ISSN 2707-1898 (print)
Український журнал інформаційних технологій
Ukrainian Journal of Information Technology
http://science.lpnu.ua/uk/ujit
https://doi.org/10.23939/ujit2024.02.011
Article received 12.10.2024 p.
Article accepted 19.11.2024 p.
UDC 004.02++519.85+004.4

Correspondence author
V. V. Voityshyn
volodymyr.v.voityshyn@lpnu.ua

В. М. Теслюк, А. Є. Батюк, В. В. Войтишин

Національний університет "Львівська політехніка", м. Львів, Україна

МЕТОД ПОБУДОВИ НОРМАЛІЗОВАНОГО РОЗКЛАДУ РЕАЛІЗАЦІЇ ЗАДАЧ ПРОЄКТУ З РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ SCRUM-КОМАНДИ БЕЗ ДИФЕРЕНЦІАЦІЇ СПЕЦІАЛІЗАЦІЙ

З погляду бізнесу одним із ключових аспектів проєктів із розроблення програмного забезпечення є потреба у прогнозованості часу та ресурсів, необхідних для їх успішного виконання. Від цієї оцінки істотно залежить перебіг реалізації та її результати – часто наслідками недооцінки стають неуспішність чи навіть провали проєктів. Розклад реалізації задач є однією із основних складових оцінки проєкту. Посилення конкуренції у сфері розроблення програмного забезпечення накладає все жорсткіші вимоги на оцінки проєктів – що вищі точність та рівень деталізації оцінки та що менше робочого часу витрачено на її підготовку, то більші шанси випередити конкурентів. Тому автоматизація побудови розкладу реалізації задач проєкту є актуальним науковим та технічним завданням. У статті висвітлено метод побудови розкладу реалізації задач проєкту із розроблення програмного забезпечення, що виконує scrum-команда без диференціації спеціалізацій інженерів-розробників. Обмеження на ресурси враховують, збалансовуючи оцінки трудомісткості задач та спроможності розроблення проєктної команди на кожному зі спринтів. Вимірювання як трудомісткості, так і спроможності розроблення здійснюють із застосуванням підходу "нормалізації", абстрагуючись від рівня компетентності інженерів-розробників. Послідовність внесення задач у розклад враховує залежності між ними та ґрунтується на використанні методу диспетчеризації (List Scheduling). У результаті застосування розробленого методу отримують множину допустимих розкладів реалізації, для порівняння яких визначено функцію штрафу, що мінімізує час простоювання проєктної команди, враховуючи, що простоювання команди на початку проєкту менш бажане, ніж на етапі завершення. Розроблений метод, який рекомендовано застосовувати разом із методом поетапного оцінювання, дає змогу скоротити робочий час, який експерти витрачають на підготовку оцінки проєкту. Завдяки простоті поточної версії методу відповідну програмну реалізацію можна здійснити без написання вихідного коду, наприклад, із звикористанням лише табличного редактора. Серед наступних кроків з розвитку методу найпріоритетнішими є підтримка scrum-команд із диференційованими та змішаними спеціалізаціями, а також додавання можливості переходу від нормалізованого до "повного" розкладу реалізації, враховуючи життєвий цикл задач проєкту та структуру робочого часу інженерів-розробників.

Ключові слова: теорія розкладів, розклад реалізації проєкту, scrum-команда; метод поетапного оцінювання, метод диспетчеризації.

Вступ / Introduction

Незважаючи на високий рівень ризиків та невизначеності, пов'язаний з розробленням програмного забезпечення (ПЗ), успішність таких проєктів значною мірою залежить від прогнозованості обсягу часу та ресурсів, необхідних для їх реалізації. Типовою є ситуація, коли цей прогноз (у сфері розроблення ПЗ його називають "оцінкою", англ. *estimate*) необхідно надавати до початку реалізації проєкту. Зазвичай така оцінка охоплює ієрархічну структуру задач та оцінку їх трудомісткості, склад команди, оцінку тривалості проєкту, бюджет, а також розклад реалізації проєктних задач. Низька якість цієї оцінки, зокрема, неналежна точність (як правило, це заниження необхідних обсягів ресурсів та часу) призводить до серйозних проблем на етапі реалізації проєкту та навіть може стати причиною його неуспішного завершення. І навіть більше, стійка тенденція до зростання конкуренції у сфері розроблення ПЗ накладає ще жорсткіші вимоги щодо оцінок проєктів. Зокрема, що вищий рівень деталізації оцінки та що менше робочого часу витрачено на її підготовку, то вища конкурентоспроможність. Тому автоматизація побудови розкладу реалізації проєкту (результатом якої, зокрема, стане зменшення робочого часу, який витрачають експерти) є актуальною науковою та технічною задачею.

У практиці проєктного менеджменту найвідомішими підходами до побудови та візуалізації розкладу виконання проєкту є діаграми Ганта та мережеві діаграми [8], [9], [18]. Незважаючи на поширеність цих підходів та програмних засобів для роботи з ними (одним із найвідоміших із яких є Microsoft Project), побудова такого розкладу, як правило, – трудомістке завдання, що ускладнюється необхідністю враховувати обмеження на ресурси (наприклад, спроможність команди щодо розроблення, як правило, змінюється упродовж реалізації проєкту). Типовою проблемою є й те, що "класичні" методи та засоби побудови розкладу

реалізації не до кінця враховують той факт, що виконання проєкту відбуватиметься із застосуванням agile-підходів (наприклад, scrum).

У математичній науці методи побудови розкладу реалізації проєкту і вивчають у межах дисципліни "дослідження операцій", зокрема, у теорії розкладів. У науковій літературі відповідний клас задач отримав назву "побудова розкладу реалізації задач проєкту з уврахуванням обмежень на ресурси" (англ. "Resource-Constrained Project Scheduling Problem", RCPSP) [1], [3]. До сьогодні уже розроблено велику кількість методів розв'язання задач цього класу [4], [7], однак через складність та різноманітність відповідних прикладних задач жоден із запропонованих методів не претендує на універсальність.

