DOI: https://doi.org/10.26565/2304-6201-2025-65-04

УДК (UDC) 004.4:005.8:005.3

Лисицький Костянтин Євгенович *к.т.н., доцент, доцент кафедри математичного моделювання та аналізу даних; Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 4, Харків-22, Україна, 61022; e-mail: constantin.lisickiy@karazin.ua;* **https://orcid.org/0000-0002-7772-3376**

Солдатенко Денис Сергійович *студент; Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 4, Харків-22, Україна, 61022; e-mail: soldatenko2020ks13@student.karazin.ua;* **https://orcid.org/0009-0001-0486-0265**

Моделювання структури ролей в системі управління проєктними завданнями

Актуальність. У сучасних інформаційних системах управління проєктами чітке структурування ролей є критично важливим для ефективної організації командної роботи. Методології Scrum, XP та DSDM передбачають розподіл повноважень і доступу до ресурсів відповідно до функціональних ролей учасників. Це особливо актуально в умовах масштабних проєктів із багаторівневою структурою доступу, де правильне моделювання ролей сприяє підвищенню продуктивності, безпеки та контрольованості процесів розробки. Недостатня формалізація ролей може призвести до дублювання функцій, конфліктів у командах та зниження загальної ефективності управління. Тому розробка гнучкої та адаптивної моделі розподілу ролей є важливим етапом при створенні сучасних систем планування та контролю завдань.
Мета. Метою статті є моделювання структури ролей у системі управління проєктними завданнями з урахуванням основних методологій управління проєктами, таких як Scrum, XP, DSDM. У статті аналізуються ключові ролі та обов'язки в рамках кожної з цих методологій, а також порівняння цих моделей з точки зору їх придатності для системи планування задач. Крім того, стаття має на меті дослідити, яка з моделей є найбільш оптимальною для організації задач у системах з багаторівневим доступом.
Методи дослідження. Порівняльний аналіз, моделювання.
Результати. Проведено порівняльний аналіз трьох методологій управління проєктами — Scrum, XP та DSDM — з метою визначення їхньої ефективності в контексті побудови системи управління завданнями. На основі структури ролей, запропонованої у Scrum, розроблено багаторівневу модель розподілу ролей у команді, яка враховує рівень кваліфікації кожного учасника. Запропоновану модель було вдосконалено шляхом введення додаткових ролей, що дало змогу краще адаптувати систему до реальних умов роботи сучасних IT-команд. Для кожної ролі були визначені зони відповідальності та рівні доступу до функціоналу системи, що дозволяє забезпечити гнучке керування процесами, підвищену безпеку та зручну масштабованість. Окрему увагу приділено деталізації ролей розробників залежно від їхнього професійного рівня (молодший, середній, старший), що позитивно впливає на ефективність командної взаємодії, контроль якості виконаних завдань та процес наставництва у команді.
Висновки. Розроблено модель структури ролей у системі управління проєктними завданнями на основі Scrum з урахуванням розширень, що відповідають реаліям сучасної IT-індустрії. Визначено вплив кожної ролі на функціонування системи управління та виявлено, що саме поєднання базових ролей Scrum з додатковими спеціалізованими ролями дозволяє формувати стійку багаторівневу систему, здатну адаптуватися до змін середовища проєкту.

Ключові слова: *роль, структура ролей, система управління завданнями, Agile-методології, права доступу.*

Як цитувати: Лисицький К. Є., Солдатенко Д. С. Моделювання структури ролей в системі управління проєктними завданнями. *Вісник Харківського національного університету імені В. Н. Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління*. 2025. вип. 65. С. 46-56. https://doi.org/10.26565/2304-6201-2025-65-04

How to quote: Lysytskyi K. Y., Soldatenko D. S. “ Modeling the structure of roles in a project task management system”, *Bulletin of V. N. Karazin Kharkiv National University, series Mathematical modelling. Information technology. Automated control systems*, vol. 65, pp. 46-56, 2025. https://doi.org/10.26565/2304-6201-2025-65-05[in Ukrainian]

1 Вступ

У сучасних умовах розвитку інформаційних технологій системи управління проєктами відіграють ключову роль у забезпеченні ефективної організації колективної праці [1]. Все більше компаній та команд переходять до використання проєктно-орієнтованого підходу, де важливою складовою стає чіткий розподіл обов’язків, визначення зон відповідальності та контроль за

виконанням завдань. Саме тому структура ролей у таких системах має суттєве значення для підтримки дисципліни, ефективної комунікації та керованості робочих процесів.

Однією з основних вимог до сучасної системи управління проєктними задачами є гнучкість у визначенні прав доступу до інформації, інструментів та функціоналу відповідно до ролі, яку виконує користувач у рамках проєкту [2]. Це дозволяє не лише забезпечити безпечне середовище для виконання завдань, а й оптимізувати розподіл навантаження, зменшуючи кількість помилок, дублювання функцій та конфліктів у команді.

Сьогодні існує низка підходів до формування командної структури у проєктному середовищі, кожен з яких пропонує власну модель ролей та принципи взаємодії. Вибір тієї чи іншої моделі значною мірою залежить від розміру команди, специфіки задач, рівня формалізації процесів та організаційної культури. Проте, незалежно від методології, чітке моделювання ролей залишається критичним фактором для досягнення прозорості в управлінні, підвищення продуктивності та адаптивності до змінних умов середовища [3].

