**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ МЕДИЦИНИ ТА БІОТЕХНОЛОГІЙ ІМЕНІ С. З. ҐЖИЦЬКОГО**

Факультет менеджменту, бізнесу та публічного адміністрування

Кафедра менеджменту ІТ-сфери

Допускається до захисту
“_____”_____2025 р.
В.о. зав. кафедри _____
(підпис)
к.е.н., доцент Олена КІНДРАТ
наук. ступ., вчене звання ім’я та прізвище

# КВАЛІФІКАЦІЙНА РОБОТА

студента **Хомишина Дмитра Андрійовича**

на тему:

## *«Управління ІТ-проєктами з використанням Agile та Scrum: переваги, ризики, практика»*

на присвоєння кваліфікації – *Магістр з менеджменту ІТ-сфери*

Керівник роботи _____ к.е.н., доцент **Кіндрат О.В.**
(підпис) (вчене звання, прізвище та ініціали)

Львів – 2025

Львівський національний університет ветеринарної медицини та біотехнологій імені С.З. Гжицького

| | |
|---|---|
| Факультет | Менеджменту, бізнесу та публічного адміністрування |
| Кафедра | Менеджменту ІТ-сфери |
| Галузь знань | 07 «Управління та адміністрування» |
| Спеціальність | 073 «Менеджмент» |
| Освітньо-професійна програма | «Менеджмент ІТ-сфери» |

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

_____

“____”_____20___року

# З А В Д А Н Н Я
# НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ
## **Хомишину Дмитру Андрійовичу**

Тема роботи

## **«Управління ІТ-проєктами з використанням Agile та Scrum: переваги, ризики, практика»**

керівник роботи Кіндрат Олена Василівна, к.е.н., доцент _____
( прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «___»_____2025 року №___

2. Строк подання здобувачем роботи_____

3. Вихідні дані до роботи: навчально-методична, наукова та довідкова література, що відповідає темі кваліфікаційної роботи, інформаційні звтіи компанії ТОВ “*****”, Накази та внутрішні регламенти ТОВ «*****» (непублічні документи).

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

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ ЗА МЕТОДОЛОГІЯМИ AGILE ТА SCRUM. 1.1. Сутність та еволюція управління ІТпроєктами в умовах цифрової трансформації 1.2. Філософія та принципи гнучкого управління (Agile): маніфест та цінності 1.3. Методологія Scrum: ключові ролі,артефакти та події (практика) 1.4. Порівняльний аналіз переваг та ризиків використання Scrum і традиційних методологій (Waterfall) РОЗДІЛ 2.

АНАЛІЗ ПРАКТИКИ ВПРОВАДЖЕННЯ AGILE ТА SCRUM В ІТ-КОМПАНІЇ. 2.1. Організаційно-економічна характеристика діяльності ТОВ «*****» 22.2. Аналіз методологічної бази управління ІТ-проєктами в компанії. 2.3. Оцінка переваг та ідентифікація проблемних зон при застосуванні гнучких методологій у діяльності компанії. 2.4. Практика реалізації ІТ-проєктів: аналіз успішних кейсів та отриманих результатів. РОЗДІЛ 3. ПРОЄКТНІ ПРОПОЗИЦІЇ З УДОСКОНАЛЕННЯ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ НА ОСНОВІ ПРИНЦИПІВ AGILE ТА SCRUM. 3.1. Мінімізація типових ризиків ІТ-проєктів в Scrum-середовищі. 3.2. Рекомендації щодо оптимізації процесів Scrum (спринти, ретроспективи, залучення стейкхолдерів).
3.3. Розробка моделі показників ефективності (KPI) ІТ-проєктів в Agile-середовищі.3.4. Очікувані результати від впровадження запропонованих заходів та їх економічне обґрунтування.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) 1.еволюція підходів до управління ІТ-проєктами в умовах цифрової трансформації 2. цінності та принципи гнучкого управління (Agile-маніфест) 3. структура фреймворку Scrum (ролі, артефакти та події) 4. порівняльна характеристика методологій Scrum і Waterfall 5. порівняння Scrum, Kanban та Scrumban як моделей управління ІТ-проєктами 6. організаційна структура та модель управління ІТ-проєктами в ТОВ «*****» 7. SWOT-аналіз застосування Agile та Scrum у діяльності компанії 8. система ключових показників ефективності (KPI) ІТ-проєктів в Agile-середовищі.

6. Дата видачі завдання_____

**КАЛЕНДАРНИЙ ПЛАН**

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Отримання завдання. Вивчення рекомендованої літератури. Вивчення об'єкту дослідження. | травень | виконано |
| 2 | Розділ 1. Теоретичні основи управління it-проєктами за методологіями agile та scrum | липень | виконано |
| 3 | Розділ 2. Аналіз практики впровадження agile та scrum в it-компанії | серпень | виконано |
| 4 | Розвіл 3. Проєктні пропозиції з удосконалення управління it-проєктами на основі принципів agile та scrum | Серпень-вересень | виконано |
| 5 | Кінцеве оформлення кваліфікаційної роботи. Підготовка до захисту. Пробний захист на випусковій кафедрі. | Жовтень-листопад | виконано |

Здобувач _____ _____Хомишин Д.А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Кіндрат О.В.
(підпис) (прізвище та ініціали)

# АНОТАЦІЯ

**Хомишин Д. А. «Управління ІТ-проєктами з використанням Agile та Scrum: переваги, ризики, практика»**: Кваліфікаційна робота: (073 «Менеджмент») / ЛНУВМБ імені С. З. Ґжицького. Кафедра менеджменту ІТ-сфери. Наук. кер.: О.В. Кіндрат, к.е.н., доцент. Львів, 2025. 64 с.

У першому розділі роботи розглянуто теоретичні засади гнучких методологій управління проєктами (Agile та Scrum), їхню філософію, принципи та основні цінності. Проаналізовано ключові ролі, артефакти та події в рамках фреймворку Scrum. Визначено основні переваги, які надають ці підходи (гнучкість, швидка реакція на зміни, орієнтація на клієнта), а також потенційні ризики їхнього впровадження.

У другому розділі здійснено аналіз практичного застосування Agile та Scrum на прикладі ІТ-компані. Проведено порівняльний аналіз успішних і неуспішних проєктів, керованих за цими методологіями. Досліджено використання метрик для оцінки прогресу (Velocity, Burndown Charts) та методи виявлення і мінімізації ризиків (наприклад, технічного боргу та зміни обсягу робіт).

У третьому розділі запропоновано практичні рекомендації та інструменти для підвищення ефективності управління ІТ-проєктами за допомогою Agile та Scrum. Розроблено модель адаптації Scrum для великих розподілених команд. Запропоновано механізми управління ризиками. Надано поради щодо покращення Daily Scrum та Retrospective з метою постійного вдосконалення процесів.

Робота складається з трьох розділів, висновків і списку використаних джерел. Робота містить 3 рисунків, 15 таблиць, і викладена на 64 сторінках.

**Ключові слова:** Agile, Scrum, управління ІТ-проєктами, ризики, Velocity, Burndown Chart, командна взаємодія, ітеративна розробка.

© Д.А. Хомишин, 2025

© ЛНУВМБ імені С.З. Ґжицького, 2025

# ANNOTATION

**Khomyshyn D. A. "Management of IT Projects Using Agile and Scrum: Advantages, Risks, and Practice**": Qualification Paper: (073 "Management") / Stepan Gzhytskyi National University of Veterinary Medicine and Biotechnologies D epartment of IT Management. Scientific supervisor: O. V. Kindrat, Ph.D. in Economics, Associate Professor. Lviv, 2025. 64 p.

The first chapter of the work examines the theoretical foundations of flexible project management methodologies (Agile and Scrum), their philosophy, principles, and core values. Key roles, artifacts, and events within the Scrum framework are analyzed. The main advantages offered by these approaches (flexibility, rapid response to change, client orientation), as well as the potential risks of their implementation.

The second chapter analyzes the practical application of Agile and Scrum using the example of an IT company. A comparative analysis of successful and unsuccessful projects managed by these methodologies is conducted. The use of metrics for progress evaluation (Velocity, Burndown Charts) and methods for identifying and minimizing risks (e.g., technical debt and scope creep) are investigated.

The third chapter proposes practical recommendations and tools for improving the efficiency of IT project management using Agile and Scrum. A model for adapting Scrum for large, distributed teams is developed. Mechanisms for risk management are proposed. Advice is provided on improving the Daily Scrum and Retrospective meetings for continuous process refinement.