Об'єкт дослідження – побудова розкладу реалізації задач проєкту з розроблення ПЗ.

Предмет дослідження – методи та засоби побудови розкладу реалізації задач проєкту з розроблення ПЗ з урахуванням обмежень на ресурси.

Мета роботи – розроблення методу побудови розкладу реалізації задач проєкту з розроблення ПЗ із урахуванням обмежень на ресурси. Ключовою вимогою до цього методу є те, що він повинен бути комплементарним до методу поетапного оцінювання проєктів із розроблення ПЗ [21], [22], а також методу підтримки прийняття рішень щодо складу команди та тривалості проєкту [20], [22]. Зокрема, рекомендовано застосовувати метод на етапі детального оцінювання, який безпосередньо передує початку реалізації проєкту.

Для досягнення зазначеної мети визначено такі основні *завдання дослідження*:

- проаналізувати літературні джерела та відомі методи побудови розкладу реалізації задач проєкту;
- сформулювати постановку задачі, користуючись термінологією та теоретичними основами, викладеними у працях [20], [21], [22];
- розробити метод побудови нормалізованого розкладу реалізації задач проєкту з розроблення ПЗ для scrum-команд без диференціації спеціалізацій;
- продемонструвати можливості розробленого методу на прикладі.

Матеріали та методи дослідження. Поточне дослідження ґрунтується на теоретичних основах оцінювання проєктів із розроблення ПЗ, викладених у працях [20], [21], [22], та методах теорії розкладів, зокрема, List Scheduling. Для демонстрації застосування розробленого методу побудови розкладу реалізації задач проєкту використано метод поетапного оцінювання [20], [21], [22] та приклад із [22]. Для програмної реалізації методу застосовано табличний редактор Microsoft Excel.

Аналіз останніх досліджень та публікацій. Метод побудови розкладу реалізації задач проєкту із розроблення програмного забезпечення, наведений у цій статті, призначений для визначення того, в які часові проміжки відбуватиметься реалізація кожної із задач, враховуючи їх трудомісткість та спроможність розроблення проєктної команди. У науковій літературі це завдання висвітлено, зокрема, в таких галузях, як проєктний менеджмент та математична наука.

У проєктному менеджменті відповідна задача належить до дисципліни планування проєктів [18]. Класичними підходами до побудови розкладу реалізації проєктних задач є застосування діаграм Ганта [8], [9] або мережевих діаграм [18]. Зокрема, мережеві діаграми використовують у таких відомих методах, як PERT (англ. "Project Evaluation and Review Technique") [5], [6] та CPM (англ. "Critical Path Method") [14].

У математичній науці побудову розкладу реалізації задач проєкту вивчають у межах дисципліни "Дослідження операцій", зокрема, у теорії розкладів. Задачу, метод розв'язування якої подано в цій статті, зараховують до класу задач побудови розкладу виконання робіт проєкту із урахуванням ресурсних обмежень (англ. "Resource-Constrained Project Scheduling Problem", RCPSP) [1], [3], [12], [13]. У літературі виділяють такі основні групи методів розв'язування задачі RCPSP: точні методи (до них, зокрема, належать методи лінійного, цілочисельного, цільового та динамічного програмування), евристичні та метаевристичні методи [1], [7], [15], [23]. В основі низки підходів до розв'язування задачі RCPSP – ідея методу List Scheduling (в україномовному варіанті трапляється також назва "метод диспетчеризації") [10], [11], [19], що ґрунтується на жадібному алгоритмі розстановки задач у розкладі реалізації відповідно до їх пріоритету.

Незважаючи на те, що методів розв'язання спеціальних випадків задачі RCPSP чимало, жоден із них не є універсальним (що пояснюється насамперед складністю та різноманіттям відповідних задач). Актуальність розроблення методу побудови розкладу реалізації задач проєкту із розроблення ПЗ зумовлена, передусім, вимогою його комплементарності до вже розроблених авторами методів поетапного оцінювання проєктів з розроблення ПЗ [21], [22] та методу підтримки прийняття рішень щодо складу команди та тривалості проєкту [20], [22].

Результати дослідження та їх обговорення / Research results and their discussion

Поточне дослідження є логічним продовженням розроблення методів оцінювання проєктів із розроблення ПЗ [20], [21], [22]. Зокрема, в цій статті використано такі терміни: елемент оцінювання (англ. "estimable item"); ієрархічна структура елементів оцінювання, ICEO (англ. "estimable items breakdown structure", EIBS); нормалізований робочий час розроблення, НРЧР (англ. "normalized development working time", NDWT); нормалізована спроможність розроблення, НСР (англ. "normalized development capacity", NDC); нормалізована оцінка розробки, НОР (англ. "normalized development estimate", NDE); команда інженерів-розробників без диференціації спеціалізацій (англ. "non-differentiated specializations"), з диференційованими спеціалізаціями (англ. "differentiated specializations") та зі змішаними спеціалізаціями (англ. "mixed specializations").

З метою уніфікації уточнимо терміни, які вживатимемо у тексті:

1. Замість "проєктна задача" використовуватимемо універсальніший термін "елемент оцінювання". Відповідно декомпозицію змісту проєктних робіт називатимемо "ієрархічна структура елементів оцінювання" (ICEO) [22].

2. Підхід нормалізації, призначений для уніфікації вимірювання трудомісткості, як базу використовує одиницю людино-години [21], [22]. Для елементів оцінювання використовується нормалізована оцінка розробки (НОР), що ураховує лише оцінку часу основної розробки, відкидаючи такі супровідні активності, як наприклад, виправлення дефектів чи написання модульних тестів. Своєю чергою, для вимірювання продуктивності проєктної команди вживають такі терміни, як "нормалізований робочий час розробки" (НРЧР) та "нормалізована спроможність розробки" (НСР), що визначаються незалежно від рівня компетентності інженера-розробника.