У цій статті основну увагу приділено вивченню підходів до побудови структур ролей у командній роботі, аналізу переваг та обмежень наявних моделей, а також визначенню принципів, які дозволяють забезпечити баланс між централізованим контролем та автономією учасників проєкту. Дослідження спрямоване на формування концептуальної моделі, що може бути використана при розробці програмних систем планування задач у багаторівневому командному середовищі.

2 Теоретичні засади моделювання ролей в управлінні проєктами

Управління проєктами як комплексна дисципліна охоплює широкий спектр діяльності, спрямованої на досягнення визначених цілей у встановлені терміни та з урахуванням наявних ресурсів. Однією із фундаментальних ознак ефективного управління є чітке визначення та розподіл відповідальності між учасниками проєктної команди [4]. Саме тут на перший план виходить концепція ролі.

У контексті управління проєктами, роль являє собою не просто формальну посаду, а скоріше динамічний набір очікувань, поведінки, компетенцій та обов’язків, які асоціюються з конкретною функцією або завданням у рамках проєкту. Визначення ролі передбачає окреслення того, що повинна робити людина, як вона повинна це робити, та який рівень відповідальності вона несе за результати своєї діяльності [4, 5]. На відміну від статичної посадової інструкції, роль може змінюватися залежно від фази проєкту, поточних потреб команди та індивідуальних навичок учасників.

У сучасних системах управління проєктами, особливо тих, що базуються на Agile-методологіях, концепція ролі часто тісно пов’язана з принципами самоорганізації та крос-функціональності команд. У таких підходах ролі можуть бути менш ієрархічними та більш орієнтованими на співпрацю та спільне досягнення цілей [6]. Ефективне моделювання ролей у цих контекстах вимагає не лише визначення обов’язків, але й урахування необхідних компетенцій, стилів комунікації та здатності до командної роботи.

Таким чином, концепція ролі в контексті управління проєктами є фундаментальним елементом для побудови ефективних та гнучких організаційних структур. Чітке розуміння сутності ролі, її відмінностей від посади та її значення для комунікації, відповідальності та прийняття рішень є критично важливим для успішної реалізації проєктів будь-якої складності.

3 Аналіз структур ролей в основних методологіях управління проєктами

Сучасні методології управління проєктами, що належать до сімейства Agile, пропонують відмінні від традиційних підходів структури ролей у проєктних командах. У цьому розділі буде проведено детальний аналіз моделей ролей, що використовуються в трьох ключових Agile-методологіях: Scrum, Extreme Programming (XP) та Dynamic Systems Development Method (DSDM), з особливим акцентом на їхніх обов'язках та рівнях доступу в контексті систем управління проєктними завданнями.

Розуміння цих моделей є важливим кроком для подальшого порівняльного аналізу та моделювання оптимальної структури ролей для систем з багаторівневим доступом.

3.1 Модель ролей в Scrum

У основі Scrum-команди знаходяться три ключові ролі, кожна з яких виконує певні функції для забезпечення успішної розробки продукту.

Власник Продукту (Product Owner) є відповідальним формування та управління беклогом продукту, що означає визначення, пріоритезацію та уточнення елементів беклогу. Він також бере участь у плануванні релізів та ітерацій (спринтів), визначаючи цілі спринту та елементи беклогу, які будуть реалізовані в рамках поточного спринту. Крім того, власник продукту є основним комунікатором між командою розробників та зацікавленими сторонами, такими як клієнти, користувачі або керівництво. Він збирає їхні потреби, враховує відгуки та формує їх у вимоги до продукту. Наприкінці кожного спринту власник продукту перевіряє виконану роботу та приймає або відхиляє результати, переконуючись, що робота відповідає визначеним критеріям прийняття [7].

Щодо рівня доступу в системі управління проєктними завданнями, власник продукту, як правило, має найвищі права. Він повинен мати повний доступ на читання та редагування беклогу продукту, включаючи можливість створювати нові елементи, змінювати їх пріоритетність, деталізувати описи тощо. Він також має доступ до інструментів планування спринтів, щоб визначати цілі спринту та обирати елементи беклогу для включення. Для відстеження прогресу, власник продукту має доступ до інформації про стан виконання завдань, оцінки складності та будь-які блокувальні фактори. Крім того, він може залишати коментарі та надавати зворотний зв'язок безпосередньо в системі, а також мати доступ до звітів та аналітики.

Скрам-майстер (Scrum Master) виступає в ролі керівника та менеджера для скрам-команди, допомагаючи їй працювати ефективно та дотримуватися принципів Scrum. Його обов'язки включають планування спринту, щоденні скрам-мітинги, огляд спринту та ретроспектива спринту. Він також відповідає за усунення будь-яких перешкод, які заважають команді досягати цілей спринту, чи то технічні проблеми, або організаційні питання. Крім того, скрам-майстер бере під свою опіку команду, навчаючи її принципам Scrum та допомагаючи самоорганізовуватися та вдосконалювати свої внутрішні процеси [7].

Щодо рівня доступу, скрам-майстер зазвичай має повний доступ на читання беклогу продукту та спринту, щоб розуміти контекст роботи команди. Він може мати можливість редагувати статус завдань, щоб відображати їхній поточний стан, та додавати коментарі для комунікації та фіксації інформації. Скрам-майстер також має доступ до інструментів планування спринтів та може переглядати звіти та метрики, щоб виявляти потенційні проблеми та сприяти вдосконаленню.