The work consists of three chapters, conclusions, and a list of references. The work contains 3 figures, 15 tables, and is presented on 64 pages.

**Keywords:** Agile, Scrum, IT project management, risks, Velocity, Burndown Chart, team interaction, iterative development.

© D. A. Khomyshyn, 2025

© Stepan Gzhytskyi National University of Veterinary Medicine and Biotechnologies, 2025

**ЗМІСТ**

# ВСТУП

Сучасна епоха інформаційних технологій ставить перед управлінням проєктами нові виклики: тут вже недостатньо просто дотримуватись послідовності виконання завдань. Потрібна здатність швидко, майже миттєво, адаптуватись до змінюваних вимог ринку та очікувань клієнтів. У таких умовах традиційні, інертні моделі управління проєктами, як Waterfall, все частіше поступаються місцем Agile-методологіям. Серед них фреймворк Scrum став найбільш популярним і визнаний світовим стандартом. Його популярність пояснюється акцентом на інкрементальній доставці, швидкому зворотному зв'язку та самоорганізації команд, що є критично важливим для підтримки конкурентоспроможності та досягнення переваг у висококонкурентному середовищі. Здатність швидко постачати робочий функціонал і зменшувати "вартість затримки" (Cost of Delay) є прямою економічною вигодою, яку має забезпечувати ефективний Scrum.

Актуальність цієї роботи заключається у критичній необхідності вийти за рамки поверхневого "імітування Agile" (або "Scrum-but"), це коли команди лише дотримуються зовнішніх властивостей скраму (спринти, щоденні зустрічі, використання Story Points), але не генерують відчутної, вимірюваної бізнес-ефективності. Це імітування скраму призводить до "театралізованого Agile", де процеси витрачають час, але не забезпечують кращого результату. Незважаючи на широке застосування скраму та Agile, багато IT-проєктів стикаються з внутрішніми проблемами, які підривають ефективність роботи, та демонструють неефективність ключових властивостей скраму(дейлі, ретроспектива і тд), де наприклад: ретроспективи перетворюються на формальне обговорення скарг команди, замість того щоб стати механізмом який буде визначати основні причини проблем та генерувати чіткі, відповідальні та вимірювані в SMART Action items, які стануть невід'ємною частиною наступного спринта. Відсутність уніфікованої системи ключових показників ефективності (KPI), що ускладнює об'єктивну оцінку справжнього "життя" проєкту. Коли майже все вимірюється швидкістю виконання завдань,

менеджмент отримує спотворену картину, ігноруючи критично важливі показники, як-от час, витрачений на переробку завдань, або стабільність виконання завдань (Velocity consistency). І, як наслідок, команди виявляються нездатними системно керувати якістю свого коду, що призводить до зростання технічного боргу та збільшення часу, необхідного для відновлення після збоїв (MTTR). Саме ці прогалини, що виникають між теорією Scrum та його практичною реалізацією, вимагають глибокого аналізу та розробки конкретних, керованих даними, інструментів для їх усунення.

Мета цієї роботи - розробити та обґрунтувати комплексний, практично орієнтований підхід до управління IT-проєктами за допомогою Scrum, зосереджений на підвищенні прозорості, системній мінімізації ризиків та підвищенні якості, що вимірюється не внутрішніми зусиллями, а зовнішніми бізнес-результатами. Досягнення цієї мети передбачає виконання низки завдань, що охоплюють як теоретичний аналіз переваг та ризиків Scrum, так і розробку конкретних інструментів, спрямованих на корекцію його типових недоліків. Сюди входить визначення та класифікація критичних KPI - як операційних, що показують ефективність команди (Velocity, Lead time, Rework), так і стратегічних, що відображають бізнес-успіх (Value realization, MTTR) - для створення надійного кількісного фундаменту управління, який дозволить перейти від інтуїтивного менеджменту до менеджменту, заснованого на фактах. Потім буде розроблено архітектуру дворівневого Дашборду "Health of project", який забезпечить адаптовану прозорість, що є критично важливим для взаємодії з клієнтом: деталізовану для внутрішнього використання(командою та Скрам мастером) та консолідовану для стейкхолдерів, мінімізуючи розбіжність очікувань щодо прогресу та цінності. Особлива увага буде приділена перетворенню ретроспектив на дієвий, керований даними механізм постійного вдосконалення, який використовує аналіз коливань KPI для виявлення кореневих причин проблем, а не лише їхніх симптомів. Додатково будуть сформульовані практичні рекомендації по управлінню технічним боргом - шляхом фіксованої інвестиції у беклог

(рекомендовано 10-15% часу) для забезпечення довгострокової стійкості кодової бази - та впровадження кількісних Exit criteria як надійного "ментального бар’єра" проти поспішного випуску недосконалого продукту. Ці критерії мають чітко визначати мінімально прийнятний рівень якості, перш ніж функціонал може бути представлений клієнту. Об'єктом дослідження є сам процес управління проєктами за Scrum у його реальних, складних проявах, тоді як предметом - методи та інструменти підвищення його ефективності та якості, які трансформують інтуїтивні рішення в рішення, керовані даними. Методологічною основою є системний аналіз, моделювання та синтез, що дозволяють інтегрувати метрики, управління якістю та коучингові практики в єдину, дієву систему, практична значущість якої полягає у можливості прямого застосування для трансформації практики Scrum із рівня виконання процесів на рівень досягнення вимірюваних бізнес-результатів та забезпечення стійкого розвитку аутсорсингового бізнесу. Впровадження цієї моделі не лише підвищить внутрішню ефективність, але й позитивно вплине на задоволеність клієнтів та моральний дух команди, знижуючи плинність кадрів і, як наслідок, загальні операційні витрати компанії.

Це рішення виростає з реальної потреби - IT-сфера шукає такий спосіб керувати процесами, щоб можна було працювати в масштабах, не втрачаючи моторики Scrum. Замість того, щоб просто повторювати день за днем основні властивості скраму, команда отримує інструмент для сталого випуску класних цифрових продуктів. Отже, матеріал стає опорою для стрибка від самої лише активності до реальних підсумків. А це якраз те, що потрібно організаціям, які хочуть перемагати серед жорстокої конкуренції в сучасному онлайн-середовищі.

# РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ УПРАВЛІННЯ IT-ПРОЄКТАМИ ЗА МЕТОДОЛОГІЯМИ AGILE ТА SCRUM

## 1.1. Сутність та еволюція управління IT-проєктами в умовах цифрової трансформації

Керувати IT-проєктами - це складна справа, де важливо точно підбирати методи, інструменти й уміння задля конкретного кінцевого результату. Головне - добитися успіху всередині жорстких меж: обсягу роботи, строків, коштів і стандартів якості, хоч би що трапилося - технічні проблеми чи раптові зміни в технологіях.

Зазвичай інженерні справи передбачають чіткий план, тоді як цифрові речі живуть за своїми правилами. Програми й онлайн-сервіси - не те, що можна потримати в руках, бо це повітряні конструкції з коду. Через це замовник не одразу чітко сформулює, що хоче отримати, тож спочатку все пливне в тумані. З одного боку, IT світ міняється так швидко, ніби кожен день новий закон приймають. Технології розвиваються, користувачі вимагають іншого - і команда має працювати щолихвилини щоб надати кращий результат. Готовність підлаштовуватись тепер не просто бонус, а запорука того, щоб продукт не випав з гри. Усе залежить не стільки від інструментів, а радше від людей - їх навичок, уміння розв’язувати завдання, головне - добре говорити одне з одним і працювати разом. Менеджер має керувати процесами, але також надихати, допомагати команді взаємодіяти й заохочувати їх рухатися далі.