3. У цьому дослідженні терміни "графік реалізації проєкту" та "розклад реалізації елементів оцінювання" взаємопов'язані, але мають різні значення. Під графіком реалізації проєкту розумітимемо часовий проміжок із ієрархічною структурою фаз, упродовж якого здійснюється виконання проєкту. У табл. 2 подано приклад графіка реалізації, що містить чотири фази, кожна із яких складається з певної кількості спринтів. Розклад реалізації елементів оцінювання визначає часові проміжки (на графіку реалізації проєкту), упродовж яких відбувається реалізація кожного з цих елементів оцінювання. Наприклад, у розкладі реалізації вказано, що виконання певного елемента оцінювання здійснюється упродовж другого, третього та п'ятого спринтів.

Як правило, елементи оцінювання, які входять до складу певної ICEO, мають різні залежності, що впливають на їх місце на графіку реалізації проєкту. Такі залежності розділимо на дві категорії: залежності від графіка реалізації проєкту та залежності між елементами оцінювання.

Нехай маємо елемент оцінювання $x \in X$, де X – ICEO; $x[\text{start}] \in H$ та $x[\text{end}] \in H$ планові час початку та завершення реалізації x. Для визначеності вважатимемо, що графік реалізації проєкту H – повністю впорядкована множина спринтів, упродовж яких здійснюється реалізація проєкту. Очевидно, що $x[\text{start}] \le x[\text{end}]$. Тоді можливі такі залежності часу початку та завершення реалізації елемента оцінювання від графіка реалізації проєкту:

$$x[\text{start}] \in \Delta_{[x]}, \tag{1}$$

$$x[\text{end}] \in \Delta^{[x]}, \tag{2}$$

$$x[\text{end}] - x[\text{start}] \in \Delta_{[x]}^{[x]}. \tag{3}$$

Зауважимо, що використання операції належності множині у (1)–(3) дає змогу визначати як точне значення виразів у лівих частинах, так і діапазон таких значень. Множина Δ здебільшого є підмножиною цілих чисел – $\Delta \subset \mathbb{Z}$ (наприклад, коли час початку та завершення є порядковими номерами спринтів). Зауважимо, що можна припустити існування ситуацій, коли Δ є підмножиною дійсних чисел. Наприклад, якщо необхідно, щоб реалізація x розпочалася у спринті 1, то (1) запишеться як $x[\text{start}] = 1$. У випадку, якщо тривалість реалізації x не повинна перевищувати чотири спринти, то (3) матиме вигляд $x[\text{end}] - x[\text{start}] \le 3$. На практиці залежності від графіка реалізації проєкту можуть бути зумовлені зовнішніми (відносно проєкту) залежностями, наприклад, необхідністю очікування певних результатів від паралельного проєкту.

Залежності другої категорії визначають співвідношення між часом початку та завершення різних елементів оцінювання. Нехай маємо два елементи оцінювання $x_1 \in X$ та $x_2 \in X$, де X – ICEO; $x_1[\text{start}]$, $x_2[\text{start}]$ – планований час початку реалізації цих елементів, а $x_1[\text{end}]$, $x_2[\text{end}]$ – плановий час завершення їх реалізації. Тоді можливі такі часові залежності між часом початку та завершення реалізації:

$$x_1[\text{start}] - x_2[\text{start}] \in \Delta_{[x_1, x_2]}, \tag{4}$$

$$x_1[\text{start}] - x_2[\text{end}] \in \Delta_{[x_1]}^{[x_2]}, \tag{5}$$

$$x_1[\text{end}] - x_2[\text{start}] \in \Delta_{[x_2]}^{[x_1]}, \tag{6}$$

$$x_1[\text{end}] - x_2[\text{end}] \in \Delta^{[x_1, x_2]}. \tag{7}$$

Наприклад, якщо реалізація x_2 повинна розпочатися не пізніше, як через два спринти після завершення реалізації задачі x_1, то вираз (6) запишеться так: $x_1[\text{end}] - x_2[\text{start}] \in \{-2, -1, 0\}$. У складніших випадках можливе одночасне поєднання кількох типів залежностей, наприклад, реалізації x_1 та x_2 повинні розпочатися та завершитися одночасно: $x_1[\text{start}] - x_2[\text{start}] = 0$ і $x_1[\text{end}] - x_2[\text{end}] = 0$.

На практиці найпоширеніша залежність, що є частковим випадком (6):

$$x_1[\text{end}] - x_2[\text{start}] < 0, \tag{8}$$

тобто реалізація x_2 повинна розпочатися після завершення x_1 (так звана залежність типу "завершення – початок").

Варто зазначити, що реалізація всіх типів залежностей (1)–(7) у методі побудови розкладу – доволі складне завдання, тому в поточній версії методу підтримуються лише залежності типу (8).

Одним із ключових завдань, яке потрібно вирішити під час побудови розкладу реалізації елементів оцінювання, є визначення тривалості реалізації кожного із елементів оцінювання, тобто значень $x[\text{start}]$ та $x[\text{end}]$, де $x \in X$, X – ICEO. Зауважимо, що НОР елемента оцінювання, $x[\text{NDE}]$, істотно впливає на тривалість його реалізації, але, очевидно, не тотожна із цією тривалістю. На те, скільки триватиме реалізація елемента оцінювання, впливає кілька факторів, основні з яких такі:

1) НОР елемента оцінювання;

2) залежності елемента оцінювання від графіка реалізації проєкту (1)–(3);

3) часові залежності від інших елементів оцінювання (4)–(7);

4) НСР команди (у часовому проміжку реалізації елемента оцінювання);

5) життєвий цикл елемента оцінювання, що, зокрема, визначає розподіл робочого часу, витраченого на супровідні активності розроблення [21], [22].

