doi: 10.32620/oikit.2024.100.10

УДК 004.4:004.9:519.8 І. В. Шевченко, О. В. Лучшева

Інкрементний підхід до планування роботи студентів над спільним програмним проєктом

Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут»

На сьогоднішній день відомі продуктові і аутсорсингові IT-компанії використовують гнучкі (Agile) моделі для життєвого циклу (ЖЦ) розроблення програмного забезпечення, переваги яких полягають у швидкій адаптації до змін, постійному зворотному зв'язку, покращені якості продукту, підвищені залученості команди, фокусі на цінності для клієнта, прозорості та контролі, а також у зменшенні внутрішніх і зовнішніх ризиків. Всі гнучкі моделі є ітеративними, тобто процес розроблення поділяється на окремі однотипні ітерації, які мають різні назву в різних моделях. Ознайомлення і опрацювання на практиці здобувачами першого (бакалаврського) рівня вищої освіти галузі інформаційних технологій сучасних Agile-моделей є важливим завданням для майбутніх IT-фахівців. Для виконання цього завдання було обрано найпопулярнішу Agile-модель Scrum, яка також є ще і інкрементною, тобто такою, що на кожній ітерації видає додаткову готову до експлуатації функціональність. Розглянуті основні ролі учасників і їх обов'язки, артефакти і церемонії Scrum-процесу. У статті пропонується застосувати Scrum-модель для ітераційного і інкрементного розроблення програмного забезпечення, яке буде розв'язувати задачу багатокритеріальної оптимізації з використанням теоретико-ігрового підходу. Подано постановку і загальний алгоритм розв'язання задачі багатокритеріальної оптимізації з використанням теоретико-ігрового підходу, а також приклад формулювання Product Backlog'у, який містить дванадцять функціональних вимог, що мають бути реалізовані студентською Scrum-командою за дванадцять ітерацій (спринтів). Описано рекомендований процес розбиття здобувачів на Scrum-команди і призначення ролей учасникам. Також наведено приклад інкрементної реалізації на основі Scrum-моделі ЖЦ розв'язання задачі багатокритеріальної оптимізації з використанням теоретико-ігрового підходу.

Ключові слова: програмне забезпечення; життєвий цикл розроблення програмного забезпечення; Agile; SCRUM; багатокритеріальна оптимізація.

Вступ

На сьогоднішній день відомі продуктові і аутсорсингові IT-компанії використовують гнучкі (Agile) моделі для життєвого циклу розроблення програмного забезпечення (Software Development Lifecycle, SDLC). Які ж основні переваги гнучких моделей? Це перш за все швидка адаптація до змін, постійний зворотний зв'язок, покращена якість продукту, підвищена залученість команди, фокус на цінності для клієнта, прозорість та контроль, а також зменшення внутрішніх і зовнішніх ризиків. Особливістю Agile-моделей SDLC є їх ітеративний характер. Більш детально наведені переваги Agile-моделей будуть розглянуті нижче.

Ознайомлення і опрацювання на практиці здобувачами першого (бакалаврського) рівня вищої освіти галузі інформаційних технологій сучасних Agile-моделей є важливим завданням для майбутніх IT-фахівців. Для виконання цього завдання було обрано найпопулярнішу Agile-модель SCRUM, яку пропонується застосувати під час ітераційного і інкрементного розроблення програмного забезпечення для розв'язання задачі багатокритеріальної оптимізації на основі теоретико-ігрового підходу.

1. Постановка задачі

У статті буде обґрунтовано можливість використання гнучкої Scrum-моделі життєвого циклу розроблення програмного забезпечення в навчальному процесі першого (бакалаврьского) рівня вищої освіти для здобувачів галузі «Інформаційні технології», наведено приклад формулювання Product Backlog'у і змісту спринтів для опанування здобувачами Scrum-процесу під час ітеративного і інкрементного розроблення програмного забезпечення для розв'язання задачі багатокритеріальної оптимізації на основі теоретико-ігрового підходу.

2. Особливості Agile-методології SDLC

Agile-методологія – це загальний підхід до розроблення програмного забезпечення, який базується на гнучкості, швидкій адаптації до змін і постійному вдосконаленні. Основні принципи Agile-методології прописані в документі «Agile Manifesto», акцент в якому зроблено саме на «людях та співпраці», «працюючому продукті», «співпраці з замовником», «готовності до змін». Тобто ця методологія орієнтована на тісну взаємодію з клієнтами і часті випуски працюючого програмного забезпечення.