Розвиток управління IT-проєктами йшов поряд з швидким ростом технологій, а ще - через зміни в тому, чого хочуть компанії. Через це світ переходить від жорстких схем до гнучких, крокових підходів. З 70-х до 90-х переважали старі методи, найпопулярніший із них - Водоспадна модель (Waterfall). У ньому процес ділився на частини: збирання вимог, планування, дизайн, код, тести й запуск. Кожну вимогу потрібно було закрити до кінця, перш ніж братися за наступну. Такий спосіб давав ясність, повні записи всього що виконано й легший контроль за ходом розробки - особливо коли задачі не

міняються й усе давно відомо. Коли почали активно розвиватися технології, стало ясно - метод занадто дорогий і громіздкий. Зміни вносити важко, особливо ближче до фінішу проєкту. До того ж, зворотний зв’язок дуже повільний. замовник бачив результат тільки в самому кінці. найчастіше - коли поточні потреби ринку уже сильно змінилися.[12, с. 99-100]

На початку 2000-х неспроможність Waterfall нормально працювати з мінливими вимогами й високою невизначеністю запустила створення гнучких методів. Справжньою причиною стали ділові потреби - потрібно було швидше давати користь клієнтам і скорочувати терміни виходу продукту на ринок. Усе це призвело до появи Agile-маніфесту у 2001-му[1], де чітко окреслили нові правила: важливіше люди й їхня комунікація, а не формальні процедури; живий результат ліпший за купу паперів; тісний контакт з замовником і відкритість змінам мають бути нормою.[13 с. 71-72] Відтоді керування проєктами перестало обмежуватися просто слідуванням плану - тепер це безперервна адаптація, контроль змін і можливих загроз на всіх етапах.

У час цифрових змін - коли нові технології проникають у бізнес - керування ІТ-проєктами стає справою номер один, а не просто черговою задачею. Такі компанії, як ***** роблять ставку на швидке запускання продуктів маленькими кроками, що дає сильну перевагу перед конкурентами. Головне вже не просто закрити етапи(як у Waterfall), а приносити користь без затримок і зайвих дій. Увагу тепер переносять не тільки на код і системи, але й на людей: потрібно слухати замовника, тримати контакт весь час, водночас втягувати його в сам процес. Дуже важлива культура яка дозволяє пробувати нове й швидко вчитися. Ті, хто працює добре, не приховують помилки - навпаки, обговорюють їх голосно, щоб стати кращими. Коли все міняється за лічені секунди, методи на кшталт Agile чи Scrum уже не просто модні слова - це основа того, як люди організовують справи й думають про бізнес. Головне - оперативно підлаштовуватися під клієнтів і ринок. Керування проєктами тепер дає конкретну користь: видимий результат, який можна прорахуавати. [25, с. 57–60]

## 1.2. Філософія та принципи гнучкого управління (Agile): маніфест та цінності

Гнучке управління - це не лише інструменти, це погляд на створення програм через призму цінностей, змін та повторень. Саме в 2001 році такий підхід узаконили завдяки Маніфесту гнучкої розробки [1], який виник як реакція на слабкість старих суворих моделей у швидкозмінному IT-середовищі. Чотири ключові принципи документа задали нову логіку: замість нерухомих процесів і паперів - живі люди й реальні результати.

По-перше, люди й їхня взаємодія важливіші за інструменти чи процеси. Ця основа показує - успіх залежить не від суворих правил, а від того, як добре команда спілкується між собою та замовником. У Agile роблять ставку на живу, просту розмову, де кожен має голос. Коли учасники можуть самі приймати рішення, реакція на проблеми швидша, аніж коли треба чекати дозволу через бюрократичний діалог.

Ще один важливий напрям - це більша увага до робочого продукту, а не до повної документації. У Agile головним є стабільна передача програмного забезпечення, яке працює, перевірене й готове до використання, адже саме так воно задовольняє потреби користувача. На противагу традиційним підходам, де спочатку складають детальні описи, у гнучких методах документи мають допоміжне значення. Поступове надання якісних частин дозволяє замовнику бачити реальний прогрес, отримувати програму раніше і давати зворотний зв'язок у термін - це набагато краще за бездоганні, проте абстрактні звіти.

Третя цінність говорить - співпраця з клієнтом важливіша за сухі умови договору. Це міняє взаємини замовника й виконавця: тепер вони не формальні, а справжньо партнерські. Замість жорсткого дотримання старих положень контракту, Agile робить ставку на постійний контакт із замовником протягом проекту. Клієнт чи його представник (на кшталт Product owner) входить до команди та допомагає визначити пріоритети. Завдяки такому підходу продукту менше загрожує стати непотрібним через деякий час, і обсяги робіт можна гнучко коригувати.

Зрештою, важливіше - це гнучкість замість сліпої вірності плану. У Agile зміни сприймаються не як помилка, а як шанс стати кращим на ринку, бо показують нові потреби чи реальні уроки. Завдяки коротким етапам розробки команда може оперативно підлаштовуватись, не переробляючи все з нуля. Такий підхід дає змогу пристосовуватись на льоту, на противагу жорстким моделям, що покладаються лише на прогнози.

Дванадцять принципів пояснюють ідеї маніфесту й допомагають командам діяти ефективно, зосереджуючись на трьох головних аспектах - замовнику, колективі та якості продукту. Головне в Agile - робити клієнта задоволеним своїм продуктом, щоб розробляти корисний софт стабільно, завдання ділимо на маленькі готові частини. Такий підхід приймає зміну вимог навіть наприкінці проєкту, адже команда має швидко реагувати на нові зміни до вимог. Через це робочий результат краще показувати регулярно - кожні декілька тижнів або місяців, щоб отримувати оперативні коментарі й не витрачати час на функції, які ніхто не буде використовувати, саме гнучкий та якісний продукт стає справжньою ознакою якості.

Успіх залежить не лише від окремих людей, а й від командної взаємодії - бізнес і розробники мають працювати разом кожного дня під час проєкту. Коли компанія співпрацює з міжнародними клієнтами, важливо мати добре налагоджені інструменти для дистанційної координації. Проєкти краще створювати навколо активних учасників, забезпечуючи їм потрібну допомогу, адже найкращі результати дають самостійні команди, які самі обирають шляхи виконання завдань[13, с. 75-76]. Якщо говорити про передачу повідомлень, то безпосередній контакт залишається найточнішим способом - живе спілкування переважає над довгими формальними документами через свою чіткість, оперативність та відкритість.

Методологія скрам робить кцент на технічній якості. Agile дозволяє працювати в сталому темпі, адже реалістичне навантаження запобігає виснаженню (burnout). Особливо важливо зберігати чіткий план та увагу до деталей, простота ж - не плутанина, а мистецтво прибирати зайве, що дає

гнучкість системі. Щоб удосконалюватись, команда періодично оцінює свою продуктивність за допомогою ретроспектив. Такий аналіз стає основою для корекції процесів - саме так команда розвивається.

Філософія Agile будується на злитті ітеративного й інкрементального методів розробки - це безпосередньо підтримує принцип «адаптивності до нових умов». Ітерації означають поділ процесу на короткі проміжки часу, скажімо двотижневі спринти, протягом кожного команда проходить весь етап від планування до тестування, щоб швидше отримати результат та зменшити невизначеність. З іншого боку, при інкрементальному способі кожен закінчений спринт додає до продукту окремий готовий фрагмент функціоналу. Цей фрагмент уже перевірений, працює самостійно і може передаватись у клієнту. Такий синтез дає змогу створювати мінімально життєздатний продукт (MVP) - найпростішу версію, яка задовольняє базові очікування перших клієнтів і допомагає збирати дані для майбутніх покращень проєкту.[14]

Усвідомлення сильних і слабких сторін потрібне для ефективного впровадження Agile. Основні переваги - це зростання задоволеності клієнтів через постійну взаємодію й раннє надання якісного продукту, ризики значно падають завдяки швидкому знаходженню помилок ще на початку проєкту, тим більше коли вимоги міняються часто. Також важливий фактор - гнучкість: пріоритети можна перебудовувати оперативно, адже умови на ринку не стоять на місці, особливо за межами однієї країни [28, с. 105-106]. Завдяки самоорганізації, довірі всередині команди й чіткому фокусі на результаті люди починають активніше долучатись до процесу, відчуваючи свою роль. Проте методологія має обмеження: складно точно планувати строки й бюджет на тривалий термін наперед, тож фінансистам доводиться приймати невизначеність як норму. Окрім цього, підхід вимагає багато від клієнта чи Product owner’a - потрібна постійна участь та оперативні рішення, що не завжди реалізується на практиці. Водночас ефективність залежить від команди: важливий достатній рівень досвіду, самоконтроль і зрілість. Розуміння цих факторів критичне для успішного запровадження

найпоширенішої Agile-практики - Scrum, який детально описано далі.

