МАГІСТЕРСЬКА РОБОТА

МР.ІПм – 32.00.00.000 ПЗ

Група ІПм-22-6

Бандура Степан

2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Бандура Степан Романович

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

МАГІСТЕРСЬКА РОБОТА

«Моделі та засоби доменних специфікацій проектів, що керуються методологією SCRUM»

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Бандура С.Р.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Піх Володимир Ярославович, к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

В.о. завідувача кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Рецензент

проф. Шекета В.І.

(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
В.о. зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ 04 ” вересня 2023 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Бандура Степан Романович

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи **“**Моделі та засоби доменних специфікацій проектів, що керуються методологією SCRUM”**”**

керівник проекту (роботи) Піх Володимир Ярославович, к.т.н., доцент

затверджені наказом закладу вищої освіти від “ 18 ” грудня 2023 р. № 738/7

2. Строк подання студентом проекту (роботи) 12 січня 2024 р.

3. Вихідні дані до проекту (роботи) Теоретичні концепції та формальні моделі побудови та функціонування інформаційних та програмних технологій певного класу

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Теоретичні основи побудови Scrum проектів
2. Побудова мовної моделі
3. Оцінка отриманих результатів

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Метод Scrum (рис. 1.1)
2. Структура артефакту Scrutn (рис. 2.4)
3. Визначення зв'язку Sprint. (рис. 2.10)
4. структура проекту (рис. 3.1) .
5. Розподіл досвіду учасників опитування (рис. 3.9)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Нормоконтроль	доц., к.т.н. Вовк Р.Б.	
Перевірка на плагіат	доц., к.т.н. Вовк Р.Б.	

7. Дата видачі завдання 04 вересня 2023 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури	01.10.2023	виконано
2	Теоретичні основи побудови Scrum проектів	25.10.2023	виконано
4	Побудова мовної моделі для Scrum проектів	22.11.2023	виконано
6	Оцінка отриманих результатів	15.12.2023	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	12.01.2024	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 89 с., 25 рис., 15 табл., 45 джерел

Тема: Моделі та засоби доменних специфікацій проектів, що керуються методологією SCRUM

Об'єкт дослідження: проекти, що використовують методологію SCRUM для управління розробкою програмного забезпечення.

Мета роботи: вивчення та оптимізація використання моделей та засобів доменних специфікацій у проектах, які керуються методологією SCRUM.

Предмет дослідження: моделі та засоби доменних специфікацій, які застосовуються в проектах, управління якими здійснюється згідно з методологією SCRUM.

Результати дослідження: виконана розробка ефективних моделей доменних специфікацій на основі моделей, які оптимізують роботу команд у SCRUM-проектах, визначення ефективності. Виконана оцінка та визначення ефективності розроблених моделей в контексті конкретних проектів, підвищення ефективності управління проектами.

Висновок

Дослідження спрямоване на розкриття та вивчення засад побудови SCRUM-проектів, а також розробку та апробацію моделей та засобів доменних специфікацій у цьому контексті.

МЕТОДОЛОГІЯ SCRUM, СИНТАКСИС, СЕМАНТИКА, ЕКСПЕРЕМЕНТ, ДИЗАЙН ЕЛЕМЕНТІВ, ДОМЕН, ОПИТУВАННЯ, ДОМЕННО-СПЕЦИФІКОВАНА МОВА

ABSTRACT

Master's thesis: 89 pages, 25 figures, 15 tables, 45 sources

Topic: Models and means of domain specifications of projects guided by the SCRUM methodology

Research object: projects using the SCRUM methodology to manage software development.

The purpose of the work: study and optimization of the use of models and means of domain specifications in projects guided by the SCRUM methodology.

The subject of the study: models and means of domain specifications, which are used in projects, which are managed according to the SCRUM methodology.

Research results: the development of effective models of domain specifications based on models that optimize the work of teams in SCRUM projects, determination of effectiveness. Assessment and determination of effectiveness of developed models in the context of specific projects, improvement of project management efficiency.

Conclusion

The research is aimed at revealing and studying the basics of building SCRUM projects, as well as the development and testing of domain specification models and tools in this context.

SCRUM METHODOLOGY, SYNTAX, SEMANTICS, EXPERIMENT, DESIGN OF ELEMENTS, DOMAIN, SURVEY, DOMAIN-SPECIFIC LANGUAGE

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

DSL - Доменно-орієнтована мова

BPMN - Business Process Modeling Notation

UML - уніфікованої мови моделювання

ВСТУП

Актуальність роботи

Актуальність теми дослідження "Моделі та засоби доменних специфікацій проектів, що керуються методологією SCRUM" обумовлена кількома факторами:

- Зростання популярності SCRUM: SCRUM став однією з найбільш використовуваних методологій управління проектами в області розробки програмного забезпечення. З розвитком агільних підходів SCRUM здобуває все більше прихильників серед компаній і розробників.

- Потреба в оптимізації процесів розробки: З використанням SCRUM виникає потреба в пошуку ефективних моделей та засобів для розробки доменних специфікацій, які враховуватимуть особливості цієї методології.

- Інновації в області розробки мов та моделей: Розвиток методів розробки доменно-специфікованих мов і моделей відбувається швидко, і виникає необхідність в їхньому адаптуванні та впровадженні в SCRUM-проекти.

- Комплексність SCRUM-проектів: SCRUM використовується в різноманітних проектах, іноді складних та обширних. Застосування доменних специфікацій може полегшити та уточнити роботу команд над такими проектами.

Зростання конкуренції: У сфері розробки програмного забезпечення конкуренція зростає, і компанії шукають нові підходи та інструменти для підвищення ефективності та якості продуктів.

Таким чином, дослідження цієї теми є актуальним та важливим для подальшого розвитку методологій управління проектами та покращення процесів розробки програмного забезпечення.

Порівняння роботи з відомими розв’язаннями проблеми

Для порівняння результатів дослідження із відомими розв'язаннями проблеми можна взяти до уваги існуючі підходи та інструменти в області розробки програмного забезпечення, зокрема в контексті управління проектами за методологією SCRUM та використанням доменних специфікацій. Ось деякі аспекти для порівняння:

- Ефективність SCRUM-проектів: Порівняти ефективність SCRUM-проектів, що використовують розроблені моделі та засоби доменних специфікацій, із тими, які використовують стандартні підходи та інструменти.

- Сприйняття командою: Оцінити, як команди, що використовують нові моделі, сприймають та працюють з ними порівняно із тими, що використовують існуючі рішення.

- Результативність управління проектом: Порівняти результативність управління проектами за допомогою розроблених моделей та засобів з тими, які використовують звичайні методи та інструменти.

- Легкість впровадження: Визначити, наскільки легко впроваджувати та адаптувати розроблені рішення до реальних SCRUM-проектів порівняно з існуючими рішеннями.

- Практичність та використання: Оцінити практичність та придатність розроблених моделей та засобів у реальних умовах роботи, порівняно з вже існуючими інструментами та підходами.

Під час порівняння слід також врахувати відгуки та рекомендації користувачів, аналіз ринку інструментів у даній галузі, а також можливість впровадження розроблених моделей у реальні проекти.

Мета і задачі дослідження

Метою дослідження є вивчення та оптимізація використання моделей та засобів доменних специфікацій у проектах, які керуються методологією SCRUM.

Досягнення мети включало розв’язання таких задач:

- Розгляд теоретичних основ SCRUM
- Побудова Мовної Моделі
- Вивчення та розробка моделей доменних специфікацій,
- Оцінка та аналіз отриманих результатів.

Об’єкт дослідження: проекти, що використовують методологію SCRUM для управління розробкою програмного забезпечення.

Предмет дослідження: моделі та засоби доменних специфікацій, які застосовуються в проектах, управління якими здійснюється згідно з методологією SCRUM.

Методи дослідження

Для реалізації поставленої мети будуть використовуватися такі методи: літературний аналіз, вивчення наявної літератури з питань SCRUM-проектів, методів розробки доменно-специфікованих мов та моделей доменних специфікацій, експеримент, методи опитування, аналіз результатів.

Наукова новизна отриманих результатів

Отримані результати дослідження мають наукову новизну у засобах і моделях доменних специфікацій, розроблених спеціально для SCRUM-проектів. Це включає розробку ефективних методів та підходів до моделювання вимог та процесів в контексті даної методології. Також, результати дослідження можуть включати нові аспекти впровадження доменних специфікацій у SCRUM-проекти та їхній вплив на продуктивність та результативність.

Практичне значення одержаних результатів включає в себе особистий внесок студента, а саме розробку ефективних моделей доменних специфікацій, визначення їх ефективності для підвищення ефективності управління проектам

Структура та обсяг магістерської роботи

Магістерська робота викладена на 89 сторінках друкованого тексту, який складається із вступу, трьох розділів, висновків, списку використаних джерел (45 найменування) і містить 25 рисунків та 15 таблиць.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕОРЕТИЧНІ ОСНОВИ ПОБУДОВИ ПРОЕКТІВ НА ОСНОВІ МЕТОДОЛОГІЇ SCRUM

1.1. Огляд методології Scrum

Scrum — це фреймворк Agile процесів, який використовувався для управління розробкою складних продуктів з початку 1990-х. У 1993 році Джефф Сазерленд разом з Джоном Скумніоталсом і Джеффом МакКенною представили підхід Scrum у Easel Corporation і були першими, хто називав його одним словом Scrum. А потім Кен Швабер і Джефф Сазерленд спільно представили статтю, в якій розробляється концепція Scrum, на конференції з об’єктно-орієнтованого програмування, систем, мов і додатків '95 (OOPSLA' 95), що відбулася в 1995 році в Остіні, штат Техас. Відтоді кілька користувачів Scrum, експертів і авторів продовжували об’єднувати свій досвід і найкращі галузеві практики в методології Scrum.

Scrum використовує адаптивний, ітеративний, поетапний і простий підхід для продуктивної реалізації проектів найвищої можливої цінності. Структура Scrum складається з команд Scrum та їхніх пов’язаних ролей, подій, артефактів і правил. Кожна складова в структурі відіграє певну роль, щоб гарантувати успіх Scrum. Конструкція Scrum працює в ітераціях, які називаються спринтами, які містять планування спринту, щоденні Scrum, роботу з розробки, огляд спринту та ретроспективу спринту. Figurei ілюструє огляд потоку проекту Scrum (рис. 1.1).

Цикл розробки Scrum-проекту починається із зустрічі зацікавлених сторін, після чого створюється заява про бачення проекту. Потім власник продукту створює резервний журнал продукту, який містить пріоритетний список вимог до продукту, написаний у формі історій користувачів. Кожен спринт має нараду з планування спринту, під час якої власник продукту та команда планують, що робити для наступного спринту; високопріоритетні історії користувачів розглядаються для включення в беклог спринту.

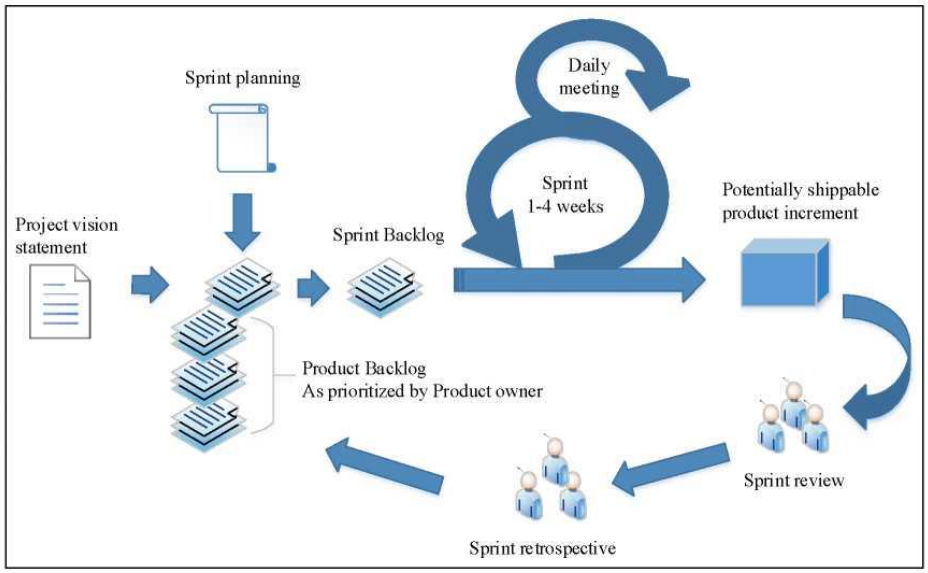

Рис. 1.1 Графічне представлення методології Scrum

Спринт зазвичай триває 1-4 тижні. Під час виконання кожного спринту команда Scrum щодня проводить коротку, цілеспрямовану зустріч для відстеження та обговорення прогресу роботи; зустріч називається «Денна зустріч Scrum». Наприкінці спринту команда розробників проводить нараду з огляду спринту, під час якої вони демонструють результати для власника продукту та відповідних зацікавлених сторін. Результатом наради для перегляду є переглянутий беклог продукту, який містить адаптовані елементи для наступного спринту. Цикл спринту завершується ретроспективною зустріччю спринту, де Команда розробників перевіряє їх ефективність і обговорює шляхи підвищення ефективності. Потім команда Scrum переходить до наступного спринту, поки не завершить весь проект.

1.2 Особливості команди в Scrum

У цьому розділі обговорюється організація команди Scrum, щоб зрозуміти ролі та обов’язки членів команди в контексті проекту Scrum. Важливо розуміти визначені ролі та обов’язки для точної реалізації проектів Scrum. Ролі Scrum можна умовно розділити на дві категорії: основні ролі та неосновні ролі.

У Scrum є три основні ролі, які виконують проект у процесі Scrum. Три основні ролі: власник продукту, Scrum Master і команда розробки. Чітке розуміння ролі

основної команди Scrum має важливе значення для забезпечення успішної реалізації проектів Scrum.

Власник продукту - Власник продукту відповідає за максимізацію цінності продукту та роботи Команди Розробників. Власником продукту є одна особа, а не комітет. Власник продукту представляє думку клієнтів і керує невиконаними продуктами. Власник продукту має замовити елементи в журналі продукту та переконатися, що резерв продукту є видимим і зрозумілим для всіх. Нікому не дозволяється брати роботу з іншого джерела чи вимог, і команда розробників має виконувати рішення власника продукту в резерві продукту.

Scrum Master - Scrum Master є фасилітатором, який відповідає за те, щоб Команда Розробників зрозуміла теорію, практику та правила Scrum. Scrum Master скеровує, полегшує та допомагає членам команди, які беруть участь у проекті, визначити, які з їхніх взаємодій із Scrum-командою можуть призвести до максимізації цінності та покращення їх. Scrum Master є слугою-лідером команди Scrum. Scrum-майстер обслуговує Власника продукту в кількох аспектах, зокрема: керування зустрічами Scrum із командою розробників, розуміння того, як організувати резервну роботу продукту, щоб максимізувати цінність, і допомагає команді розробників зрозуміти рішення власника продукту щодо резервної роботи продукту. Scrum-майстер обслуговує команду розробників, навчаючи теорії Scrum і усуваючи перешкоди в прогресі проекту.

Команда розробки. Команда розробки складається з групи професіоналів, які відповідають за створення результатів проекту відповідно до вимог, визначених Власником продукту. Команда розробників є самоорганізованою та багатофункціональною, яка володіє всіма необхідними навичками для створення продукту.

Неосновні ролі – це ті ролі, які не є обов’язковими для проекту Scrum. Однак непрофільні ролі також можуть відігравати важливу роль у деяких конкретних проектах Scrum.

Замовник - замовник - це особа або організація, яка забезпечує вимоги та набуває продукт або послугу. У деяких проектах Scrum замовник може

безпосередньо взаємодіяти з власником продукту, щоб визначити історії користувача. Вони також беруть участь у зустрічі Sprint Review для перевірки результатів, продемонстрованих командою розробки. Користувач – це люди, які безпосередньо використовують продукт або послугу проекту. Вони можуть бути присутніми наприкінці процесу проекту та висловити свою думку щодо продукту, щоб допомогти команді Scrum покращити його.

1.3. Дослідження процесів в Scrum

Процес Scrum стосується потоку та пов'язаних з ним дій проекту Scrum. Відповідно до посібника SBOK™, проект Scrum складається з 5 фаз і 19 процесів. Ці етапи та дії наведені в таблиці 1.1.

Таблиця 1.1.

Етапи процесів в SCRUM

Фази	**процеси**
Ініціювати	Створення бачення проекту
	Визначте Scrum Master
	Команда розробки форм
	Розробка Epic(s)
	Створення резерву пріоритетних продуктів
	Проведіть планування спринту
План і кошторис	Створюйте історії користувачів
	Схвалення, оцінка та фіксація історій користувачів
	Створення завдань
	Кошторисні завдання
	Створити беклог спринту
Реалізувати	Створення кінцевих результатів
	Проводьте щоденну зустріч Scrum
	Наречений Пріоритетний продукт Беклог
Огляд і ретроспектива	Скликати Scrum Scrums
	Продемонструйте та перевірте спринт
	Ретроспективний спринт
Завершення	Доставка результатів
	Проект «Ретроспектива».

Ініціювати – як ініціація проекту, ця фаза включає процеси створення команди, визначення продукту та створення Backlog Product.

• Створення бачення проекту: у цьому процесі буде визначено власника продукту, який спілкуватиметься з клієнтами. Бізнес-кейс обговорюється для створення заяви про бачення проекту для команди Scrum для розуміння проекту.

• Визначте Scrum Master: Scrum Master визначається за допомогою конкретних критеріїв відбору компанії.

• Команда розробки форми: члени групи розробки вибираються для створення продукту в цьому процесі. Зазвичай Власник продукту повинен співпрацювати з Scrum Master, щоб вибрати відповідних розробників для конкретного проекту.

• Розробка Epic(s): Epics розробляються на цій стадії, коли більшість історій користувачів написані в описі функціональних можливостей високого рівня та вимоги приблизно визначені.

• Створити беклог пріоритетного продукту: у цьому процесі визначені Epics розбиваються на менші та встановлюються пріоритетами власника продукту для створення пріоритетного беклогу продукту.

• Проведення планування спринту: основна команда Scrum обговорює вимоги в Беклозі продукту для створення розкладу планування спринту, який є поетапним розкладом розгортання для керівництва командою щодо створення артефактів. У цьому процесі також буде визначено довжину спринту.

План і оцінка. На цьому етапі власник продукту детально описує історії користувачів, перелічені в Backlog Product. Історії користувачів розбиваються на конкретні завдання, які має виконати команда розробників. Команда Scrum оцінює зусилля, необхідні для розробки функціональності, описаної в історіях користувачів.

• Створення історій користувачів: Історії користувачів зазвичай докладно описуються власником продукту, щоб переконатися, що вимоги клієнта чітко зображені та можуть бути зрозумілі команді розробників.

• Схвалення, оцінка та фіксація історій користувачів: у цьому процесі Власник продукту вибирає та затверджує історії користувачів для спринту. Потім команда Scrum обговорює та оцінює зусилля, необхідні для впровадження функціональних можливостей, описаних у кожній історії користувача. Нарешті, команда розробників

бере на себе зобов’язання створити та надати продукти, зображені в історіях користувачів, у майбутньому Спринті.

• Створення завдань: схвалені, оцінені та зобов’язані історії користувачів розбиваються на конкретні завдання та створюється список завдань.

• Оцінка завдань: основна команда Scrum обговорює та оцінює зусилля, необхідні для виконання кожного завдання зі списку завдань.

• Створити беклог спринту: у цьому процесі команда Scrum проводить нараду з планування спринту, під час якої команда розробників співпрацює з власником продукту, щоб створити беклог спринту, що містить вибрані завдання, які потрібно виконати в майбутньому спринті.

Впровадити – команда розробників виконує завдання та створює продукт проекту на цьому етапі. Щоб сприяти виконанню завдань, Scrum-майстер повинен забезпечити проведення щоденних Scrum-нарад з командою розробників для перевірки проблем у роботі та відстеження прогресу на шляху до мети спринту. А журнал продукту підтримується та постійно адаптується для досягнення цілі продукту разом із просуванням проекту.

• Створення результатів: Команда розробників виконує завдання, перелічені в Backlog Sprint для створення продукту. Scrum-дошка часто використовується, щоб допомогти команді відстежувати хід роботи. Завдання згруповані в три розділи — NotCheck out», —Checkout» і —Dne», на дошці Scrum, керуючи командою розробників щодо виконання завдань.

• Проведення щоденної зустрічі Scrum: Команда розробників використовує щоденну зустріч Scrum для перевірки перешкод, відстеження прогресу та організації роботи. Ця зустріч сприяє спілкуванню та співпраці між членами команди.

• Беклог пріоритетних продуктів Groom: Беклог пріоритетних продуктів постійно оновлюється та коригується. Може відбутися нарада з перегляду спринту, під час якої основна команда Scrum обговорюватиме зміни для оновлення Backlog Product.

Огляд і ретроспектива. Ця фаза зосереджена на перегляді результатів, які були зроблені в поточному спринті, та обговоренні шляхів покращення ефективності та

продуктивності команди розробників. У великих організаціях цей етап також включає проведення зустрічей Scrum of Scrums.

• Скликати Scrum of Scrums: у цьому процесі Scrum-команди проводять зустріч Scrum of Scrums для співпраці та перевірки залежностей, відстеження відповідного прогресу та перевірки перешкод між командами. Цей випадок стосується лише великих проектів, у яких задіяно кілька команд Scrum.

• Продемонструвати та підтвердити спринт: під час наради з огляду спринту команда розробників демонструє результати спринту власнику продукту та клієнтам, які вирішуватимуть, приймати чи ні результати.

• Ретроспективний спринт: у цьому процесі команда розробників має можливість перевірити себе та створити план покращення, який буде виконано під час майбутнього спринту. Scrum Master заохочує команду розробників вдосконалювати, використовуючи структуру Scrum, її процес і методи, щоб зробити її більш ефективною та продуктивною для наступного спринту.

Випуск - фаза випуску полягає в тому, щоб доставити прийняті результати замовникам і визначити відповідні документи проекту.

• Відвантажити результати: у цьому процесі прийняті результати передаються клієнтам. І тоді команда Scrum завершує цей спринт.

• Ретроспективний проект: Команда Scrum співпрацює з клієнтами, щоб ретроспективно переглянути цей спринт, визначити відповідні документи та засвоїти отримані уроки.