Команда Розробників (Development Team) є групою професіоналів, які відповідають за фактичну розробку та поставку продукту наприкінці кожного спринту. Scrum не визначає внутрішніх ролей у команді розробників, оскільки розподіл обов’язків між учасниками здійснюється відповідно до рішень компанії або керівництва, з урахуванням навичок і досвіду кожного фахівця [7].

Рівень доступу для команди розробників у системі управління проєктними завданнями забезпечує їм можливість ефективно виконувати свою роботу. Це включає повний доступ на читання беклогу спринту, можливість оновлювати статус завдань, за які вони відповідають, можливість оцінювати складність завдань та додавати коментарі для комунікації. За потреби, члени команди можуть мати можливість створювати підзадачі для кращого управління своєю роботою та доступ до пов'язаної документації.

3.2 Модель ролей в Extreme Programming (XP)

В основі XP-команди знаходяться кілька ключових ролей, які забезпечують ефективну розробку програмного забезпечення.

Замовник (Customer) в XP відіграє ключову роль, бувши безпосередньо залученим до процесу розробки. Його основні обов'язки включають визначення функціональних вимог до системи у формі історій користувачів, встановлення пріоритетів цих історій для кожної ітерації, а також надання зворотного зв'язку команді розробників щодо розробленого програмного забезпечення. Замовник бере активну участь у плануванні ітерацій, визначаючи, які історії користувачів будуть реалізовані [8].

Щодо рівня доступу в системі управління проєктними завданнями, замовник має можливість переглядати всі історії користувачів, їхні описи, критерії прийняття та поточний статус виконання. Він також має можливість додавати нові історії користувачів та змінювати їхній пріоритет. Замовник повинен мати доступ до інформації про прогрес команди, бачити, які історії користувачів знаходяться в розробці, які завершені, а які ще не розпочаті. У деяких системах

замовнику може бути надано право формально приймати або відхиляти завершені історії користувачів.

Програміст (Programmer) є іншою ключовою роллю, яка відповідальна за проєктування, розробку та тестування програмного коду. Однією з рис XP є практика парного програмування, коли двоє програмістів працюють разом за одним комп'ютером над однією задачею. Це сприяє обміну знаннями, підвищенню якості коду та зменшенню кількості помилок. Програмісти також беруть активну участь у плануванні ітерацій, оцінюючи складність історій користувачів та розбиваючи їх на конкретні завдання. Вони несуть відповідальність за постійну інтеграцію коду та проведення тестування [8].

Щодо рівня доступу, програмісти мають повний доступ на читання всіх елементів беклогу, можливість оновлювати статус завдань, за які відповідають, оцінювати складність завдань та додавати коментарі. Оскільки XP заохочує спільну роботу, всі програмісти в парі можуть мати однакові права на редагування коду та пов'язаних з ним завдань у системі управління проєктними завданнями. Вони також можуть мати можливість створювати нові завдання, що виникають у процесі розробки, та пов'язувати їх з відповідними історіями користувачів.

Тренер (Coach) в XP виконує роль, схожу на Скрам-майстра в Scrum, але з більшим акцентом на технічні практики. Його обов'язки включають навчання команди практикам XP (таким як парне програмування, тестування через розробку, рефакторинг, безперервна інтеграція), полегшення комунікації та співпраці в команді, а також допомогу у розв’язанні проблем. Тренер також стежить за дотриманням принципів XP та сприяє постійному вдосконаленню процесів [8].

Щодо рівня доступу, Тренер зазвичай має повний доступ на читання всієї проєктної інформації, включаючи беклог, плани ітерацій, статуси завдань та звіти про прогрес. Він може мати можливість редагувати статуси завдань, додавати коментарі та нотатки, а також використовувати інструменти для візуалізації прогресу команди. Його рівень доступу спрямований на забезпечення прозорості та підтримку ефективної роботи команди.

Відстежувач (Tracker) є роллю, відповідальною за збір та аналіз метрик проєкту. Його обов'язки включають збір даних про швидкість команди, час виконання завдань, кількість помилок та інші важливі показники. Відстежувач використовує ці дані для виявлення тенденцій, прогнозування майбутнього прогресу та надання зворотного зв'язку команді для поліпшення процесів [8].

Щодо рівня доступу, відстежувач має доступ на читання всіх даних, необхідних для збору метрик, включаючи статуси завдань, оцінки, журнали роботи тощо. Він також має доступ до інструментів для генерації звітів та аналізу даних. Він не має прав на редагування основних елементів беклогу або завдань, але його доступ сфокусований на отриманні інформації для моніторингу та аналізу роботи команди.

3.3 Модель ролей в Dynamic Systems Development Method (DSDM)

DSDM визначає низку ролей, які можна умовно поділити на бізнес-ролі та технічні ролі, а також ролі, що забезпечують управління та підтримку процесу.

Бізнес-спонсор (Business Sponsor) є ключовою бізнес-роллю, яка забезпечує фінансування та підтримку проєкту на високому рівні. Його основні обов'язки включають визначення бізнес-цілей проєкту, забезпечення наявності необхідних ресурсів та прийняття ключових бізнес-рішень. Бізнес-спонсор не є щоденним учасником розробки, але його підтримка є критично важливою для успіху проєкту [9, 10].

Щодо рівня доступу в системі управління проєктними завданнями, бізнес-спонсор зазвичай має високорівневий доступ до інформації про проєкт, включаючи бізнес-вимоги, плани, прогрес, звіти про витрати тощо. Він може мати можливість затверджувати ключові рішення та зміни в проєкті, а також переглядати загальну аналітику проєкту.

