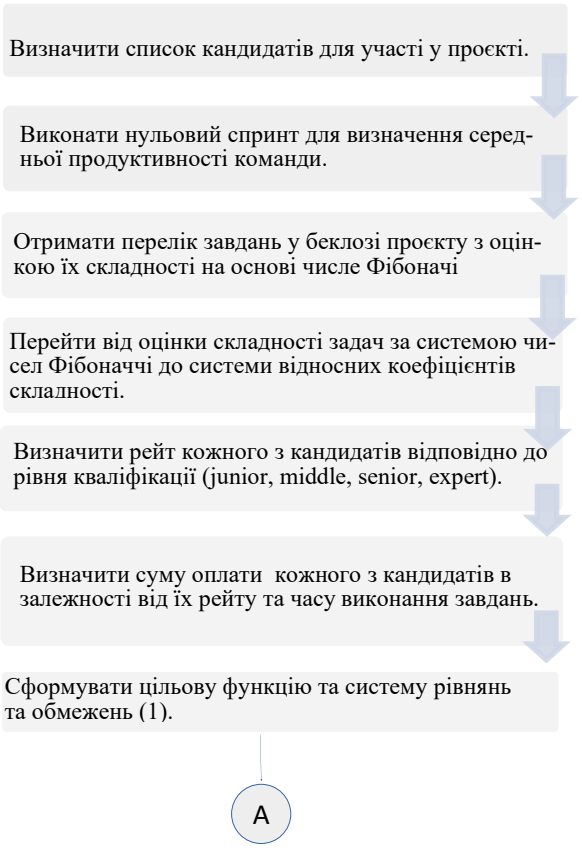
P відповідає значенню середньої продуктивності команди, що визначається протягом нульового спринту проєкту.

Кожне завдання (історія) містить оцінку складності O, що наведено в табл.1. Значення оцінки складності має діапазон від 1 до 1,5, де 1 – найлегше завдання, а 1,5 – найскладніше. Нехай професійний рівень кожного з виконавців Pr має діапазон від 1 до 4, де 1 – рівень знань та практичних навичок початківця (junior), 2 – рівень знань та практичних навичок спеціаліста, що має певний досвід (middle), 3 – рівень знань та практичних навичок спеціаліста, що має вагомий досвід (senior), 4 – рівень знань та практичних навичок експерта (expert). Нехай рейт виконавця R – величина, що визначає заробіток спеціаліста за одиницю часу. Рейт має співвідносну величину для всіх кандидатів на здійснення проєкту, наприклад у.о./год. Права частина нерівностей системи є обмеженням, що визначає максимальну заробітну плату кандидата (виконавця) за спринт. Вона має значення максимального рейту виконавця за спринт (2–4 тижні, відповідно до значення його тривалості). Максимальне значення коригується із часом, який виконавець може використати для поточного проєкту (в умовах роботи на декількох проєктах). Значення коефіцієнта β_{nm}буде дорівнювати нулю, якщо кандидат не має навичок (компетенцій) для здійснення конкретного завдання – історії Product backlog проєкту. Варто зазначити, що x мають два індекси, перший відповідає номеру рівняння, тобто номеру виконавця (n), а другий – номеру задачі в Product backlog проєкту (m). Використовуємо індекси для того, щоб визначити, хто із кандидатів проєкту буде вирішувати відповідні завдання в межах обмеження

$$\sum_{i=1}^{n} \sum_{j=1}^{m} X_{ij} \le 1. \qquad (3)$$

Функцією обмеження є попередження ситуації, у якій декілька виконавців будуть вирішувати одне й те саме завдання в проєкті. Отже, після пошуку максимуму цільової функції на першій ітерації отримуємо перелік історій, які будуть належати Sprint backlog першого спринту, та список їхніх виконавців (якщо значення x_{ij}відповідає нулю, задача не належить до спринту виконавця, якщо 1, то задача належить спринту). Для визначення переліку історій наступного спринту потрібно з Product backlog проєкту видалити перелік історій попереднього етапу. Розподіл задач із Product backlog проєкту згідно зі спринтами проєкту здійснюється доти, доки кількість задач у Product backlog проєкту буде менша за кількість кандидатів.

На рис. 2 наведена загальна схема алгоритму вибору оптимального складу виконавців для проєкту.