### 1.3. Методологія Scrum: ключові ролі, артефакти та події

Скрам - це найуживаніший спосіб реалізувати гнучкі технології на практиці. Працює краще всього там, де доводиться мати справу з непередбачуваними завданнями при створенні складних систем.[21] Основа методу - живий досвід разом із здоровим підходом до управління процесами. Скрам це про відкритість, нагляд і готовність швидко все змінювати при потребі. Суть полягає у трьох пунктах: Перш за все – видимість: кожен повинен бачити стан справ, документи та просування вперед. По-друге - перевірка: треба часто аналізувати хід роботи й те, що вже зроблено, аби помічати проблеми раніше. Ну і по-третє - поправки: якщо щось явно йде не так, слід терміново змінювати план чи сам механізм. Уся система побудована навколо трьох частин, які постійно взаємодіють: конкретні ролі, очікувані результати й заплановані перевірки.

У Scrum є три основні ролі - вони разом складають спроможну й незалежну команду.[1] Продукт овнер займається тим, аби продукт давав якомога більше користі. Він керує списком завдань: описує задачі, встановлює пріоритети, пояснює деталі задач і слідкує, щоб усе було зрозуміло сформульовано. Рішення про те, що потрапить у нову версію, приймає саме він. Він ще й пов'язує запити замовників із внутрішньою групою розробників, адже має все чітко та детально пояснити розробникам. Далі – це Скрам-майстер: людина, яка слідкує, чи використовуються правила Scrum правильно.[38] Ця особа не начальник, а радше помічник для команди, допомагає краще працювати за допомогою порад, знімає труднощі, налагоджує контакт між власником продукту, робочою групою та рештою учасників. Група розробки гарантує, що після кожного спринта виходить справжній результат – частина продукту, придатна до використання. Колектив сам організує свою діяльність, функціонує автономно і об’єднує всі професії, потрібні для просування проєкту далі. Вони разом складають плани на спринт,

перевіряючи пункти беклогу й поліпшуючи їхню якість. Гарний результат - це справа кожної людини в команді.

Артефакти - це практичні інструменти, що показують, що вже зроблено й на чому варто зосередитися, допомагаючи час від часу підправляти хід роботи. Беклог - це список нових завдань для продукту - функцій, покращень, виправлення помилок або технічних доробок - живий і гнучкий, адже пріоритети постійно міняються. Команда за методологією Scrum саме звідси бере те, над чим працювати далі. Елементи списку часто записують через призму користувацьких потреб, потім їх аналізують, конкретизують і поновлюють раз у раз - такий підхід має назву Refinement. Тож нагорі завжди лишаються найбільш термінові й добре опрацьовані пункти. Список на спринт береться з основного списку задач - обирають те, що треба зробити зараз. До нього додають чіткий план, як саме це втілити у працездатний результат. Ця мета стає загальною точкою відліку для кожної людини в команді. План живий: команда його корегує кожен день, хто хоче - той допомагає. На виході – нова версія продукту, що включає все, що було зроблено раніше й тепер. Важливо, щоб воно справді працювало, могло вийти в світ і відповідало правилам «готово». Ці правила - не просто формальність, а реальна гарантія того, що якість на місці.

Scrum має п’ять ключових зустрічей, усі вони обмежені за часом - це допомагає швидше приймати рішення. Основою є спринт, навколо якого будуть інші події; він триває до чотирьох тижнів, далі - жодних продовжень, мета лишається такою самою, бо команда повинна зберігати концентрацію. Коли починається новий етап, проводять планування спринта: тут домовляються про те, що саме хочуть зробити, а також прораховують кожен крок. Така зустріч триває не довше восьми годин для місячного спринта, адже важливо не губити час даремно. Кожного ранку команда швидко збирається - так званий щоденний скрам, що займає від 5 до 15 хвилин. На ньому обговорюють, що вдалося зробити учора, а потім планують справи на день. Зустрічаються всі учасники команди, найчастіше стоячи, бо так

продуктивніше. Після кожного спринта проходить огляд - не довший за чотири години при місячному циклі. Учасники демонструють стейкхолдерам результат інкременту спринта(продукт), потім разом розглядають беклог продукту - так обирають найбільш необхідну справу далі. Це шанс побачити стан продукту та оперативно внести коригування. Зустріч ретроспективи спринта триває максимум три години, увага при цьому скерована на те, як команда працювала і чи все було ок з процесами Scrum. Обговорюють, що давалося легко, а де варто щось змінити просто на майбутньому спринті. Кожна сесія допомагає групі навчатися один одного, адже ефект досягається через регулярне покращення не лише коду, але й взаємодії всередині колективу.

## 1.4. Порівняльний аналіз переваг та ризиків використання Scrum і традиційних методологій (Waterfall)

Для обґрунтування вибору методології Scrum компанією ТОВ «*****» необхідно провести порівняльний аналіз цього гнучкого підходу з традиційними послідовними моделями, з яких найпоширенішою є Waterfall (Каскадна модель). Це дозволить чітко визначити переваги та ризики кожної моделі в контексті сучасного управління ІТ-проєктами.[19, с. 105-106].

Також існують інші методології розробки ПЗ, але ми будемо фокосуватися на основних методолігіях адже більшість інших методологій вже застарілі окрім можливо Канбану який також часто використовується в розробці ПЗ, але зараз ми розглянемо основі відмінності між Scrum та Waterfall.

Основна відмінність між Scrum та Waterfall криється у філософії та підході до планування та реалізації[20, с. 51-55]. Ці контрасти найкраще проілюстровані у порівняльній таблиці.

Таблиця 1.

Порівняння Scrum та Waterfall

| **Критерій порівняння** | **Scrum (Agile-фреймворк)** | **Waterfall (Традиційна модель)** |
|---|---|---|
| **Філософія** | Адаптивна, орієнтована на швидку реакцію на зміни, емпіризм. | Прогностична, орієнтована на суворе дотримання плану, що базується на вихідних вимогах. |
| **Життєвий цикл** | Ітеративний та інкрементальний (короткі Спринти), що створює робочий продукт частинами. | Лінійний, послідовний (фази: вимоги, дизайн, реалізація, тестування) з однією точкою випуску. |
| **Участь клієнта** | Висока, постійна, щоденна співпраця. | Низька, обмежена початковим етапом збору вимог та кінцевим прийняттям. |
| **Зміни вимог** | Вітаються на будь-якому етапі, інтегровані через механізм Product Backlog Refinement. | Високозатратні, складні та небажані, оскільки можуть призвести до повного перепланування. |

Застосування Scrum забезпечує фірмам із розробки складних продуктів важливі переваги через гнучкість у процесах.

Scrum допомагає створювати продукт більш якісно та швидко для замовника. На відміну від Waterfall, коли продукт показують лише після довгого часу - тут нові робочі частинки з’являються кожні 1–4 тижні. Завдяки цьому клієнт може постійно все перевіряти, тестувати й коментувати. Через таку взаємодію команда швидше реагує на зміни, а результат краще відповідає поточним очікуванням ринку і замовника - що особливо важливо в умовах

гострої конкуренції.[14, с. 120-121]

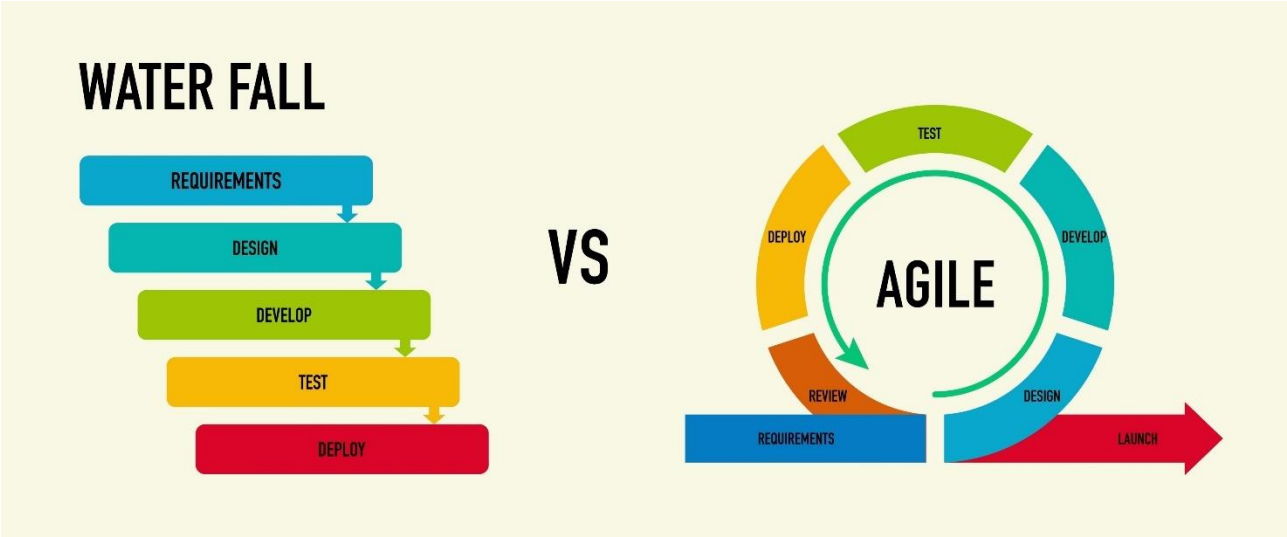


*Рис 1. Waterfall vs Agile*

Методологія scrum допомагає зменшити ризики, підвищує контроль. У Waterfall помилки на етапі вимог часто знаходять лише при тестуванні - це ускладнює виправлення помилок й тягне за собою затримки. Scrum через короткі спринти і щоденні мітинги швидко виявляє проблеми, технічний борг або перешкоди. Ретроспективи ж постійно покращують процеси, адже команда аналізує спілкування й взаємодію.