Бізнес-амбасадор (Business Ambassador) представляє потреби кінцевих користувачів та бізнес-інтереси на щоденній основі. Його обов'язки включають надання детальних вимог, участь у пріоритезації функціональності, надання зворотного зв'язку щодо розробленого програмного забезпечення та участь у приймальному тестуванні. Бізнес-амбасадор є активним членом команди розробки [9, 10].

Щодо рівня доступу, бізнес-амбасадор має повний доступ на читання та редагування вимог, можливість їх пріоритезації, а також можливість залишати коментарі та надавати зворотний

зв'язок щодо завдань розробки. Він також має доступ до інформації про прогрес розробки та результати тестування.

Технічний координатор (Technical Coordinator) відповідає за забезпечення технічної узгодженості проєкту, підтримку архітектури системи, контроль якості коду. Він є лідером технічної команди та забезпечує прийняття технічних рішень [10].

Щодо рівня доступу, технічний координатор має повний доступ до всієї технічної інформації в системі управління проєктними завданнями, включаючи завдання розробки, технічну документацію, архітектурні рішення, результати тестування та метрики якості коду. Він має право редагувати технічні завдання, встановлювати стандарти та контролювати їхнє дотримання.

Лідер команди (Team Leader) відповідає за щоденне управління командою розробки, планування завдань, моніторинг прогресу, розв’язання проблем та забезпечення ефективної комунікації всередині команди [9, 10].

Щодо рівня доступу, лідер команди має повний доступ на читання всієї інформації, пов'язаної з роботою команди, включаючи вимоги, завдання, статуси, коментарі та прогрес. Він має можливість редагувати статуси завдань, призначати завдання членам команди, керувати планом роботи команди та генерувати звіти про прогрес.

Розробник (Developer) відповідає за проєктування, розробку, тестування та інтеграцію програмного коду. Як і в Scrum, команда розробників в DSDM є крос-функціональною та самоорганізованою [9, 10].

Щодо рівня доступу, розробники мають повний доступ на читання вимог та завдань, за які вони відповідають, можливість оновлювати їхній статус, додавати коментарі та створювати підзадачі. Вони також мають доступ до пов'язаної технічної документації та інструментів розробки.

Крім цих основних ролей, DSDM також може включати інші ролі, такі як *Тестувальник (Tester)*, який відповідає за планування та проведення тестування, документування дефектів та верифікацію їх виправлення [9, 10]. Його рівень доступу в системі управління проєктними завданнями включатиме можливість переглядати вимоги, створювати та редагувати тестові сценарії та випадки, фіксувати результати тестування та відстежувати статус дефектів.

Інтегратор (Integrator) відповідає за об'єднання різних компонентів системи, розроблених окремими членами команди, та забезпечення їхньої сумісної роботи [10]. Його рівень доступу може включати права на перегляд та редагування інформації про компоненти системи, залежності між ними та результати інтеграційного тестування

Фасилітатор (Facilitator) відповідає за організацію та проведення ефективних зустрічей та воркшопів [10]. Його рівень доступу може включати можливість планування зустрічей у системі, фіксування прийнятих рішень та відстеження виконання домовленостей. Кожна з цих ролей матиме відповідний рівень доступу до системи управління проєктними завданнями, що чітко корелює з їхніми обов'язками та потребами в інформації для ефективного виконання покладених на них функцій у проєкті.

4 Порівняльний аналіз моделей ролей для систем управління проєктними завданнями

Згідно з дослідженням Zippia, 71% компаній вже впровадили Agile-методології, що демонструє зростаючу популярність цих методів у сфері розробки програмного забезпечення. Це свідчить про те, що все більше організацій обирають гнучкі підходи до управління проєктами, прагнучи досягти більшої ефективності, адаптивності та швидкості реагування на зміни у вимогах. Крім того, великий успіх Agile-методологій можна побачити у відсотках досягнутих результатів: 64% таких проєктів виявляються успішними, а 60% з них забезпечують зростання доходів компаній [11].

Таблиця 1. Відсоток впровадження різних Agile-методологій серед компаній
Table 1. Percentage of implementation of various Agile methodologies among companies

Фреймворк	Частка впровадження (%)
Scrum	66%
Scrumban	9%
XP Hybrid	9%
Kanban	6%
Iterative	4%
Інші	5%

Згідно з даними в таблиці 1, Scrum є найбільш популярною Agile-методологією, з часткою впровадження 66% [12]. Такий показник є свідченням того, що Scrum не лише широко застосовується в галузі розробки програмного забезпечення, але й демонструє універсальність у реалізації проєктів різного масштабу та складності.

Для порівняння, такі методології як XP Hybrid та Scrumban мають частку впровадження лише по 9%, що свідчить помітно меншу популярність. Хоча вони й мають свої переваги, зокрема в умовах тісної співпраці з клієнтом, вони не демонструють такої ж широкої застосовності, як класичний Scrum. Значно меншу частку займає група інших методологій, включаючи такі як DSDM, які разом становлять приблизно 5%. Це може пояснюватися специфічною спрямованістю цих фреймворків на управління ризиками та змінами, що робить їх менш універсальними та складнішими в інтеграції в стандартні процеси розробки.