Очевидно, що значення $x[\text{start}]$ та $x[\text{end}]$, для кожного $x \in X$ не є достатніми для побудови розкладу реалізації – крім цього, необхідно вказати, скільки часу на реалізацію витрачатиметься на кожному із спринтів. *Розкладом реалізації елементів оцінювання* (англ. "normalized schedule of estimable items implementation") $X = \{x_1, x_2, \ldots, x_m\}$ називатимемо матрицю:

$$J = \begin{pmatrix} M_{h_1}^{[x_1]} & M_{h_2}^{[x_1]} & \cdots & M_{h_k}^{[x_1]} \\ M_{h_1}^{[x_2]} & M_{h_2}^{[x_2]} & \cdots & M_{h_k}^{[x_2]} \\ \cdots & \cdots & \ddots & \cdots \\ M_{h_1}^{[x_m]} & M_{h_2}^{[x_m]} & \cdots & M_{h_k}^{[x_m]} \end{pmatrix}, \quad (9)$$

де $H = \{h_1, h_2, \ldots, h_k\}$ – графік реалізації проєкту, заданий як повністю впорядкована множина спринтів, $M_h^{[x]}$ – повний робочий час (ураховуючи супровідні активності розробки [21], [22]), який інженери-розробники витрачають на реалізацію елемента оцінювання $x \in X$ упродовж спринта $h \in H$.

Оскільки трудомісткість елементів оцінювання та спроможність розроблення проєктної команди оцінюють згідно з підходом нормалізації [21], [22], то для побудови розкладу (9) зручно спершу отримати так званий нормалізований розклад реалізації. Нехай маємо множину елементів оцінювання $x \in X$, для кожного із яких визначено НОР $x[\text{NDE}]$. Також маємо графік реалізації проєкту, виражений як повністю впорядкована множина спринтів H; для кожного зі спринтів $h \in H$ відома НСР $C_h \geq 0$. Тоді *нормалізованим розкладом реалізації елементів оцінювання* (англ. "normalized schedule of estimable items implementation") називатимемо матрицю:

$$J_N = \begin{pmatrix} U_{h_1}^{[x_1]} & U_{h_2}^{[x_1]} & \cdots & U_{h_k}^{[x_1]} \\ U_{h_1}^{[x_2]} & U_{h_2}^{[x_2]} & \cdots & U_{h_k}^{[x_2]} \\ \cdots & \cdots & \ddots & \cdots \\ U_{h_1}^{[x_m]} & U_{h_2}^{[x_m]} & \cdots & U_{h_k}^{[x_m]} \end{pmatrix}, \quad (10)$$

де $U_h^{[x]} \geq 0$ – нормалізований робочий час розроблення, витрачений на реалізацію елемента оцінювання $x \in X$ на спринті $h \in H$. Справедливі такі умови балансу для рядків:

$$\sum_{h \in H} U_h^{[x]} = x[\text{NDE}], \quad (11)$$

а також стовпців матриці:

$$\sum_{x \in X} U_h^{[x]} \leq C_h. \quad (12)$$

Для кожного $x \in X$ існує умова, що лімітує мінімальний та максимальний обсяги нормалізованого робочого часу розроблення, що можна витратити на реалізацію елемента оцінювання упродовж спринта:

$$\left[U^{[x]}\right]_{\min} \leq U_h^{[x]} \leq \left[U^{[x]}\right]^{\max}, h \in H. \quad (13)$$

Важливою умовою є те, що реалізація елемента оцінювання не може перериватися.

Підкреслимо, що вирази (10)–(13) застосовні до команди інженерів-розробників без диференціації спеціалізацій. Для команд із диференційованими чи змішаними спеціалізаціями $\sum_{x \in X} U_h^{[x]} \leq C_h$, $x_1[\text{end}] - x_2[\text{start}] < 0$, $x \in X$ будуть векторами, елементи яких відповідатимуть спеціалізаціям інженерів-розробників.

Легко помітити, що нормалізований розклад реалізації елементів оцінювання передбачає такі спрощення:

1) враховується лише нормалізований робочий час розробки $U_h^{[x]}$, але не враховується робочий час, витрачений на супровідні активності розробки;

2) ІСЕО X є множиною лише простих елементів оцінювання (тобто не враховується ієрархічність ІСЕО);

3) не враховується ієрархічність графіка реалізації проєкту.

Варто підкреслити, що нормалізований розклад реалізації елементів оцінювання (10) слугує своєрідним "скелетом", на основі якого будують так званий "повний" розклад (9), який враховує супровідні активності розроблення та рівні компетентності учасників проєктної команди. Побудова "повного" розкладу не належить до завдань цього дослідження.

Для побудови нормалізованого розкладу реалізації елементів оцінювання необхідні такі вхідні дані:

1) команда інженерів-розробників без диференціації спеціалізацій;

2) графік реалізації проєкту, виражений як повністю впорядкована множина спринтів H, із можливістю додавання нового спринта, якщо необхідно збільшити тривалість проєкту;

3) для кожного зі спринтів $h \in H$ відома НСР C_h;

4) ІСЕО X є множиною елементів оцінювання, для кожного з яких визначено НОР $x[\text{NDE}]$;

5) сумарна НСР команди $C = \sum_{h \in H} C_h$ достатня для реалізації всіх елементів оцінювання із H:

$$\sum_{h \in H} C_h \geq \sum_{x \in X} x[\text{NDE}]; \quad (14)$$

6) між елементами оцінювання $x \in X$ допускаються залежності типу "завершення – початок" (8); вважають, що відповідний граф залежностей не містить циклів.

Завдання методу полягає у такій побудові нормалізованого розкладу реалізації елементів оцінювання (10), щоб виконувалися обмеження на ресурси (11)–(13) і забезпечувалася нерозривність реалізації кожного з елементів оцінювання $x \in X$. Такий розклад називатимемо допустимим. Очевидно, що можливе існування одного і більше допустимих розкладів. Якщо ж допустимий розклад отримати не вдається, необхідно збільшити кількість спринтів у графіку реалізації проєкту H (що, очевидно, гарантує існування допустимого розкладу).