По-третє, у Scrum команди взаємодіють більш ефективніше (Team Empowerment). Самоорганізація тут - основа, бо люди самі вирішують, як краще працювати. Такий підхід додає відповодальності й заангажованості, а ще пробуджує нові ідеї. У Waterfall навпаки - строга верствова система, де все контролює начальство. Це часто гасить ініціативу сильних спеціалістів і обмежує творчий простір.

Навіть якщо Scrum має чіткі переваги, з ним пов’язані окремі ризики - їх варто враховувати, особливо під час розширення.

Один із головних ризиків Scrum - це проблеми з довготривалим плануванням [14, с. 21–22]. У Waterfall усі деталі, строки й бюджет встановлюють на початку процесу, що створює вигляд чіткого контролю. Через те, що Scrum спирається на крокове планування, важко точно передбачити загальний обсяг завдань або фінансові потреби на майбутнє, тому

фінансовим партнерам треба більше гнучкості та впевненості.

Ще один ризик - надмірна залежність від ролей (Ризик ресурсів). Ефективність Scrum значною мірою базується на фаховості й включеності ключових осіб, зокрема Product owner (PO) та Scrum master (SM). Коли PO не чітко формулює пріоритети або важкодоступний для команди, спринти втрачають ефект. Через недостатній досвід SM перешкоди можуть залишатись нерозв’язаними, що призводить до безладу, який часто називають «ScrumBut» - коли методологія застосовується частково чи формально.[6, с. 21–22]

Треба звернути увагу на ризики, пов’язані з обсягом завдань. У Waterfall висока ймовірність Scope Creep - неочікуване нарощування завдань через офіційні правки, які важко врахувати пізніше. Водночас у Scrum такий зріст менш імовірний: нові задачі потрапляють до Product backlog та борються за пріоритет разом із рештою. Це дозволяє тримати обсяги під контролем протягом чітко обмеженого часу спринта.

# РОЗДІЛ 2. АНАЛІЗ ПРАКТИКИ ВПРОВАДЖЕННЯ AGILE ТА SCRUM В ІТ-КОМПАНІЇ

## 2.1. Організаційно-економічна характеристика діяльності ТОВ «*****»

ТОВ «*****» - це частина міжнародної ІТ-компанії ***** із Данії. Робота українського відділення зосереджена на важливих техноглогічних проєктах: тут створюють програми, допомагають у розвитку бізнесу, запускають цифрові системи.[16] Основні клієнти - компанії з країн Європи, Америки. Протягом часу фірма отримала досвід у реалізації проектів будь-якої складності: маленькі замовлення поєднуються з великими глобальними завданнями. Цей досвід допоміг створити гнучку систему реагування на запити клієнтів - паралельно з цим сформовано команду фахівців, які впевнено працюють у різнокультурних умовах. Бізнес-підхід ТОВ «*****» будується навколо комплексного підходу: компанія пропонує повний цикл послуг замість простого аутсорсингу персоналу. Фокус - на технологічному партнерстві із акцентом саме на якісну розробку програмного забезпечення. Основна діяльність охоплює широкий масив ІТ-напрямків: тут і кастомні проєкти для FinTech, електронної комерції й медичних платформ, окремо - тестування й запуск ІТ-продуктів із контролем якості та автоматизацією процесів. Ще один напрям - консультації з управління проектами й бізнес-аналітики, разом із технічною підтримкою (від L1 до L3) уже запущених систем.

Організаційна структура ТОВ «*****» відповідає сучасним ІТ-компаніям, які працюють за гнучкими підходами. Матрична модель дозволяє співробітникам належати до функціональних підрозділів - таких як HR, розробка чи тестування - з одночасним входженням у проєктні Scrum-групи. Ці команди формуються тимчасово, орієнтуючись на конкретні завдання. Такий підхід поєднує глибину експертизи всередині напрямів із оперативною взаємодією в рамках проєктів. Загальне стратегічне керівництво покладено на директора, до складу якого входять провідні відділи компанії. HR-відділ займається підбором, адаптацією й розвитком працівників - пошук ведеться

систематично. У Delivery-відділі (PMO) керують портфелем проєктів, тут також координують реалізацію продуктів, саме цей напрям охоплює Scrum masters і Project managers. Ресурсну основу для команд створюють технічні департаменти - Development і QA працюють у тісному зв'язку. Функції бухгалтерії та адмінвідділу полягають у фінансовому облікові, контролі, логістиці тощо, вони забезпечують внутрішню стабільність. Організація активно застосовує гнучкі методики: Agile, Scrum чи Kanban замінюють жорсткі схеми, що дозволяє швидше реагувати на запити міжнародного ринку й полегшує комунікацію.

Таблиця 2.1

Приблизний розподіл персоналу за функціональними підрозділами

| **Підрозділ** | **Кількість осіб** | **Частка, %** |
|---|---|---|
| Розробка (Development) | 115 | 46.7% |
| Забезпечення якості (QA) | 35 | 14.2% |
| Проєктний менеджмент (PM/Scrum masters) | 22 | 8.9% |
| Бізнес-аналіз (BA/PO) | 18 | 7.3% |
| Продажі та маркетинг | 20 | 8.1% |
| Адміністративний та Фінансовий персонал | 36 | 14.6% |
| РАЗОМ | 246 | 100% |

Корпоративна культура будується на відкритості та довірі - також важливою є взаємоповага та постійний розвиток. Політика з персоналом у «*****» спрямована на підтримку людей як основного капіталу. Навчання організовує HR-відділ: працівники проходять курси, тренінги - це стосується і технічних навичок, а також гнучких компетентностей. Командна діяльність разом із корпоративними заходами допомагає краще спілкуватись і ефективніше співпрацювати. Компанія послідовно впроваджує принципи різноманіття та інклюзивності, даючи однакові шанси кожному. Зокрема

реалізує проєкти соціального характеру - пожертвування на потреби громади чи фінансування навчальних програм, що покращує сприйняття бренду серед кандидатів на роботу. Умови для комфортної роботи формуються систематично, у тому числі враховується емоційний стан персоналу. Це створює основу для справжнього європейського типу корпоративної культури, яка дозволяє командам запроваджувати нові рішення.

Фінансова міцність ТОВ «*****» ґрунтується на роботі з платоспроможними ринками та довгостроковому співробітництві з замовниками. Навіть якщо точних цифр не розголошують, вивчення діяльності дає можливість зрозуміти основні економічні принципи компанії. Конкурентоздатність підтримується за рахунок професійного складу кадрів разом із раціональною структурою витрат. Серед головних сильних сторін - швидке запускання продуктів, що досягається через Agile-підхід, а також сталість у якості завдяки відповідності стандартам ЄС. Клієнти охоплюють різні сфери - FinTech, торгівля, медицина, це зменшує загрозу втрат через спад у одній галузі. Багато хто повертається для нових проектів, до того ж довготривалі угоди (більше двох років) показують: клієнти задоволені співпрацею. Гнучкі підходи покращують результати не лише за рахунок скорочення зайвої документації та чіткої черги завдань у Scrum, а й шляхом економії часу - адже часте коригування курсу запобігає дорогим переробкам. Отже, ТОВ «*****» працює за моделлю, яка поєднує мобільність, свіжі ідеї й раціональне використання коштів, тож її можна брати за основу досліджень про гібридні Agile-практики.