Наведемо принципові особливості Agile-методології [1, 2]:

1) ітеративний підхід: процес розроблення розбивається на короткі цикли або ітерації, які зазвичай тривають від двох до чотирьох тижнів; кожна ітерація завершується робочим продуктом, який можна демонструвати клієнтам;

2) інкрементне розроблення: продукт розробляється поетапно, з кожною ітерацією додаються нові функції або вдосконалюються існуючі; це дозволяє отримувати зворотний зв'язок від користувачів і робити необхідні корективи на ранніх етапах;

3) тісна взаємодія з клієнтами: клієнти активно залучені в процес розроблення, що дозволяє точно розуміти їхні потреби та швидко реагувати на зміни в вимогах;

4) крос-функціональність команд: команди зазвичай складаються з фахівців різних напрямків (розробники, тестувальники, аналітики та ін.), що сприяє більш ефективній співпраці та швидкому вирішенню проблем;

5) самоорганізація команд: команди мають високу ступінь автономії та відповідальності, що дозволяє їм приймати рішення без необхідності очікувати схвалення від керівництва;

6) постійне вдосконалення: Agile-методологія передбачає регулярні ретроспективи, де команда аналізує свою роботу, виявляє проблеми та шукає способи їх вирішення для покращення подальшого процесу розроблення;

7) фокус на цінностях для клієнта: основною метою є створення продукту, який приносить максимальну цінність для клієнта (кінцевого користувача або замовника), це досягається шляхом пріоритизації вимог та зосередження на найбільш важливих функціях;

Назвемо основні моделі SDLC, що базуються на принципах Agile-методології: Scrum, Kanban, Lean, Extreme Programming (XP), Crystal та інші, в тому числі і їх поєдання, наприклад, Scrumban.

Зазначимо, що поєднання різних Agile-методологій дозволяє одночасно використовувати їх найкращі практики (best practices), що в свою чергу призводить до підвищення ефективності та якості розроблення програмного забезпечення.

3. Особливості SCRUM-моделі SDLC

Scrum [3–6] – це одна з найпопулярніших Agile-моделей SDLC, яку також називають і фреймворком для управління програмними проєктами.

Scrum, як і будь-яка Agile-модель SDLC, є ітеративною моделлю. Які ж основні особливості ітеративного підходу в Scrum? Розроблення програмного продукту здійснюється через повторювані ітерації (рис. 1), які в Scrum називаються спринтами (sprints). Кожен спринт триває фіксований період часу, зазвичай від двох до чотирьох тижнів. Довжина всіх спринтів має бути однаковою і фіксованою, щоб забезпечити передбачуваність і стабільність у процесі розроблення.

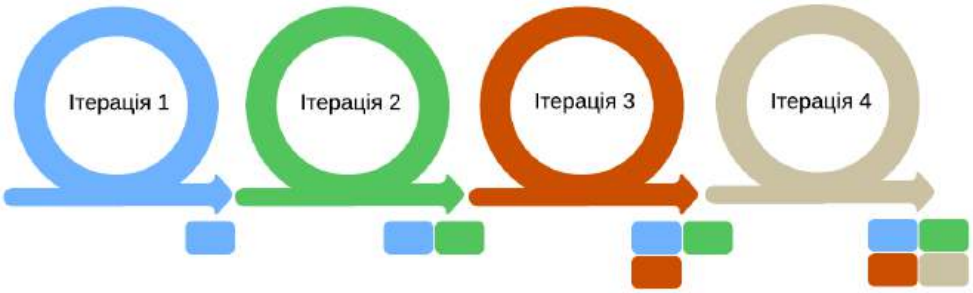

Рис. 1 Ітеративність і інкрементність Scrum-процесу

У Scrum-процесі виділяють учасників з наступними ролями:

1) Product Owner, який відповідає за максимізацію цінності продукту, а також за управління Product Backlog'ом;

2) Scrum Master, який забезпечує правильне використання всіх Scrum-церемоній, а також допомагає команді усувати будь-які перешкоди;

3) самоорганізована команда розробки, яка виконує завдання і несе відповідальність за результат.

Розглянемо три основних артефакти Scrum-процесу: Product Backlog, Sprint Backlog і Increment.