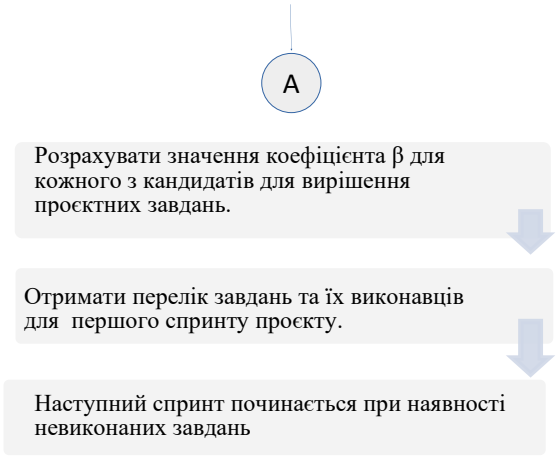


Рис. 1. Алгоритм вирішення

Процес розподілу завдань серед виконавців розпочинається з актуалізації інформації про них (назва, складність) та кандидатів їхнього вирішення (ім'я, прізвище, рівень професійних знань та рейт). У програмному продукті користувач вибирає завдання та кандидатів на участь у проєкті (рис. 2), натискає кнопку «Розрахувати». Також у межах обмежень додає максимальні суми, які учасники мають отримати як заробітну плату за майбутній спринт (час для вирішення завдання) (рис. 3).

Результат наведено на рис. 4.

Рис. 2. Актуалізація завдань та виконавців

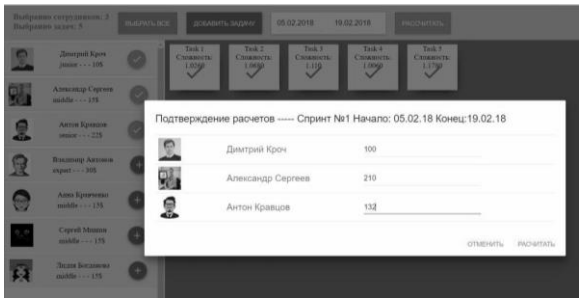


Рис. 3. Обмеження щодо максимальних витрат за спринт

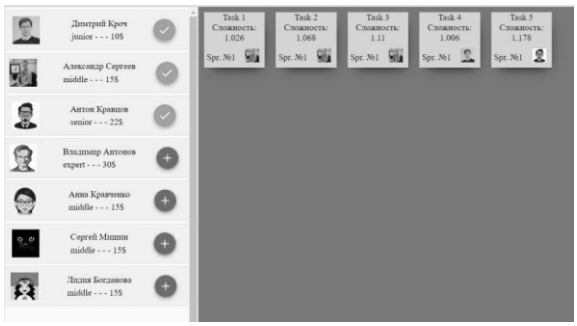


Рис. 4. Результати розподілу завдань

Якщо завдання не належить до першого спринту, воно залишається без виконавця. Таким чином, потрібно декілька ітерацій, щоб усі завдання були розподілені згідно зі спринтами проєкту (рис. 5–6).

Рис. 5. Варіант розподілу завдань для одного спринту

Рис. 6. Варіант розподілу завдань за кількома спринтами

Приклад використання алгоритму. У Product backlog є три завдання та три кандидати для вирішення їх у першому спринті. Кожен із кандидатів має різний рівень професійних знань (junior, middle та senior відповідно). Для пошуку середнього показника продуктивності команди необхідно здійснити нульовий спринт, у якому кожному із членів команди пропонуємо вирішити тестове

завдання, щоб отримати показник продуктивності кожного з кандидатів.

Показник продуктивності – це час (у годинах), який знадобився кожному із кандидатів команди на вирішення тестового завдання в нульовому спринті. Задля більшої точності та детального аналізу продуктивності команди можна запропонувати декілька тестових завдань та здійснити повноцінний нульовий спринт – від двох до чотирьох тижнів. Оскільки цей приклад містить лише п'ять завдань у Product backlog, необхідно вирішити лише одне тестове завдання для визначення продуктивності команди. Середня продуктивність P – це середнє арифметичне часу вирішення тестового завдання кожним із кандидатів. Нехай за результатами нульового спринту ми отримуємо $P = 5(\text{год/завдання})$. У цьому випадку маємо п'ять завдань у Product backlog x_1, x_2, x_3, x_4, x_5, що мають відповідні показники складності за шкалою чисел Фібоначчі: $x_1: 5, x_2: 13, x_3: 20, x_4: 1, x_5: 40$.