Існує кілька мов метамоделювання, запропонованих для вдосконалення процесу розробки, таких як Unified Modeling Language (UML, OMG, 2007), Business Process Modeling Notation (BPMN, OMG, 2005), ECore (Budinsky, 2003) [17] і Graph-Object-Property-Role-Relationship (GOPRR, Kelly 1997) [18]. Alegria [19] та ін. пропонує використовувати мета-модель Systems Process Engineering (SPEM) для опису дорожньої карти процесу розробки Agile. SPEM був запропонований як стандарт Object Management Group (OMG, 2008) для моделювання процесу проектування. Більшість цих мов є загальними. Незважаючи на те, що ці мови дуже популярні, проблема, що стоїть за ними, полягає в тому, що загальні мови не

орієнтовані на певну область, наприклад, на конкретний метод розробки програмного забезпечення. Різні методи розробки програмного забезпечення мають різну філософію проектування та власні особливості, які не завжди можна точно виразити загальною мовою. Багато інструментів моделювання не засновані на стандартних мовах, оскільки вони не можуть моделювати користувацькі елементи залежно від конкретної області [14]. Загальна мова, така як UML, не забезпечує механізму для визначення елементів за допомогою точок зору користувача, які можуть допомогти розробникам зрозуміти та змоделювати складний і специфічний системний процес.

Однак останнім часом доменно-специфічна мова (DSL) стає все більш популярною для захоплення потужних абстракцій добре вивчених областей застосування [20]. DSL дозволяє специфікувати програмне забезпечення з індивідуальної точки зору. Це підвищує рівень абстракції та наближає реалізацію до словника, зрозумілого розробникам і кінцевим користувачам [21]. Це також може зробити моделювання процесів більш простим для розуміння клієнтами та допомогти їм відстежувати прогрес.

Позначення DSL можна класифікувати на дві групи: графічні позначення та текстові позначення. Порівняно з текстовими нотаціями, графічні нотації легше зрозуміти та використовувати людям, які не мають технічних знань, наприклад новачкам або учням молодших класів. Крім того, згідно з дослідженням [22], графічна мова дозволяє описувати більшість бізнес-процесів із простішою семантикою та більш абстрактним синтаксисом. Тому в цій дипломній роботі ми запропонували визначити графічний DSL для моделювання процесу Scrum.

Насправді DSL завжди використовується для моделювання конкретної програми, але для моделювання процесу розробки програмного забезпечення рідко зустрічається в літературі. Однак визначення процесу розробки програмного забезпечення є дуже важливим для розробників, щоб застосувати відповідні методи впровадження програмного забезпечення. Scrum як найпопулярніший метод широко застосовувався для розробки продуктів. Але у звіті CHAOS за 2015 рік, опублікованому Standish Group, лише близько 39% Agile-проектів можна назвати

успішними (вирішення всіх Agile-проектів з 2011 по 2015 фінансовий рік у новій базі даних CHAOS). Успіх проекту означає, що Команда Розробників досягає своїх цілей із задовільним результатом, у межах запланованого бюджету та вчасно. Незважаючи на те, що гнучкі підходи призвели до більш успішних проектів і менш явних провалів, ніж традиційні методи розробки, компанії все ще стикаються з серйозними ризиками та проблемами, застосовуючи гнучкі методи. Під час наших перших інтерв'ю респондент також зазначив, що члени команди не мають спільного розуміння того, як застосовувати метод Scrum на початку, а надзвичайні ситуації завжди вирішувалися неналежним чином. Scrum також має невід'ємну проблему — відсутність опису процесу реалізації проекту. Команди розробки використовують Burn-Chart для керування завданнями та відстеження прогресу проектів. Тим не менш, продукт і реалізація завжди неправильно розуміються розробниками. Scrum не надає користувачам інструмент для перевірки деталей прогресу розробки, того, що було зроблено, деталей результатів. Ці проблеми вирішуються шляхом посилення комунікації. Розробники використовують щоденні зустрічі, щоб пояснити свої завершені та поточні завдання, а також їм потрібно знати, що зробили інші. Але п'ятнадцятихвилинної зустрічі недостатньо, щоб допомогти їм охопити всі деталі прогресу, і конфлікти кодів трапляються через відсутність опису проекту . Тому в цьому проекті ми пропонуємо використовувати графічний DSL для моделювання процесу розробки програмного забезпечення та допомогти розробникам зрозуміти метод Scrum та пов'язані з ним проекти.

Ми віримо, що запровадження чітко визначеного DSL для методу Scrum може не лише допомогти недосвідченим користувачам вивчити та застосувати метод Scrum, але й полегшить Групі розробників вирішення проблем, які існували під час впровадження, та зменшить ризики невдачі.

Щоб відповісти на запитання дослідження, ми провели вивчення літератури, контрольований експеримент та опитування.

Вивчення літератури — це підхід, який використовується для виявлення, оцінки та інтерпретації доступних досліджень, що мають відношення до конкретного дослідницького питання чи тематичної області. У цій дипломній роботі

вибрано дослідження літератури, щоб зібрати знання про метод Scrum (RQ1) і DSL (RQ2) . Експеримент — це систематичний і науковий підхід, який передбачає маніпулювання однією або декількома змінними та вимірювання впливу цих маніпуляцій на змінні. Проводиться експеримент, щоб ми могли оцінити запропоновану мову в академічному середовищі та відповісти на RQ3 . Метод опитування може забезпечити кількісний опис інформації для порівняння та пояснення знань [23]. Його можна використовувати, розробивши анкету та надавши її учасникам онлайн. Загалом опитування вимагає менше часу для авторів на створення та для учасників на відповіді. Опитування використовується для збору думок користувачів Scrum щодо запропонованої мови для моделювання процесу Scrum. Результат опитування дає нам змогу дослідити потенційне використання запропонованої мови в промисловому середовищі (RQ4) .

1.4. Метод розробки доменно-специфікованої мови

Щоб з’ясувати елементи, їхні властивості та правила, які використовуються для представлення процесу Scrum, було проведено дослідження літератури. У результаті проведення дослідження літератури було отримано набір елементів Scrum та їхні зв’язки, визначені в літературі, щоб дати відповідь на RQ1.

Перш ніж визначити передбачувану мову опису Scrum, ми дослідили методи, які зазвичай використовуються для визначення графічних мов, щоб вибрати метод, який підходить для визначення мови. Пізніше, на основі результату RQ , ми витягли основні будівельні блоки та правила запропонованої мови для опису процесу Scrum. Потім ми визначили синтаксис і семантику запропонованої мови відповідно до описаного методу розробки DSL і відповіли на RQ2 .

Далі ми провели як кількісні, так і якісні методи для оцінки корисності запропонованої мови. Ми провели експеримент в академічному середовищі та опитування в промисловості. Для кількісного дослідження контрольований експеримент може надати прямий результат того, чи вважається запропонована мова корисною для вивчення методу Scrum і розуміння конкретних проектів Scrum.

Типом цього плану експерименту є план порівняння. Піддослідними були студенти коледжу. Обробками були методи опису, запропонована мова та непропонована мова. Інструментами цього експерименту були навчальні матеріали Scrum, які складаються з двох частин: вступ до методу Scrum та опис конкретного проекту Scrum. Ми створили два типи описів цих навчальних матеріалів. Один описаний традиційним способом використовував текстові документи. Інший описаний за допомогою розробленої мови моделювання Scrum. Перед проведенням експерименту ми провели пілотний експеримент, щоб перевірити та вдосконалити дизайн експерименту, щоб переконатися в правильності результату експерименту. Усі суб'єкти без досвіду роботи зі Scrum були випадковим чином розділені на дві групи. Одна група вивчала структуру Scrum і приклади проектів, використовуючи навчальні матеріали, описані запропонованою мовою, а інша використовувала текстові навчальні матеріали. Після ознайомлення з навчальними матеріалами Scrum усі суб'єкти повинні були відповісти на кілька запитань, щоб визначити, наскільки суб'єкти вивчають метод Scrum (RQ3.1) і розуміють конкретні проекти Scrum (RQ3.2) . Аналізуючи результати експерименту, ми перевірили, чи корисна запропонована мова під час навчання Scrum, і відповіли на RQ3 .

Тим часом опитування було також проведено в галузі для оцінки потенційного використання запропонованої мови. Учасників попросили порівняти ці два типи описів і оцінити, чи допомагає запропонована мова їм зрозуміти процес Scrum.

Учасниками опитування є користувачі Scnun з промислових команд Semin. Зібрані дані були оброблені в якісному маімері, щоб визначити переваги та обмеження запропонованої мови з точки зору користувача Scrum (RQ4).

Загальні методи дослідження та кроки представлені на малюнку 2. Рис. слід читати зліва направо та зверху вниз.

Через обмеження часу та ресурсів використання цих двох методів для відповіді на запитання RQ1 є недоцільним. В додаток. Scrum — це ітеративний метод із невеликою кількістю простих елементів і елементів. Ми можемо в достатній мірі зібрати будівельні блоки та ніли, вивчаючи літературу та без неї необхідність

перегляду всіх відповідних документів. Крім того, ми також розглядали можливість використання інтерв'ю з промисловими експертами для збору відповідних знань.

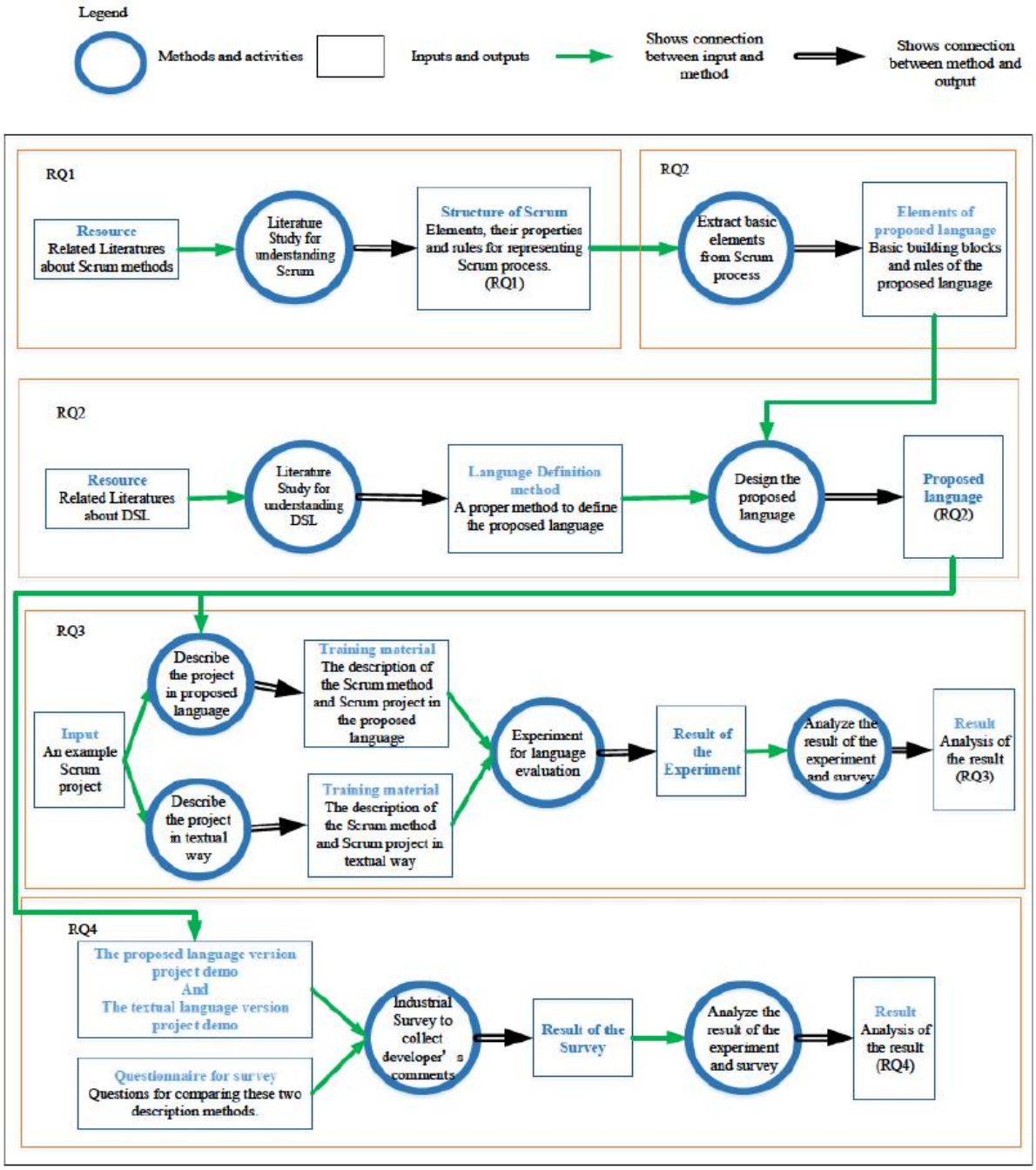

Рис. 1.2. Методи дослідження та етапи

Однак Scrum — це інкрементна та ітераційна структура розробки програмного забезпечення з невеликою кількістю простих правил, проведення вивчення літератури достатньо для вилучення елемента Scrum, тоді як використання інтерв'ю промислового експерта вимагає більше додаткових ресурсів. Крім того, проведення

промислових інтерв'ю потребує від промислових експертів більших часових зусиль, які нам недоступні.

Для фази аналізу [37] представив два поширені способи проведення аналізу домену: FODA (Feature-Oriented Domain Analysis) і FAST (Family-Oriented Abstractions, Specification, and Translation). Крім того, Sohs-Martmez [14] повідомив про своєрідний підхід, заснований на експертній думці, до проведення фази аналізу. Метод FAST вимагає збору пов’язаних DSL або програм для аналізу спільності. Однак, згідно з нашим початковим дослідженням, існує небагато пов’язаних DSL або програм, які можуть бути направлені. Тому FAST є неправильним підходом у цьому проекті. Підхід, заснований на експертній думці, потребує багато експертних ресурсів Scrum і витрат часу. Але ми не можемо отримати доступ до достатньої кількості експертів Scrum і DSL. Отже, цей підхід також виключається. У цьому дослідженні ми вибрали FODA для проведення аналізу DSL; FODA вимагає набору структурованих функцій, які можна охопити в літературі Scrum. Для нас це здійсненний і розумний метод виконання частини аналізу DSL.

Для фази проектування [25, 38] запропонували два підходи до розробки DSL: з нуля або налаштувати існуючу мову загального призначення. Очевидно, що розробка DSL з нуля вимагає більше зусиль і досвіду в порівнянні з повторним використанням існуючої мови. Насправді існує подібна мова загального призначення, яку можна використовувати для моделювання, BPMN. Повторне використання BPMN може зменшити наші зусилля щодо розробки, а наявна мова також може надати багато цінних визначень або концепцій, до яких ми можемо звернутися. Тому ми вирішили повторно використати деякі концепції з BPMN у нашому дослідженні для розробки запропонованої мови.

Щоб оцінити корисність запропонованої мови в навчанні Scrum (RQ3), можна провести кейс-стаді. Результат дослідження прикладу може продемонструвати, як запропонована мова сприяє навчанню Scrum. Однак, з метою оцінки корисності запропонованої мови, тематичне дослідження не є належним способом у цьому дослідженні; оскільки кейс-стаді — це метод, який складається зі спостереження реальної словесної ситуації в природному середовищі. Він може проілюструвати, як

працює запропонована мова, але не може надати результат того, чи є запропонована мова корисною чи ні порівняно з існуючими мовами моделювання.

Щоб дослідити потенційне використання запропонованої мови в галузі (RQ4), можна провести експеримент. Експеримент може дати результат того, чи доцільно застосовувати запропоновану мову в промисловості. Однак, щоб отримати корисні дані для подальшого аналізу, експеримент потрібно провести в проекті Scrum у промисловому середовищі, але він був недоступний для нас. Крім того, для команди Scrum використання нової техніки моделювання вимагає додаткових ресурсів. Після розслідування ми визнали цей варіант неможливим. Проведення такого експерименту можна розглядати як контрольне дослідження.

Відповідно до Memik [37], повний процес розробки DSL складається з п’яти етапів: рішення, аналіз, проектування, впровадження та розгортання. Розробка DSL не є послідовним процесом, це означає, що ці п’ять фаз завжди впливають одна на одну і залежать одна від одної. Наприклад, на дизайн DSL часто впливають міркування реалізації [37]. У цьому розділі різні фази представлені окремо.

• Рішення.

На етапі прийняття рішення розробники DSL повинні прийняти кілька важливих рішень, наприклад, який тип DSL (графічний або текстовий) повинен бути, яка область проблем DSL, який існуючий DSL слід вибрати для прийняття для нового DSL [37] тощо. В [38] запропонував розробникам DSL «уважно вирішити, використовувати графічну чи текстову реалізацію». Розробники повинні зважувати переваги та недоліки кожного рішення, тому що неправильне рішення може втратити багато інвестиційних зусиль і призвести до переробки на інших етапах [38]. Проте є ще багато рішень, які не можна прийняти на початку, тому що рішення мають залежати від інших умов на інших етапах.

На етапі аналізу необхідно визначити предметну область і зібрати знання про предметну область [37]. Вхідними даними цієї фази є різні джерела знань, такі як база даних літератури, і рішення з фаз прийняття рішень, такі як проблемна область DSL. Результатами на етапі аналізу є зібрані знання, включаючи літературу про

існуючі пов’язані DSL, основну предметно-специфічну термінологію та виконання семантики [37, 38].

• Дизайн

Як називається цей етап, DSL має бути повністю визначений і пояснений на етапі проектування [38]. Вхідними даними цього етапу є вихідні дані фази аналізу, а виходами є повна модель домену, включаючи: визначення домену, повну термінологію домену, детальний опис концепцій домену та моделі функцій, що представляють «спільність і варіативність концепцій домену [37] ».

Існує дві шкали класифікації методів проектування DSL [37]:

1) формальний характер опису дизайну,

2) ступінь успадкування від існуючих DSL.

Щоб бути більш конкретним, у неофіційному описі дизайну DSL специфікація зазвичай пояснюється природною мовою та серією діаграм. Навпаки, формальний дизайн DSL записується в ефективних методах семантичного визначення [37]. Для другого виміру є два способи розробки нового DSL: складання та повторне використання існуючих DSL або винахід абсолютно нового DSL [37, 38]. Очевидно, що розробка абсолютно нової мови вимагає багато зусиль. Однак повторно використовувати існуючі DSL для розробки нового DSL набагато легше. Однак вибрані мови повинні відповідати принципу безперебійності, щоб бути складеними.

• Реалізація

Для виконуваного DSL після фази проектування слід визначити техніку реалізації (повернутися до фази прийняття рішення). На етапі реалізації слід створити виконуване робоче середовище DSL або генератор додатків. Вхідними даними цього етапу є повне визначення DSL на етапі проектування та підхід до реалізації, визначений на етапі прийняття рішення. Результатом є виконуваний компілятор DSL, генератор додатків або інтерпретатор мови [37, 39].

• Розгортання

Фаза розгортання, яка також називається фазою використання [39], на фазі розгортання реалізований компілятор або генератор додатків повинен отримати екземпляр DSL як вхідні дані та виконати або створити відповідну програму [37].

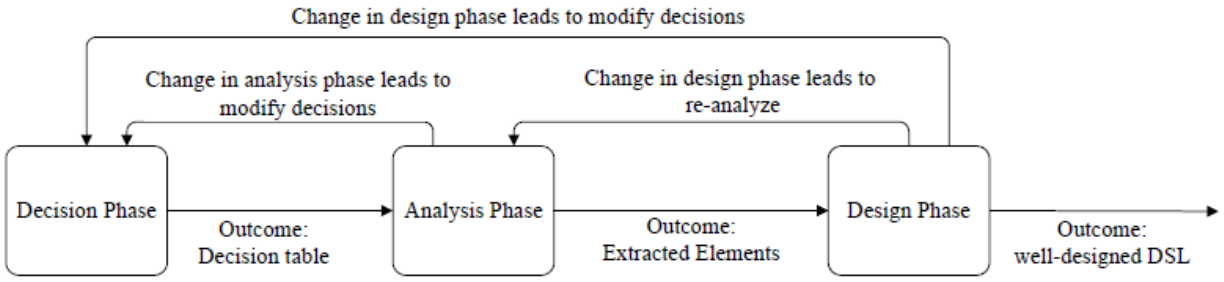

Рис. 1.3. Підхід до розробки DSL

У нашій роботі з розробки DSL, оскільки Scrum є типом методу розробки програмного забезпечення замість виконуваної програми, ми вирішили розробити невиконуваний DSL. Таким чином, етапи впровадження та розгортання не включені в запропонований процес розробки DSL. Рис. 1.3 ілюструє процес і підходи, які ми використовували для розробки запропонованого DSL.

1.5. Опис сутності та етапів експерименту дослідження

У цьому розділі описано етапи процесу експерименту, який складається з визначення обсягу та планування. Перш за все, ми розбираємо експеримент відповідно до проблеми, мети та цілей. Наступним кроком є планування, де ми визначаємо дизайн експерименту та розглядаємо прилади.

• Визначення обсягу

Визначення обсягу є першою діяльністю процесу експерименту. Це пояснює, чому проводиться експеримент. На етапі визначення обсягу визначається основа експерименту. Належний обсяг експерименту гарантує, що намір експерименту може бути виконано під час процесу проведення експерименту.

Метою етапу визначення обсягу є визначення мети та цілей експерименту. Обсяг експерименту визначається визначенням його цілей. І мета формулюється виходячи з проблеми, яку ми хочемо вирішити. У цьому дипломному проекті основна проблема полягає в тому, щоб дослідити, чи та як запропонована мова для опису проекту Scrum сприяє вивченню та розумінню процесу Scrum. Тому метою

цього експерименту є оцінка корисності запропонованої мови моделювання Scrum порівняно з іншими мовами опису.

Текстова мова як загальний метод завжди використовується для опису та представлення процесу Scrum. У цьому експерименті ми планували порівняти ефективність навчання і ефективність між цими двома різними мовами, запропонованою мовою моделювання Scrum і текстовою мовою, у представленні процесу Scrum розробникам програмного забезпечення.

Для чіткого визначення мети експерименту ми використали шаблон мети, запропонований у [40]. Використовуючи шаблон цілі, ми можемо гарантувати, що важливі аспекти експерименту точно визначені перед плануванням і проведенням. Мета цього експерименту підсумовується так: проаналізуйте мову тексту та запропоновану мову моделювання Scrum з метою оцінки щодо ефективності та результативності з точки зору студентів у контексті студентів бакалаврату та магістратури програмної інженерії, які вивчають процес Scrum.

. Об’єктом дослідження є методи, які використовуються для опису процесу Scrum, та їх ефективність для того, щоб допомогти студентам зрозуміти та вивчити процес Scrum. Методи, оцінені в цьому експерименті, це текстовий спосіб і запропонована мова моделювання Scrum. Цільовий. Метою цього експерименту є оцінка впливу цих двох різних методів на розуміння та вивчення процесу Scrum, зокрема щодо корисності запропонованої мови. Порівнюючи результати оцінювання, ми можемо зробити висновки про те, як і чи сприяє запропонована мова навчанню процесу Scrum.