### 2.2. Аналіз методологічної бази управління ІТ-проєктами в компанії

Методологія керування проєктами в ТОВ «*****» ґрунтується на принципах Agile - цей напрям обрано через його придатність для роботи з мінливими та складними умовами світового ринку. Основу становить гнучка стратегія, поєднуючи елементи Scrum і Kanban - такий підхід часто називають Scrumban. Така модель враховує реальність комерційного аутсорсингу, де

жорстке дотримання класичних правил Scrum не завжди можливе, замість цього команди мають простір для адаптації. Зокрема, терміни проведення подій коригуються разом із замовником, якщо того вимагають потреби проєкту чи ситуація на ринку.

Суть «М’якого Scrum» - це поєднати структуру Scrum і вільний рух Kanban. Тут є чіткі ролі, спринти та наради, але водночас додаються елементи потоку: видимість завдань, межі для активних справ (WIP) і плавна подача результату.[24] Такий спосіб допомагає ***** тримати порядок і темп: короткі цикли по два тижні дають регулярні оновлення продукту. У той самий час команда може швидко реагувати - особливо коли йдеться про технічну підтримку чи раптові задачі, як-то критичні помилки, у таких випадках працює дошка типу Kanban з контролем навантаження, щоб не гальмувати процес. Головне правило такого підходу: формальні норми Scrum можна міняти, якщо так буде краще для замовника. Наприклад, ранкові збори скорочують до 5–7 хвилин там, де команда сильна, автономна і добре координується. Вигідно аутсорсити ще й тому, що на малих проєктах Scrum Master інколи замінюється керівником проекту чи техлідом. Зміни в спринті теж можливі - особливо при серйозних помилках або нових потребах ключових клієнтів. У такому разі мета спринта коригується через діалог продукт овнера з командою розробників, щоб адаптація була під контролем. В загалі м’який скрам допомагає командам бути більш гнучкими у випадках коли необхідно щось швидко і ефективно поміняти або змінити напрям розробки, часто м’який скрам є необхідністю для того щоб продовжувати працювати ефективно.

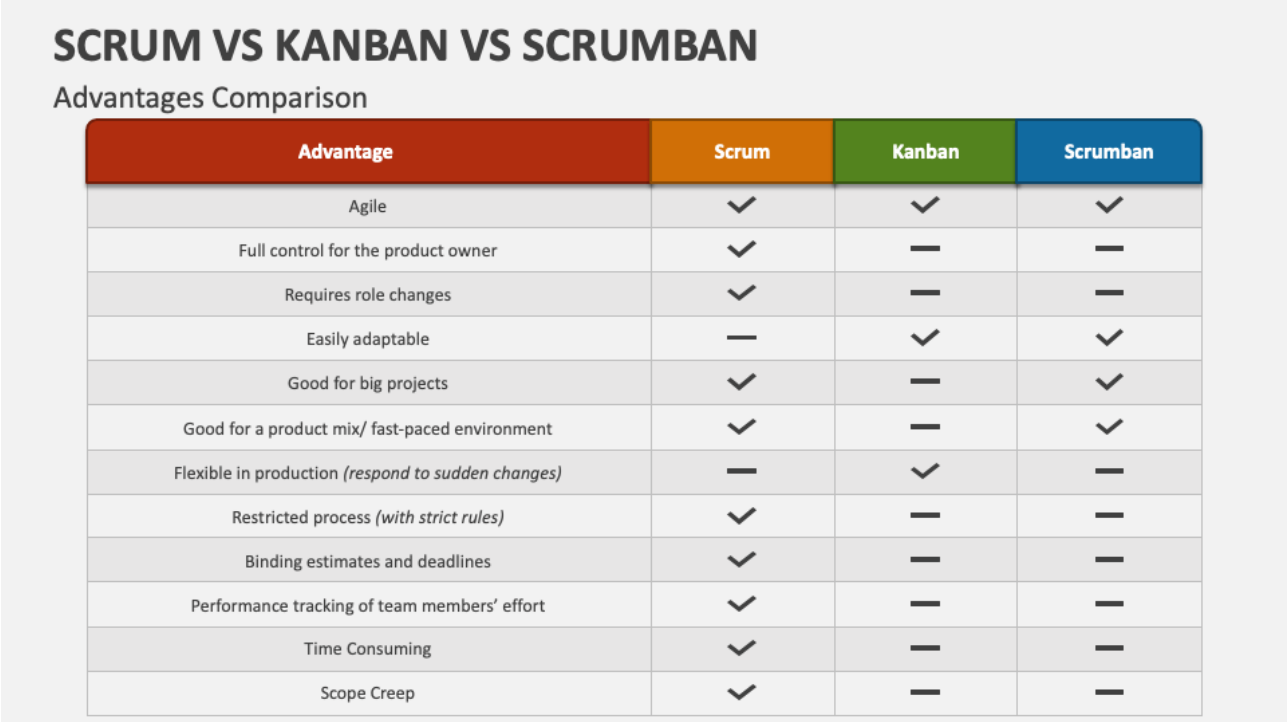


*Рис 2. Scrum vs Kanban vs Scrumban*

ТОВ «*****» гнучко впроваджує основні ролі, адаптуючи їх під матричну організацію. Команда розробки - це зазвичай 3–9 фахівців із різним досвідом, які самостійно планують свою роботу. Функцію власника продукту часто обирає замовник або кваліфікований аналітик від *****, той забезпечує чіткий акцент на бізнес-користь. Скрам-мастер у компанії найчастіше супроводжує кілька команд одночасно, працюючи радше як провідник процесу, аніж контролер, що відображає принцип м'якого управління. Разом з Jira, Asana або Trello часто але не завжди використовують сучасні інструменти для керування процесами - це дає чіткий огляд усіх трьох Scrum-артефактів. Формат Product backlog постійно оновлюється та розставляє пріоритети, враховуючи довгострокову стратегію продукту. Під час планування спринта завдання деталізують до технічного рівня в межах Sprint backlog. Definition of done дотримуються неухильно, саме ця практика гарантує якість результату, що особливо важливо в FinTech й Healthcare.

Упровадження гібридних Agile-підходів чітко позначилось на роботі ТОВ «*****». Завдяки постійним спринтам клієнти отримують якійсний продукт швидше за конкурентів - це скорочує час виходу продукту на ринок,

даючи перевагу у глобальному масштабі. Щоденні зустрічі Daily scrum разом із регулярними оглядами спринтів значно покращили взаєморозуміння між командою, Product owner та стейкхолдерами, сприяючи кращій комунікації. Самоорганізація колективів дала змогу брати більшу відповідальність, самостійно приймати рішення - як результат, потреба в детальному контролі практично зникла. Також регулярні ретроспективи дають змогу команді аналізувати свої процеси й вносити в них поправки. Наприклад, коли команда систематично не встигає реалізувати заплановане навантаження - вона переглядає оцінювання часу на розробку або скорочує тривалість спринта, це демонструє емпіричний підхід на практиці та сприяє розвитку культури поступового покращення розробки.

ТОВ «*****» використовує гнучку основу - не сувора слідування стандартам, а поєднання Scrum і Kanban. Такий підхід дає можливість комбінувати чіткий ритм та ролі з оперативною адаптивністю. Завдяки цьому метод працює добре у IT-аутсорсингу, де вимоги часто міняються.

## 2.3. Оцінка переваг та ідентифікація проблемних зон при застосуванні гнучких методологій у діяльності компанії

Застосування гнучкого підходу «Ненав'язливий Scrum», розглянутого раніше, дає ТОВ «*****» чіткі переваги на глобальному ринку. Проте цей практичний метод водночас викриває окремі внутрішні небезпеки й слабкі місця - їх потрібно постійно контролювати та корегувати.