Якщо існує більше ніж один допустимий розклад реалізації елементів оцінювання, необхідно визначити критерій їх порівняння. Таким критерієм виберемо функцію штрафу від нормалізованого робочого часу простою команди:

$$\Theta = \sum_{h \in H} \left[\theta_h \left(C_h - \sum_{x \in X} U_h^{[x]} \right) \right] \to \min, \qquad (15)$$

де $\theta_h \geq 0$ – коефіцієнт штрафу, значення якого залежить від фази реалізації проєкту h. Для прикладу, коефіцієнт штрафу θ_h дає змогу визначити, що простій команди (спричинений залежностями елементів оцінювання) є критичнішим на початкових фазах реалізації проєкту, ніж на завершальних.

Для побудови нормалізованого розкладу реалізації елементів оцінювання необхідно виконати певні кроки, що ґрунтуються на методі List Scheduling [10], [11], [19].

Крок 1. Побудувати повністю впорядковану множину елементів оцінювання S, скориставшись залежностями (8). Для прикладу, першим елементом такої множини є елемент оцінювання, який не має залежностей від інших елементів оцінювання. Легко помітити, що в загальному випадку можливо побудувати доволі велику кількість варіантів такої повністю впорядкованої множини елементів оцінювання. Для зручності викладення вважатимемо, що множина S "працює" за принципом стека, тобто в разі вибору першого елемента цей елемент видаляється із S.

Крок 2. Вибрати перший елемент оцінювання із S та задати розподіл його нормалізованого робочого часу розроблення за спринтами, дотримуючись виконання обмежень (11)–(13), намагаючись розмістити реалізацію якомога раніше, а також враховуючи те, що реалізація не може перериватися.

Крок 3. Якщо розміщення елемента оцінювання неможливе (через недостатність доступного НСР команди), необхідно додати ще один спринт згідно із правилами, визначеними графіком реалізації проєкту. Після цього повторити крок 2.

Крок 4. Завершити виконання, якщо у множині S більше не залишилося елементів оцінювання.

Очевидно, що запропонований вище підхід завжди забезпечує побудову допустимого розкладу реалізації елементів оцінювання. Кількість варіантів такого розкладу визначається кількістю можливих значень S, що генерується на кроці 1. Порівняння отриманих варіантів розкладу здійснюватимемо згідно із (15).

Приклад застосування методу. Для демонстрації розробленого методу скористаємося прикладом із [22] (із незначними модифікаціями). Елементи оцінювання, їх НОР та залежності типу "завершення – початок" (8) подано в табл. 1. Графік реалізації проєкту, зокрема оцінку НСР, отримані із застосуванням методу поетапного оцінювання [21], [22], наведено у табл. 2.

Табл. 1. Приклад елементів оцінювання / Example of estimable items

Ідентифікатор	Елемент оцінювання	НОР, людино-години	Залежності
EI-1	Табличне подання event-даних, згенерованих бізнес-процесом оцінювання	470,4	EI-9, EI-6
EI-2	Візуалізація концептуальної схеми бізнес-процесу оцінювання	218,4	EI-9
EI-3	Метод побудови актуальної схеми бізнес-процесу оцінювання	806,4	EI-9
EI-4	Візуалізація актуальної схеми бізнес-процесу оцінювання	882,0	EI-9, EI-3, EI-6
EI-5	Повідомлення щодо відхилень під час виконання бізнес-процесу	218,4	EI-9, EI-6
EI-6	Реалізація інтеграції даних	352,8	EI-9
EI-7	Безпека та управління користувачами	302,4	EI-9
EI-8	Адміністрування та конфігурація	201,6	EI-9
EI-9	Налаштування базової структури проєкту	92,4	

Табл. 2. Приклад графіка реалізації проєкту / Example of project implementation schedule

	Фаза проєкту				Сума
	Початок проєкту	Розширення команди	Основна фаза реалізації	Стабілізація	
Кількість спринтів	2	2	15	2	21
НСР фази, людино-години	170,0	290,0	2960,0	150,0	3570,0
НСР спринта, людино-години	85,0	145,0	197,3	75,0	

Після застосування до розглянутого прикладу методу побудови розкладу реалізації елементів оцінювання, отримано результати, наведені на рис. 1. Легко побачити, що під час побудови розкладу реалізації було необхідно додати один спринт (номер 20) до основної фази реалізації, що, своєю чергою, призвело до збільшення нормалізованого часу простою проєктної команди до 222.0 людино-годин.

Незначно змінивши послідовність елементів оцінювання, а саме помінявши місцями EI-1 та EI-7, отрима-

ємо дещо інший варіант розкладу реалізації елементів оцінювання, поданий на рис. 2. У цьому новому варіанті тривалість проєкту становить 21 спринт, що збігається із початковою оцінкою тривалості у табл. 2. Важливо підкреслити, що у другому варіанті нормалізований час простою проєктної команди становить 4,7 людино-годин (що менше ніж 222,0 людино-години у першому варіанті). Той факт, що варіант 2 є "кращим" за варіант 1, підкреслює і значення функції штрафу (15): $\Theta_1 = 32,6$ та $\Theta_2 = 0,0$.

Очевидно, що, змінюючи послідовність елементів оцінювання (не порушуючи залежностей між ними), можна отримати й інші варіанти розкладу реалізації. Формальною ознакою для визначення "кращих" варіантів є функція штрафу (15).

Обговорення результатів дослідження. Згідно з концепцією конуса невизначеності [2], метод поетапного оцінювання проєктів із розроблення ПЗ передбачає послідовну підготовку трьох типів оцінок: попередньої, проміжної та детальної. Кожна наступна з них є детальнішою та точнішою за попередню [20], [21], [22]. В контексті методу поетапного оцінювання розроблений метод побудови розкладу реалізації елементів оцінювання доцільно застосовувати на третьому етапі (детального оцінювання), коли рівень деталізації ICEO достатній. На етапі реалізації проєкту побудований розклад буде взято за основу для розподілу проєктних задач за спринтами.