Ефективність і дієвість є основними якостями методів опису, які вивчаються в цьому експерименті. Вони є основним ефектом цих двох мов опису, який допомагає розробникам вивчити та зрозуміти процес Scrum. Ми вимірюємо та кількісно оцінюємо ефективність та результативність навчання, коли суб’єкти вивчають процес Scrum різними методами опису.

Цей експеримент вивчається з точки зору студентів. Він зосереджений на оцінці здатності цих двох мов допомогти студентам зрозуміти та вивчити процес Scrum. Погляди студентів відрізняються від поглядів дослідників. Дослідників

завжди хвилює принцип або філософія Scrum, а не його застосування. Студенти більше зосереджуються на застосуванні Scrum і тому, як його розуміти та застосовувати. Тому результати оцінювання мови опису інтерпретуються з точки зору студентів.

Контекст цього експерименту полягає в тому, що суб’єкти вивчають Scrum різними мовами. Суб’єкти – студенти, які спеціалізуються на інженерії програмного забезпечення.

• Планування

Після визначення обсягу експерименту нам потрібно скласти план, який буде керувати виконанням експерименту, щоб переконатися, що експеримент можна провести належним чином. Планування готує те, як слід проводити експеримент. Фаза планування складається з 6 заходів, і вхідними даними для цієї фази є визначення мети експерименту. Виходячи з визначення мети, ми повинні спочатку визначити контекст експериментів. Контекст – це середовище, в якому проводився експеримент. Після цього слід сформулювати гіпотезу та вибрати змінні незалежні та залежні. Потім проводиться відбір предмета. Тип плану експерименту визначається на основі гіпотези та обраних змінних. Нарешті, приладова техніка забезпечує практичну реалізацію експерименту.

На етапі планування готується детальний і повний проект для проведення експерименту.

Контекст: Цей експеримент планується провести в університеті. Це автономний експеримент (не в промисловому середовищі). Він зосереджений на навчанні Scrum в освітньому середовищі. Ідея експерименту полягає в тому, що суб’єкти спочатку вивчають Scrum, а потім відповідають на кілька запитань на основі навчальних матеріалів. Ми вимірюватимемо час, який вони витратили на вивчення Scrum, і кількість запитань, на які вони правильно відповіли, щоб оцінити ефективність та ефективність різних методів лікування.

Використовувати студентів як суб'єктів дешевше і легше контролювати, ніж професіоналів. Крім того, представлення Scrum для студентів означає, що вони можуть навчитися новому методу розвитку з цього експерименту. Це може бути

цікавішим для піддослідних і легше надихнути їх взяти участь у цьому експерименті. Ще одна причина для проведення експерименту в освітньому середовищі полягає в тому, що попередній досвід розробки професіоналів з галузі може вплинути на них, коли вони вивчать і розумітимуть процес Scrum, описаний різними мовами. Менший досвід розробки може змусити студентів більше зосередитися на навчальному матеріалі (створеному різними мовами опису), щоб вивчити Scrum. Таким чином, порівняно з галузевим випадком, освітнє середовище легше надати безпосередній результат оцінки описуваних мов і з меншою кількістю факторів, що вводять в оману.

Перевірка гіпотез є важливим аспектом експерименту, щоб знати, що буде оцінюватися в експерименті. Гіпотеза повинна бути викладена формально і чітко. Якщо гіпотезу можна спростувати за даними, зібраними під час експерименту, то висновки можна зробити на основі перевірки гіпотези.

Як зазначено у визначенні мети, ми збираємося порівняти як ефективність, так і результативність, коли йдеться про вивчення процесу Scrum із використанням двох різних мов опису, щоб оцінити корисність запропонованої мови. Перший метод — текстова мова, а другий — запропонована мова моделювання Scrum. Мова тексту є природною мовою. Пропонована мова є графічним DSL, створеним спеціально для моделювання процесу Scrum.

Гіпотезу можна виразити такими ситуаціями:

1. Суб’єкти використовують підготовлені навчальні матеріали для вивчення процесу Scrum і відповідають на запитання на основі матеріалів, щоб перевірити, наскільки вони дізналися про Scrum. Якщо суб’єкти з групи, яка використовує текстовий навчальний матеріал, і групи, яка використовує графічний навчальний матеріал, мають однакову ефективність навчання, це означає, що який би тип навчальних матеріалів вони не використовували, не впливає на вивчення та розуміння процесу Scrum.

2. В іншому можливому випадку суб’єкти, які використовують графічні навчальні матеріали, краще або гірше розуміють процес Scrum, ніж суб’єкти, які використовують текстові матеріали. Очікується, що вони мають вищу або нижчу

ефективність і результативність, ніж студенти, які використовують текстові матеріали. Іншими словами, учасники, які використовують графічні матеріали, витрачають більше або менше часу на вивчення процесу Scrum (ефективність), а також можуть правильно відповісти на більше або менше питань (ефективність), ніж інша група. Якщо запропонована мова має позитивний результат, це означає, що запропонована мова може покращити розуміння Scrum і допомогти користувачам вивчити процес Scrum, і її можна вважати корисною для представлення розробникам. Якщо запропонована формулювання має негативний результат, ми повинні зробити висновок, що запропонована формулювання марна.

Виходячи з цих можливих ситуацій гіпотез, формальні гіпотези можуть бути сформульовані як: Ефективність: Гіпотези означають, що ми повинні протестувати та перевірити статистичну значущість того, що дві описувані мови мають різну ефективність і ефективність у процесі навчання Scrum. Якщо ми хочемо зробити висновок, що запропонована мова корисна для вивчення процесу Scrum, нам потрібно відкинути нульову гіпотезу, а потім підтвердити, що запропонована мова має позитивний результат щодо допомоги користувачам у вивченні процесу Scrum.

Щоб підтвердити статистичну значущість різниці мов, необхідно зібрати наступні дані для оцінки результатів експерименту.

• TMEC: Час, виміряний у хвилинах, який був потрібний для ознайомлення з матеріалами та відповідей на запитання.

• NRESP: кількість правильних відповідей для кожного предмета на питання про Scrum.

• TMEC/NRESP: час, виміряний у хвилинах на відповідь, який необхідний для отримання однієї правильної відповіді.

Ефективність завжди вимірюється часом виконання завдань, який суб’єкти виконують для читання та вивчення навчальних матеріалів. Ефективність можна розрахувати, підрахувавши кількість правильних відповідей суб’єктів, коли вони намагалися виконати завдання. Для іншого питання існує інший дизайн і мета оцінювання. Таким чином, підрахунок кількості правильних відповідей може допомогти нам з’ясувати, якою мірою суб’єкти засвоїли знання з матеріалів. Крім

того, ці дані також можна використовувати для розрахунку ефективності (TMEC/NRESP). Насправді показник точності також є показником, який можна використовувати для розрахунку ефективності. Однак у цьому експерименті цих деревних типів даних достатньо для перевірки різниці статистичної значущості, тому ми не використовували рівень точності задач.

Гіпотези та заходи визначають, який тип статистичного тесту слід використовувати. Для кожного показника (TMEC, NRESP і TMEC/NRESP) необхідно порівняти два способи обробки (мову тексту та мову графіки). По-перше, ми використовуємо критерій Шапіро-Вілка для обчислення кожного показника, щоб перевірити, чи підкоряється він нормальному розподілу. Якщо розподіл можна класифікувати як нормальний, використовується t-тест. Т-тест є одним із найбільш часто використовуваних параметричних тестів. Тест використовується для порівняння двох незалежних вибіркових середніх значень, і дизайн повинен складатися з одного фактора з двома методами лікування [40]. Його можна використовувати, щоб перевірити, чи є значна різниця між двома способами лікування. З іншого боку, якщо тест вказує на те, що показник розподіляється не так, як зазвичай, для аналізу даних можна використати критерій Манна-Уітні. Тест Манна-Уїтні є непараметричною альтернативою t-тесту. Він завжди використовується, якщо припущення, зроблені t-тестом, є невизначеними [40].

Якщо результати тесту вказують на те, що суб’єкти, які працюють з текстовими та графічними матеріалами, мають значну різницю у вивченні та розумінні Scrum, і, крім того, графічна мова може підвищити ефективність та ефективність навчання Scrum, ми можемо зробити висновок, що запропонована мова корисно познайомитися з процесом Scrum. Навпаки, якщо результати тесту показують, що між цими двома мовами опису Scrum немає різниці, а графічні матеріали мають негативний результат, ми розуміємо, що запропонована мова марна.

Незалежні та залежні змінні: незалежні змінні – це ті змінні, якими маніпулюють та змінюють їх під час експерименту. Це також методи лікування, які ми хочемо дослідити. Мета цього експерименту — дослідити різні описи Scrum методи, застосовані до двох об'єктів (навчальних матеріалів). Отже, у цьому

експерименті є одна незалежна змінна, мова опису Scrum, з двома значеннями: текстова мова та графічна мова (пропонована мова).

Різні методи лікування можуть дати різні результати. Ефект лікування вимірюється залежними змінними. Залежними змінними цього експерименту є ефективність та результативність навчання. Ефективність вимірюється TMEC і TMEC/NRESP. Ефективність вимірюється NRESP. Це означає, що ми маємо забезпечити, щоб піддослідні могли чітко записати час, який вони витратили на навчання, і правильно позначити відповіді на запитання. Змінні в цьому експерименті підсумовані в таблиці 1.2.

Таблиця 1.2.

Змінні експерименту

	Name	Values	Description
Independent variables	Scrum describing method	Textual language	The inspection objects are two types of training materials described in the textual and graphical way separately.
		Graphical language	
Dependent variables	Efficiency	TMEC	The time spent by each reviewer in learning the training materials and answering the question is recorded by all subjects.
		TMEC/NRESP	The time used to generate one correct answer is also recorded.
	Effectiveness	NRESP	The number of correct answers marked by each subject should be recorded.
Main Confounding variables	Scrum Experience	The level of Scrum knowledge	All subjects should in the same level of Scrum knowledge.

Вибір предметів: це дослідження проводилося в академічному середовищі. Для вибору суб'єктів для проведення цього експерименту ми використали стратегію зручності та оцінки вибірки. Зручна вибірка – це техніка безімовірнісної вибірки для запрошення найближчих і найбільш зручних осіб для участі в експерименті [40]. У цьому проекті зручно відбирати та звертатися до студентів у ВНЗ. Для того, щоб цей експеримент був репрезентативним для всього студентського населення, ми повинні запросити студентів з різною освітою. Таким чином, суб'єктами, відібраними в

цьому експерименті, були як студенти бакалаврату, так і магістри, які спеціалізуються на інженерії програмного забезпечення (і суміжних). Важливо, щоб суб'єкти відчували вільність відмовитися від запрошення без будь-якого примусу. Учасники повинні пообіцяти, що вони серйозно підуть до цього експерименту.

Вибірка суджень – це ще одна стратегія неімовірнісної вибірки для відбору суб'єктів із сукупності. Це упереджений метод, який використовується для відбору суб'єктів, коли деякі особи з популяції є кращими суб'єктами, ніж інші. Причина для проведення вибірки судження полягає в тому, що відмінності в рівні досвіду Scrum суб'єктів можуть серйозно вплинути на результат експерименту. У цьому експерименті суб'єкти повинні були виконати деякі завдання Scrum, і ми вимірювали вплив лікування на них. Якщо суб'єкти добре знають Scrum, вони витрачатимуть менше часу на вивчення матеріалів і легко відповідатимуть на запитання на основі навчальних матеріалів, незалежно від того, який тип навчальних матеріалів вони використовували. Таким чином, досвід розробки Scrum або рівень знань суб'єкта є переважним фактором, що змішує в цьому експерименті.

За цим фактором ми використали метод вибірки суджень, щоб відібрати студентів, які не мають досвіду чи знань про Scrum.

Насправді, мета цього дослідження полягає в тому, щоб допомогти розробникам краще зрозуміти Scrum, для проведення цього експерименту краще використовувати професійних розробників.

План експерименту: щоб спланувати експеримент, ми маємо розглянути гіпотезу, щоб побачити, який статистичний аналіз ми використали, щоб відхилити нульову гіпотезу. На основі статистичних припущень, щоб оцінити різницю значущості між двома різними мовами опису, ми розробили дизайн експерименту як рандомізацію та балансування.

У цьому експерименті ми хочемо дослідити, чи запропонована мова для опису процесу Scrum має вищу ефективність навчання, ніж текстовий спосіб опису процесу Scrum. Фактором є метод опису, а методами лікування є запропонована мова та мова тексту. Відповідно до мети експерименту ми вирішили застосувати до цього експерименту рандомізований дизайн. Ми використовуємо однакові об'єкти

для обох процедур і випадковим чином розподіляємо суб’єктів для кожної процедури. Кожний суб'єкт повинен використовувати тільки одну процедуру на одному об'єкті [40].

Навчальний матеріал складається з двох частин: вступу до процесу Scrum та прикладу проекту застосування Scrum. Наприкінці кожної частини є вікторина з кількома запитаннями, щоб перевірити, наскільки випробуваний навчився після вивчення поданого матеріалу. Збираючи та аналізуючи дані вікторини, ми могли виміряти зручність вивчення кожного навчального матеріалу.

Перша частина навчального матеріалу є коротким вступом до основних ідей Scrum. Він має на меті надати суб’єктам базові знання та розуміння Scrum. Ця частина включала опис процесу Scrum, артефакти Scrum і ролі команди Scrum. Щоб забезпечити точне ознайомлення зі знаннями, цей матеріал редагується та керується на основі Навчального посібника Scrum Master [41]. Навчальний посібник є посібником, який допоможе учню Scrum скласти професійний іспит Scrum Master (PSM). Порівняно з іншою літературою, вступну схему та контекст цього навчального посібника легше зрозуміти та засвоїти. Як професійний посібник для PSM, він більше зосереджений на застосуванні, а не на чистій філософії або теорії Scrum; отже, воно ближче до орієнтування реальних умов розвитку. Питання вікторини першої частини також вибрані з навчального посібника. Маленька вікторина містить 5 питань. Ці запитання призначені для того, щоб оцінити, чи розуміють суб’єкти основні елементи, зв’язок і процес виконання методу Scrum. У вікторині було два типи питань:

1. Перевірка розуміння побудови та відповідальності ролей команди Scrum;

2. Питання про те, як організувати діяльність під час ітераційного циклу Scrum.

Результати вікторини дозволять нам виміряти, наскільки суб’єкти дізналися та зрозуміли про Scrum. Це також опосередковано відображає те, який тип навчального матеріалу є кращим для навчання Scrum.

Друга частина містить приклад проекту процесу Scrum, який суб’єкти можуть навчитися застосовувати метод Scrum на практиці. Приклад проекту можна розглядати як об’єкт експерименту, який було переглянуто з документу конкретного

проекту Scrum. Важливо вибрати відповідний об’єкт експерименту, який можна використовувати як середовище для оцінки можливостей навчання двох мов моделювання Scrum. Ми вибрали конкретний студентський проект Scrum як об’єкт експерименту, а потім описали цей приклад проекту двома способами: текстовим і графічним. Причина, по якій ми вибрали студентський проект, полягає в тому, що промисловий проект Scrum надто складний для розуміння новачками. Крім того, важко отримати доступ до конкретного проекту Scrum в галузі через деякі комерційні причини. Студентський проект має більше переваг перед промисловим проектом. Загалом, масштаб студентського проекту є відносно невеликим, тому учням буде легше отримати деталі проекту протягом обмеженого часу.

Зважаючи на обмежений час виконання експерименту, ми лише представляємо огляд проекту та проілюструємо два Спринти як приклади, щоб учасники могли вивчити та зрозуміти процес Scrum. Ми також навмисно пропускаємо деякі важливі елементи процесу Scrum, щоб визначити, чи суб’єкти розуміють Scrum, і знайти помилки в застосуванні. Тест, який додається до другої частини, містить 7 запитань, які зосереджені на перевірці розуміння суб’єктом Scrum, зручність вивчення навчальних матеріалів і простоти використання для пошуку детальної інформації про проект. Продукт проекту створювався ітеративно, для динамічної зміни частини, чи може суб’єкт легко знайти правильну інформацію в навчальних матеріалах. Типи запитань можна підсумувати так:

1. Запит про те, скільки днів у певному спринті;

2. Запит вмісту завдань оновлення;

3. Перевірка завершеності процесу або дій відповідно до визначення методу Scrum.

Вибірка «сніжної кулі» — це тип розширеної випадкової вибірки, ми не можемо контролювати вимоги до досвіду Scrum.

• Пілотно-експертний огляд

Для того, щоб обстеження було якіснішим та отримало точніший результат. Ми провели пілотне опитування з досвідченими користувачами Scrum, які мають більше трьох років досвіду Scrum. Під час пілотного опитування анкета пілотного

опитування надсилається вибраним користувачам Scrum електронною поштою. Потім ми зібрали від них відповіді. Одночасно були отримані і відгуки анкети опитування. Після цього ми вдосконалили анкету відповідно до отриманих відповідей та відгуків.

• Анкета опитування

Анкета складається з двох частин. Перша частина - це приклад проекту, описаний як текстовим, так і графічним способом. Друга частина — це набір запитань про думки щодо порівняння двох типів виразів.

Таблиця 1.3.

Питання опитування

Питання	**Причина**
Як давно ви використовуєте Scrum?	Це питання використовується для визначення рівня досвіду розробника.
Як ви навчилися Scrum?	Це питання використовується для аналізу найпоширенішого методу навчання Scrum сьогодні.
Як ви вважаєте, чи може графічна мова допомогти вам зрозуміти процес чи проект Scrum?	Це запитання використовується для вимірювання цінності графічної мови для досвідчених користувачів Scrum.
Як ви думаєте, чи може графічна мова допомогти новому учню краще зрозуміти процес Scrum?	Це запитання використовується для вимірювання цінності графічної мови для початківців Scrum.
Який спосіб (текстовий/графічний) ви б хотіли використовувати в процесі навчання та викладання Scrum.	Це запитання використовується, щоб визначити, який спосіб (текстовий чи графічний) розробник віддає перевагу використанню під час
На вашу думку, які переваги та недоліки навчання процесу Scrum за допомогою цієї графічної мови?	Це запитання використовується для збору думок користувачів Scrum щодо графічної мови.
На вашу думку, які переваги та недоліки навчання процесу Scrum з використанням текстових матеріалів?	Це запитання використовується для збору думок користувачів Scrum щодо текстової мови.
Які проблеми чи перешкоди виникають під час вивчення процесу Scrum?	Це запитання використовується для збору проблем, які виникли під час традиційного навчання Scrum.
Як ви вважаєте, чи може графічна мова допомогти вам вирішити або пом’якшити вищезазначені проблеми, і чому?	Це запитання використовується, щоб зібрати потенційні рішення вищезазначених проблем за допомогою графічної мови.
Чи будете ви використовувати цю графічну мову в майбутній роботі? чому	Це питання використовується для аналізу бажання запровадити графічну мову в промислове середовище.

Для частини запитань ми розробили 10 запитань, у тому числі чотири закритих і шість відкритих. Таблиця 1.3 показує запитання та причини, чому ми їх ставимо.

Опитування триватиме чотири тижні. Через брак ресурсів промислових користувачів Scrum і обмеження часу для нашої магістерської роботи ми вважали чотири тижні належним часом для нашого опитування.

Є два основні джерела контакту з користувачами Scrum: зв'язатися з компаніями програмного забезпечення, щоб запитати, чи є досвідчений розробник, зацікавлений, і зв'язатися з користувачами Scrum через програми соціальних мереж, такі як Facebook і LinkedIn. Анкета опитування надсилається електронною поштою.

Висновки до розділу

В даному розділі ми застосували метод якісного вмісту для аналізу даних, оскільки більшість питань опитування є суб'єктивними. Як ми зазначали раніше, метою цього опитування є збір думок промислових користувачів Scrum щодо запропонованої мови, щоб дослідити її потенційне використання в промисловості. Проводячи якісний аналіз, ми визначили переваги та недоліки запропонованої мови з точки зору користувача Scrum, а також оцінили корисність запропонованої мови.

РОЗДІЛ 2
ПОБУДОВА ДОМЕННО-СПЕЦІФІКОВАНОЇ МОВНОЇ МОДЕЛІ

2.1 Впровадження доменно-специфікованої мови

Доменно-орієнтована мова (DSL) — це тип мов, розроблених для певного домену. DSL забезпечує – суттєві переваги у виразності та простоті використання порівняно з мовами загального призначення в їх домені" [43, 37]. Враховуючи особливості конкретної області, DSL покращує обмін знаннями між експертами області та розробниками [44]. DSL розроблено для обмеженого домену. Таким чином, нотації та конструкції можуть бути виражені коротко без неоднозначних загальних мов.

Добре спроектований DSL повинен відповідати наступним принципам [44]:

- DSL забезпечує пряме відображення елементів, зв'язків і атрибутів конкретного домену.
- DSL має використовувати звичайні терміни конкретного домену. Умови мають бути зрозумілими під час спілкування розробників із експертами домену.
- DSL не може містити жодних неоднозначних виразів.

Щодо виконуваності можна класифікувати за чотирма шкалами [37]. Перший тип DSL має чітко визначену семантику виконуваних файлів, таку як HTML і SQL. Ці типи DSL можна виконувати безпосередньо. Тоді другий тип - це генератор додатків. Користувачі вводять сформульовану мову (екземпляр DSL), а генератор DSL генерує відповідну програму. Третій класичний тип DSL в основному не призначений для виконання, але він може бути корисним для створення додатків. Наприклад, BNF не призначений для прямого виконання або генерації програми, однак його можна використовувати для визначення інших DSL або —ct як мови введення для генератора аналізатора [37]». Останній тип DSL розроблений як невиконувана мова, наприклад, деякі DSL просто призначені для представлення доменно-спеціальної структури даних [37, 45]. Цей тип DSL може отримати користь від різних типів інструментів, таких як адаптовані редактори.

У цьому проекті ми пропонуємо запровадити графічний DSL для моделювання процесу Scrum. Оскільки Scrum — це метод розробки, а не виконувана програма. Тому доцільно розробити невиконуваний DSL для опису процесу Scrum. Графічний DSL має бути розроблений у три етапи: прийняття рішення, аналіз і проектування. У цьому документі етапи розробки та розгортання були видалені з процесу розробки DSL. Наступні розділи проілюстрували детальний процес розробки запропонованого DSL.

На етапі прийняття рішення ми вирішили набір критичних проблем, таких як проблемна область DSL, тип DSL тощо. Ці проблеми визначили наш подальший напрямок розвитку.

Після початкових інтерв'ю результати показали, що всі респонденти позитивно ставляться до ідеї запровадження DSL для процесу Scrum. Вони вважають, що впровадження DSL може допомогти їм зрозуміти метод Scrum і конкретні проекти Scrum. Тому ми прийняли перше рішення, яке полягає в запровадженні DSL для представлення процесу та концепцій Scrum.