Product Backlog є впорядкованим списком усіх робіт, що необхідно виконати для створення програмного продукту. Product Backlog є динамічним документом, який постійно оновлюється і коригується Product Owner'ом, адже саме він відповідає за його управління та підтримку, забезпечуючи його актуальність і відповідність цілям проєкту. Елементи Product Backlog'у найчастіше представлені у вигляді user stories, задач, дефектів або можуть використовуватися і інші форми представлення. Також важливим моментом існування Product Backlog'у є його пріоритизація, тобто його впорядкованість Product Owner'ом за пріоритетом, де найбільш важливі елементи знаходяться зверху списку, а значить мають бути виконані першими.

Sprint Backlog є набором елементів з Product Backlog'у, оцінених за трудомісткістю і відібраних командою для виконання в поточному спринті під час його планування. Кожен елемент Sprint Backlog'у деталізується до рівня конкретних завдань, які необхідно виконати для його завершення.

Increment є сукупністю усіх завершених елементів Product Backlog'у, виконаних під час спринту. У кінці кожного спринту команда зобов'язана представити працюючу версію програмного забезпечення, яка містить усі додані, готові до випуску функції, що можуть бути представлені замовникам чи

користувачам.

Важливою складовою Scrum-процесу є різні церемонії, за виконанням яких слідкує Scrum Master, а саме:

1) зустріч «Sprint Planning», яка проводиться на початку кожного спринту, під час неї оцінюються і визначаються елементи Product Backlog'у для реалізації в поточному спринту;

2) щоденні короткі зустрічі «Daily Meeting», на яких команда фіксує наявний прогрес і існуючі перешкоди;

3) зустріч «Sprint Review», яка проводиться наприкінці спринту, під час якої Product Owner'у демонструють результат роботи, отриманий під час поточного спринту, а також отримують зворотний зв'язок;

4) зустріч «Sprint Retrospective», яка є ключовою зустріччю наприкінці кожного спринту, під час якої команда може оцінити свою роботу, процеси та взаємодію протягом завершеного спринту, виявити успішні практики, які варто зберегти, і знайти можливості для покращення своєї роботи у майбутніх спринтах.

На рис. 2 наведено візуальне представлення Scrum-процесу в цілому: учасники (Product Owner, Scrum Master, Scrum team), артефакти (Product Backlog, Sprint Backlog, Increment) і церемонії (Strint Planning, Daily Meeting, Sprint Review і Sprint Retrospective.

Рис. 2. Типова SCRUM-модель SDLC

Для опанування інкрементного процесу розроблення програмного забезпечення було вирішено запропонувати здобувачам доволі складну алгоритмічну задачу, яку можна програмно реалізовувати ітеративно, тобто поступово нарощуючи функціональність. Такою задачею стала задача багатокритеріальної оптимізації, яку необхідно розв'язати з використанням теоретико-ігрового підходу.

4. Задача багатокритеріальної оптимізації і її розв'язання з використанням теоретико-ігрового підходу

Наведемо загальну постановку задачі багатокритеріальної оптимізації.

Дано k лінійних критеріїв оптимальності F_t (максимізації або мінімізації лінійних цільових функцій):

$$F_t = \sum_{j=1}^{n} C_{tj} x_j \to extr \quad \left(t = \overline{1,k}\right), \tag{1}$$

а також система з m лінійних обмежень, з яких r обмежень – це обмеження-

нерівності, а інші (m-r) – обмеження-рівності:

$$\sum_{j=1}^{n} a_{ij}x_j \le a_i, \quad i = \overline{1,r}\ ,$$
$$\sum_{j=1}^{n} a_{ij}x_j = a_i, \quad i = \overline{r+1,m}. \tag{2}$$

Необхідно знайти вектор розв'язку $\overline{X} = (\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n)$ при умові, що s змінних x_i є невільними, тобто на їх знак накладено обмеження ≥0, а інші (n-s) змінних є вільними:

$$x_j \ge 0, \quad j = \overline{1,s}, \quad s \le n \tag{3}$$

Метод розв'язання задачі багатокритеріальної оптимізації з довільною кількістю лінійних критеріїв оптимізації полягає у розв'язанні однокритеріальних задач у вигляді задач лінійного програмування окремо для кожного критерію оптимізації та подальшому отриманні компромісного розв'язку, який відповідає мінімальному сумарному незадоволенню всім k критеріям оптимальності.