Фаза реалізації		Початок проєкту		Розширення команди		Основна фаза реалізації																Стабілізація		Сума
# спринта		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	
НСР, людино-години		85,0	85,0	145,0	145,0	197,3	197,3	197,3	197,3	197,3	197,3	197,3	197,3	197,3	197,3	197,3	197,3	197,3	197,3	197,3	197,3	75,0	75,0	**3766,8**
# ЕО	НОР, л-г	Розклад реалізації елементів оцінювання																						
EI-9	92,4	85,0	7,4																					
EI-3	806,4		40,0	50,0	50,0	70,0	70,0	90,0	90,0	90,0	90,0	90,0	76,4											
EI-6	352,8		37,6	40,0	50,0	70,0	70,0	60,0	25,2															
EI-4	882,0													150,0	150,0	150,0	150,0	150,0	132,0					
EI-1	470,4									107,3	107,3	107,3	120,9	27,6										
EI-7	302,4			55,0	45,0	57,3	57,3	47,3	40,5															
EI-2	218,4													19,7	47,3	47,3	47,3	47,3	9,5					
EI-5	218,4																		55,8	162,6				
EI-8	201,6																			34,7	166,9			
НСР - НОР, людино-години		0,0	0,0	0,0	0,0	0,0	0,0	0,0	41,6	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	30,4	75,0	75,0	**222,0**
Коеф. штрафу, θ		0,95	0,91	0,86	0,82	0,77	0,73	0,68	0,64	0,59	0,55	0,50	0,45	0,41	0,36	0,32	0,27	0,23	0,18	0,14	0,09	0,05	0,00	
θ · (НСР - НОР)		0,00	0,00	0,00	0,00	0,00	0,00	0,00	26,47	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	2,76	3,41	0,00	**32,6**

Рис. 1. Приклад нормалізованого розкладу реалізації елементів оцінювання: варіант 1 / Example of normalized schedule of estimable items implementation: option 1

Фаза реалізації		Початок проєкту		Розширення команди		Основна фаза реалізації															Стабілізація		Сума
# спринта		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
НСР, людино-години		85,0	85,0	145,0	145,0	197,3	197,3	197,3	197,3	197,3	197,3	197,3	197,3	197,3	197,3	197,3	197,3	197,3	197,3	197,3	75,0	75,0	**3569,5**
# ЕО	НОР, л-г	Розклад реалізації елементів оцінювання																					
EI-9	92,4	85,0	7,4																				
EI-3	806,4		40,0	50,0	50,0	70,0	70,0	90,0	90,0	90,0	90,0	90,0	76,4										
EI-6	352,8		37,6	40,0	50,0	70,0	70,0	60,0	25,2														
EI-4	882,0													150,0	150,0	150,0	150,0	150,0	132,0				
EI-7	302,4			55,0	45,0	57,3	57,3	47,3	40,5														
EI-1	470,4								41,6	107,3	107,3	107,3	106,9										
EI-2	218,4												14,0	47,3	47,3	47,3	47,3	15,2					
EI-5	218,4																	32,1	65,3	121,0			
EI-8	201,6																			76,3	75,0	50,3	
НСР - НОР, людино-години		0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	24,7	**24,7**
Коеф. штрафу, θ		0,95	0,90	0,86	0,81	0,76	0,71	0,67	0,62	0,57	0,52	0,48	0,43	0,38	0,33	0,29	0,24	0,19	0,14	0,10	0,05	0,00	
θ · (НСР - НОР)		0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	**0,0**

Рис. 2. Приклад нормалізованого розкладу реалізації елементів оцінювання: варіант 2 / Example of normalized schedule of estimable items implementation: option 2

Розроблений метод побудови розкладу реалізації елементів оцінювання не рекомендується застосовувати, коли:

1) реалізація проєкту істотно залежить від зовнішніх факторів;

2) є ризики, що істотно впливають на перебіг реалізації проєкту;

3) передбачаються складні залежності між елементами оцінювання;

4) реалізація проєкту організована без використання спринтів (тобто не використовується scrum).

У цих ситуаціях рекомендовано розглянути можливість використання, наприклад, діаграм Ганта [8], [9], [18], методів PERT [5], [6] та CPM [14], а у складніших випадках – методу GERT [17].

У найпростішому варіанті розроблений метод побудови розкладу реалізації елементів оцінювання можна реалізувати у табличному редакторі (що і було

продемонстровано на прикладі вище). Можливість застосування табличного редактора зумовлена, зокрема, тим, що для проєктної команди не передбачена диференціація спеціалізацій. У випадку команди із диференційованими чи змішаними спеціалізаціями, коли значення НОР та НСР є векторами, розмірність яких відповідає кількості спеціалізацій інженерів-розробників, використання табличного редактора істотно ускладнюється – в цьому випадку ефективнішою буде реалізація із застосуванням однієї із мов програмування (наприклад, Python, C#, Java тощо).

Окремо варто звернути увагу на те, що Microsoft Project, незважаючи на його широкі можливості з розроблення розкладу реалізації проєктів, доволі складно застосувати для програмної реалізації пропонованого методу. Ця складність зумовлена, передусім, відсутністю вбудованих функцій для контролю виконання умов балансу НСР та НОР (11), (12), а також недостатньою гнучкістю під час визначення тривалості задачі залежно від призначених для її реалізації ресурсів.

Вважаємо доцільними такі кроки з розвитку розробленого методу побудови розкладу реалізації елементів оцінювання:

1) конкретизувати підхід до визначення тривалості реалізації елемента оцінювання та розподілу його нормалізованого часу розроблення між спринтами (крок 2 методу);

2) додати підтримку команд із диференційованими та змішаними спеціалізаціями;

3) розширити метод можливістю переходу від нормалізованого розкладу реалізації до "повного", враховуючи життєвий цикл елементів оцінювання та структуру робочого часу інженерів-розробників [21], [22];

4) дослідити можливість визначення критичного шляху (в тому сенсі, як його розуміють у контексті методу CPM [14]);

5) дослідити можливість додавання інших типів залежностей елементів оцінювання, визначених (1)–(7);

6) дослідити можливість застосування методу гілок та меж [16], який потенційно може оптимізувати розроблений метод, "відсікаючи" "безперспективні" варіанти розкладів реалізації.