Для типу представлення DSL ми порівняли графічний DSL і текстовий DSL, результат був очевидним. В [38] перерахував переваги та недоліки цих двох типів представництва. —Текстові представлення, наприклад, зазвичай мають перевагу швидшої розробки та не залежать від платформи та інструменту. [38]» З іншого боку, —графічні моделі забезпечують кращий огляд і легші для розуміння моделі. [38]» Нарешті ми вирішили розробити графічний DSL, оскільки кращий огляд і легке розуміння — це саме те, чого ми хочемо.

Business Process Modeling Notation (BPMN) — стандартна нотація для моделювання процесу [46, 47]. Це дуже популярне графічне позначення для представлення домену процесу. Є тисячі документів і статей, які говорять про BPMN. Враховуючи запропонований DSL має аналогічну мету: ілюструвати процес. Єдина відмінність полягає в тому, що BPMN призначений для всіх бізнес-процесів, а запропонований DSL зосереджений лише на процесах Scrum . Тому ми вирішили повторно використати деякі основні концепції з BPMN. Звичайно, пропонований DSL повинен бути представлений у графічному вигляді.

Крім того, запропонований DSL призначений для того, щоб допомогти учням Scrum краще зрозуміти процес і концепції Scrum. Немає виконуваної програми, яку можна створити. Тому ми вирішили розробити невиконуваний DSL. Потім у майбутній роботі можна змінити запропонований DSL на виконуваний DSL, щоб виконати вимоги в інших проблемах домену. У таблиці 2.1 показано рішення, прийняті на цьому етапі.

Таблиця 2.1.

Рішення, прийняті для розробки DSL

Рішення	Причина
Представте DSL для представлення процесу та концепцій Scrum	Існують певні прогалини домену та потреби, які вимагають репрезентативного підходу для пом'якшення.
Повторне використання концепцій з BPMN	Є дві переваги, якщо ми повторно використовуємо деякі елементи BPMN. По-перше, це може зменшити зусилля щодо розробки, тому що нам не потрібно розробляти новий мовний елемент і пов'язані з ним правила. По-друге, це буде легше сприйняти людям, які раніше вивчали BPMN, тому що їм не потрібно вивчати нові поняття для використання цієї мови.
Графічний DSL	Графічний DSL легше зрозуміти початківцям.
Невиконуваний DSL	Немає необхідності впроваджувати та розгортати на поточному етапі.

2.2. Формальний аналіз домену

На цьому етапі ми провели формальний аналіз домену. Згідно з [37], формальний результат аналізу домену містить чотири частини:

1) визначення домену,

2) термінологію домену,

3) опис концепцій домену.

4) моделі ознак.

Отже, ми представляли нашу роботу, виконуючи цю формальну структуру. У цьому розділі ми спочатку визначили, як ми проаналізували та класифікували вибрані документи. Потім на основі документів ми конкретно пояснили область домену та термінологію домену. Крім того, для кожного терміна ми описали його

поняття та чому ми витягли його з літератури. Нарешті, ми демонструємо модель функцій для завершення цього формального аналізу.

На початку фази аналізу ми спочатку провели дослідження літератури, щоб вибрати потрібну літературу. Крім того, на етапі прийняття рішення ми вже визначили мету DSL, яка полягає в тому, щоб допомогти учню Scrum отримати краще розуміння під час процесу навчання. Тому статті розділені на дві категорії: використання Scrum у проектах і навчання Scrum для студентів. Статті першої категорії використовувалися для виділення елементів Scrum, а статті другої категорії використовувалися для виявлення ефективніших підходів до навчання та навчання Scrum. Потім ми склали функції на основі освіти та запропонований DSL.

• Пропонований обсяг DSL

Запропонований DSL забезпечує графічне позначення та модель для представлення Scrum-процесу слухачам Scrum і покращення розуміння учнями концепцій Scrum і конкретного проекту Scrum. Виходячи з цього опису, пропонований DSL є невиконуваною мовою, що залежить від домену. Отже, генератор додатків або компілятор не входить до сфери DSL.

• Термінологія домену та опис понять домену

Ми витягли елементи Scrum з цих документів. Щоб зробити вихідні дані більш узгодженими, ми об’єднуємо термінологію домену та описи концепцій домену. Для кожного вилученого терміна ми разом описали поняття домену та атрибути для кожного елемента.

Розділ демонструє витягнуті елементи та їхні атрибути.

• Відставання продукту

Усі вимоги, які є основою для фреймворку Scrum, слід вибирати з Backlog продукту. Беклог продукту — це список вимог [24]. Кожна вимога повинна мати властивість, щоб ілюструвати її важливість і послідовність розвитку. Крім того, кожна вимога має коротке пояснення того, що це за вимога. Белог продукту підтримується власником продукту [15]. Вимоги, перелічені в резервній частині продукту, називаються —usr story” [24, 26, 28] або – baklog item” [24, 27, 28].

Причина вибору продукту backlog як елемент дуже очевидна. Кожен проект Scrum повинен мати беклог продукту. Це основна концепція методу Scrum.

• Sprint Backlog

Sprint Backlog — це також набір вимог. Вимоги вибираються з Backlog продукту на Sprint Planning Meeting. Однак, Sprint backlog не тільки є підмножиною продукту backlog, але й надає набагато більш конкретні деталі для кожної вимоги [27], наприклад, що таке «done» для кожної функції [15]. Власник продукту повинен конкретно описати команді розробників вибрані елементи резервної роботи продукту. Відкладення спринту містить список усього, що потрібно зробити під час спринту. Теоретично, після початку спринту всі нові оновлення повинні бути додані в наступному спринті. На практиці, під час спринту, деякі невеликі, але термінові вимоги також можуть бути додані до Sprint Backlog. Крім того, може бути кілька команд, які мають спільний доступ до одного продукту, але лише одна команда бере певний спринт-беклог [15].

• Товар Backlog Item

Цей елемент є атрибутом Product Backlog. У списку історій Backlog продукту коротко описані історії користувачів і загальний приблизний час розробки.

Таблиця 2.2.

Список атрибутів елемента Backlog продукту

Атрибут	Причина і опис
Ім'я	Коротка описова назва історії може стисло пояснити мету елемента резервного виконання продукту [28].
Важливість	Важливість використовується для впорядкування елементів історії. Лише власник продукту може вибрати цей атрибут. Цей атрибут є важливим, оскільки вся робота з упорядкування базується на цьому атрибуті [24].
Початкова оцінка часу	Цей атрибут призначений для приблизної оцінки вимог користувача на початковому рівні. Цей атрибут може допомогти Product Owner і Scrum Master організувати scrum-діяльність [28]. Цей номер має надати команда розробників, яка збирається розробляти цей елемент. Оцінка не обов'язково має бути дуже точною, оскільки її часто використовують для порівняння один з одним сюжетних елементів. Наприклад, якщо для елемента історії оцінюється один раз, а для іншого елемента історії – два. Це лише означає, що для завершення другої історії користувача потрібно більше часу. Це не означає, що друга користувальницька історія повинна витрачати час вдвічі, ніж перша [28].

Sprint Backlog Item

Sprint Backlog Item подібний до Product Backlog Item. Однак опис історій користувачів у спринті має бути більш конкретним і добре оціненим. Бэклог спринту містить список усіх дій, які потрібно виконати в цьому спринті [27]. Під час наради з планування спринту власник продукту конкретно пояснює вимоги користувача. Потім команда розвитку слід ретельно оцінювати витрати часу. Ці дані мають бути записані та використовуватимуться в усьому спринті.

Таблиця 2.3. Список атрибутів Sprint Backlog Item

Атрибут	Причина і опис
Ім'я	Коротка описова назва історії може стисло пояснити мету елемента резервного виконання продукту [28].
Важливість	Цей атрибут використовується для вимірювання пріоритету розвитку в спринті. Значення цього атрибута має бути узгоджено на нараді з планування спринту власником продукту. Під час спринту власник продукту може оновити це значення, щоб налаштувати та контролювати прогрес розробки [27].
Оцінка часу	Цей атрибут подібний до атрибута в «Списку невиконаних історій продукту». Однак цей атрибут є більш точною оцінкою часу. Під час наради з планування спринту Команда розробників має ретельно оцінити час для кожної розповіді про відставання в спринті [28].
Сценарій	Цей атрибут є описом високого рівня того, як історія може бути продемонстрована під час спринту [3].
Критерії	Є багато літератури, яка наголошує на важливості визначення «Готово» для історії [24, 15, 27, 28, 33]. Власне, «Дне» — це своєрідний критерій. Коли артефакт програмного забезпечення відповідає критерію, ця історія вважатиметься готовою».
Категорія	Цей атрибут вказує, до якої вимоги відноситься цей елемент. Наприклад, цей атрибут може бути -testing" —dsign" —otimization" і так далі [28].
Заявник	З боку власника продукту він/вона може продовжувати контактувати з клієнтом або зацікавленою стороною, щоб отримати більше деталей про вимоги. Що стосується розробки, команда може використовувати цей атрибут, щоб відстежити людину, якій дійсно потрібен цей предмет, і отримати більш конкретні та чіткі відгуки за допомогою особистого обговорення [28].
ID відстеження помилок	Якщо проект має окрему систему відстеження помилок, цей атрибут буде корисним для відстеження будь-якої відповідності між елементом і помилками.

Команда розробників Scrum зазвичай визначає завдання для виконання невиконаних історій Sprint. Відповідно до Quaglia [26], усі завдання повинні бути визначені в рамках спринту. Потім члени команди вирішують, хто що робитиме з

огляду на поставлене завдання. Незважаючи на те, що цей елемент не визначено в офіційних документах Scrum, це все одно є вагомою причиною для вилучення цього елемента, оскільки існує багато повідомлень про використання цього завдання на практиці [26]. Зазвичай завдання призначається лише одному розробнику.

Таблиця 2.4.

Список атрибутів завдання

Атрибут	**Причина і опис**
Ім'я	Коротка описова назва історії може коротко пояснити мету завдання.
Розрахунковий час	Цей атрибут використовується для вимірювання часу, витраченого на
Фактично витрачений час	Фактичний час використовується для запису фактичного часу, витраченого на це завдання. Таким чином, ці дані можуть бути обрані та використані для вимірювання подібних затрат часу на виконання завдання.

У документах та офіційному посібнику не надто багато дискусій. Єдине визначення: «Icrement — це сума всіх Елементів Беклогу Продукту, виконаних у спринті [15]». Крім того, новий приріст має бути повністю закінчений. Власник продукту може вирішити відпустіть чи ні. По суті, Inc rement — це набір елементів продукту. Цей список використовується для відображення елемента «готово».

Спринт — це період часу, протягом якого команда Scrum працює після проведення зустрічі з планування спринту [27]. Це серце Scrum, і всі роботи з розробки виконуватимуться в Sprint.

Таблиця 2.5.

Список атрибутів спринту

Атрибут	**Причина і опис**
Мета	Ціль **спринту** — це ціль спринту, яка може бути досягнута шляхом виконання спринту. Під час зустрічі з планування спринту мета спринту створюється власником продукту. Він надає вказівки Групі розробників і дає команді більше гнучкості для виконання своєї роботи [15].
Відставання в спринті	Цей атрибут є ядром спринту. Усі вибрані історії користувачів і пов’язана інформація будуть представлені в резервному журналі Sprint.
Тривалість часу	Цей атрибут використовується для запису часу, який буде витрачено на цей спринт.

У методі Scrum зустріч є дуже важливим способом спілкування всередині команди Scrum і контролю над ходом розробки [15]. Тому ми вилучили цей елемент, щоб вказати всі зустрічі. В атрибуті «Категорія» слід заповнити певний вид категорії зустрічі.

Таблиця 2.6.

Список атрибутів зустрічі

Атрибут	**опис**
Тема	Цей атрибут використовується для ілюстрації основної теми цієї зустрічі.
Час початку	Цей атрибут використовується для запису часу початку зустрічі.
Тривалість Довжина	Різні види зустрічей мають різну тривалість, наприклад, нарада з планування спринту завжди триває близько 2 годин [28]. Однак для щоденної зустрічі Scrum потрібно лише 15 хвилин [27, 28]. Тому необхідно уточнювати тривалість зустрічі.
Категорія	У теорії Scrum існує три типи базових зустрічей: нарада з планування спринту, щоденна нарада scrum, нарада для огляду спринту та ретроспектива спринту [15]. Беручи до уваги практичну ситуацію, може бути більше різних видів зустрічей; ми розробляємо цей атрибут для визначення категорії зустрічі.
Учасники	Цей атрибут відноситься до всіх людей, які збираються брати участь у зустрічі. Також в цьому атрибуті повинні бути вказані ролі учасників.
Ресурси/Вхід	Цей атрибут визначає всі ресурси або вхідні матеріали, необхідні для цієї зустрічі. Наприклад, під час оглядової наради спринту вхідною інформацією для оглядової наради є програмний артефакт, розроблений у цьому спринті
Результат/Вихід	Цей атрибут визначає всі результати або вихідні матеріали, необхідні для цієї зустрічі. Наприклад, мета спринту може бути результатом зустрічі з планування спринту [27].

Учасник

Цей елемент представляє одну особу, яка бере участь у цьому проекті Scrum. Звичайні Scrum-ролі складаються з власника продукту, Scrum-майстра та команди розробки. Власник продукту має лише можливість підтримувати беклог продукту, наприклад, додавати нову історію користувача в беклог продукту та встановлювати важливість усіх історій користувача [24, 28].

Scrum Master — це один або група людей, які до пома гають власнику Scrum і команді розробників організовувати та проводити заходи Scrum [15].

Теоретично Команда Розробників має бути багатофункціональною та самоорганізованою командою [24, 15]. Це означає, що команда розробників повинна

мати можливість вирішувати всі технічні проблеми. Однак на практиці іноді scrum-команди є напівфункціональними [28, 33]. Тому деякі зовнішні учасники можуть бути залучені, щоб допомогти розробці завершити спринт.

Отже, цей учасник може бути власником продукту, Scrum-майстром, командою розробників або іншими спеціальними ролями. На практиці трьох ролей, запропонованих Scrum guide [15], недостатньо для справжнього проекту scrum. Наприклад, іноді клієнтам потрібно безпосередньо спілкуватися з розробниками [33], цей вид діяльності не входить до діяльності Scrum. Scrum Master не має жодної відповідальності та досвіду для організації такого роду діяльності. Ось чому деякі компанії вводять більше ролей для вирішення подібної проблеми. Найбільш типовою роллю є продуктовий менеджер, яка походить від традиційних моделей розробки [29, 30]. У деяких ситуаціях клієнтам потрібно безпосередньо спілкуватися з командою розробки. Таким чином, до цього проекту залучено замовника [27, 33]. Отже, елемент «учасник» повинен мати здатність представляти ці різні ролі.

Таблиця 2.7.

Список атрибутів учасників

Атрибут	**Причина і опис**
Роль	Цей атрибут використовується для визначення ролі проекту Scrum, наприклад Scrum Master, Product Owner і Project manager [29, 30].
контакт	У проектах scrum акціонерам завжди потрібно спілкуватися один з одним. Тому необхідний атрибут для запису того, як зв’язатися з акціонером.

Функціональна модель

Як ми пояснювали раніше, мета запропонованого DSL — описати Scrum для початківців Scrum. Отже, запропоновані функції DSL можна спростити для демонстрації різних знань. Існує три категорії знань, які пропонований DSL повинен представляти учням.

Перша категорія – термінологія. У цій частині DSL пояснить значення всіх елементів Scrum. Також у цій частині будуть продемонстровані відповідні легенди елементів. Друга частина – Scrum Model. Ілюструючи концептуальний прототип

Scrum, ця частина в основному описує теорію Scrum, принципи Scrum і зв'язок між елементами Scrum.

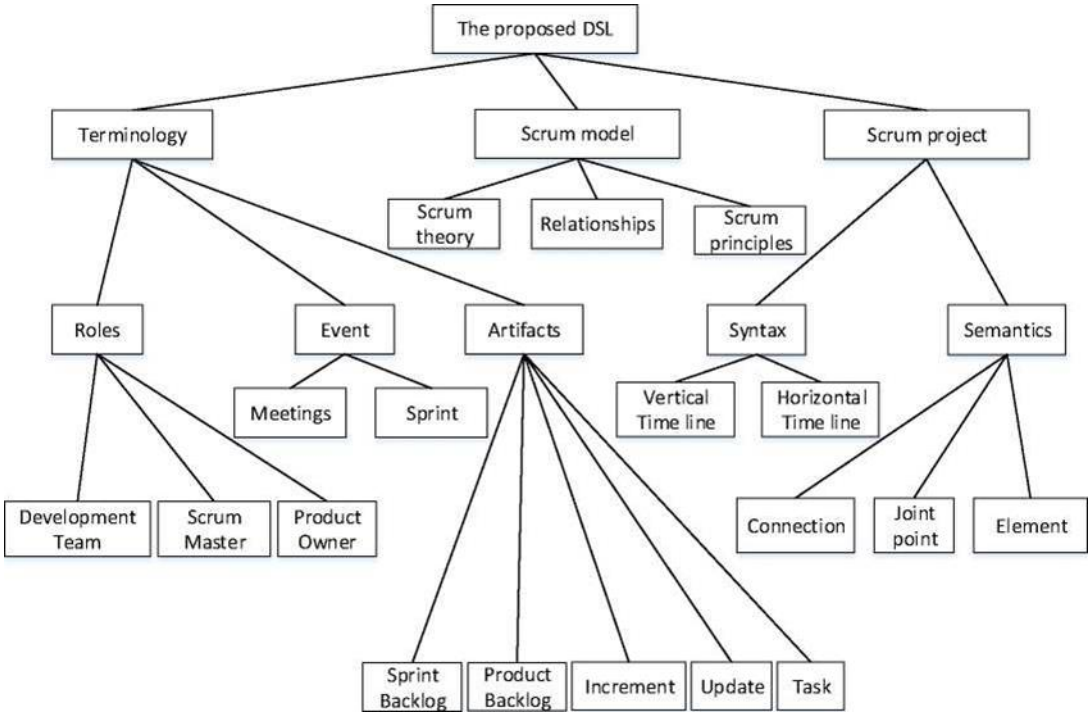

Рис. 2.1. Структура функції DSL

Останньою частиною запропонованого DSL є розділ проекту Scrum. Ця частина використовується для об'єднання двох вищевказаних частин у конкретний проект. Користувач мови може легко побудувати модель процесу проекту Scrum відповідно до синтаксису та семантики запропонованого DSL. Синтаксис стосується того, як побудувати юридичний мовний вираз, тоді як семантика стосується значення побудованого виразу. Базуючись на цій частині, DSL забезпечує зручний спосіб керування та вираження конкретних проектів Scrum.

2.3 Етап проектування доменно-специфікованої мови

У цьому розділі описуються повні деталі дизайну DSL. Відповідно до [37] формальне визначення DSL має складатися з регулярного виразу; граматика

атрибутів, специфікації синтаксису та семантична специфікація. Наступні підрозділи демонструють ці вимоги в різних аспектах.

Ми вже продемонстрували загальну структуру функцій. Структуру мови можна розділити на наступні три частини: термінологія, модель Scrum і конкретний проект Scrum.

У частині - Terminology всі елементи Scrum класифікуються за трьома різними категоріями: роль, подія та артефакт, відповідно до офіційних рекомендацій Scrum [15]. В офіційному визначенні Scrum існує лише три ролі: команда розробників, майстер Scrum і власник продукту. Однак, щоб пропонований DSL міг проілюструвати структуру та конфігурацію групи розробників, наприклад, скільки розробників і скільки часу кожен розробник може внести в команду, ми додали новий елемент ролі - deeloper" на наступному рівні. команди розвитку. Рис. 5 ілюструє структуру «ole» у запропонованому DSL.

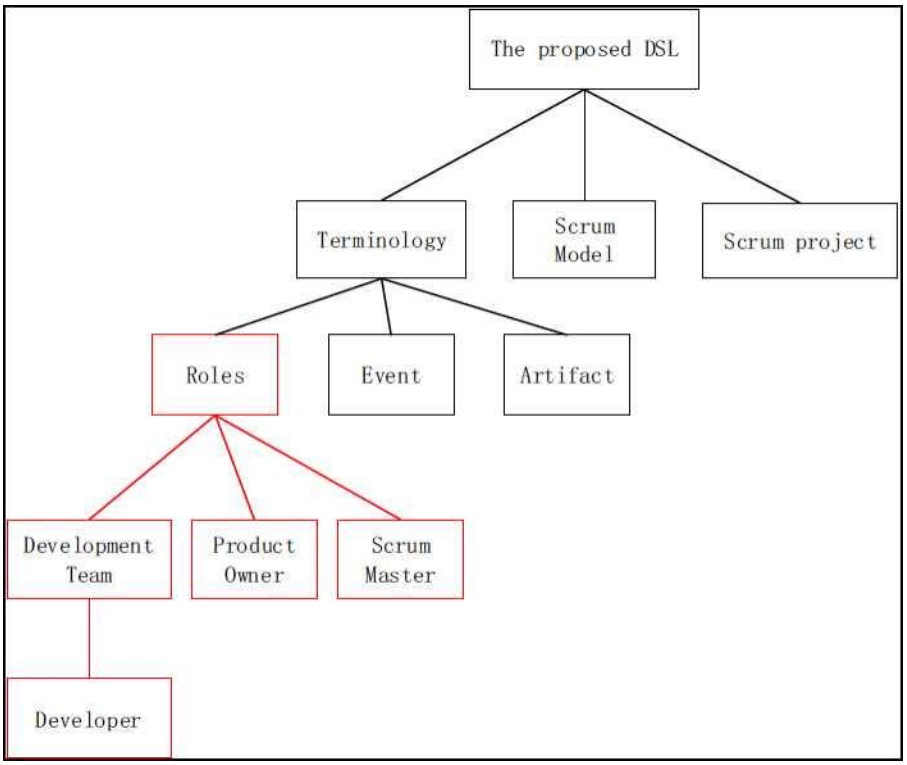

Рис. 2.2. Структура ролей Scrum

В офіційному посібнику Scrum [15] є п'ять типів подій: спринт, ретроспектива спринту, оглядова зустріч Splint, щоденна зустріч Scrum і планування спринту . У

той час як ретроспективу спринту, оглядову зустріч спринту, щоденну зустріч Scrum і зустріч планування спринту можна класифікувати як «зустріч», оскільки всі вони проводяться як стиль зустрічі з однаковими атрибутами елементів. Єдині відмінності полягають у призначенні та тривалості вікна. Тому ми додали новий рівень — ioeting" між - Event" і зустрічами, щоб зробити структуру більш розумною та зрозумілою. На рис. 6 показано структуру — Eent".