Компромісний розв'язок шукають у вигляді:

$$\overline{X}_{\text{компр}} = \sum_{t=1}^{k} \lambda_t \overline{X}_t^{(o)}, \qquad \sum_{t=1}^{k} \lambda_t = 1, \tag{4}$$

де $\overline{X}_t^{(o)}$ – оптимальний розв'язок для критерію оптимальності F_t;

λ_t – ваговий коефіцієнт для критерію оптимальності F_t.

Задача пошуку вагових коефіцієнтів компромісного розв'язку формулюється у вигляді ігрової задачі та розв'язується як пара двоїстих задач лінійного програмування.

Наведемо загальний алгоритм розв'язання задачі багатокритеріальної оптимізації з використанням теоретико-ігрового підходу, описаний у [7]:

Крок 1: Розв'язати за допомогою симплекс-методу k однокритеріальних задач у вигляді задачі лінійного програмування, знайти k оптимальних розв'язків.

Крок 2: Побудувати матрицю мір неоптимальності кожного знайденого оптимального розв'язку до інших критеріїв оптимальності (цільових функцій).

Крок 3: Сформулювати ігрову матричну задачу, де стратегіями поведінки першого гравця будуть виступати оптимальні розв'язки для кожного критерію оптимальності (цільової функції), а стратегіями поведінки другого гравця – самі критерії оптимальності (цільові функції).

Крок 4: Знайти розв'язок матричної гри у змішаних стратегіях, змішана стратегія для першого гравця буде дорівнювати ваговим коефіцієнтам λ_t.

Крок 5: Розрахувати значення вектору компромісного розв'язку за формулою (4).

5. Реалізація SCRUM-моделі SDLC для інкрементної програмної реалізації задачі багатокритеріальної оптимізації

Застосуємо основні ролі, артефакти і церемонії Scrum-процесу для інкрементної програмної реалізації описаного вище загального алгоритму розв'язання задачі багатокритеріальної оптимізації з використанням теоретико-ігрового підходу.

Product Owner'ом, який буде формувати Product Backlog і якому будуть демонструватися результати кожного спринту команди на зустрічі Sprint Review,

буде виступати викладач.

Здобувачі діляться на невеликі крос-функціональні команди і самостійно розподіляють ролі у Scrum-команді, в тому числу і призначають Scrum Master'а, який буде слідкувати за дотриманням всіх церемоній Scrum-процесу. Для збільшення ефективності навчального процесу пропонується з деякою періодичністю змінювати ролі учасників Scrum-команди (наприклад, кожні три спринти), але не змінювати склад команди, що забезпечить кожному здобувачу можливість виконати кожну роль.

Як Product Owner сформуємо єдиний для всіх команд Product Backlog, тобто перелічимо основні функціональні вимоги і впорядкуємо їх в порядку реалізації, тобто пріоритизуємо його (табл. 1). Кожна команда отримає копію сформованого Product Backlog'у. Присутність Product Owner'а, тобто викладача, є обов'язковою на таких зустрічах, як «Sprint Planning» і «Sprint Review», що легко можна реалізувати, наприклад, через використання сесійних залів (кімнат) у програмі для організації відеоконференцій Zoom, коли за кожною командою закріплюється відповідна зала, а викладач може переключатися між кімнатами.

Таблиця 1

Формування Product Backlog

	Функціональні вимоги
FR1.	Має бути реалізовано синтаксичний аналіз (парсинг) вхідних даних, а також заповнення і виведення симплекс-таблиць як на екран, так і у файл *.txt.
FR2.	Має бути реалізовано виконання для симплекс-таблиці процедури модифікованого жорданового виключення із заданим розв'язувальним елементом. Протокол розрахунку має генеруватися як на екран, так і у файл *.txt.
FR3.	Має бути реалізовано пошук опорного розв'язку задачі лінійного програмування. Протокол розрахунку має генеруватися як на екран, так і у файл *.txt.
FR4.	Має бути реалізовано пошук оптимального розв'язку задачі лінійного програмування з однорідної системою обмежень у вигляді нерівностей. Протокол розрахунку має генеруватися як на екран, так і у файл *.txt.
FR5.	Має бути реалізовано підготовчий етап видалення нульових рядків (0-рядків) перед пошуком опорного розв'язку задачі лінійного програмування зі змішаною системою обмежень. Протокол розрахунку має генеруватися як на екран, так і у файл *.txt.
FR6.	Має бути реалізовано підготовчий етап видалення вільних змінних перед видаленням 0-рядків, а також обрахунок видалених змінних для їх повернення в оптимальний розв'язок задачі лінійного програмування. Протокол розрахунку має генеруватися як на екран, так і у файл *.txt.
FR7.	Має бути реалізовано формулювання пари двоїстих задач і пошук розв'язку пари двоїстих задач лінійного програмування на парній симплекс-таблиці. Протокол розрахунку має генеруватися як на екран, так і у файл *.txt.
FR8.	Має бути реалізовано пошук змішаних стратегій гравців матричної гри шляхом формулювання пари двоїстих задач і їх розв'язання. Протокол розрахунку має генеруватися як на екран, так і у файл *.txt.