Підкреслимо, що заявлені вище перспективи розвитку розробленого методу передбачають його застосування у контексті методу поетапного оцінювання проєктів із розроблення ПЗ [20], [21], [22].

Наукова новизна отриманих результатів дослідження – розвинено метод List Scheduling, використаний для побудови розкладу реалізації елементів оцінювання, із урахуванням як обмежень на ресурси умови балансу між НОР змісту проєктних робіт та НСР проєктної команди; сформульовано критерій якості розкладу реалізації елементів оцінювання, який є функцією штрафу, що мінімізує нормалізований час простоювання проєктної команди.

Практична значущість результатів дослідження – розроблений метод побудови розкладу реалізації елементів оцінювання проєкту з розроблення ПЗ, будучи комплементарним до методу поетапного оцінювання, дає змогу зменшити витрати робочого часу експертів на підготовку детальної оцінки проєкту; програмну реалізацію поточної версії методу можна виконати із мінімальними затратами, застосовуючи лише табличний редактор і не вимагаючи написання вихідного коду.

Висновки / Conclusions

Усі завдання дослідження виконано повною мірою, унаслідок чого розроблено метод побудови нормалізованого розкладу реалізації елементів оцінювання проєкту із розроблення ПЗ, що є подальшим розвитком методу List Scheduling [10], [11], [19]. Основна ідея методу полягає у розподілі проєктних задач на графіку реалізації так, щоб НОР елементів оцінювання не перевищувала НСР проєктної команди на кожному зі спринтів. Розроблений метод є комплементарним до методу оцінювання проєктів із розроблення ПЗ, описаного у працях [20], [21], [22].

За результатами дослідження зроблено такі висновки:

1. Встановлено, що розв'язана задача належить до класу задач побудови розкладу реалізації проєкту з урахуванням обмежень на ресурси (англ. "Resource-Constrained Project Scheduling Problem", RCPSP). У науковій літературі подано методи розв'язування різних часткових випадків цієї задачі.

2. Визначено, що розроблений метод, застосований на етапі детального оцінювання, дає змогу зменшити витрати часу експертів на підготовку розкладу реалізації елементів оцінювання проєкту з розроблення ПЗ.

3. Визначено, що розроблений метод гарантовано будує хоча б один допустимий розклад (за рахунок збільшення тривалості проєкту на 1 спринт у разі потреби).

4. З'ясовано, що в разі застосування повного перебору кількість можливих варіантів розкладу доволі велика (в разі відсутності залежностей між елементами оцінювання ця кількість становитиме $n!$, де n – кількість елементів оцінювання). Отже, актуальна задача оптимізації методу, зокрема, дослідження можливості застосування методу гілок та меж для відкидання "неперспективних" варіантів.

5. Встановлено, що завдяки простоті поточної версії методу програмну реалізацію можна здійснити із застосуванням табличного редактора і немає потреби в написанні вихідного коду.

6. Із окреслених перспектив розвитку методу найпріоритетніші такі: підтримка scrum-команд із диференційованими і змішаними спеціалізаціями; додавання можливості переходу від нормалізованого до "повного" розкладу реалізації елементів оцінювання, що ґрунтується на життєвому циклі елементів оцінювання та структурі робочого часу інженерів-розробників.

References

1. Blazewicz, J., Lenstra, J. K., & Kan, A. H. G. R. (1983). Scheduling subject to resource constraints: Classification and com-