Таблиця 2. Переваги, недоліки та приклади застосування Agile-фреймворків
Table 2. Advantages, disadvantages and examples of Agile frameworks application

Фреймворк	**Сфера застосування**	**Переваги**	**Недоліки**
Scrum	Розробка продуктів, зокрема: медичні додатки, вебзастосунки, ігри, розважальні сервіси	Різні ролі сприяють самоуправлінню. Щоденні зустрічі Scrum допомагають відслідковувати призначення завдань. Ретроспективи спринтів усувають слабкі місця і сприяють наданню рекомендацій.	Успіх проєкту залежить від професіоналів та експертів процесу Scrum. Власник продукту та клієнти повинні мати достатньо знань і досвіду роботи з процесом Scrum. Відсутність покриття системних тестів.
XP	Підходить для малих або середніх проєктів, проєктів, які вимагають тісної співпраці між командою розробників і клієнтом. Наприклад, пошукова система Google, веб-сервіси Amazon.	Якість коду та продуктивність покращуються завдяки функції рефакторингу.	Залучення клієнта займає багато часу, що може призвести до стресу. Немає підтримки для розподілених команд, тому потрібне навчання для членів команди.
DSDM	Краще підходить для управління проєктами, контролю ризиків, технік розробки. Наприклад, для розробки фінансових додатків.	Забезпечує швидку розробку додатків, сприяє управлінню проєктами, контролю ризиків та технікам розробки.	Проблеми з адмініструванням під час розробки, не враховує критичність проєкту, немає конкретних вказівок щодо розміру команди та довжини ітерацій.

Згідно з даними в таблиці 2, Scrum є найбільш ефективним фреймворком для побудови системи управління задачами, зокрема завдяки чітко визначеним ролям [13].. Порівняно з XP, де акцент зміщено на безпосередню взаємодію та спільну роботу. Ця ясність у визначенні ролей є ключовою перевагою при проєктуванні багаторівневої системи управління задачами, оскільки дозволяє точно відобразити ієрархію доступу та повноважень. На відміну від DSDM, який хоч і враховує управління змінами та ризиками, проте не надає такого ж рівня деталізації у визначенні прав та обов'язків ролей у контексті виконання конкретних завдань, Scrum забезпечує більш практичну основу для побудови системи управління.

Щодо аналізу ефективності фреймворків, то було використано результати кількісного опитування фахівців галузі інформаційних технологій. Дані, наведені в таблиці 3, базуються на оцінках учасників, зібраних через структуровану анкету. Респонденти — проєктні менеджери, розробники, Scrum-майстри, бізнес-аналітики — представляли понад 30 компаній, які працюють

у сфері розробки ПЗ. Було враховано різноманітні типи проєктів: від стартапів до масштабних корпоративних систем. Респонденти оцінювали рівень задоволення основними аспектами управління проєктом: вартістю, часом, ризиками, якістю та управлінням людськими ресурсами. Отримані числові значення було нормалізовано до відсоткової шкали (0–100%), що дозволяє більш показово порівнювати фреймворки між собою [14].

Таблиця 3. Порівняння Scrum, XP та DSDM за критеріями управління проєктами
Table 3. Comparison of Scrum, XP, and DSDM according to project management criteria

Критерій	Scrum	XP	DSDM
Управління вартістю (Cost Management)	75%	60%	65%
Управління часом (Time Management)	85%	60%	70%
Управління ризиками (Risk Management)	84%	78%	71%
Управління якістю (Quality Management)	83%	65%	75%
Управління людськими ресурсами (HR Management)	87%	70%	75%

Результати аналізу трьох найбільш поширених Agile-методологій — Scrum, XP та DSDM — демонструють помітну перевагу фреймворку Scrum у більшості ключових аспектів управління проєктами. Так, найвищі показники в управлінні часом і людськими ресурсами досягнуто саме в межах Scrum, що пояснюється високим рівнем організованості процесів, структурованою природою спринтів та регулярною командною взаємодією. Завдяки цьому забезпечується стабільне планування, швидка адаптація до змін і постійна координація всередині команди.

Методологія DSDM демонструє відносну збалансованість, зокрема у сфері управління якістю та людськими ресурсами, однак поступається Scrum за більшістю показників. Це пов’язано з тим, що DSDM більше орієнтований на формалізовані бізнес-процеси та має вищу вхідну складність, що потребує ретельнішої підготовки і чітко окреслених ролей ще до початку реалізації.

XP, хоча і відомий своєю гнучкістю та орієнтованістю на потреби клієнта, виявляє нижчі результати у більшості категорій, за винятком управління ризиками. Його перевага у цій сфері над методологією DSDM зумовлена короткими ітераціями, високим рівнем зворотного зв’язку і постійною участю замовника в процесі розробки. Однак, недостатній рівень структурованості у підходах до управління часом та вартістю може ускладнювати реалізацію великих або довгострокових проєктів.

Узагальнюючи результати порівняльного аналізу, можна стверджувати, що Scrum є найбільш адаптивним і збалансованим серед досліджуваних фреймворків. Його гнучка модель, чіткий розподіл відповідальностей та підтримка прозорої командної взаємодії створюють ефективну основу для впровадження до багаторівневої системи управління доступом. У контексті розробки системи управління завданнями з гнучкою ієрархією ролей, саме Scrum забезпечує найкращі передумови для ефективного планування, контролю і реалізації функціоналу відповідно до вимог безпеки, масштабованості та динаміки сучасних IT-проєктів.