Продовження таблиці 1

	Функціональні вимоги
FR9.	Має бути реалізовано пошук оптимальних розв'язків k задач лінійного програмування з їх збереженням. Протокол розрахунку має генеруватися як на екран, так і у файл *.txt.
FR10.	Має бути реалізовано побудову матриці неоптимальності кожного знайденого оптимального розв'язку до інших критеріїв оптимальності (цільових функцій). Протокол розрахунку має генеруватися як на екран, так і у файл *.txt.
FR11.	Має бути реалізовано формулювання і розв'язок матричної гри на основі матриці неоптимальності, де стратегіями поведінки першого гравця будуть виступати оптимальні розв'язки, а стратегіями другого гравця – критерії оптимальності (цільові функції). Протокол розрахунку має генеруватися як на екран, так і у файл *.txt.
FR12.	Має бути реалізовано встановлення значень вагових коефіцієнтів оптимальних розв'язків і обрахунок компонентів вектору компромісного розв'язку задачі багатокритеріальної оптимізації. Протокол розрахунку має генеруватися як на екран, так і у файл *.txt.

Під час зустрічі «Sprint Planning» під наглядом Scrum Master'а команда має обрати саму верхню функціональну вимогу, розбити її на окремі завдання (tasks) і розподілити завдання між учасниками. У даному випадку під час планування спринту не використовується оцінювання функціональних вимог, яке є зазвичай важливим етапом цієї зустрічі, і допомагає команді зрозуміти обсяг необхідних робіт і забезпечує реалістичність планів на спринт. Вважається, що під час одного спринту, тривалість якого через обмеженість академічного семестру обирається рівною 1 тижню, команда зможе реалізувати, протестувати і виправити помилки для однієї функціональної вимоги.

На рис. 3 наведено приклад результату розроблення програмного забезпечення розв'язання задачі багатокритеріальної оптимізації на основі теоретико-ігрового підходу, який було отримано після завершення 12-ти спринтів.

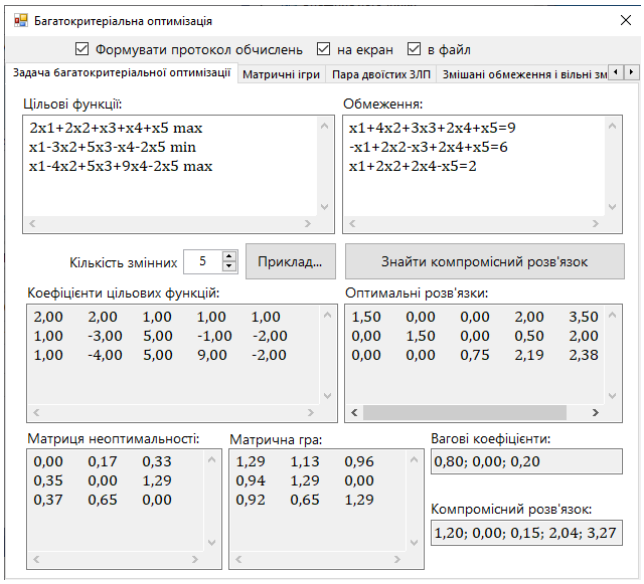

Рис. 3. Приклад результату розроблення програмного забезпечення розв'язання задачі багатокритеріальної оптимізації з використанням Scrum-процесу

Нижче буде наведено автоматично згенерований протокол обчислень, який було отримано за допомогою розробленого програмного забезпечення (проміжкові етапи розв'язання задач лінійного програмування було вилучені з протоколу для скорочення його довжини):

Пошук оптимального розв'язку **першої** задачі лінійного програмування:

Вхідна симплекс-таблиця:

	-x1	-x2	-x3	-x4	-x5	1
0 =	-1,00	2,00	-1,00	2,00	1,00	6,00
0 =	1,00	4,00	3,00	2,00	1,00	9,00
0 =	1,00	2,00	0,00	2,00	-1,00	2,00
F1 =	-2,00	-2,00	-1,00	-1,00	-1,00	0,00

Опорний розв'язок: X = (0,00; 1,50; 0,00; 0,50; 2,00).
Оптимальний розв'язок: X1* = (1,50; 0,00; 0,00; 2,00; 3,50).
Max (F1) = 8,50.