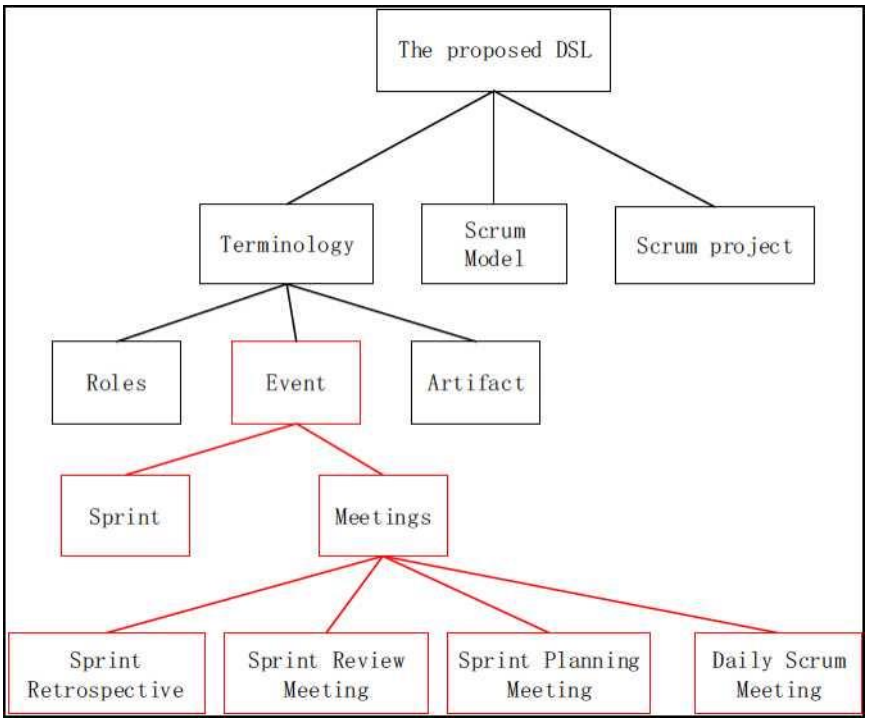

Рис. 2.3. Структура події Semin.

Артефакт» містить Беклог Продукту, Беклог Спринту та Інкремент, кожен із яких — це набір —ЇЄГ ІСТОРІЇ», І ЄДИНА різниця полягає в тому, що Беклог Продукту та Інкремент складаються з Елементів Беклогу Продукту, а Беклог Спринту містить лише Sprint Відставання [15]. Різниця полягає в тому, що елементи Sprint Backlog мають більш детальні описи, ніж елементи Backlog Product.

Як ми представили раніше. Scrum складається лише з кількох принципів і концепцій. Він не надає надто багато деталей. Тоді як призначення запропонованого DSL полягає в тому, щоб конкретно продемонструвати проекти Scrum. Щоб пом’якшити цю прогалину, ми додали нове завдання артефакту, щоб проілюструвати більш конкретну інформацію про те, як розділяється та вирішується

елемент невиконаних завдань Sprint. З іншого боку, як ми всі знаємо. Scrum є однією з гнучких методологій. Найбільшою перевагою Agile є гнучкість. Щоб підкреслити гнучкість, ми знову додали ще один новий елемент —updte». Елемент можна використовувати для вираження оновлень продукту або спринту в спринті або між двома спринтами.

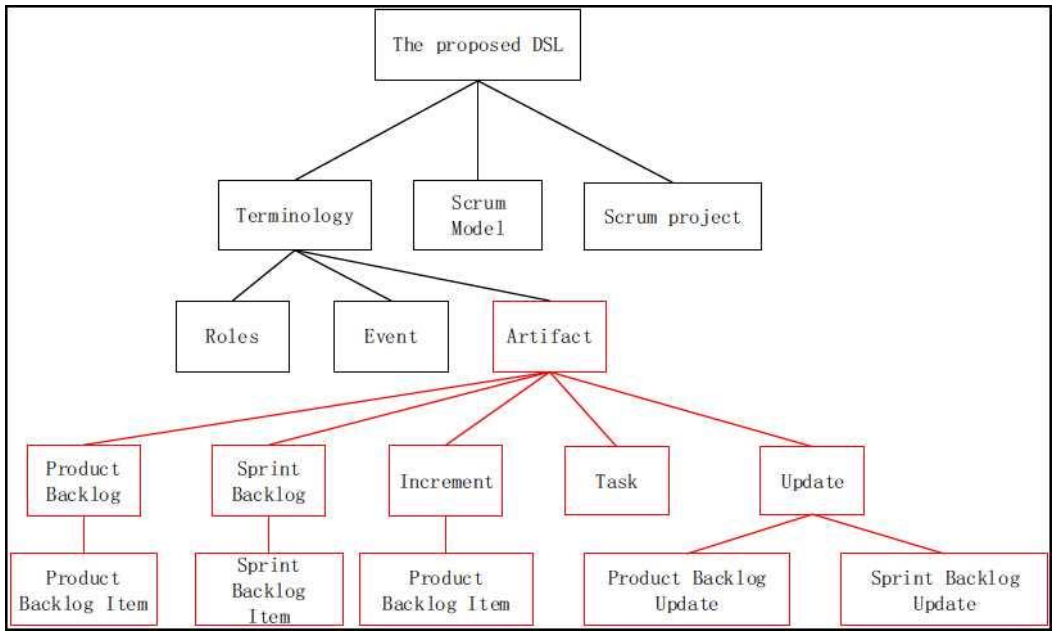

Рис. 2.4. Структура артефакту Scrutn

Друга частина під назвою «Сумова модель», насправді, ця частина використовується лише для демонстрації теорії Scrutn на абстрактному рівні. Проектування цієї частини має освітню мету. Якщо ми хочемо представити лише графічний DSL для моделювання конкретного проекту Scrum, ця частина взагалі буде абсолютно марною, тому що ця частина не належить до жодної важливої частини DSL, привіт іншим словом, навіть цю частину видалено; запропонований DSL все ще завершений. Частина «Модель оглушення» не визначає жодного терміна, синтаксису чи семантики. Це постійна модель для демонстрації концепцій Scrutn. Проте, з іншого боку, необхідно продемонструвати модель Scrutn для початківців Scrum, які не знайомі зі Scrutn. Треба враховувати, що більшість користувачів є новачками в Scrum. Навіть якщо ми познайомимо користувачів із усіма термінами Scrutn, усім синтаксисом і семантикою, вони все одно не знатимуть,

як працює проект Scrutn. Отже, частину « Модель Saitn» слід ввести, щоб покращити розуміння користувачами теорії Scrum. In — Sami Model», запропонований DSL пояснює теорію Scrum, принципи Scrumn і зв'язок між ними. На рисунку 2.5 показана структура — Scrum Model.

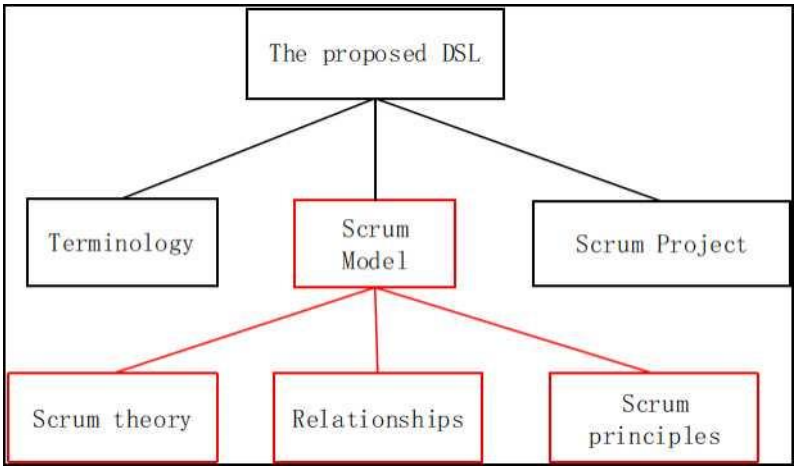

Рис. 2.5. Структура моделі Scrum

Остання частина — проект Scrum. Ця частина містить підхід до керування та ілюстрації конкретних проектів Scrum.

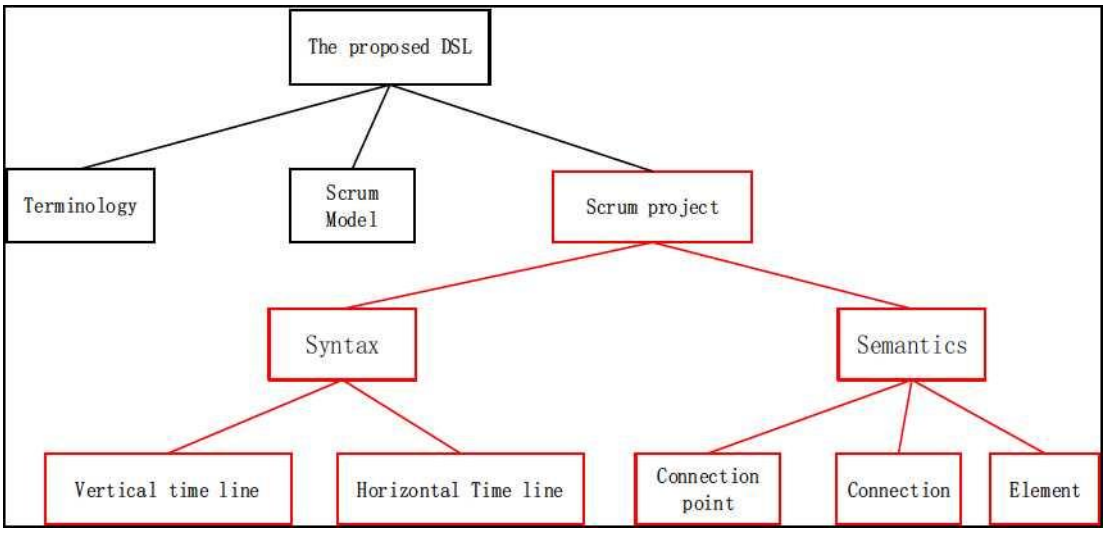

Рис. 2.6. Структура проекту Scnun

Крім того, у цій частині також визначено синтаксис і семантику. Крім того, цю частину можна розглядати як композицію «Термінології» та «Самської моделі». По-перше, користувачі вивчають елементи та терміни Scrum із розділу «Terminology». Потім теорія та принципи Scrum використовуються для створення абстрактного прототипу проекту Scrum у свідомості користувачів. Нарешті, користувачі використовують синтаксис і семантику для керування або створення конкретного проекту Scrum у запропонованому DSL. Рис. 2,6 ілюструє структуру частини проекту «Scrum».

2.4 Термінологія та дизайн елементів

У цьому розділі ми описали деталі елементів та їхні атрибути, вибрані з вилучених елементів. Крім того, легенда, розроблена для кожного елемента, також була проілюстрована в таблиці 2.8. Фактично, атрибути кожного елемента були розроблені так, щоб бути достатніми та досить гнучкими, щоб дозволити користувачам вибрати необхідні елементи для опису елемента. Користувачам не потрібно використовувати всі атрибути.

Таблиця 2.8.

Розроблений елемент для запропонованої мови

Елемент	**Опис і атрибут**
Учасник	**Опис:** цей елемент використовується для позначення власника продукту, Scrum Master і розробника.
	Атрибут: ***Ідентифікатор та ім'я*** *:* використання унікального ідентифікатора або імені для визначення учасника. ***роль:*** Цей атрибут використовується для визначення ролі в команді; це може бути власник продукту, Scrum Master або розробник. ***Контактна інформація:*** Цей атрибут використовується для вказівки контактної інформації учасників, такої як номер телефону або робоча адреса. ***Примітка:*** Цей атрибут використовується для визначення інформації, яка є необхідною, але не включена в цей елемент.
розвиток Команда	**Опис:** цей елемент використовується, щоб показати структуру розробників у команді Scrum Development.

	Атрибут : ***Ідентифікатор та ім'я*** *:* використання унікального ідентифікатора або імені для визначення команди розробників.
спринт	**Опис:** елемент Sprint використовується для запуску нового Sprint у запропонованому DSL.
	Атрибут: ***ID :*** використання унікального ідентифікатора для визначення спринту. ***Тривалість:*** Цей атрибут використовується для визначення часу, необхідного для завершення спринту.
Зустріч	**Опис:** ми повторно використали поле активності BPMN як елемент зустрічі в запропонованому DSL, щоб указати деталі зустрічі.
	Атрибут: ***Зустріч:*** Може бути лише один із чотирьох варіантів-кандидатів [щоденна нарада Scrum, нарада з планування спринту, нарада з огляду спринту, ретроспектива спринту] ***Тривалість:*** Цей атрибут використовується для визначення часу, витраченого на завершення зустрічі. ***Тема:*** Атрибут визначає тему зустрічі. ***Учасник:*** Атрибут визначає учасника, яке значення може бути розробником або командою розробки.

У пропонованому DSL є тип елемента під назвою «Sticture Element». Як і прийменники англійської мови, цей тип елементів не має реального значення без використання з іншими нонімальними елементами.

Таблиця 2.9.

Елементи структури для побудови проекту Scrum

Структура елемент	**опис**
Хронологія продукту	**Часова лінія продукту має бути вертикальною.** The Основна концепція походить від BPMN, привіт BPMN, ця стрілка означає потік послідовності, і вона використовується для показу порядків діяльності [47]. У запропонованому DSL aiiow має подібне значення. Ми розробили часову шкалу продукту, щоб показати порядок Splint і Updates.
Лінія часу шини	**Лінія часу спринту має бути горизонтальною.** Так само, як і час продукту, концепція тимчасової лінії Splint також повторно використовується з BPMN. Шкала часу Splint показує порядок усіх типів зустрічей.
Підключення	З'єднання використовується для з'єднання двох нонімальних елементів або нонімального елемента та точки з'єднання.

2.5. Модель Scrum базового абстрактного синтаксису DSL

Ця модель Scrum використовується для побудови методу Scrum за допомогою базового абстрактного синтаксису запропонованого DSL. Це може допомогти користувачам Scrum краще зрозуміти процес Scrum замість простого відображення самого методу Scrum.

Спринт починається з наради з планування спринту, тривалість наради з планування спринту зазвичай становить від 4 до 8 годин [15]. На цій зустрічі необхідно вирішити такі питання:

- Що можна зробити в наступному спринті?
- Як виконати роботу, визначену в першому питанні?

Під час наради з планування вибирається кілька елементів резервної версії продукту, і власник продукту має вказати деталі цих елементів. Нарешті, ці Елементи Беклогу Продукту стають Елементами Беклогу Спринту. Крім того, майте на увазі, що згідно з принципами Scrum лише власник продукту має право керувати Беклогом продукту та Беклогом спринту. У деяких ситуаціях інші члени команди Scrum вимагають працювати з Backlog Product і Sprint Backlog, включаючи додавання, видалення та модифікацію. Але Власник продукту вирішує схвалювати це чи ні.

Після наради з планування спринту спринт офіційно розпочинається. Спринт – це вікно часу, яке зазвичай становить від 2 до 4 тижнів. Під час спринту щоденна зустріч Scrum проводиться щодня. Зазвичай щоденна зустріч Scrum триває 30 хвилин. Під час щоденної зустрічі Scrum кожен із членів команди розробників повинен відповісти на такі запитання:

- Що він/вона зробили вчора для досягнення результату в спринті ?
- Що він/вона зробить сьогодні для досягнення мети спринту?
- Чи існує якась перешкода, яка заважає йому/їй або команді розвитку досягти цілі спринту?

Власник продукту зазвичай відсутній на щоденних зустрічах Scrum. І під час Спринту лише Команда Розробників може змінити Беклог Спринту. Через 2–4 тижні

спринт завершується, а елементи резервного журналу продукту «dne» вибираються в Increment. Буде проведено оглядову нараду спринту, щоб перевірити приріст і адаптувати Backlog продукту. Нарада для перегляду спринту завжди проводиться чотири години (для спринту на один місяць). Елементи Backlog Product, які не були виконані, слід повернути до Backlog Product і змінити їх порядок. Після оглядової зустрічі спринту буде проведено тригодинну (на один місяць спринту) ретроспективну зустріч спринту, щоб перевірити саму команду. Мета Sprint Retrospective:

- Щоб перевірити останній спринт щодо людей, стосунків, процесу та інструментів.
- Підводячи підсумок, Sprint Backlog Items пройшли добре та забезпечують потенційне покращення
- Планувати покращення.

Синтаксис - це граматика мови. Він визначає порядок елементів і структуру вираження. Ми визначили синтаксис за допомогою діаграми класів з уніфікованої мови моделювання (UML). У цьому розділі, по-перше, ми показали зв'язок успадкування серед нормальних елементів запропонованого DSL. Крім того, нам потрібно вказати, що версія UML передує 2.5, отже, немає перекриття між дочірніми класами. Наприклад, розробник більше не може бути власником продукту.

Як ми можемо бачити на діаграмі, «учасник» є суперкласом і має три дочірніх класи: — Developer — Product Owner — Srum Master. Дочірні класи не мають нових атрибутів, але вони представляють різні ролі в Scrum. Декілька розробників складають «Команду розробників», яка є однією з трьох ролей Scrum.

Суперклас - Зустріч має чотири класи; вони також не мають більше атрибутів. Але всі ці дитячі класи займають різні частини в проекті. Наприклад, нараду з планування спринту слід використовувати лише в перший день спринту; навпаки. Оглядова зустріч спринту та ретроспектива спринту зазвичай проводяться в останній день спринту.

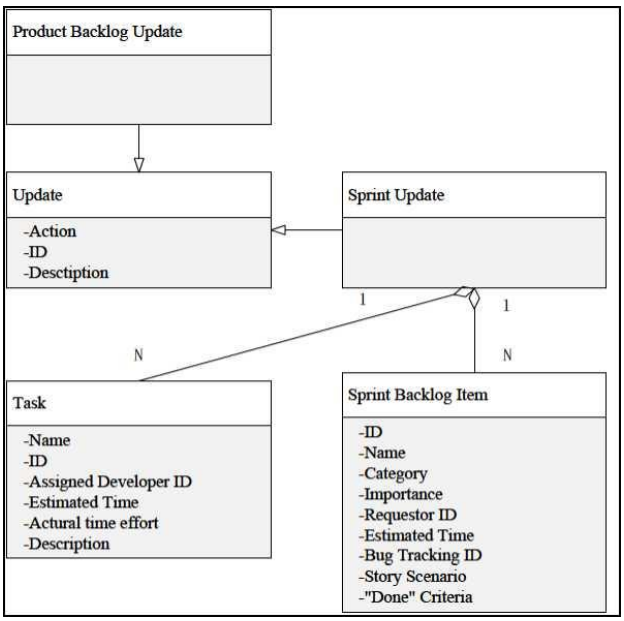

Рис. 2.8. Визначення зв’язку Update

Суперклас — це нова концепція, представлена запропонованим DSL і має два дочірніх класи. Дочірні класи не надають нових атрибутів, але вони мають інше значення в проекті Scrutn. Оновлення спринту — це операція на шкалі часу спринту. Це відбувається під час спринту та зазвичай створюється командою розробників. Оновлення спринту може змінити два типи елементів: «Завдання» та — надрукувати елемент невиконаної роботи».

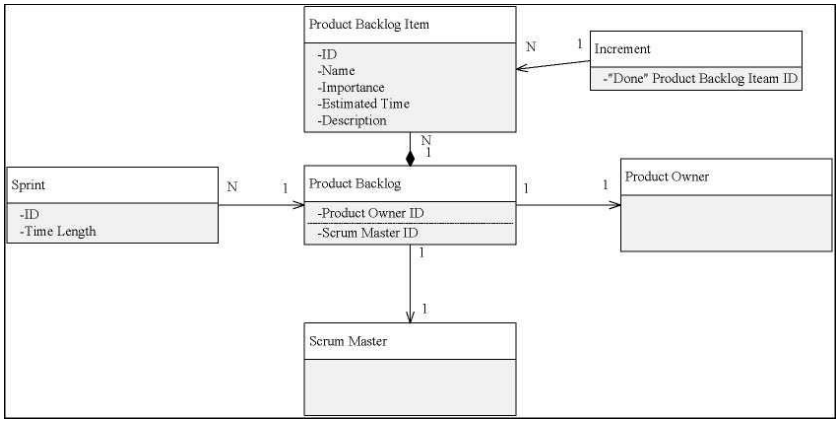

Рис. 2.9. Визначення зв’язку для Backlog продукту

На цій діаграмі ми можемо помітити, що Беклог Продукту складається з кількох Елементів Беклогу Продукту. З іншого боку, Backlog продукту може

підключитися лише до одного власника продукту та одного Scrum Master. Крім того, кілька спринтів можуть спільно використовувати один продукт-беклог. Крім того, подібно до Backlog Product, Increment також містить набір елементів Backlog Product. Різниця полягає в тому, що всі елементи Backlog продукту в Increment є — doe».

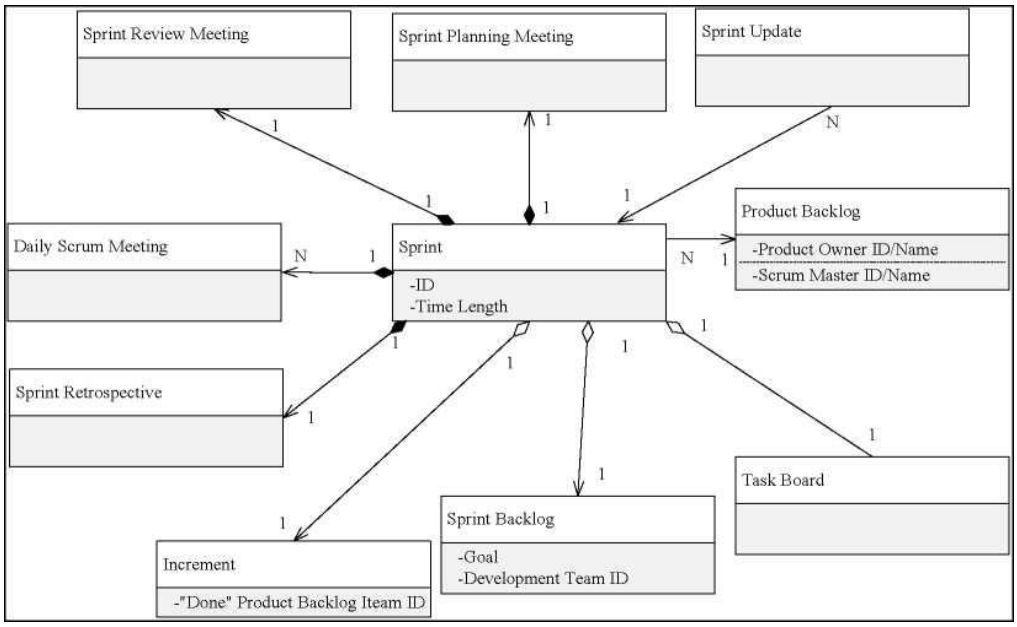

Рис. 2.10. Визначення зв'язку Sprint

У запропонованому DSL Sprint є найскладнішим елементом. Існують нічні різні елементи, які складають Sprint, і Sprint також посилається на Backlog продукту. Точніше кажучи, спринт має композиційний зв’язок із чотирма зустрічами, тому що без спринту зустрічі не мають сенсу й не проводитимуться. Крім того, Sprint має агрегаційний зв’язок із Increment, Sprint Backlog і Task Board. За винятком панелі завдань, інші елементи необхідні для спринту, але вони є артефактами, і навіть якщо спринт скасовано, ці артефакти все ще існують. Крім того, за винятком щоденної зустрічі Scrum і оновлення спринту, спринт може мати зв’язок лише з одним із кожного елемента.