Для вирішення будемо використовувати систему коефіцієнтів складності завдань у Product backlog (табл. 1):

$$O_1 = 1.026;\ O_2 = 1.068;\ O_3 = 1.11;\ O_4 = 1.006;\ O_5 = 1.178.$$

Визначимо рейт кожного із кандидатів на вирішення завдань відповідно до рівня їхніх професійних знань. Ми маємо три рівні: junior, middle та senior, отже, їхні рейти розподілено так:

$$R_1 = 10(\text{у. о./год});\ R_2 = 15(\text{у. о./год});\ R_3 = 22(\text{у. о./год}).$$

Визначимо максимальне значення витрат (оплати) кожного з кандидатів на вирішення завдань для спринту Це значення залежить від часу, який кандидат може використати під час участі в спринті. В умовах можливості part-time перший кандидат може відпрацювати 10 годин у новому проєкті, другий кандидат – 14 годин, а третій – 6 годин. Максимальні витрати на оплату кандидатів для спринту становитимуть:

$$C_1 = 100(\text{у. о./спринт});$$
$$C_2 = 210(\text{у. о./спринт});$$
$$C_3 = 132(\text{у. о./спринт}).$$

Для вирішення завдання будемо використовувати програмну реалізацію алгоритму. Необхідно обрати кандидатів серед наявних або створити профілі нових кандидатів на участь у проєкті, перевірити рейт на актуальність, визначити перелік завдань для Sprint backlog, їхню складність, значення середньої продуктивності команди, заповнити праву частину рівнянь – значення максимальної оплати, яку може отримати виконавець за спринт (рис. 7). Результат розподілу завдань на основі розв'язання діофантового рівняння наведено на рис. 8. Перше, друге та третє завдання вирішує другий кандидат (middle), а четверте та п'яте завдання вирішує третій кандидат (senior). Якщо всі завдання були розподілені, тоді процес буде здійснено в першому спринті.

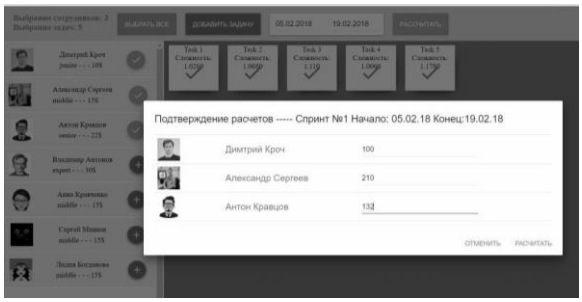


Рис. 7. Вхідні параметри системи

Рис. 8. Результати розподілу завдань

Висновки

У роботі було розроблено та досліджено цілочисельну модель програмування, яка враховує мету управління проєктами в межах SCRUM: максимізація щодо вирішення завдань проєкту, максимізація щодо планування *user stories* (історій користувача) з високим пріоритетом. У процесі дослідження було автоматизовано алгоритм вирішення завдання оптимізації. Модель та алгоритм були протестовані на етапі планування спринтів Scrum в IT-компанії.

Література

1. Методи управління людськими ресурсами при формуванні команд мультипроєктів та програм: монографія / Доценко Н. В. та інші. Харків: ХНУМГ ім. О. М. Бекетова, 2015. 201 с.
2. Хеблов І. А. Розвиток SCRUM-технологій проактивного управління проектами з критичними ризиками [Текст]: автореф. дис. на здобуття наук. ступеня канд. техн. наук: 05.13.22. Одеса, 2017. 20 с.
3. Golfarelli, Matteo & Rizzi, Stefano & Turricchia, Elisa. (2013). Multi-sprint planning and smooth replanning: An optimization model. Journal of Systems and Software. 86. 2357–2370. 10.1016/j.jss.2013.04.028.
4. A multi-objective agile project planning model and a comparative meta-heuristic approach, Information and Software Technology / Nilay Ozcelikkan, Gulfem Tuzkaya, Cigdem Alabas-Uslu, Bahar Sennaroglu. Volume 151. 2022. 107023. ISSN 0950–5849, https://doi.org/10.1016/j.infsof.2022.107023.
5. Nundlall C., Nagowah S. D. (2021). Task allocation and coordination in distributed agile software development: a systematic review. International Journal of Information Technology. Vol. 13. Pp. 321–330.
6. M. Perkusich, L. Chaves e Silva, A. Costa et al (2020). Intelligent software engineering in the context of agile software development: a systematic literature review. Information and Software Technology. Vol. 119. Pp. 106241–106247.
7. Ijaz F, Aslam W. (2019). Identification of dependencies in task allocation during distributed agile software development, Sindh Univ Res J SURJ (Sci Ser). Vol. 51. No. 01. Pp. 31–36.
8. Zhe Wang. Estimating Productivity in a Scrum team: a Multi Agent Simulation, In Proceedings of the 11th International Conference on Computer Modeling and Simulation (ICCMS 2019). ACM, New York, NY, USA. Pp.239–245.
9. Y. C. Cavalcanti, I. d. C. Machado, P. A. d. M. S. Neto, E. S. de Almeida, and S. R. d. L. Meira, "Combining rule-based and information retrieval techniques to assign software change requests," in Proceedings of the 29th ACM/IEEE international conference on Automated software engineering. 2014. Pp. 325–330.
10. D. Zhang and J. J. Tsai, "Machine learning and software engineering," Software Quality Journal. Vol. 11. No. 2. Pp. 87–119. 2003.