5 Моделювання структури ролей для системи управління проєктними завданнями

Вибір моделі ролей, заснованої на фреймворку Scrum, є доцільним рішенням для системи управління завданнями завдяки численним перевагам, які ця структура забезпечує для команд, що працюють в умовах динамічного та гнучкого управління проектами. Як зазначено в розділі 4, в системах, де застосовуються Agile-методології, важливо мати чітко визначені ролі, щоб кожен учасник мав конкретну відповідальність і доступ до необхідних ресурсів для ефективної роботи. У Scrum структура ролей ретельно опрацьована та розподілена, що забезпечує оптимальну взаємодію між учасниками команди та дозволяє кожному з них працювати на своєму місці без перекриття обов'язків або порушення процесу (рис.1) [15].

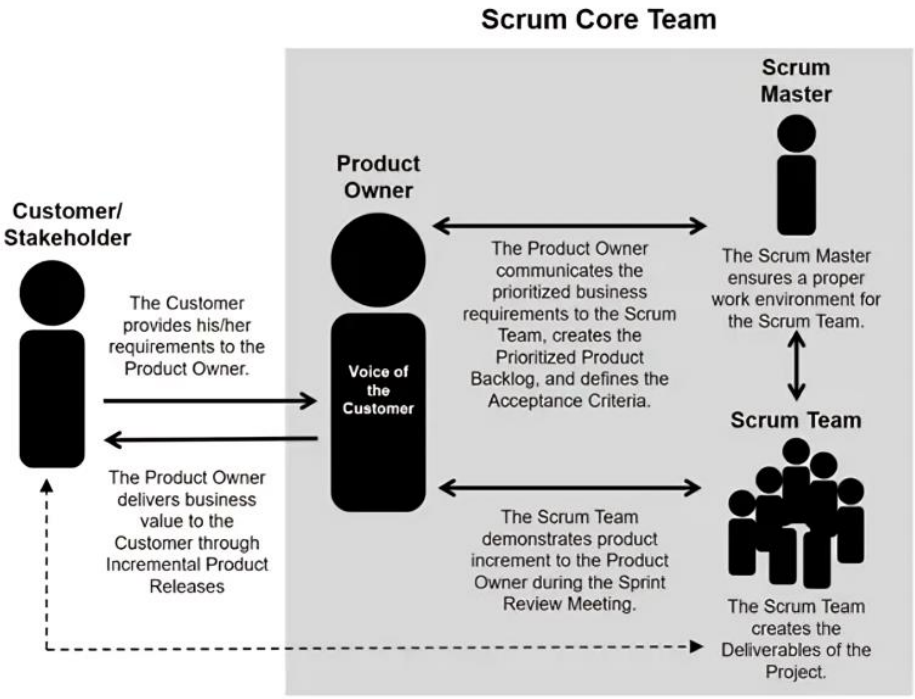

Рисунок 1. Структура ролей та зв'язків у Scrum
Figure 1. Structure of roles and relationships in Scrum

У системі управління завданнями важливо не тільки визначити чіткі ролі, але й встановити відповідні права доступу до кожного завдання, документа або процесу, щоб уникнути конфліктів. Ролі в Scrum чітко визначені та структуровані, що дозволяє спростити управління проектом. До того ж кожна роль має свій рівень доступу до завдань та їхнього виконання, що є основою для створення системи управління завданнями.

Таблиця 4. Оптимальна структура ролей для системи управління завданнями з багаторівневим доступом
Table 4. Optimal role structure for a task management system with multilevel access control

Роль	Опис ролі	Права доступу
Адміністратор (Admin)	Повний контроль над усіма модулями системи, включаючи управління користувачами та правами	Повний доступ до всіх завдань, функцій системи, включаючи створення, редагування, видалення, моніторинг
Власник продукту (Product Owner)	Визначає пріоритети, бачення продукту та вимоги до реалізації	Доступ до всіх завдань, редагування та встановлення пріоритетів, планування спринтів, але без адміністративних дій
Скрам-майстер (Scrum Master)	Координує команду та підтримує Scrum-процеси	Доступ до всіх завдань, редагування технічних та організаційних моментів, контроль процесу
Менеджер (Development Team)	Керує проектними ресурсами та координує роботу команд	Доступ до завдань своєї команди, перегляд, контроль виконання, часткове редагування в межах своєї зони відповідальності
Тімлід (Development Team)	Керує підрозділом розробки, відповідає за розподіл завдань та підтримку команди	Доступ до завдань команди, розподіл обов’язків, редагування технічних задач та контроль виконання
Старший розробник (Development Team)	Відповідає за складні технічні рішення, підтримує менш досвідчених колег	Доступ до технічних завдань, їх виконання та редагування, перегляд чужих завдань для надання зворотного зв’язку

Розробник середнього рівня (Development Team)	Має досвід для самостійного виконання задач середньої складності	Доступ до своїх завдань, виконання та часткове редагування, участь у командних завданнях
Молодший розробник (Development Team)	Виконує прості задачі під наглядом старших колег	Доступ лише до призначених завдань, право їх виконання, обмежене редагування
Фулстек розробник (Development Team)	Працює одночасно над фронтендом і бекендом системи	Повний доступ до задач, що охоплюють як клієнтську, так і серверну частини, їхнє редагування та виконання
Бекенд розробник (Development Team)	Займається серверною логікою та базами даних	Доступ до бекенд-завдань, право їх виконання та редагування
Фронтенд розробник (Development Team)	Працює над користувацьким інтерфейсом	Доступ до фронтенд-завдань, право їх виконання та редагування
Інженер з якості (Development Team)	Проводить автоматизоване та ручне тестування	Доступ до тестових завдань, створення тест-кейсів, перегляд інших завдань для перевірки
Тестувальник (Development Team)	Перевіряє функціональність реалізованих завдань	Доступ до завершених завдань для тестування, додавання коментарів та звітів про помилки