Рис. 2.11 демонструє абстрактний синтаксис Task. Це ілюструє, що завдання можна призначити лише одному розробнику. Крім того, Завдання може належати лише одній дошці завдань, але дошка завдань може містити більше одного Завдання.

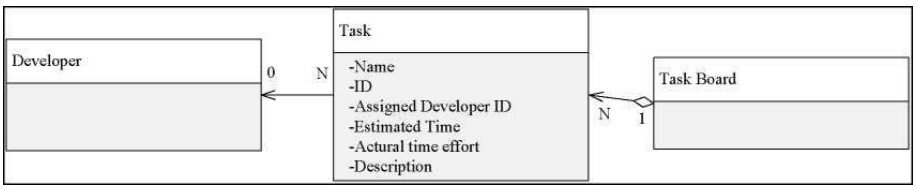

Рис. 2.11. Визначення зв'язку завдання

Рис. 2.12 демонструє абстрактний синтаксис Sprint Backlog Item. Він показує, що Елемент Беклогу Спринту має композиційний зв’язок із Беклогом Спринту, це означає, що якщо Беклоґ Спринту скасовано, усі Елементи Беклогу Спринту в цьому Беклозі Спринту не існують.

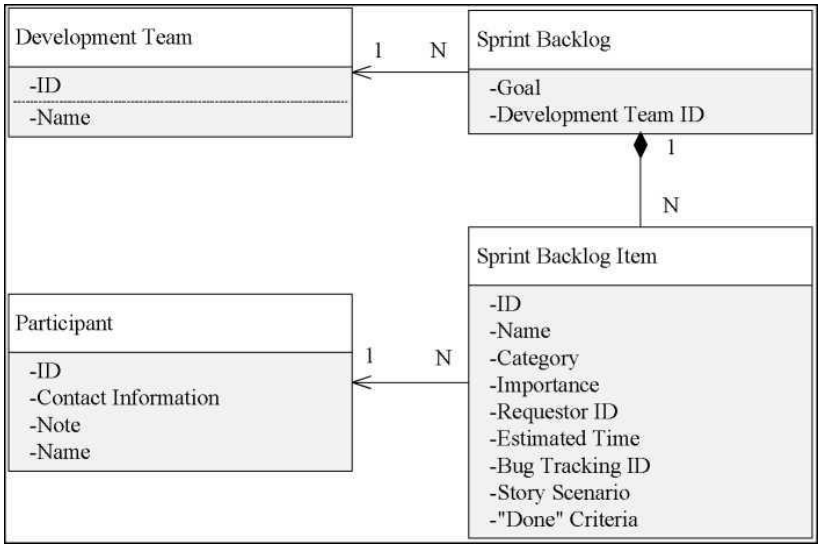

Рис. 2.12. Визначення зв’язку Sprint Backlog.

Крім того, для елемента резерву спринту лише один учасник може бути призначений запитувачем. З іншого боку, Spring Backlog можна призначити лише одній команді розробки.

Висновки до розділу

В даному розділі виконана побудова доменно-специфікованої мовної моделі, розглянуто впровадження доменно-специфікованої мови. Здійснено формальний аналіз домену та досліджено етапи проектування доменно-специфікованої мови.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ МОДЕЛЕЙ ДОМЕННИХ СПЕЦИФІКАЦІЙ ПРОЕКТІВ, ЩО КЕРУЮТЬСЯ МЕТОДОЛОГІЄЮ SCRUM

3.1 Розробка синтаксису та семантики

У цьому розділі ми показали приклад проекту, щоб спростити проект, ми показуємо лише Sprint як приклад. На початку прикладу ми демонструємо огляд усього проекту на рисунку 3.1.

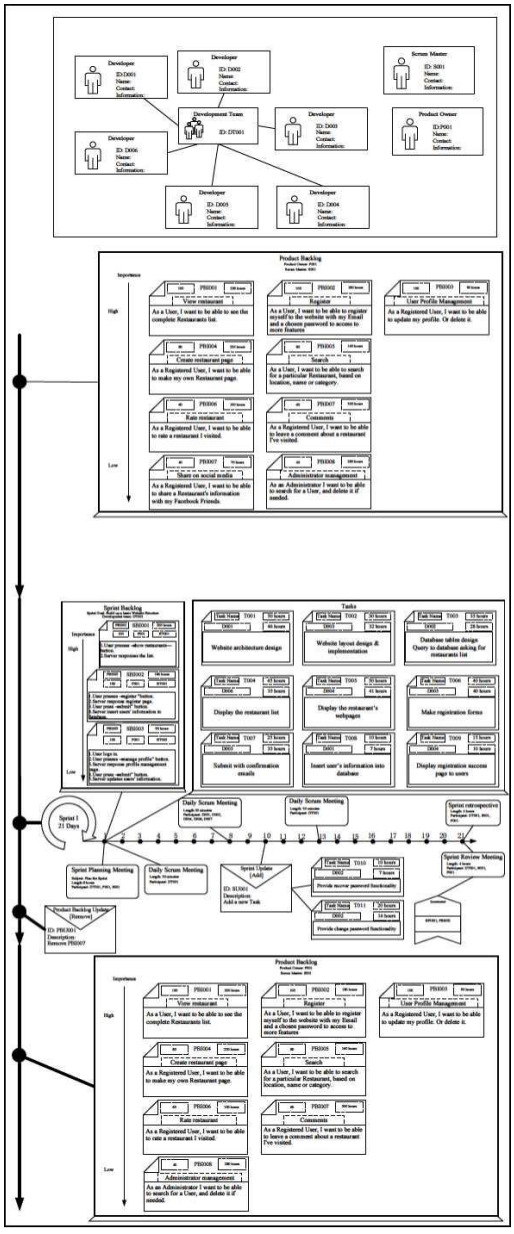

Рис. 3.1 Структура проекту

На рис. 3.2 ми продемонстрували структуру команди Scrum. Як ми бачимо, в команді Scrum є один Scrum Master і один Product Owner. І в команді розробки 6 розробників.

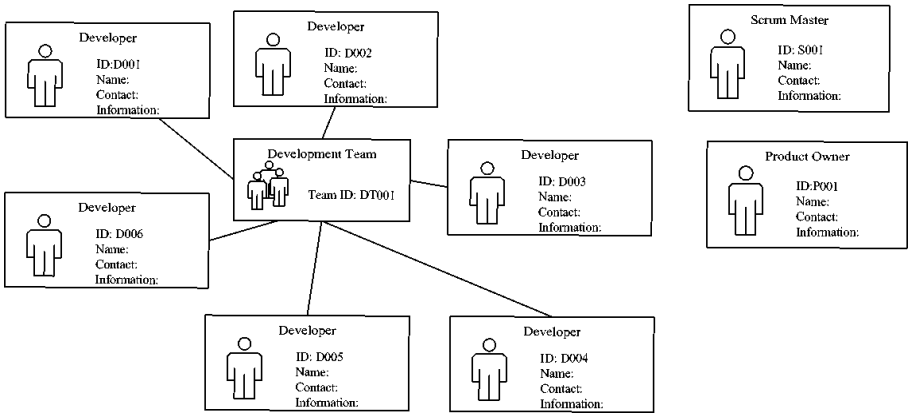

Рис. 3.2. Приклад команди Scrum

На рис. 3.3 показано деталі Backlog продукту. Беклог продукту та його елементи вказуються перед спринтом. У цьому Беклозі продукту є 9 пунктів. Елементи впорядковані за значенням їх важливості зверху вниз. Предмети з однаковими важливість перераховані в одному рядку. Три елементи, важливість яких становить 100, перераховані в першому рядку. Це означає, що вони мають однаковий пріоритет для розвитку. Наступні два елементи зі значенням важливості 80 перераховані в нижньому рядку. Інші предмети впорядковуються за тим самим правилом.

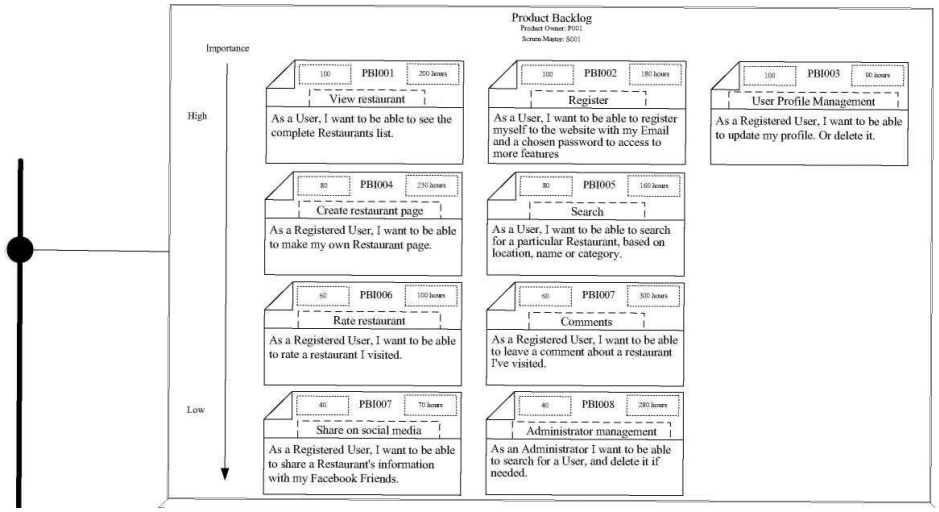

Рис. 3.3. Беклог продукту

На рис. 3.4 показано конкретний спринт. У перший день Спринту на нараді з планування Scrum визначаються пункти Беклогу Спринту, а також встановлюються завдання. Потім щодня проводяться щоденні зустрічі Scrum, а на 8-й день розробник № 3 відсутній і повертається на 13-й день. Завдання № 10 і № 11 оновлюються на 10-й день. Наприкінці цього спринту проводяться ретроспектива Сплінту та оглядова нарада спринту. Крім того, обід на нараді з перегляду Splint використовується для перегляду Приросту, розробленого в спринті. Після цього Сплінта беклог продукту оновлюється.

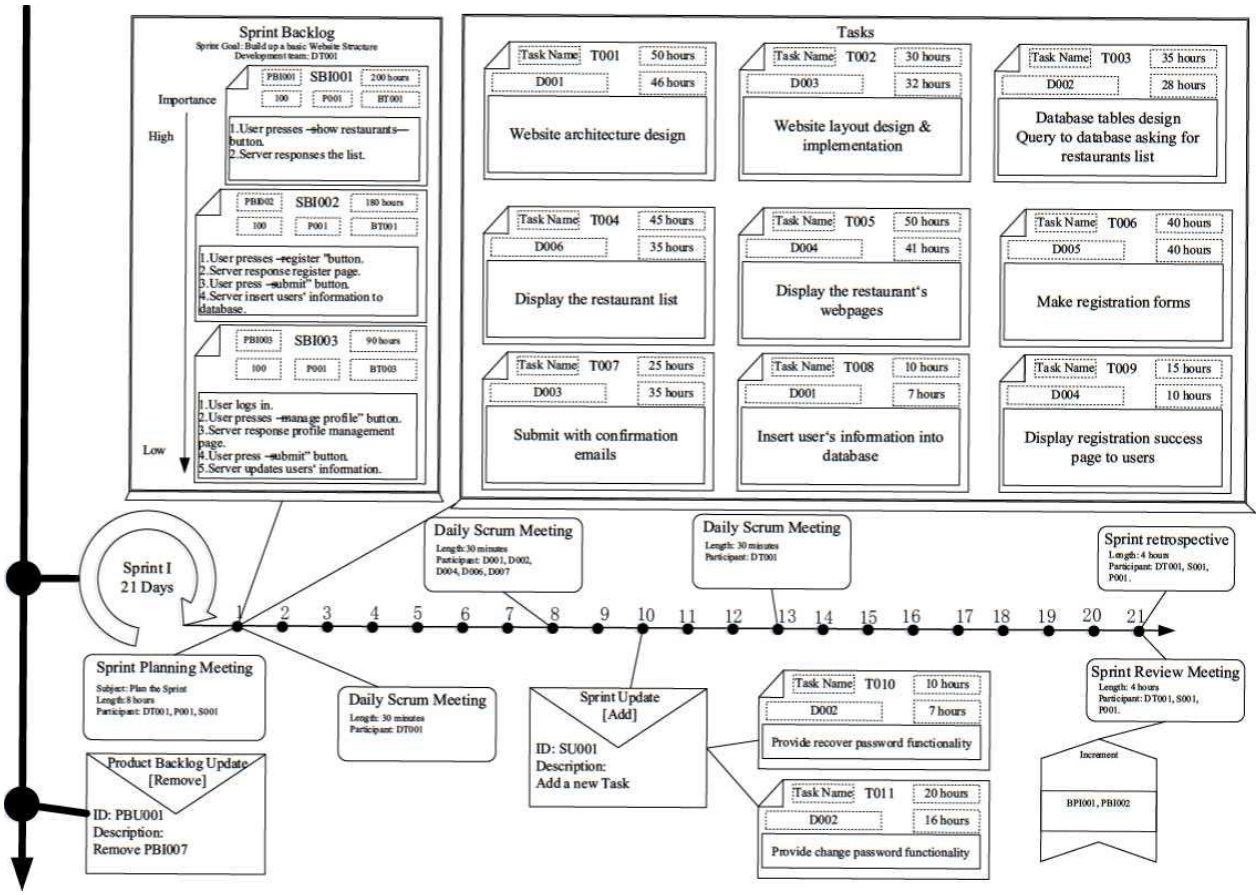

Рис. 3.4. Приклад спринта

На рис. 3.5 показано статус Backlog продукту після спринту. Порівняно з попереднім Беклогом продукту, перші три елементи розробляються, а пункт номер 7 видаляється після Спринту.

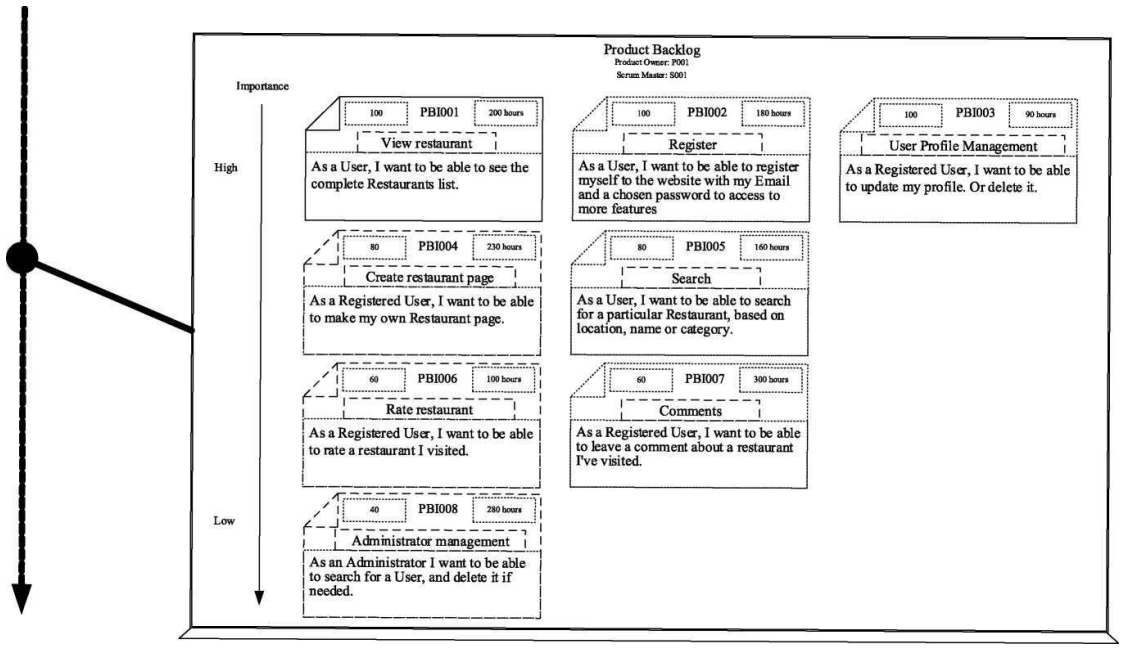

Рис. 3.5. Оновлений резерв продукту

3.2. Процеси підготовки імітаційного моделювання та експерименту

Підготовка. Перш за все, ми запросили студентів взяти участь у цьому експерименті. Ми розіслали лист-запрошення студентам, які навчаються в магістратурі, на електронну пошту. Мотивація для студентів, які беруть участь у цьому експерименті, полягає в тому, що вони можуть отримати деякі додаткові знання про Scrum, які будуть корисними для їхнього майбутнього навчання та освіти. Для того, щоб маніпулювати змішуючим фактором (попередній досвід або знання про Scrum), кандидатами, які ми обрали, були студенти, які мало знали про Scrum. Усі піддослідні проводили цей експеримент добровільно, а не примусово.

Ми склали план часу і забронювали кілька кімнат для піддослідних, повідомили їм, коли і де вони можуть проводити експеримент. Вони можуть вибрати одну доступну точку часу та провести експеримент у призначених кімнатах. У цьому експерименті час є важливим вимірюванням у цьому експерименті; ми підготували піддослідних цифровий годинник, за яким вони могли легко дізнатися час. виконання.

Завдання полягало в тому, щоб вивчити та відповісти на навчальні матеріали. Навчальний матеріал включає дві частини: вступ до методу Scrum та конкретний приклад проекту Scrum. Для графічної версії метод Scrum був описаний запропонованою мовою. Через обмеження можливостей ми не проводили лекції запропонованою мовою для предметів. Це означає, що суб'єкти мають вивчати запропоновану мову, також використовуючи матеріали. Вивчення методу Scrum і конкретного проекту слід виконувати разом. Ми не надали жодного пояснення знань у навчальних матеріалах. У цьому експерименті суб'єкти могли навчитися методу Scrum лише за допомогою експериментальних інструментів. Піддослідні могли робити перерви під час експерименту, якщо вони відзначали час.

Перевірка даних. Дані цього експерименту були зібрані від 16 студентів бакалаврату та 18 студентів магістратури. Однак після оцінки навчальних матеріалів, які вони закінчили, дані чотирьох студентів мають бути видалені, через що дані були визнані недійсними або екстремальними. Ці дані не можуть відображати нормальну продуктивність населення. Результати чотирьох студентів були зняті через:

- Дані одного студента заповнено не повністю; запитання другої частини не мають відповідей.
- Один учасник не записав час.
- Дані двох студентів було видалено, оскільки вони витратили занадто багато часу, щоб завершити експеримент. Дані можна розглядати як викид, який слід видалити. Середній час виконання становить близько 30 хвилин, але вони витратили більше години, щоб завершити експеримент.

Після фільтрації даних ми залишили 14 студентів бакалаврату та 16 магістрантів для статистичного аналізу та інтерпретації результатів.

Експеримент проводився на 34 студентах, але було виявлено, що лише 30 відповідей можна було взяти до аналізу. Результати експерименту показують, що використання графічної мови покращує зрозумілість процесу Scrum. Ми хотіли б розглянути результат з різних точок зору: ефективності та ефективності навчання Scrum.

Ефективність навчання Scrum вимірювалася кількістю правильних відповідей. Точний відсоток відповідей показує, наскільки суб’єкти зрозуміли процес Scrum після вивчення наданих матеріалів. Ми розробили 13 запитань для суб’єктів, щоб оцінити їхнє розуміння, і кожне запитання було зосереджено на одній концепції або використанні методу Scrum. Кожне запитання має чотири або п’ять варіантів. Суб'єкти повинні перевірити правильність чи неправильність кожного вибору; на запитання можна вважати правильну відповідь, якщо всі чотири варіанти правильно оцінені. Це може переконатися, що суб’єкти не зможуть пройти запитання з неправильним або двозначним розумінням. Слід пояснити, що шосте питання було вилучено через неналежне оформлення, оскільки питання не відповідало інформації матеріалів. Тому ми пропустили це питання, коли збираємо відповідні дані. Для кожного запитання використовується стовпчаста діаграма як представлення кількості суб’єктів, які правильно відповіли на це запитання, у різних типах матеріалів на рис. 3.6.

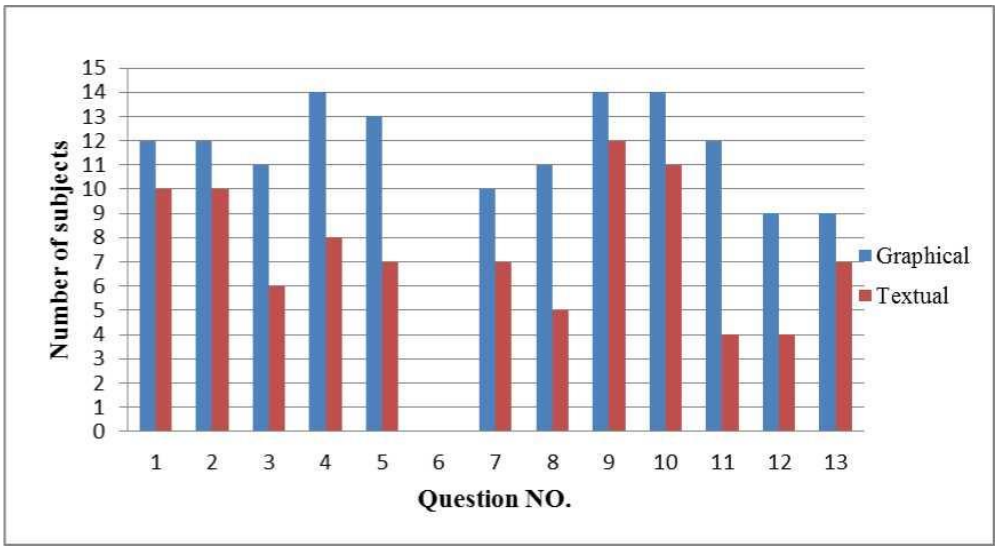

Рис. 3.6. Кількість суб'єктів, які правильно відповіли на кожне питання

На рис. 3.7 показано, що суб’єкти, які користувалися графічною версією, краще розуміють, ніж користувачі текстової версії. Запитання NO.1-NO.5 використовуються для перевірки розуміння основної ідеї Scrum. Для запитання 4 і 5

предмети в графічна версія працює набагато краще порівняно з текстовою. Ці два запитання ставляться, щоб перевірити розуміння використання Backlog продукту та структури команди Scrum. Це вказує на те, що графічна мова працює краще в аспекті пояснення організаційної структури Scrum і використання артефактів. У питаннях № 7-13 користувачі графічної мови демонструють величезну перевагу, ніж текстова версія. Ці запитання зосереджені на детальній інформації та процесі прикладу проекту Scrum. Очевидно, що графічну мову легше використовувати для покращення розуміння процесу проектів Scrum. Піддослідні здатні точно вловлювати інформацію проекту за допомогою графічних мовних матеріалів. Графічна мова справді покращила розуміння процесу Scrum. Графічні позначення є більш наочними, ніж діаграми чи таблиці, щоб пояснити концепції та продемонструвати інформацію про процес Scrum.