Завдяки гібридному підходу до управління, ***** змогла покращити робочі процеси і краще взаємодіяти з клієнтами. Головна вигода - швидкий запуск продуктів на ринок (Time-to-market), адже регулярні двотижневі спринти створюють чіткий темп та гарантоване постачання працездатного ПЗ. Такий формат дає можливість замовникам оперативно отримувати якісний продукт і контролювати фінансові витрати - це особливо важливо у швидкоплинних сферах типу FinTech або E-commerce. Окрім того, елементи

Kanban, як-от обмеження поточної роботи (WIP) і акцент на пропускну здатність, допомагають командам техпідтримки скоротити простої й швидше реагувати на серйозні помилки чи невеликі, проте термінові завдання, що забезпечує стабільність передачі якості. Не менш важливим є покращення стандартів якості й видимості процесів для всіх учасників. Хоч правила й не суворі, якість обов'язково висока - це непорушна умова. Definition of done (DoD) чітко дотримується, тож кожен інкремент повністю перевірено та готовий до запуску, що значно зменшує технічний борг. Загалом процес підтримують регулярні заходи в рамках Scrum: Огляд спринта або Daily Scrum - разом із Jira-дошками для наочності роботи. Це забезпечує стейкхолдерам і клієнтам прозорість просування проєкту. Команда все частіше проявляє самостійність і внутрішню мотивацію - Team empowerment помітно зріс. Крос-функціональні команди мають свободу організовувати себе самі, через що посилюється відповідальність і активність. Фахівці обирають оптимальні способи реалізації завдань, отже приймають кращі технічні рішення. Ретроспективи проводяться систематично - там можна без страху говорити про труднощі. Саме так команда постійно аналізує свої процеси й коригує їх, формуючи середовище поступового розвитку.

Хоч гібридний підхід добре працює, через неявне впровадження та особливості аутсорсингу з'являються ризики - їх потрібно постійно стримувати. Можливе порушення Scrum-принципів (ScrumBut)[21], якщо, наприклад, функції Scrum Master бере на себе Tech Lead чи Project manager. Хоч це економить кошти, такий крок загрожує ідеї лідерства-служби, провокуючи мікроменеджмент і слабшу самоорганізацію команди. Крім того, коли терміни чи замовник тиснуть, деякі важливі заходи - типу Ретроспектив чи Огляду спринта - можуть скоротити або взагалі скасувати, це призводить до накопичення проблем і зменшення схильності до сталої модернізації процесів. Також виникають ускладнення з довготривалим плануванням - важко точно передбачити строки й бюджет на період більше 3–6 місяців. Через це фінансові стейкхолдери часто наполягають на чітких цифрах, що не завжди

узгоджується з підходом Agile, Delivery-командам тоді доводиться шукати рівновагу між адаптивністю і економічною прозорістю. Неперервна можливість коригувати курс часом призводить до того, що Product owner постійно знаходиться під тиском, аби додавати нові задачі прямо в спринт - явище, відоме як "розтягування спринта", - через що ціль спринта може порушитися, а команда - перевтомитися. Окрім того, організації треба враховувати особливості взаємодії з віддаленими партнерами й представниками різних культур.[24] Робота з людьми з інших часових поясів робить важким узгодження часу для нарад - командам доводиться включатися раніше або залишатися після закінчення дня. Через неоднакове ставлення до спілкування чи влади деякі колеги сором’язливо діляться зауваженнями на ретро, так само й клієнт може стримуватися, отримуючи пряму оцінку.

Для системної оцінки поточної ситуації проведемо SWOT-аналіз впровадження гнучких методологій у ТОВ «*****».

Таблиця 2.2

SWOT аналіз

| **Категорія** | **Внутрішні фактори** | **Зовнішні фактори** |
|---|---|---|
| **Сильні сторони (Strengths)** | S1. Гнучкий гібридний підхід: Успішне поєднання Scrum (ритм) та Kanban (потік). S2. Кваліфікація команди: Наявність крос-функціональних та самоорганізованих команд. S3. Високий стандарт якості (DoD): Забезпечення надійності продукту. | O1. Зростаючий попит на швидкість: Глобальний ринок цінує швидкий Time-to-Market, який забезпечує Agile. O2. Інтеграція інструментів: Широке застосування Jira, що полегшує прозорість та віддалене управління. O3. Зрілість ринку: Більшість міжнародних клієнтів вже розуміють та підтримують Agile-процеси. |

Продовження таблиці 2.2

| | | |
|---|---|---|
| **Слабкі сторони (Weaknesses)** | W1. Ризик розмивання ролей: Змішування функцій PM/SM, що знижує фокус на коучингу. W2. Складнощі з оцінкою: Проблеми з надійним довгостроковим прогнозуванням. W3. Залежність від PO: Критична залежність успіху від доступності та компетентності Product Owner'a. | T1. Жорсткі вимоги клієнта: Вимога фіксованої ціни/часу (Fixed-Price) від клієнтів, що суперечить гнучкості. T2. Висока конкуренція: Конкуренти також використовують Agile, що вимагає постійного вдосконалення процесів. T3. Геополітичні та економічні ризики: Зовнішні чинники, що впливають на стабільність команди та процесу. |

У ***** успішне керування проєктами залежить від якості спілкування - адже багато команд працюють дистанційно або гібридно. З цією метою застосовують не лише пошту, Slack чи Teams для поточних запитань (щоб уникнути проблем із часовими зонами), а й регулярні онлайн-зустрічі. Такі наради передбачають короткі щоденні сесії типу Daily Scrum через вебкамери, де важлива демонстрація екрану з Jira. Документи систематично оновлюють у Confluence та внутрішніх базах знань - це забезпечує доступ до інформації без надмірної комунікації. Крім того, компанія розробляє практики, щоб послабити труднощі, пов’язані з міжкультурним форматом роботи та віддаленістю. Серед них - чіткі правила зворотного зв’язку, які допомагають уникати конфліктів через культурні відмінності під час прямої спілкування, особливо на ретроспективах. Водночас активно застосовують візуалізацію процесів: дошки та діаграми Burn-down або Burn-up, щоб знизити важливість мовового бар'єра й очевидно показувати просування в розробці.

Загалом, гнучкі підходи добре працюють для ТОВ «*****», адже

посилюють внутрішню мобільність та орієнтацію на клієнтів. Проте результат залежатиме від того, як компанія стежитиме за своїми слабкими місцями - особливо щодо балансу між легкою формою Scrum і чіткою дисципліною.

## 2.4. Практика реалізації IT-проєктів: аналіз успішних кейсів та отриманих результатів

Робота ТОВ «*****» чітко показує - гібридний метод «Ненав'язливий Scrum» справді працює для міжнародних аутсорсингових проєктів. У цьому розділі - аналіз головних показників успіху та два реальні приклади, де гнучкість з дисципліною створили вигоду для бізнесу.

Замість звичайних підходів, де головне - це терміни й бюджет, у ***** використовують інший спосіб: тут важливо, наскільки швидко щось робиться, якісно воно вийшло та чи задоволений замовник (реальна користь). Оцінюють ефективність за допомогою конкретних показників.

Таблиця 2.3

Таблиця ефективності

| **Категорія метрик** | **Ключові показники (KPI)** | **Мета застосування** |
|---|---|---|
| **Швидкість постачання** | Velocity (Швидкість команди) | Вимірювання обсягу цінності (зазвичай, у Story Points), який команда успішно завершує за один спринт. Використовується для прогнозування майбутніх постачань. |
| | Sprint goal success rate | Відсоток спринтів, у яких була повністю досягнута Ціль спринта. Відображає стабільність та передбачуваність команди. |

Продовження таблиці 2.3

| | | |
|---|---|---|
| **Якість продукту** | Defect density | Кількість виявлених дефектів на одиницю коду або на один Story point. Допомагає відстежувати ефективність DoD та тестування. |
| | Time Spent on Rework | Відсоток часу спринта, витрачений на виправлення багів, знайдених після випуску, що відображає технічний борг. |
| **Задоволеність клієнта** | NPS (Net Promoter Score) | Оцінка ймовірності того, що клієнт порекомендує компанію (вимірюється після великих релізів). |
| | Cycle Time (Kanban-метрика) | Середній час, необхідний для переведення завдання з колонки «В роботі» до «Готово». Критичний для підтримки та термінових завдань. |

Використання цих метрик базується на простому правилі: стале значення швидкості - навіть не найвище - краще за нестабільне. Через це команда зможе планувати справжні, досяжні цілі спринту, що підвищує частку успішних завершень спринта і посилює впевненість замовника у передбачуваності команди. Для клієнта стабільність має велике значення, адже пов’язана з надійністю колективу та його здатністю дотримуватися домовленостей. Разом ці показники дають чіткий огляд не лише темпу роботи, але й рівня якості, а також ефекту на бізнес, роблячи процес прозорим замість невідомого «чорного ящика».

Фокус на сталій швидкості роботи допомагає замовникові краще організовувати процеси - наприклад, просвітницькі заходи чи рекламні кампанії, адже терміни появи нових можливостей стають зрозумілішими. У разі, коли команда досягає мети спринта принаймні у дев’яти із десяти випадків, це підвищує впевненість замовника та полегшує контроль над

очікуваннями. Саме така системність часто має найвищу вагу під час взаєморозрахунків ззовні.