Структура ролей для системи управління завданнями з багаторівневим доступом (табл.4) базується на моделі Scrum, яка, як було обґрунтовано в попередньому розділі, є найбільш ефективною та адаптивною до сучасних вимог проєктного менеджменту. Її ключова перевага — це чіткий розподіл ролей і повноважень, що дозволяє створити команду, здатну ефективно реагувати на зміни та забезпечувати стабільну реалізацію проєкту. Основні ролі Scrum — це *власник продукту (Product Owner)*, *скрам-майстер (Scrum Master)* і *команда розробників (Development Team)*, і саме на них базується фундамент запропонованої структури. Проте, у процесі практичного застосування цієї моделі, виникає необхідність розширити її за рахунок додаткових ролей, які хоч і не входять до оригінального складу Scrum, проте суттєво підсилюють функціональність системи.

Насамперед варто звернути увагу на роль *адміністратора*, яка не передбачена Scrum, однак є критично важливою для системи управління завданнями з точки зору технічного адміністрування, контролю безпеки, управління користувачами, доступом і загальними системними параметрами. Її присутність дозволяє відокремити рівень технічного супроводу від проєктної роботи, що підвищує стабільність системи.

Додаткова роль *менеджера* також не є частиною класичної Scrum-моделі, однак її включення до системи дає змогу реалізувати ефективну взаємодію між кількома командами, що часто необхідно в корпоративному середовищі. Менеджер може не брати участі у щоденних Scrum-активностях, однак його функціонал важливий для зовнішнього управління ресурсами, планування бюджету або інтеграції із бізнес-процесами компанії.

Деталізація розробницької команди через введення ролей *тімліда, старшого, середнього, молодшого розробника*, а також спеціалізацій (*фулстек, фронтенд, бекенд*) дозволяє значно точніше контролювати доступ до завдань залежно від рівня кваліфікації та сфери компетенції. Така деталізація не лише відповідає реальним практикам IT-команд, але й дозволяє гнучко розподіляти обов’язки, уникати дублювання функцій і забезпечувати ефективне наставництво, що є важливим для сталого розвитку команди.

Так само важливими є ролі *інженера з якості* та *тестувальника*, які в класичному Scrum розглядаються як частина кросфункціональної команди. Проте, в реальних умовах проєктів, особливо з високими вимогами до безпеки, виокремлення цих ролей дозволяє побудувати

незалежний процес контролю якості, що знижує ризики дефектів, дозволяє проводити спеціалізовані аудити і забезпечує вищу якість кінцевого продукту.

Таким чином, поєднання базової Scrum-структури з розширеним переліком ролей дозволяє створити систему управління завданнями, яка поєднує переваги формального управління, саморганізованих команд і ролей, адаптованих до реалій сучасного розробницького середовища. Це підвищує як технічну ефективність, так і управлінську контрольованість проєктів різного рівня складності.

6 Висновки

У даній статі проведено порівняльний аналіз різних моделей ролей для систем управління проєктними завданнями, зокрема на основі фреймворків Agile, таких як Scrum, XP та DSDM. Було детально розглянуто структуру ролей у цих методологіях та їхню ефективність у контексті управління завданнями. Результати дослідження показали, що модель ролей Scrum є найбільш підхожою для інтеграції в систему управління завданнями завдяки чітко визначеним ролям, що дозволяє забезпечити ефективну взаємодію між учасниками команди та забезпечити контроль над виконанням завдань.

Зокрема, Scrum забезпечує високу гнучкість і адаптивність завдяки чітко розподіленим ролям: власнику продукту, скрам-майстру та команді розробників. Така структура дозволяє створити самоуправлінську команду, яка здатна оперативно реагувати на зміни та досягати високої продуктивності. У порівнянні з іншими фреймворками, такими як XP і DSDM, Scrum надає більш структурований підхід до управління завданнями та дозволяє ефективно керувати процесом завдяки регулярним зустрічам та ретроспективам.

Крім того, впровадження додаткових ролей, таких як адміністратор та менеджер, а також спеціалізацій серед розробників, значно підвищує функціональність та масштабованість системи. Це дозволяє створити багаторівневу структуру управління з чітким контролем доступу та забезпечити належний рівень безпеки та ефективності в процесі реалізації проєкту.

Таким чином, результати дослідження підтверджують доцільність використання Scrum як основи для побудови системи управління завданнями з багаторівневим доступом. Вибір цієї моделі ролей дозволяє оптимізувати управління проєктами, покращити ефективність командної роботи та забезпечити високу якість виконання завдань у складних і масштабних проєктах.