Пошук оптимального розв'язку **другої** задачі лінійного програмування:

Вхідна симплекс-таблиця:

	-x1	-x2	-x3	-x4	-x5	1
0 =	-1,00	2,00	-1,00	2,00	1,00	6,00
0 =	1,00	4,00	3,00	2,00	1,00	9,00
0 =	1,00	2,00	0,00	2,00	-1,00	2,00
F2 =	1,00	-3,00	5,00	-1,00	-2,00	0,00

Опорний розв'язок: X = (0,00; 1,50; 0,00; 0,50; 2,00).
Оптимальний розв'язок: X2* = (0,00; 1,50; 0,00; 0,50; 2,00).
Min (F2) = -9,00.

Пошук оптимального розв'язку **третьої** задачі лінійного програмування:

Вхідна симплекс-таблиця:

	-x1	-x2	-x3	-x4	-x5	1
0 =	-1,00	2,00	-1,00	2,00	1,00	6,00
0 =	1,00	4,00	3,00	2,00	1,00	9,00
0 =	1,00	2,00	0,00	2,00	-1,00	2,00
F3 =	-1,00	4,00	-5,00	-9,00	2,00	0,00

Опорний розв'язок: X = (0,00; 1,50; 0,00; 0,50; 2,00).
Оптимальний розв'язок: X3* = (0,00; 0,00; 0,75; 2,19; 2,38).
Max (F3) = 18,69.

Отримали k=3 оптимальних векторів:

	x1	x2	x3	x4	x5
X1*:	1,50	0,00	0,00	2,00	3,50
X2*:	0,00	1,50	0,00	0,50	2,00
X3*:	0,00	0,00	0,75	2,19	2,38

Матриця коефіцієнтів цільових функцій:

	x1	x2	x3	x4	x5
C1:	2,00	2,00	1,00	1,00	1,00
C2:	1,00	-3,00	5,00	-1,00	-2,00
C3:	1,00	-4,00	5,00	9,00	-2,00

Пошук матриці неоптимальних розв'язків:

0,00	0,17	0,33
0,35	0,00	1,29
0,37	0,65	0,00

Матрична гра A:

1,29	1,13	0,96
0,94	1,29	0,00
0,92	0,65	1,29

Пошук сідлової точки: нижня ціна гри: A[1, 3] = 0,96; верхня ціна гри: A[1, 1] = 1,29.
Сідлову точку матричної гри не знайдено!

Розв'язання матричної гри симплекс-методом:

Постановка **прямої** задачі лінійного програмування:

Z = q1 + q2 + q3 -> max

при обмеженнях:

1,29 * q1 + 1,13 * q2 * q3 <= 1
q1 + 1,29 * q2 <= 1
q1 * q2 + 1,29 * q3 <= 1
q1, q2, q3 >= 0

Постановка **двоїстої** задачі лінійного програмування:

W = p1 + p2 + p3 -> min

при обмеженнях:

1,29 * p1 * p2 * p3 >= 1
1,13 * p1 + 1,29 * p2 * p3 >= 1
p1 + 1,29 * p3 >= 1
p1, p2, p3 >= 0

Симплекс-таблиця:

	t1, -q1	t2, -q2	t3, -q3	W, 1
p1	r1 = 1,29	1,13	0,96	1,00
p2	r2 = 0,94	1,29	0,00	1,00
p3	r3 = 0,92	0,65	1,29	1,00
1	F = -1,00	-1,00	-1,00	0,00

Оптимальний розв'язок прямої задачі Z: Q* = (0,00; 0,40; 0,57).
Оптимальний розв'язок двоїстої задачі W: P* = (0,77; 0,00; 0,20).
Max(Z) = Min(W) = 1,03.