References

1. Kelly Waters. How to Implement Scrum in 10 Easy Steps. Available at: https://www.101ways.com/how-to-implement-scrum-in-10-easy-steps.
2. Using the Fibonacci scale in Agile estimation. Available at: https://www.lucidchart.com/blog/fibonacci-scale-for-agile-estimation.
3. Pablo Viola (2019). How to implement Scrum in 7 steps and not die trying. Available at: https://www.hexacta.com/how-to-implement-scrum-in-7-steps-and-not-die-trying/
4. 2020 Scrum Guide Changes and Updates Explained. Available at: https://www.scruminc.com/2020-scrum-guide-changes-updates-explained.
5. 11 steps of Scrum. Available at: https://luminous-men.com/post/11-steps-of-scrum.
6. Ken Schwaber, J. Sutherland, Scrum Development Process, in OOPSLA Business Object Design and Implementation Workshop. Springer, London, 1997. [Electronic resource]. Available at: http://www.jeffsutherland.org/oopsla/schwapub.pdf.
7. Martijn van Asseldonk. How Scrum motivates people [Electronic resource]. Available at: https://www.scrum.org/resources/blog/how-scrum-motivates-people.
8. 5 Steps to Implementing Scrum for Life [Electronic resource]. Available at: https://www.scrum.org/resources/5-steps-implementing-scrum-life.
9. A guide to the Scrum Body of Knowledge (SBOK Guide), 2016 Edition, [Text]. SCRUMstudy, a brand of VMEdu, Inc., Phoenix, Arizona USA, 2016. 340 p.
10. Stephanie Ockerman. Maximize Scrum with the Scrum Values [Electronic resource]. Available at: https://www.scrum.org/resources/blog/maximize-scrum-scrum-values-focus-part-1-5.

Знахур Сергій Вікторович, к.е.н., доц. каф. інформаційних систем, тел. +38 050-300-43-90, serhii.znakhur@gmail.com
Знахур Людмила Володимирівна, викл. каф. інформаційних систем, тел. +38050-65-11-282, razinalv@gmail.com
Харківський національний економічний університет ім. С. Кузнеця, пр. Науки, 9-а, м. Харків, 61166, Україна.

Optimizing team composition and workload using the SCRUM framework

Abstract. Problem. *The following tasks were solved in the work: an analysis of existing solutions for optimizing the distribution of tasks in SCRUM was given; the mathematical statement of the problem of optimizing the distribution of tasks in SCRUM and the algorithm for its solution, the options for using the algorithm for the formation of the optimal team composition and distribution of tasks between performers.* ***Goal.*** *The purpose of the article on optimizing the distribution of tasks in SCRUM is to define the algorithm of the process of solving the issues of task allocation (user story) Product backlog and to help SCRUM teams work more efficiently, speed up the development process and achieve better results in the shortest possible time.* ***Methodology****. The*

most popular software engineering methodology is the SCRUM framework. In Scrum planning, the assignment of user stories to sprints requires consideration of multiple objectives to use the limited resources more effectively. ***Results.*** *In this paper, a mixed-integer programming model is developed which considers the following objectives: maximizing the sprint capacity usage, maximizing the assignment of user stories with high priority to primary sprints. The results are to contribute to both theory and practice of SCRUM planning. The proposed model is applied to the small, medium, and big-sized instances of the problem taken from a real-life system.* ***Originality****. To manage a portfolio of projects using the SCRUM approach, a planning algorithm is proposed, which includes the following: SCRUM information support; selection and formation of target parameters and restrictions; calculation of the coefficients of the complexity of tasks; solution of the Diophantine equation.* ***Practical value.*** *In the process of research, the algorithm for solving the optimization problem was automated. The model and algorithm were tested for the planning phase of SCRUM sprints in the company's IT.*

Key words: *project, team, tasks, planning, backlog, SCRUM.*

Znakhur Serhii, phd, prof. of dep. information systems, phone +38 050-300-43-90, serhii.znakhur@gmail.com

Znakhur Liudmyla, assistant professor of dep. information systems, phone. +38050-65-11-282, razinalv@gmail.com

Simon Kuznets Kharkiv National University of Economics, 61166, Ukraine, Kharkiv, 9a Nauki Ave.