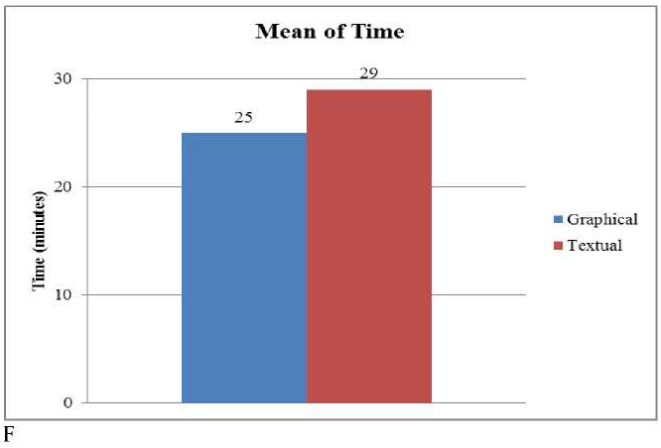

Рис. 3.7. Витрачений час (у хвилинах) на заповнення навчальних матеріалів

Немає сумніву, що час є важливим показником для оцінки ефективності навчання Scrum. Рис. 3.7 показує, що для суб’єктів, які працюють з графічними матеріалами, час, необхідний у середньому для завершення навчальних матеріалів, менше, ніж для суб’єктів, які працюють з текстовими матеріалами. Графічний матеріал легше зрозуміти під час вивчення процесу Scrum.

Крім того, суб’єкти, які працювали з текстовими матеріалами, витрачали вдвічі більше часу, ніж суб’єкти, які працювали з графічними матеріалами, щоб дати правильну відповідь (рис. 3.8). Це означає, що суб’єкти могли швидко зрозуміти та застосувати базові знання процесу Scrum. Результат вказує на те, що графічна мова є більш ефективним способом допомогти суб’єктам вивчити метод Scrum і зрозуміти конкретний проект Scrum.

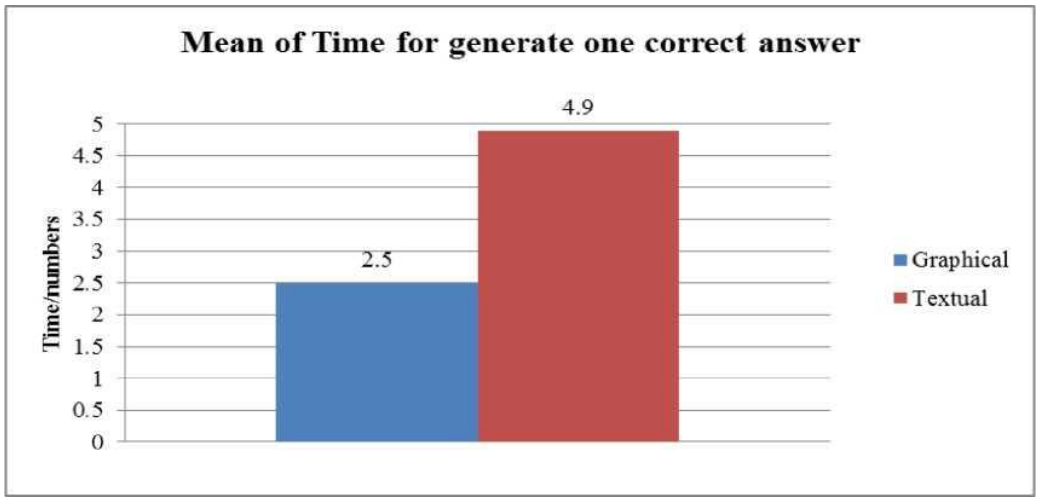

Рис. 3.8. Час, витрачений (у хвилинах) на правильну відповідь

Детальна описова статистика для експерименту проілюстрована в таблиці 3.1. Результат вказує на те, що нульові гіпотези щодо ефективності та ефективності навчання можуть бути відхилені, що підтверджується перевіркою статистичної значущості.

Згідно з планом експерименту, статистичні дані слід аналізувати з вивчення методу Scrum і застосування процесу Scrum у конкретному прикладі проекту. Ми зібрали й узагальнили дані, а також проаналізували результат використання графічної та текстової мови щодо знань Scrum (метод Scrum, RQ3.1) і застосування Scrum у проектах (конкретний проект RQ3.2). Для кожного аспекту є три змінні NRESP, TMEC і TMEC/NRESP, які використовуються для оцінки тренувального ефекту. Відмінності між графічними матеріалами та текстовими також представлені у вигляді значень у таблиці 3.1.

Таблиця 3.1.

Результат експерименту

Variable		Mean		Difference: Graphical-Textual
		Graphical	Textual	
Scrum method part	NRESP	4.40	3.13	1.27
	TMEC	11.53	10.40	1.13
	TMEC/NRESP	2.67	3.80	-1.13
Scrum project example part	NRESP	5.73	3.33	2.40
	TMEC	13.13	18.6	-5.47
	TMEC/NRESP	2.27	6.60	-4.33

Стовпці в таблиці 3.1 містять назву використовуваного тесту статистичної значущості, покращення зрозумілості та перевірку гіпотез. Використання тесту статистичної значущості для кожної змінної визначається результатом нормальності. Таким чином, ми використали тест Шапіро-Вілка 2 для перевірки нормальності кожної змінної, а межею вибрано значення 0,01 (якщо значення результату перевищує 0,83, набір даних слід розглядати як нормальність). Для кожної змінної, якщо як текстові, так і графічні дані є нормальними, t-критерій слід використовувати для перевірки рівня значущості; в іншому випадку ми використовували тест Манна-Уітні для обробки даних.

Після проведення тестів статистичної значущості для кожної змінної результати показують, що спостережувана мова, текстова та графічна мова має значну різницю в розумінні конкретного проекту, за винятком вивчення базових знань Scrum.

Для вивчення основної ідеї методу Scrum можна відхилити нульові гіпотези для змінних NRESP і TMEC/NRESP. Як видно з таблиці 18 , порівняно з текстовою мовою ефективність навчання (TMEC/NRESP) і результативність (NRESP) підвищилися на 41% і 30% при використанні графічної мови для навчання суб’єктів. Однак при засвоєнні базових знань методу Scrum графічна мова значно погіршилася, ніж текстова версія. Це означає, що суб'єкти, які працювали з графічними матеріалами, витрачали більше часу на вивчення методу Scrum. Насправді існує фактор, який може призвести до погіршення результату вивчення графічної мови, а саме вартість навчання графічної мови. Через обмеженість

ресурсів ми не проводили лекції, щоб представити графічну мову, яка використовується в матеріалах. Тому суб'єктам потрібно спочатку вивчити графічну мову, а потім вивчати Scrum на основі цих нотацій. Тому суб'єктам, які працюють з графічними матеріалами, потрібно більше часу приділяти вивченню методу Scrum. Однак після того, як вони вивчили графічну мову, додаткові витрати на навчання повинні бути усунені. Отже, у другій частині експерименту суб'єкти, які працювали з графічними матеріалами, показали значне покращення TMEC, вони витрачали лише 70% часу, який суб'єкти використовували текстові матеріали, необхідні для розуміння проекту Scrum.

Для розуміння методу Scrum у конкретних проектах Scrum графічна мова також має суттєве покращення в ефективності та ефективності навчання порівняно з текстовою мовою. Особливо для змінних NRESP і TMEC/NRESP група графічних мов має більш ніж 60% збільшення порівняно з текстовою мовою. Нульові гіпотези для всіх змінних можуть бути відхилені. Це означає, що графічна мова може впливати на зрозумілість конкретного проекту Scrum.

Таблиця 3.2.

Результат перевірки статистичної значущості

Variable		Shapiro–Wilk test (Textual/Graphical)	Significance level ($p \leq 0.05$)	*H0*-accepted $H0_{effectiveness}$ & $H0_{efficiency}$	*H1*-Polarity
Scrum method part	NRESP	0.85/0.76	0.00152 (Mann-Whitney)	No	Positive
	TMEC	0.73/0.95	0.1141 (Mann-Whitney)	Yes	Negative
	TMEC/NRESP	0.88/0.74	0.03662 (Mann-Whitney)	No	Positive
Scrum project example part	NRESP	0.92/0.85	<0.0001 (t-test[3])	No	Positive
	TMEC	0.95/0.92	0.000795 (t-test)	No	Positive
	TMEC/NRESP	0.77/0.75	0 (Mann-Whitney)	No	Positive

Таблиця 3.3.

Результати покращення/погіршення

Variable		Improvement (value)	Improvement (%)
Scrum method part	NRESP	1.27	41
	TMEC	1.13	-11 (Deterioration)
	TMEC/NRESP	-1.13	30
Scrum project example part	NRESP	2.40	72
	TMEC	-5.47	30
	TMEC/NRESP	-4.33	66

Відповідно до комбінації таблиць 3.2 і таблиці 3.3 ми чітко бачимо, що графічна мова покращила розуміння процесу та проекту Scrum, і нульові гіпотези ефективності та результативності навчання слід відкинути. Результат можна продемонструвати таким чином:

H1: Ефективність (графічний) > Ефективність (текстовий)

H1: Ефективність (графічний) > Ефективність (текстовий)

Якщо графічні матеріали мають суттєву різницю з текстовими матеріалами, крім того, це може покращити ефективність навчання, ми повинні зробити висновок, що запропонована мова є корисною для того, щоб допомогти суб'єктам вивчити метод Scrum і зрозуміти конкретні проекти Scrum. Запропонована мова може покращити розуміння процесу та проектів Scrum.

У цьому експерименті існує чотири типи загроз валідності, які слід проаналізувати: валідність висновку, валідність конструкції, внутрішня валідність і зовнішня валідність. Достовірність висновку пов'язана зі здатністю зробити правильний висновок і зв'язком між методами лікування та результатами експерименту. Загрози побудувати валідність стосуються результатів експерименту для теорії, що лежить в основі експерименту. Загрози внутрішній валідності – це проблеми, які можуть вплинути на незалежну змінну щодо причинності без відома дослідника [40]. Зовнішня валідність стосується узагальнення результату експерименту в інших середовищах, таких як промислове середовище.

У цьому експерименті ми застосували прийнятні статистичні методи для обробки даних і отримання результату. Вибір статистичних методів для кожної змінної був надійним. По-перше, ми перевірили властивості набору даних добре відомим методом. Після цього ми застосували відносні методи для обчислення даних на основі результату тесту. Обробка даних є вимогливою, а методи надійними, щоб виправдати припущення.

Основною загрозою щодо валідності висновку є низька кількість вибірок, що може зменшити здатність узагальнити правильний висновок. Лише п'ятнадцять студентів брали участь у кожній експериментальній групі, що зменшило потужність використаних тестів статистичної значущості. Проте ми серйозно відбирали

суб’єктів і фільтрували дані, що могло забезпечити якість і точність даних і результату.

Основна загроза валідності конструкту полягає в тому, що перед експериментом не було проведено лекції для суб’єктів, щоб представити запропоновану мову. Це призвело до сумного результату: деякі суб’єкти в групі графічної мови витрачали більше часу на вивчення базових знань методу Scrum. Однак загальний час, який суб’єкти працювали з графічними матеріалами витратили на завершення експерименту, був меншим, ніж суб’єкти, які використовували текстові матеріали.

Через обмеженість ресурсів ми не змогли провести лекцію, щоб представити запропоновану нами мову; це справді спричинило певні перешкоди суб’єктам для розуміння запропонованої мови та матеріалів. Але вартість вивчення запропонованої мови також є важливим фактором, який слід враховувати для оцінки корисності запропонованої мови.

Результат показує, що навіть суб'єктам потрібно витрачати додатковий час на вивчення графічних позначень; вони також показали кращі результати, ніж суб'єкти, які працювали з текстовими матеріалами в цілому.

Внутрішня валідність включає три основні загрози. Перша загроза полягає в тому, що конструкція приладів може не підходити для перевірки розуміння предметів. Складність навчальних матеріалів і тестів не така ж, як складність проектних документів реального світу. Наприклад, деякі питання, що стосувалися деталей проекту, були занадто складними, щоб дати повну відповідь. Інструкція навчального матеріалу була незрозумілою; деякі суб'єкти не знали, як правильно використовувати навчальні матеріали, що призвело до того, що вони зробили помилки при заповненні відповідей на запитання, і ми не можемо точно перевірити успішність суб'єктів.

Друга загроза внутрішній валідності полягає в тому, що деяким студентам може бути важко зрозуміти зміст навчальних матеріалів через відсутність досвіду розробки. Деякі предмети також не мають достатніх знань щодо розробки програмного забезпечення. Тому їм важко зрозуміти точне значення технічних

термінів у конкретному контексті. Ця проблема серйозно вплинула на їх навчання та розуміння.

Останньою загрозою внутрішньої валідності є відсутність різноманітності у вибраній популяції. Ми щойно відібрали волонтерів, які мають менше знань про Scrum; тому що результат пілотного експерименту вказує на те, що досвід Scrum суб’єктів був фактором, що вплинув на результат експерименту. Однак це також усунуло різноманітність знань і досвіду піддослідних. Предмети з меншим досвідом Scrum можуть просто представляти нових учнів; вибірка суб'єкта може не охопити всю сукупність користувачів Scrum.

Зовнішня валідність - це відбір суб'єктів. Цей експеримент проводився в освітньому середовищі, тому суб’єктами є всі студенти, і більшість із них – новачки, які вивчають Scrum і не мають практичного досвіду розробки. Це головна відмінність між досліджуваними та промисловими розробниками, які знайомі з процесом розробки програмного забезпечення.

Однак мета цього експерименту полягає в тому, щоб оцінити, як запропонована мова покращує зрозумілість процесу Scrum. Досвід розвитку досліджуваних є фактором, який перешкоджає дослідженню, і його необхідно суворо контролювати. Крім того, вплив досвіду на цей експеримент невиразний.

Ми провели промислове опитування за допомогою анкети з кількома запитаннями, щоб зібрати думки користувачів Scrum щодо запропонованої мови. Ми використали якісний метод для обробки даних і оцінки запропонованої мови.

Тим не менш, у цьому дослідженні, через обмеження часу та контактного ресурсу, метод опитування не було достатньо виконано для оцінки запропонованої мови в галузі. Автор не мав належного ресурсу для контактів із промисловістю, щоб запросити достатньо користувачів Scrum для оцінки запропонованої мови. Крім того, більшість учасників не вивчали та не оцінювали ретельно деталі запропонованої мови через обмеження часу. Дійсно, результат опитування може не відображати належним чином ставлення всіх галузевих користувачів Scrum. Але загалом це все одно допомогло нам визначити переваги та обмеження запропонованої мови відповідно до думок промислових користувачів Scrum. Це

опитування також досліджувало потенційне використання запропонованої мови в галузі.

Проведення опитування: це опитування проводилося з 4 по 31 липня . Ми отримали контактну інформацію користувачів Scrum від наших друзів і опублікованих блогів.

Аналіз результатів: для аналізу даних ми в основному використовували метод якісного вмісту. Мета цього опитування полягає в тому, щоб дослідити, чи може запропонований DSL бути прийнятий промисловими розробниками та їх думки щодо запропонованої мови. Провівши якісний аналіз, ми узагальнили переваги та недоліки запропонованої мови з точки зору користувача Scrum.

Питання 1: Як довго ви використовуєте Scrum?

Після перевірки даних ми нарешті відібрали 17 дійсних анкет для аналізу. У цих користувачів Scrum середній досвід Scrum становить близько 2 років. Три користувачі Scrum мають 3 роки досвіду. Рис. 3.9 ілюструє Scrum розподіл досвіду.

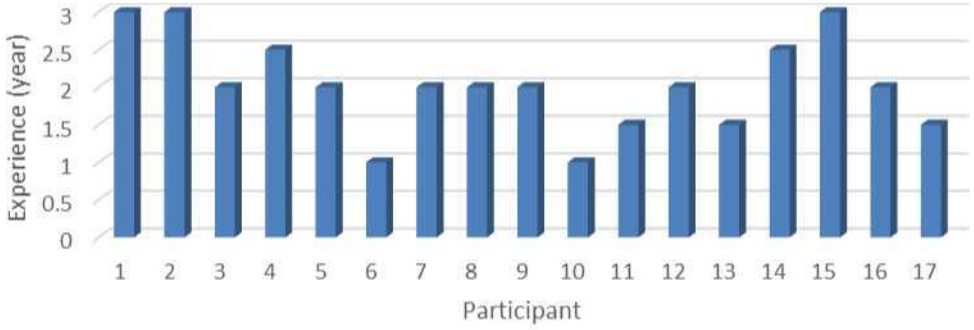

Рис. 3.9. Розподіл досвіду учасників опитування

Питання 2: Як ви навчилися Scrum?

У другому питанні ми з’ясували, що всі учасники пробували вивчати Scrum через Інтернет. Десять учасників вивчали Scrum через книги та літературу. Всього 8 учасників навчання Scrum зі школи. Усі користувачі Scrum мали досвід вивчення Scrum у проектах. Це означає, що моделювання конкретних проектів є потенційним рішенням для покращення навчання Scrum. Рис. 3.10 ілюструє результат запитання 2.

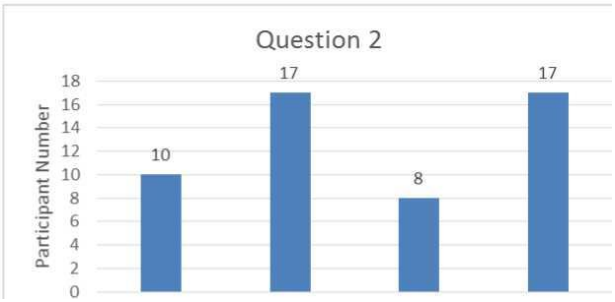

Рис. 3.10. Розподіл відповідей на запитання 2.

Запитання 3: Як ви вважаєте, чи може графічна мова допомогти вам зрозуміти процес або проект Scrum?

Для запитання 3 рис. 3.11 демонструє, що 14 із 17 людей мають позитивну думку щодо графічної мови; більше того, 5 людей цілком згодні з тим, що графічна мова може допомогти їм зрозуміти Scrum. Троє людей не знають, чи корисна графіка для вивчення Scrum. На діаграмі ми можемо помітити, що ніхто не обирає негативні думки. Це певною мірою підтверджує нашу ідею про те, що графіка може допомогти розробникам вивчити Scrum.

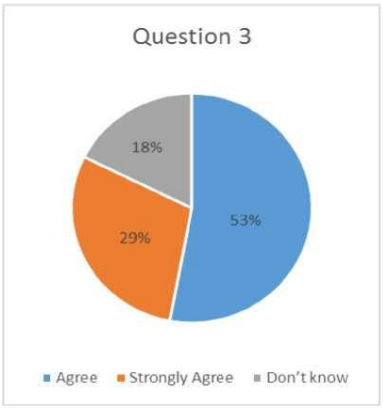

Рис. 3.11. Розподіл відповідей на запитання 3.

Запитання 4: Як ви вважаєте, чи може графічна мова допомогти новоспеченому краще зрозуміти процес Scrum?

Результат запитання 4 майже такий самий, як і запитання 3. Рис. 3.12 показує, що порівняно з відповідями на запитання 3 більше людей вважають, що графічна мова є більш корисною для початківців Scrum. Лише двоє учасників досі дотримуються нейтральної думки щодо можливостей графічної мови. Десять

користувачів Scrum вважають, що графічна мова може надати деякі покращення для учнів Scrum. І п’ятеро людей все ще твердо вірять, що графічна мова може допомогти їм зрозуміти метод Scrum.

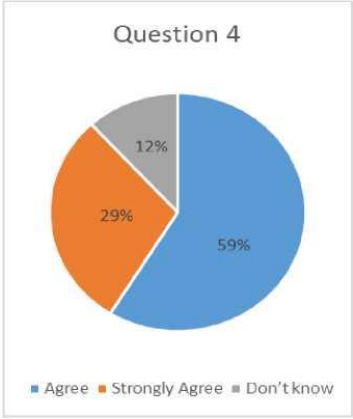

Рис. 3.12. Розподіл відповідей на запитання 4.

Запитання 5: Який спосіб (текстовий/графічний) ви б хотіли використовувати в процесі вивчення та викладання Scrum і чому?

Результат запитання 5 в основному відповідає результатам запитань вище. Однак помітно, що хоча принаймні 15 людей вважають, що графічна мова є корисною для вивчення Scrum (дані з питань 3, 4 та 5), все ще майже 2/3 людей вважають текстову мову необхідною для вивчення Scrum. На рис. 2 людини віддають перевагу навчанню за допомогою тексту, а 9 людей обирають обидва способи. Вони вважають, що поєднання текстових і графічних способів, ймовірно, краще, ніж один метод. Це вказує на те, що все ще є деякі недоліки, які надана графічна мова не подолає.

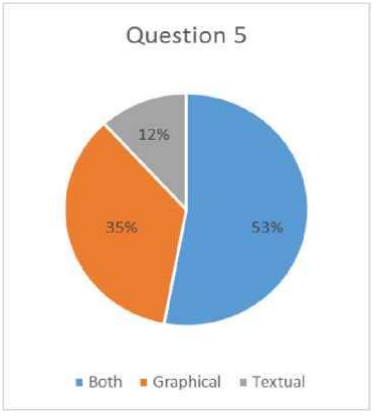

Рис. 3.13. Розподіл відповідей на запитання 5.

Питання 6: На вашу думку, які переваги та недоліки навчання процесу Scrum з використанням запропонованої графічної мови?

Це питання є відкритим. Ми узагальнили основні думки учасників на основі питання 5.