Розрахунок змішаних стратегій гравців:

Змішана стратегія 1-го гравця: 0,80; 0,00; 0,20.
Змішана Стратегія 2-го гравця: 0,00; 0,41; 0,59.

Вагові коефіцієнти оптимальних розв'язків: 0,80; 0,00; 0,20.

Компромісний розв'язок:

	x1	x2	x3	x4	x5
X(компр):	1,20	0,00	0,15	2,04	3,27

Таким чином за дванадцять спринтів було реалізовано інкрементний процес розроблення програмного забезпечення для розв'язання складної алгоритмічну задачі на прикладі задачі багатокритеріальної оптимізації, яку пропонувалося розв'язати з використанням теоретико-ігрового підходу.

6. Висновки

Обґрунтовано можливість використання гнучкої Scrum-моделі життєвого циклу розроблення програмного забезпечення в навчальному процесі першого (бакалаврського) рівня вищої освіти для здобувачів галузі «Інформаційні технології». Наведено приклад формулювання Product Backlog'у для розроблення програмного забезпечення для розв'язання задачі багатокритеріальної оптимізації з використанням теоретико-ігрового підходу, а також наведено приклад інкрементної реалізації програмного забезпечення, що відповідає заданому Product Backlog'у.

Список літератури

1. Alsaqqa, S. Agile Software Development: Methodologies and Trends. / S. Alsaqqa, S. Sawalha, H. Abdel-Nabi // International Journal of Interactive Mobile Technologies. – 2020. – Vol. 14, no. 11. – P. 246–270. – Режим доступу : https://doi.org/10.3991/ijim. v14i11.13269.

2. Кіндрат, О. Agile-методи для ефективної та продуктивної імплементації ІТ-продукту / О. Кіндрат, Г. Дутка // Наукові записки Львівського університету бізнесу та права. – 2021. – Вип. 28, С. 149–157. – Режим доступу : http://dx.doi.org/ 10.5281/zenodo.5269131.

3. Valpadasu, H. Scrum: an effective software development Agile tool / Valpadasu H., Sravanthi T., Kumar S., Padmaja Ch., Krishna C., Mahender K. // International Conference on Recent Advancements in Engineering and Management (ICRAEM-2020), IOP Conference Series: Materials Science and Engineering, 9-10 October, 2020, Warangal, India. – 2020. – Vol. 981. – Режим доступу : https://doi.org/ 10.1088/1757-899X/981/2/022060.

4. Verwijs, Ch. A Theory of Scrum Team Effectiveness / Ch. Verwijs, D. Russo // ACM Transactions on Software Engineering and Methodology. – 2023. – Vol. 32, Issue 3, article no. 74. – P. 1-51. – Режим доступу : https://doi.org/10.1145/3571849.

5. Alami, A. How Scrum adds value to achieving software quality? / A. Alami, O. Krancher // Empirical Software Engineering. – 2022. – Vol. 27, article no. 165. – P. 1-68. – Режим доступу : https://doi.org/10.1007/s10664-022-10208-4.

6. Fernandes, S. Improving the Performance of Student Teams in Project-

Based Learning with Scrum. / S. Fernandes, J. Dinis-Carvalho, A. T. Ferreira-Oliveira // Education Sciences. – 2021. – Vol. 11, Issue 8. – Режим доступу : https://doi.org/10.3390/educsci11080444.

7. Левин, С. В. Теоретико-игровой подход к решению многокритериальной задачи о назначениях / С. В. Левин, А. А. Петрик // Открытые информационные и компьютерные интегрированные технологии. – 2011. – № 50. – С. 103-110. – Режим доступу : http://dspace.library.khai.edu/xmlui/ handle/123456789/4667.

References

1. Alsaqqa, S. Agile Software Development: Methodologies and Trends. / S. Alsaqqa, S. Sawalha, H. Abdel-Nabi // International Journal of Interactive Mobile Technologies. – 2020. – Vol. 14, no. 11. – P. 246–270. – Mode of access : https://doi.org/10.3991/ijim. v14i11.13269.

2. Kindrat, O. Agile-metody dlia efektyvnoi ta produktyvnoi implementatsii IT-produkty / O. Kindrat, H. Dutka // Naukovi zapysky Lvivskoho universytetu biznesu ta prava. – 2021. – Vyp. 28, S. 149–157. – Mode of access : http://dx.doi.org/10.5281/zenodo.5269131.