plexity. *Discrete Applied Mathematics*, 5(1), 11–24. https://doi.org/ 10.1016/0166-218X(83)90012-4
2. Boehm, B. W., Abts, C., Brown, A. W., Devnani-Chulani, S., Clark, B. K., Horowitz, E., Madachy, R. J., Reifer, D. J., & Steece, B. (2000). Software Cost Estimation with COCOMO II. Prentice-Hall: Saddle River.
3. Brucker, P. (2007). Scheduling Algorithms (5. 5th ed. 2007). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-69516-5
4. Brucker, P., Drexl, A., Möhring, R., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112(1), 3–41. https://doi.org/10.1016/S0377-2217(98)00204-5
5. Bureau of Naval Weapons, United States, Special Projects Office. (1958). Program Evaluation Research Task PERT Summary Report: Phase 1 (p. 35). Special Projects Office, Bureau of Naval Weapons, Department of the Navy.
6. Bureau of Naval Weapons, United States, Special Projects Office. (1958). Program Evaluation Research Task PERT Summary Report: Phase 2 (p. 108). Special Projects Office, Bureau of Naval Weapons, Department of the Navy.
7. Ciupe, A., Meza, S., & Orza, B. (2016). Heuristic Optimization for the Resource Constrained Project Scheduling Problem: A Systematic Mapping, 619–626. https://doi.org/10.15439/2016F389
8. Gantt, H. L. (1903). A Graphical Daily Balance in Manufacture. *Transactions of the American Society of Mechanical Engineers*, 24, 1322–1331. https://doi.org/10.1115/1.4060667
9. Geraldi, J., & Lechter, T. (2012). Gantt charts revisited: A critical analysis of its roots and implications to the management of projects today. *International Journal of Managing Projects in Business*, 5(4), 578–594. https://doi.org/10.1108/17538371211268889
10. Graham, R. L. (1969). Bounds on Multiprocessing Timing Anomalies. *SIAM Journal on Applied Mathematics*, 17(2), 416–429. https://doi.org/10.1137/0117039
11. Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. H. G. R. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. *In Annals of Discrete Mathematics* (Vol. 5, pp. 287–326). Elsevier. https://doi.org/10.1016/S0167-5060(08)70356-X
12. Hartmann, S., & Briskorn, D. (2022). An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 297(1), 1–14. https://doi.org/10.1016/j.ejor.2021. 05.004
13. Herroelen, W., De Reyck, B., & Demeulemeester, E. (1998). Resource-constrained project scheduling: A survey of recent developments. *Computers & Operations Research*, 25(4), 279–302. https://doi.org/10.1016/S0305-0548(97)00055-5
14. Kelley, J. E., & Walker, M. R. (1959). Critical-Path Planning and Scheduling. Eastern Joint IRE-AIEE-ACM Computer Conference, 160–173. https://doi.org/10.1145/1460299. 1460318
15. Kolisch, R., & Hartmann, S. (1999). Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis. In J. Węglarz (Ed.), Project Scheduling (Vol. 14, pp. 147–178). Springer US. https://doi.org/10.1007/978-1-4615-5533-9_7
16. Land, A. H., & Doig, A. G. (1960). An Automatic Method of Solving Discrete Programming Problems. *Econometrica*, 28(3), 497. https://doi.org/10.2307/1910129
17. Pritsker, A. A. B. (1966). GERT: Graphical Evaluation and Review Technique. RM-4973-NASA. National Aeronautics and Space Administration under Contract No. NASr-21.
18. Project Management Institute (Ed.) (2021). A guide to the project management body of knowledge (PMBOK® guide) and the standard for project management (Seventh Edition). Project Management Institute, Inc.
19. Schutten, J. M. J. (1996). List scheduling revisited. Operations Research Letters, 18(4), 167–170. https://doi.org/10.1016/ 0167-6377(95)00057-7
20. Teslyuk, V., Batyuk, A., & Voityshyn, V. (2022). Method of Recommending a Scrum Team Composition for Intermediate Estimation of Software Development Projects. 2022 IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT), 373–376. https://doi.org/10.1109/CSIT56902.2022.10000432
21. Teslyuk, V., Batyuk, A., & Voityshyn, V. (2022). Method of Software Development Project Duration Estimation for Scrum Teams with Differentiated Specializations. Systems, 123(10), 1–19. https://doi.org/10.3390/systems10040123
22. Teslyuk, V., Batyuk, A., & Voityshyn, V. (2024). Preliminary Estimation for Software Development Projects Empowered with a Method of Recommending Optimal Duration and Team Composition. Applied System Innovation, 7(3). https://doi.org/10.3390/asi7030034
23. Türkakın, O. H., Arditi, D., & Manisalı, E. (2021). Comparison of Heuristic Priority Rules in the Solution of the Resource-Constrained Project Scheduling Problem. *Sustainability,* 13(17), 9956. https://doi.org/10.3390/su13179956

V. M. Teslyuk, A. Ye. Batyuk, V. V. Voityshyn

Lviv Polytechnic National University, Lviv, Ukraine

METHOD OF BUILDING A NORMALIZED IMPLEMENTATION SCHEDULE OF A SOFTWARE DEVELOPMENT PROJECT FOR A SCRUM TEAM WITH NON-DIFFERENTIATED SPECIALIZATIONS

From the business standpoint, predictability of the resources and time required for successful project completion is one of the key aspects of software development. This estimate significantly impacts project implementation and its outcomes – underestimation often leads to failures. An implementation schedule is one of the key ingredients of software development project estimation. Over recent years, the software development business has become even more competitive which, in turn, has imposed more demanding requirements on estimates – the more accurate an estimate is, and the less working time is spent on its preparation, the higher chance to overperform the competitors. This is why automation of such a manual assignment as project implementation schedule preparation is a relevant scientific and technical task. The current article is devoted to a method of building a software development project tasks implementation schedule. The project is sup-

posed to be done by a scrum team with non-differentiated specializations. The resource constraints are expressed as balancing of the task effort estimates and the project team development capacity for each of the sprints. In order to unify measurement of software development productivity, a “normalization” approach is applied. The order of adding tasks to a schedule considers dependencies between the tasks and utilizes the List Scheduling method. As the method's outcomes, a set of feasible project implementation schedules is produced. Over a feasible schedule, a fine objective function is defined. This objective function minimizes the project team’s idle time assuming that such idle time in the beginning of the project is “worse” than in its ending. Being applied along with the Multi-Stage Estimation method, the method proposed in the current article allows to reduce the working time spent by experts on preparation of estimates. Because of its simplicity, the software implementation of the current method version can be done without writing source code – a spreadsheet editor can be used instead. The priority ways of the further evolution of the method are the following: supporting of scrum teams with differentiated and mixed specializations; transition from a normalized project implementation schedule to the “full” one considering the project tasks' lifecycle and the structure of software developers’ working time.

Keywords: job-shop scheduling, project implementation schedule, scrum team, Multi-Stage Estimation method, List Scheduling.

Інформація про авторів:

Теслюк Василь Миколайович, д-р техн. наук, професор, завідувач кафедри автоматизованих систем управління.
E-mail: vasyl.m.teslyuk@lpnu.ua; https://orcid.org/0000-0002-5974-9310

Батюк Анатолій Євгенович, канд. техн. наук, доцент, кафедра автоматизованих систем управління.
E-mail: anatolii.y.batiuk@lpnu.ua; https://orcid.org/0000-0001-7650-7383

Войтишин Володимир Володимирович, асистент, кафедра автоматизованих систем управління.
Email: volodymyr.v.voityshyn@lpnu.ua; https://orcid.org/0000-0002-7889-2593

Цитування за ДСТУ: Теслюк В. М., Батюк А. Є., Войтишин В. В. Метод побудови нормалізованого розкладу реалізації задач проєкту з розробки програмного забезпечення для scrum-команди без диференціації спеціалізацій. Український журнал інформаційних технологій. 2024, т. 6, № 2. С. 11–19.

Citation APA: Teslyuk, V. M., Batyuk, A. Ye., & Voityshyn, V. V. (2024). Method of building a normalized implementation schedule of a software development project for a Scrum team with non-differentiated specializations. *Ukrainian Journal of Information Technology, 6*(2), 11–19. https://doi.org/10.23939/ujit2024.02.011