REFERENCES

1. D. Jefferies, “From deadlines to budgets, project management skills have become vital for everyone – here’s how to make it all easier,” The Guardian. [Online]. Available: https://www.theguardian.com/it-starts-with-smartsheet/2025/jan/31/from-deadlines-to-budgets-project-management-skills-have-become-vital-for-everyone-heres-how-to-make-it-all-easier
2. P. GmbH, “7 central advantages of project management software,” Projektron GmbH, Berlin, 2001-2025, Jan. 27, 2023. [Online]. Available: https://www.projektron.de/en/blog/details/advantages-project-management-software-3728/
3. L. Lockhart, “How to structure and manage a first-rate project team,” Float - Resource Management, Planning & Scheduling Software, Aug. 09, 2024. https://www.float.com/resources/structure-and-manage-project-team
4. Toxigon, “Why role definition matters in team strategies,” Toxigon, Dec. 13, 2024. https://toxigon.com/the-importance-of-role-definition-in-team-strategies
5. M. Levinson, M. Levinson, and M. Levinson, “Defining roles, responsibilities and skills in project staffing plan,” Your Guide to Project Management Best Practices, Jun. 26, 2023. https://mymanagementguide.com/defining-roles-responsibilities-skills-project-staffing-plan/
6. M. Chervenkova, “Agile Team Roles and responsibilities: A complete guide,” Kanban Software for Agile Project Management, Jul. 07, 2022. https://businessmap.io/blog/agile-team-roles
7. Scrum.org, “What is Scrum?,” Scrum.org. https://www.scrum.org/learning-series/what-is-scrum/
8. K. McDonald, “What is Extreme Programming (XP)? | Agile Alliance,” Agile Alliance |, Oct. 18, 2023. https://www.agilealliance.org/glossary/xp
9. The Agile Handbook, “Dynamic Systems Development Method (DSDM): driving Agile project success,” May 06, 2023. https://theagilehandbook.com/blog/dynamic-systems-development-method/

10. Agile Business, “Chapter 7: Roles and Responsibilities.” https://www.agilebusiness.org/dsdm-project-framework/roles-and-responsibilities.html
11. J. Flynn, “16 Amazing Agile Statistics [2023]: What companies use agile methodology,” Zippia, Jun. 28, 2023. https://www.zippia.com/advice/agile-statistics/
12. S. Gadani, U. Mankad, and M. Gahlawat, “Comparative Analysis of Agile Software Development Methodologies: A literature review,” Lecture Notes in Networks and Systems, p. 246, Jan. 2025, doi: 10.1007/978-981-97-9529-1_21.
13. S. Gadani, U. Mankad, and M. Gahlawat, “Comparative Analysis of Agile Software Development Methodologies: A literature review,” Lecture Notes in Networks and Systems, pp. 245–246, Jan. 2025, doi: 10.1007/978-981-97-9529-1_21.
14. I. Ahmed et al., “The Influence of Kanban Agile Methodology on Software Project Management: A Survey method,” 2021 International Conference on Engineering and Emerging Technologies (ICEET), pp. 1–6, Dec. 2024, doi: 10.1109/iceet65156.2024.10913653.
15. SCRUMstudy, “Understanding defined roles and responsibilities in a Scrum ...,” Oct. 21, 2022. https://www.scrumstudy.com/article/understanding-defined-roles-and-responsibilities-in-a-scrum-project

Lysytskyi Kostiantyn *PhD on Technical Sciences, Associate Professor;*
Associate Professor of Mathematical Modeling and Data Analysis;
V.N. Karazin Kharkiv National University 4 Svobody Sq., Kharkiv, 61022, Ukraine
e-mail: constantin.lisickiy@karazin.ua;
https://orcid.org/0000-0002-7772-3376

Soldatenko Denys *student;*
V.N. Karazin Kharkiv National University Svobody Sq 4, Kharkiv, Ukraine, 61022
e-mail: soldatenko2020ks13@student.karazin.ua;
https://orcid.org/0009-0001-0486-0265

Modeling the structure of roles in a project task management system

Relevance. In modern project management information systems, a clear role structure is critical for the effective organization of teamwork. The Scrum, XP, and DSDM methodologies involve the distribution of authority and access to resources according to the functional roles of participants. This is especially relevant in large-scale projects with a multi-level access structure, where proper role modeling helps to increase productivity, security, and controllability of development processes. Insufficient formalization of roles can lead to duplication of functions, conflicts in teams, and a decrease in overall management efficiency. Therefore, the development of a flexible and adaptive model of role distribution is an important step in creating modern systems for planning and controlling tasks.

Objective. The purpose of the article is to model the structure of roles in the project task management system, taking into account the main project management methodologies, such as Scrum, XP, DSDM. The article analyzes the key roles and responsibilities within each of these methodologies, as well as compares these models in terms of their suitability for a task planning system. In addition, the article aims to investigate which of the models is the most optimal for organizing tasks in systems with multi-level access.

Research methods. Comparative analysis, modeling.

Results. A comparative analysis of three project management methodologies - Scrum, XP, and DSDM - was conducted to determine their effectiveness in the context of building a task management system. Based on the role structure proposed in Scrum, a multi-level model of team role distribution was developed, which takes into account the skill level of each participant. The proposed model was improved by introducing additional roles, which allowed the system to better adapt to the real working conditions of modern IT teams. For each role, areas of responsibility and levels of access to the system's functionality were defined, which allows for flexible process management, enhanced security, and convenient scalability. Particular attention is paid to detailing the roles of developers depending on their professional level (junior, middle, senior), which positively affects the effectiveness of team interaction, quality control of tasks performed, and the process of mentoring in the team.

Conclusions. A model of the structure of roles in the Scrum-based project task management system has been developed, taking into account extensions that meet the realities of the modern IT industry. The influence of each role on the functioning of the management system is determined and it is found that it is the combination of basic Scrum roles with additional specialized roles that allows to form a stable multi-level system that can adapt to changes in the project environment.

Keywords: *role, role structure, task management system, Agile methodologies, access rights.*

Надійшла у першій редакції 17.02.2025, в останній - 21.03.2025.
The first version has been received on 17.02.2025, the final version - on 21.03.2025.