3. Valpadasu, H. Scrum: an effective software development Agile tool / Valpadasu H., Sravanthi T., Kumar S., Padmaja Ch., Krishna C., Mahender K. // International Conference on Recent Advancements in Engineering and Management (ICRAEM-2020), IOP Conference Series: Materials Science and Engineering, 9-10 October, 2020, Warangal, India. – 2020. – Vol. 981. – Mode of access : https://doi.org/ 10.1088/1757-899X/981/2/022060.

4. Verwijs, Ch. A Theory of Scrum Team Effectiveness / Ch. Verwijs, D. Russo // ACM Transactions on Software Engineering and Methodology. – 2023. – Vol. 32, Issue 3, article no. 74. – P. 1-51. Mode of access : https://doi.org/10.1145/3571849.

5. Alami, A. How Scrum adds value to achieving software quality? / A. Alami, O. Krancher // Empirical Software Engineering. – 2022. – Vol. 27, article no. 165. – P. 1–68. – Mode of access : https://doi.org/10.1007/s10664-022-10208-4.

6. Fernandes, S. Improving the Performance of Student Teams in Project-Based Learning with Scrum. / S. Fernandes, J. Dinis-Carvalho, A. T. Ferreira-Oliveira // Education Sciences. – 2021. – Vol. 11, Issue 8. – Mode of access : https://doi.org/10.3390/educsci11080444.

7. Levin, S. V. Teoretiko-igrovoy podkhod k resheniyu mnogokriterial’noy zadachi o naznacheniyakh / S. V. Levin, A. A. Petrik // Otkrytye informatsionnye i komp’yuternye integrirovannye tekhnologii. – 2011. – № 50. – S. 103-110. – Mode of access : http://dspace.library.khai.edu/xmlui/handle/123456789/4667.

Надійшла до редакції 23.08.2024, розглянута на редколегії 23.08.2024

Incremental approach to planning student work on a collaborative software project

As of today, well-known product and outsourcing IT companies use Agile models for the software development life cycle (SDLC), the advantages of which include rapid adaptation to changes, continuous feedback, improved product quality, increased team engagement, focus on customer value, transparency and control, as well as the reduction of internal and external risks. All Agile models are iterative,

meaning the development process is divided into separate, similar iterations, which have different names in different models. Familiarization with and practical application of modern Agile models by students at the first (bachelor's) level of higher education in the field of information technology is an important task for future IT specialists. To achieve this task, the most popular Agile model, Scrum, was chosen, which is also incremental, meaning that each iteration delivers additional functionality that is ready for use. The main roles of participants and their responsibilities, artifacts, and ceremonies of the Scrum process are reviewed. The article proposes applying the Scrum model for iterative and incremental software development that will solve a multi-criteria optimization problem using a game-theoretic approach. The formulation and general algorithm for solving the multi-criteria optimization problem using a game-theoretic approach are presented, as well as an example of formulating the Product Backlog, which contains twelve functional requirements to be implemented by the student Scrum team over twelve iterations (sprints). The recommended process for dividing students into Scrum teams and assigning roles to participants is described. An example of incremental implementation based on the Scrum model of the SDLC for solving the multi-criteria optimization problem using a game-theoretic approach is also provided.

Key words: software development; software development lifecycle; Agile; SCRUM; multi-criteria optimization.

Відомості про авторів:

Шевченко Ілона Володимірівна – канд. техн. наук, доцент кафедри інженерії програмного забезпечення, Націон. аерокосм. ун-т ім. М. Є. Жуковського «Харківський авіаційний інститут». Електронна пошта: i.shevchenko@khai.edu, ORCID 0000-0003-0100-0726.

Лучшева Оксана Вадимівна – старший викладач кафедри інженерії програмного забезпечення, Націон. аерокосм. ун-т ім. М. Є. Жуковського «Харківський авіаційний інститут». Електронна пошта: o.luchsheva@khai.edu, ORCID 0000-0003-3855-2815.

About the Authors:

Shevchenko Ilona – Ph.D, associate professor at the software engineering department, National aerospace university «Kharkiv Aviation Institute». Email: ilona.shevchenko@gmail.com, ORCID 0000-0003-0100-0726.

Luchsheva Oksana – senior lecturer at the software engineering department, National aerospace university «Kharkiv Aviation Institute». Email: o.luchsheva@khai.edu, ORCID 0000-0003-3855-2815.