Усі 17 учасників дотримуються точки зору, що графічна мова надзвичайно підвищила ефективність навчання. Крім того, 13 учасників вважають, що запропонована графічна мова може надати легший спосіб отримати огляд концепцій Scrum і конкретних проектів. Дев'ять осіб відзначили, що запропонована мова робить концепції Scrum більш чіткими та зрозумілими. Учасник написав позитивний коментар щодо навчання: «Якщо зміст, який потрібно викладати, справді нудний, то графічний спосіб може бути кращим варіантом, щоб утримати студентів від сну в класі».

Запитання 7: Які, на вашу думку, переваги та недоліки навчання?

Процес Scrum у використанні запропонованих текстових матеріалів?

Це питання також є відкритим. Текстова мова більш сувора для визначення Scrum. Крім того, текстова мова надає більше деталей щодо концепцій Scrum.

Запитання 8: Які проблеми чи перешкоди виникають під час вивчення процесу Scrum?

У цьому питанні є два типи відповідей. Для користувачів Scrum, які мають більше 2,5 років досвіду, вони вважають, що головною перешкодою для них є управління та розподіл завдань. Для користувачів Scrum, які користувалися Scrum протягом 1-2 років, основною проблемою залишається відсутність практичного досвіду. У деяких випадках вони не знають, як реалізувати концепції Scrum у реальних проектах.

Запитання 9: Як ви вважаєте, чи може графічна мова допомогти вам вирішити або пом’якшити проблему вищезазначені проблеми, і чому?

Для користувачів Scrum, які мають більше досвіду Scrum, вони вважають, що запропонована мова може певною мірою пом’якшити проблеми, але ефект все ще обмежений, оскільки запропонована графічна мова в основному зосереджена на

управлінні проектом Scrum. Але існуючі проблеми більше пов'язані з технічними рішеннями.

Для користувачів Scrum, які мають лише 1-2 роки досвіду Scrum, вони вважають, що запропонована графічна мова безперечно може допомогти їм зрозуміти Scrum і застосувати концепції Scrum у конкретному проекті Scrum.

Питання 10: Чи будете ви використовувати цю графічну мову в майбутній роботі? чому

Усі учасники тримають позитивну відповідь на це питання. Основна причина полягає у високій ефективності запропонованої мови. Для користувачів Scrum, які мають 3-річний досвід, вони вважають, що добре мати таку мову, особливо коли вони приєднуються до нового проекту або на початку вивчення методу Scrum. З іншого боку, вони також вказали на вартість вивчення та підтримки цієї мови. Вони стверджували, що ретельно збалансують вартість і ефективність запропонованої мови, якщо її впровадять у їхній проект у майбутньому.

3.3. Основні елементи методу Scrum

RQ1: Які основні поняття, елементи та їхні властивості присутні в описі процесу Scrum?

Для того, щоб отримати концепції Scrum і елементи побудови, ми провели дослідження літератури на початку цього дослідження. У дослідженні літератури ми головним чином зосередилися на зборі знань про моделювання проектів Scrum та навчання Scrum. Результат дослідження показує, що в Scrum існує три типи концепцій: ролі, події та артефакти. Зокрема, у Scrum є три ролі: Scrum Master, Product Owner і Development Team. Хоча розробник не є основною роллю в Scrum, він все ще є важливою роллю в концепції ролі Scrum, оскільки він становить команду розробників. Тому ми додали розробника як роль у запропонованому DSL.

Scrum-події: спринт, нарада з планування спринту, щоденна нарада Scrum, нарада з огляду спринту та нарада з ретроспективи спринту. Спринт складається з різних зустрічей Scrum та інших заходів розвитку. Зустріч з планування Scrum

проводиться на початку спринту і триває приблизно 6-8 годин протягом 4-тижневого спринту. Щоденна зустріч Scrum повинна проводитися на початку кожного дня спринту, тривалість становить близько 30 хвилин. Наприкінці спринту слід провести дві зустрічі: огляд спринту та ретроспективу спринту. Зустріч Sprint Review використовується для перегляду та демонстрації вимог, розроблених під час останнього спринту. А ретроспективна зустріч спринту використовується для перевірки команди розробки та вирішення технічних і організаційних проблем. Останній тип концепції — артефакти Scrum, Backlog продукту, Backlog Sprint і Increment. Два типи Backlog — це набори вимог до проекту, які мають бути виконані. Лише власник продукту має повноваження керувати резервами. Інкремент — це набір «доу» вимог. Під час наради з огляду спринту слід продемонструвати приріст.

3.4. Визначення пропонованої доменно-специфікованої мови

ЗП2: Яке визначення запропонованої графічної мови?

Після вилучення елементів Scrum ми виявили, що недостатньо описувати проект Scrum лише за допомогою вищезазначених елементів. Тому, щоб полегшити навчання Scrum і моделювання Scrum, ми вводимо кілька додаткових елементів: елемент Backlog продукту, елемент Backlog Sprint і завдання. Елемент Backlog продукту та елемент Sprint Backlog є одиницями для визначення вимог. Завдання — це розподілений елемент Sprint Backlog, призначений певному розробнику. Крім того, ми також представили два типи шкали часу для побудови структури запропонованого графічного DSL.

Щоб формально спроектувати та визначити запропоновану мову, ми дотримувалися підходу та принципу проектування DSL, про які повідомлялося. Оскільки Scrum є методом розробки, а не виконуваною програмою, доцільно розробити невиконуваний ¬DSL для опису процесу Scrum. Тому ми розробили запропонований DSL у три етапи: прийняття рішення, аналіз та фази проектування. У цьому дослідженні фази розробки та розгортання були вилучені з процесу

розробки DSL. Фаза прийняття рішення проводиться для встановлення сфери DSL. Потім на фазі аналізу накопичуються всі предметні знання та уточнюються предметні проблеми. А на етапі проектування пояснюються формальні визначення DSL .

Пропонований графічний DSL містить три частини. Перша частина — Термінологія», у цій частині показано всі елементи Scrum і мовні позначки.

Друга частина — «Модель Srum», ця частина використовується для простої демонстрації основних концепцій і принципів Scrum. Остання частина — проект Srum»; користувачі мови можуть створювати конкретні проекти Scrum, використовуючи запропоновану мову.

Щоб спроектувати та формально визначити DSL, ми повторно використали кілька концепцій з BPMN. І ми використали UML для визначення синтаксису мови та зв'язків між елементами . Цей синтаксис відображає принципи Scrum і зв'язки між ролями, подіями та артефактами Scrum. Згідно з цим визначенням синтаксису користувачі мови можуть створити проект Scrum, описаний у запропонованому графічному DSL.

Для семантичної специфікації ми використали таблицю для переліку всіх можливих комбінацій елементів і варіантів елементів, а потім пояснили значення для кожної конкретної комбінації та варіантів .

ЗП3: Як запропонована мова моделювання Scrum сприяє навчанню Scrum в академічному середовищі?

Щоб оцінити корисність запропонованої мови, ми провели контрольований експеримент в освітньому середовищі, щоб перевірити, чи може запропонована мова підвищити ефективність і ефективність навчання методу Scrum.

Результати експерименту вказують на те, що запропонована мова може допомогти новим учням Scrum точніше отримати знання про метод Scrum (RQ3.1) і отримати важливу та детальну інформацію про конкретний проект Scrum (RQ3.2). Застосовуючи тести статистичної значущості для кожного набору даних, зібраного від суб'єктів, ми отримали позитивний результат, який вказує на те, що нульові гіпотези слід відхилити, а запропонована мова значно покращує розуміння методу

Scrum та його відповідних проектів. У порівнянні з текстовою мовою запропонована мова підвищила ефективність навчання суб'єкта та ефективність розуміння методу Scrum на 41% (NRESP) та 30% (TMEC/NRESP). Проте був лише один негативний результат, який показав, що суб'єктам потрібно витрачати більше часу на розуміння методу Scrum запропонованою мовою (RQ3.1). Але коли вони знайомі із запропонованими нотаціями, ефективність навчання може помітно підвищитися. Ефективність навчання за проектом Scrum та результативність суб'єктів була покращена на 72% (NRESP), 30% (TMEC) і 66% (TMEC/NRESP) після розуміння нотацій (RQ3.2). Таким чином, можна побачити, що додатковий час, витрачений на розуміння методу Scrum, був спричинений вартістю вивчення нової запропонованої мови (RQ3.1).

3.5. Результати експерименту

ЗП4: Як запропонована мова моделювання Scrum допомагає користувачам Scrum зрозуміти та застосувати метод Scrum у галузі?

Щоб дослідити потенційне використання запропонованої мови в промисловій сфері, ми провели опитування з досвідченими розробниками, які працюють у шведських і китайських компаніях. Фактично, через обмеженість ресурсу метод опитування був проведений недостатньо в цьому дослідженні. Однак це допомогло нам визначити переваги та недоліки запропонованої мови з точки зору промислового користувача Scrum.

Ми отримали 21 анкету, а значить, є 17 дійсних анкет. Усі учасники вважають, що запропонований графічний DSL корисний для покращення розуміння методу Scrum. Крім того, вони вважають, що запропонований DSL забезпечує простіший спосіб отримати огляд конкретного проекту Scrum і робить дані проекту більш детальними читабельний і зрозумілий . Звичайно, ці переваги можуть значно підвищити ефективність вивчення концепцій Scrum і конкретних проектів Scrum.

Крім того, учасники зазначили кілька недоліків. По-перше, розширюваність не така хороша, як мова тексту. Наприклад, якщо команда Scrum хоче змінити

традиційний Scrum і додати новий тип події. Мовою тексту дуже просто додати абзац і описати цю подію. Але в графічній мові потрібно розробити нові визначення елементів і легенди. У гірших випадках структуру цієї мови може потребувати рефакторингу, щоб відповідати новому елементу. По-друге, упущено деякі визначення понять. Оскільки формально визначити концепції Scrum без текстового опису непросто, тому визначення концепцій Scrum за допомогою поєднання запропонованої DSL і текстової мови може допомогти користувачам мови краще зрозуміти концепції Scrum.

Примітно, що більший інтерес до запропонованої мови виявляють розробники з меншим досвідом роботи. Це означає, що наразі запропонований DSL більше підходить для навчання Scrum.

У цьому дослідженні ми розробили та оцінили графічний DSL для моделювання процесу Scrum. Однак, через обмеження часу та ресурсів, цей дипломний проект був лише пошуковим дослідженням, а не лонгітюдним дослідженням. Це лише початкова робота щодо оцінки запропонованої мови та перевірки результатів дослідження. Більш інтенсивну і конкретну роботу слід проводити в подальших дослідженнях.

- Перевірка запропонованої мови

Визначення запропонованої мови створено авторами на основі вивчення та аналізу літератури. Існує потреба перевірити точність визначення мови, щоб переконатися, що вона розумна та правильна для моделювання проектів Scrum. Тому було б необхідно представити цю мову експертам Scrum і експертам DSL, щоб переглянути цю мову, щоб покращити її здатність точно описувати різноманітні проекти Scrum.

- Перевірка результатів дослідження в промисловому середовищі

У цьому дослідженні експеримент проводився в освітньому середовищі зі студентами, які не мають досвіду Scrum. Немає сумніву, що студенти як предмети мають відмінності від промислових розробників, наприклад, досвід розробки. Перевірка результатів дослідження в галузевому експерименті необхідна для перевірки достовірності висновків з досвідченими розробниками. Було б цікаво

провести експеримент із реальною командою Scrum у промисловому середовищі, щоб перевірити корисність запропонованої мови.

Через обмеженість ресурсів опитування в цьому дослідженні не проводилося достатньо. Час виконання та кількість учасників були обмежені; отже, результат може не відповідати повним промисловим користувачам. Необхідно провести подальше дослідження запропонованої мови в промисловому середовищі.

Висновки до розділу

В даному розділі ми представили новий DSL моделювання Scrum, щоб допомогти користувачам Scrum зрозуміти метод Scrum і застосувати його. Однак цю мову можна використовувати не тільки для навчання Scrum, але й для управління проектами. З результатів експерименту видно, що ця мова значно покращила розуміння суб’єктами проекту. Піддослідним легко знайти детальну інформацію про проект запропонованою мовою. Тому було б цікаво дослідити застосування DSL моделювання Scrum до проектів, щоб допомогти розробникам керувати процесом розробки.

ВИСНОВКИ

Результати дослідження свідчать про покращення ефективності SCRUM-проектів при використанні розроблених моделей та засобів доменних специфікацій. Команди, що використовували нові рішення, показали кращі показники управління та розробки програмного забезпечення. Команди виразили позитивне сприйняття розроблених моделей. Їхні відгуки свідчать про те, що нові інструменти полегшують роботу, покращують взаємодію в команді та сприяють більш ефективному розвитку проектів.

Впровадження розроблених моделей позитивно вплинуло на результативність управління проектами. Спостерігалася зменшення труднощів у визначенні та розробці доменних специфікацій, що вплинуло на планування та виконання завдань.

Розроблені моделі та засоби виявилися легкими у впровадженні, адаптації та використанні в SCRUM-проектах. Їх можливість інтеграції у існуючі робочі процеси забезпечує швидке впровадження без значних перешкод.

Результати дослідження підтверджують практичність та використання розроблених моделей та засобів доменних специфікацій у реальних умовах розробки. Вони стають цінними інструментами для команд, що дотримуються SCRUM-підходу.

Розробка та впровадження моделей та засобів доменних специфікацій, специфічних для методології SCRUM, виявилися успішними. Результати дослідження свідчать про їхню ефективність у вдосконаленні процесів розробки програмного забезпечення та управління SCRUM-проектами. Вищезазначені позитивні аспекти дозволяють рекомендувати розроблені рішення для практичного використання в реальних умовах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лю, Шаоїн, «Підхід до застосування SOFL для гнучкого процесу та його застосування в розробка інструменту підтримки тестування, «Інновації в системах і розробці програмного забезпечення 6.1-2 (2010): 137-143.

2. Van Waardenburg G, Van Vliet H, «When agile meets the enterprise», [J]. Інформація та програмні технології, 2013, 55(12): 2154-2171.

3. Löffler R, Güldali B, Geisen S, «Towards Model-based Acceptance Testing for Scrum», [J]. Softwaretechnik-Trends, GI, 2010.

4. Esfahani, Hesam Chiniforooshan, Jordi Cabot, and S. K. Eric, ―Adopting Agile Methods: Can Допомога в цілеспрямованому соціальному моделюванні?» RCIS, 2010.

5. Перкушіч, Мірко, Гігго Олівейра де Алмейда та Анджело Перкушіч, «Модель для виявлення» проблеми проектів розробки програмного забезпечення на основі scrum», Proceedings of 28th Annual ACM Симпозіум з прикладних обчислень, ACM, 2013.

6. Уртадо Алегріа, Хуліо А., Марія Сесілія Бастарріка та Александр Бергель, «Аналіз програмного забезпечення» моделі процесів з AVISPA», Матеріали Міжнародної конференції з програмного забезпечення 2011 р.і системний процес, ACM, 2011.

7. Wei, Quan та ін. , ―Дослідження поєднання Scrum та UML у процесі розробки програмного забезпечення Моделювання», приладобудування та вимірювання, комп'ютер, зв'язок та управління (IMCCC), Четверта міжнародна конференція 2014 року, IEEE, 2014.

8. Швабер К. «Процес розробки Scrum», [M]//Проектування та впровадження бізнес-об'єктів, Springer London, 1997, стор. 117-134.

9. Хенрік Кніберг, «Scrum and XP from the Trenches», C4Media, видавець InfoQ.com, 5.10.2007 р.

10. Карвальо, С.К., Мотта Кардозо, Ф.Р. та інші, «Порівняльне дослідження між SCRUM і RUP». Використання вбудованого програмного забезпечення в режимі

реального часу, інформаційні технології: нові покоління (ITNG), 2013 Десята міжнародна конференція, 2013.4, стор. 734-735.

11. Noordeloos R., Manteli, C. та інші, ―From RUP to Scrum in Global Software Development: A Case Дослідження, Global Software Engineering (ICGSE), 2012 IEEE Seventh International Conference,2012.8, стор. 31-40.

12. Лехтінен, Тімо О. А. та ін., «Чому результат розробки не відповідає продукту» Owners‘ Expectations?, Agile Processes, in Engineering Software, and Extreme Programming. Springer International Publishing, 2015, стор. 93-104.

13. Moe, N.B., Dingsøyr, T., Dybå, T., «Модель командної роботи для розуміння гнучкої команди: кейс» дослідження проекту Scrum, Інформаційні та програмні технології 52, стор.480–491 (2010)

14.. Соліс-Мартінес, Хайме та ін. , ―BPMN MUSIM: Підхід до вдосконалення експертів у домені ефективність моделювання бізнес-процесів для генерації конкретного програмного забезпечення додатків», Експертні системи з додатками 41.4 (2014), стор. 1864-1874.

15. Швабер К., Сазерленд Дж. Посібник зі scrum [J], Scrum Alliance, 2011

16. Посібник до ЗВОРУ ЗНАНЬ SCRUM (посібник SBOK™), видання 2016 р., VMEdu,Inc., 2015, стор.6.

17. Budinsky, F., ―Eclipse Modeling Framework: Посібник розробника‖, AddisonWesley, Reading, M.A., 2003

18. Келлі, Стівен, «Опис GOPRR», докторська дисертація, Додаток 1 (1997).

19. Алегрія, Хуліо Аріель Уртадо, Марія Сесілія Бастарріка та Александр Бергель, «Аналізуючи Модель процесу Scrum з AVISPA, Чилійське товариство комп’ютерних наук (SCCC), 2010 XXIX Міжнародна конференція IEEE, 2010.

20. Андрес, Франсіско Перес, Хуан Де Лара та Естер Герра, «Доменно-спеціальні мови з графічні та текстові види», Міжнародний симпозіум із застосування графів Transformations with Industrial Relevance, Springer Berlin Heidelberg, 2007.

21. Мора, Беатріс та ін. , «Smml: мова моделювання вимірювання програмного забезпечення», Proceedings of the 8-й семінар з предметно-орієнтованого моделювання (DSM‘2008), 2008.

22. Лу, Руопен і Шазія Садік, «Огляд порівняльного моделювання бізнес-процесів» підходи», Міжнародна конференція з бізнес-інформаційних систем, Springer BerlinГейдельберг, 2007.67

23. Е. Дановаро, А. Джейнс і Г. Суччі, «Дзідока в розробці програмного забезпечення», Companion to 23rd ACM SIGPLAN Conf. Об'єктно-орієнтована програма. сист. ланг. апл. – OOPSLA Companion ‘08, стор. 827, 2008 рік.

24. Марченко А., Абрахамссон П., «Scrum у багатопроектному середовищі: етнографічно натхненний приклад із проблем усиновлення», [C]//Agile, 2008. AGILE'08, Конференція, IEEE, 2008, стор. 15-26.

25. Робер, Сільвен, Себастьян Жерар, Франсуа Тер’єр і Франсуа Лагард, «Легкий підхід до проектування предметно-орієнтованих мов моделювання», на 35-й конференції Euromicro у 2009 р.з програмної інженерії та передових застосувань, стор. 155-161. IEEE, 2009.

26 Куалья, Едуардо Дж. і Клаудія А. Токантінс, «Управління проектами моделювання за допомогою Scrum»,Симуляційна конференція (WSC), матеріали зимової конференції 2011 року. IEEE, 2011.

27. Полі, Даніель, Бйорн Міхалік і Дірк Бастен, «Чи мають відбуватися щоденні Scrums День? Ситуація індивідуальних принципів Scrum в компанії електронної комерції, система наук (HICSS), 2015 р. 48-а Міжнародна конференція на Гаваях. IEEE, 2015.

28. Kniberg, Henrik, «Scrum and XP from the Trenches», Lulu. com, 2015.

29. Хантке, Детлев, «Підхід до поєднання SPICE і SCRUM у розробці програмного забезпечення».Проекти», Покращення процесів програмного забезпечення та визначення можливостей, Springer International Publishing, 2015, pp:233-238.

30. Ван Ваарденбург, Гуус і Ганс Ван Вліт, «Коли гнучкість зустрічається з підприємством», Інформаціята програмні технології 55.12 (2013), pp: 2154-2171.

31. Фернандес, Жоао М. та Соня М. Соуза, «Playscrum — карткова гра, щоб навчитися спритності в сутичках». метод, Ігри та віртуальні світи для серйозних програм (VS-GAMES), 2010 Другий Міжнародна конференція з питань. IEEE, 2010.

32. Паасіваара, Марія та ін., «Навчання студентів бійці за допомогою блоків LEGO», Companion Матеріали 36-ї Міжнародної конференції з розробки програмного забезпечення. ACM, 2014.

33 Елоранта, Велі-Пекка та ін. ―Scrum Anti-Patterns--An Empirical Study‖ Розробка програмного забезпечення Конференція (APSEC), 2013 20-а Азійсько-Тихоокеанська. том. 1. IEEE, 2013.

34. Скотт, Езекіель та ін. ―Чи є стилі навчання корисними індикаторами, щоб дізнатися, як учні використовують Scrum вперше?, «Комп'ютери в людській поведінці 36 (2014), с.: 56-64.

35. Фон Вангенхайм, Крістіане Греccе, Рафаель Саві та Адріано Ферреті Боргатто, ―SCRUMIA—Навчальна гра для навчання SCRUM на комп'ютерних курсах, Journal of Systems and Програмне забезпечення 86.10 (2013), стор.: 2675-2687.

36. Алегрот, Еміль та ін. ―Навчання Scrum – що ми зробили, що будемо робити та що заважає Us, Proc. XP 2015 212 (2015), стор. 361.

37. Мернік, Мар'ян, Ян Герінг і Ентоні М. Слоан, «Коли і як розробляти предметно-спеціальні мови», ACM computing surveys (CSUR) 37.4 (2005), стор.: 316-344.

38 Karsai G, Krahn H, Pinkernell C та ін. , «Інструкції з проектування для предметно-спеціальних мов», [J]. Препринт arXiv arXiv:1409.2378, 2014.

39. Ван Деурсен, Арі, Пол Клінт і ЙостВіссер, «Доменно-специфічні мови: анотація» Бібліографія, SigplanNotices 35, №. 6 (2000), стор. 26-36.

40. Wohlin, Claes та ін. , «Експериментування в програмній інженерії», Springer Science & Business Медіа, 2012.

41. http://mplaza.pm/downloads/Scrum%20Training%20Manual.pdf

42. Лінакер, Йохан та інші, «Рекомендації щодо проведення опитувань у розробці програмного забезпечення» (2015).

43. Косар, Томаш, Судев Бохра та Мар’ян Мернік, «Доменно-спеціальні мови: систематичний mapping study, Інформаційні та програмні технології 71 (2016), стор. 77-91.

44. Дам’янов, Іво та Міла Сукалінська, «Доменно-специфічні мови на практиці», міжнар. Journal of Computer Applications 115.2 (2015).

45. WILE, D.S.2001, «Підтримка спектру DSL», J.Comput.Inform.Techn.9,4, pp:263–287.