Ця методика дає змогу ***** самостійно знаходити проблемні ділянки в робочих процесах. Наприклад, коли кількість помилок тримається на низькому рівні, проте час на переробку поступово збільшується - це явний знак нагромадження технічних боргів, які варто усунути задовго до того, як команда значно уповільниться. Також довгий час виконання задач, насамперед тих, що стосуються підтримки, часто вказує на те, що потрібно покращити обслуговування клієнтських запитів - наприклад, чіткіше прописати SLA або полегшити перевірку й запуск продукту.

Контроль цих показників допомагає замість вирішення помилок працювати разом із замовником - так, де важливі довіра й користь від розробки. Замість простого виконання командою завдань, процес поступово покращується на основі аналізу даних задля ефективних підсумків.

Для ілюстрації ефективності методологічної бази ***** розглянемо два кейси, які демонструють переваги гібридного підходу в різних галузях.

Випадок 1 - Розробка B2B FinTech-платформи (Фокус на TTM та адаптивності)

Таблиця 2.4

Опис FinTech платформи

| **Характеристика** | **Опис** |
|---|---|
| **Галузь** | Фінансові технології (FinTech), B2B-кредитування. |
| **Виклик** | Клієнт (європейський банк) вимагав швидкого запуску MVP для отримання ліцензії. Невизначеність регуляторних вимог вимагала частих змін у логіці верифікації. |
| **Методологія** | Чистий Scrum (2-тижневі спринти) з дуже тісною інтеграцією Product Owner'а клієнта. |

Продовження таблиці 2.4

| | |
|---|---|
| **Ключові дії *****| 1. Інтенсивний Refinement: Постійне обговорення вимог з PO, щоб гарантувати, що Беклог відповідає останнім регуляторним змінам. 2. Автоматизація тестування: Жорстке DoD, що включає 100% покриття критичного функціоналу автоматизованими тестами, щоб уникнути регресій при частих змінах. |
| **Отримані результати** | Time-to-Market (TTM): MVP випущений на 1 місяць раніше запланованого терміну (скорочення TTM на 25%). Адаптивність: Команда успішно інтегрувала три великі зміни в основну логіку (понад 15 Story points кожна) без зриву цільових результатів спринтів. Бізнес-цінність: Клієнт отримав ліцензію та почав генерувати дохід на 6 місяців раніше. |

Випадок 2 - Підтримка та розвиток великої E-commerce платформи (Фокус на Quality, Kanban та потоці)

Таблиця 2.5

Опис E-commerce платформи

| **Характеристика** | **Опис** |
|---|---|
| **Галузь** | Роздрібна торгівля (E-commerce), high-load платформа. |
| **Виклик** | Платформа працювала на застарілому коді, мала високий рівень Defect Density (понад 0.5 на 1000 рядків коду) та потребувала одночасної роботи над новими фічами і постійного виправлення критичних багів. |
| **Методологія** | Scrum для розробки нових фіч + Kanban для команд підтримки (L2/L3) і технічного боргу. |

Продовження таблиці 2.5

| | |
|---|---|
| **Ключові дії *****| 1. Відокремлення потоків: Розділення Беклогу на два потоки – Features (Scrum) та Maintenance (Kanban). 2. Обмеження WIP: Встановлено суворі ліміти WIP на Kanban-дошці для пріоритезації критичних багів, що запобігає розпорошенню уваги команди. 3. Спринти хардкорнингу: Включення в Product Backlog завдань з рефакторингу, що запобігає зростанню технічного боргу. |
| **Отримані результати** | Defect density: Знижено з 0.5 до 0.15 на 1000 рядків коду протягом 9 місяців. Cycle Time: Середній час вирішення критичного багу скорочено з 48 годин до 8 годин завдяки фокусу Kanban. Якість: Зростання NPS клієнта завдяки стабільності платформи та високій швидкості завантаження сторінок. |

Аналіз проєктів ***** за рік дає змогу визначити основні цифри ефективності Agile. Замість стабільності - Sprint goal success rate тримається на позначці 85–90%. Високий результат показує, що команди добре планують завдання та адекватно оцінюють навантаження. Цього досягнуто через регулярні ретроспективи, де команда коригує свої процеси. Середнє відхилення фактичної Velocity від середньої по команді рідко перевищує ±10%, що дозволяє Delivery-відділу ***** надавати клієнтам більш надійні прогнози щодо довгострокових постачань, частково мінімізуючи слабку сторону складнощів з оцінкою (W2). Що стосується якості та технічного боргу, рівень технічного боргу, оцінений за часом, завдяки включенню рефакторингу в Definition of Done та обов'язковому виділенню до 10% Story Points на вдосконалення, у середньому не перевищує 15% від загального часу розробки, що забезпечує довгострокову стійкість продуктів. Нарешті, ефективність комунікації та клієнтоорієнтованість підтверджується лояльністю клієнтів (Retainers), оскільки понад 70% клієнтів продовжують

співпрацю з ***** після завершення MVP, переходячи на довгострокову підтримку та розвиток продукту. Це прямий показник високої задоволеності клієнтів (NPS) та ефективності регулярних оглядів спринта, які забезпечують постійну синхронізацію очікувань. Регулярне збирання зворотного зв'язку під час оглядів спринта дозволяє виявляти невідповідність вимогам (Missed Requirements) у середньому через 3–4 тижні після початку розробки, тоді як у моделі Waterfall цей період міг би становити 3–6 місяців.

Досвід ІТ-проєктів у ТОВ «*****» показує - гібридний Agile працює найкраще в міжнародному аутсорсингу. Ефективність залежить не від сліпої роботи за Scrum, а від того, як його ідеї (емпіричні дані, перевірка, коригування) допомагають досягти мети - швидше запускати продукт, краще передбачати процес та тримати стабільну якість.

# РОЗДІЛ 3. ПРОЄКТНІ ПРОПОЗИЦІЇ З УДОСКОНАЛЕННЯ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ НА ОСНОВІ ПРИНЦИПІВ AGILE ТА SCRUM

## 3.1. Мінімізація типових ризиків ІТ-проєктів в Scrum-середовищі

Ефективне управління ІТ-проєктами, особливо в аутсорсинговій моделі, вимагає не лише дотримання процесів, а й мінімізації ризиків. Хоча Scrum за своєю природою є інструментом для виявлення та адаптації до ризиків (завдяки частим інспекціям), типові проблеми, такі як розмиті вимоги, перевантаження команди та технічний борг, можуть дестабілізувати навіть найзрілішу команду.

Проєктні пропозиції ***** зосереджуються на вдосконаленні процесів, що безпосередньо впливають на три ключові категорії ризиків: ризики скоупу та якості визначення, організаційні та людські ризики та технічні ризики.

Найбільш поширений ризик у гнучких методологіях – це неконтрольоване зростання обсягу робіт (Scope Creep) та ризик постачання функціоналу, який не відповідає бізнес-цінності. У ***** цей ризик мінімізується через стандартизацію того, що вважається "Готовим" та "Готовим до розробки".

Ризик W1 (Нестабільні та нечіткі вимоги) є першопричиною низької Velocity та частих переробок. Якщо команда бере в роботу нечіткий елемент з беклог, це неминуче призводить до простою, поганої роботи та зниження морального духу. Щоб мінімізувати цей ризик, пропонується ввести сувору Definition of Ready (DoR) – набір критеріїв, яким має відповідати елемент Product Backlog (PBI), перш ніж його можна взяти в спринт.

Таблиця 3.1

Критерії DoR

| **Критерій DoR** | **Мета мінімізації ризику** | **Відповідальний за верифікацію** |
|---|---|---|
| Чіткість цінності | Уникнення розробки PBI без підтвердженої бізнес-цінності. Кожне завдання має відповідати SMART-критеріям. | Product Owner |
| Наявність Exit Criteria | Гарантія, що Definition of Done (DoD) для цього PBI визначено, погоджено з клієнтом, а його критерії прийому (Acceptance Criteria) чітко описані. | Scrum Master, QA Lead |
| Технічна здійсненність | Запобігання блокуванню роботи через невідомі залежності, відсутність інтеграційних ключів чи технологічні бар'єри. | Development Team |
| Розмір та Оцінка | PBI має бути розміром story points, придатним для завершення протягом одного спринта (наприклад, 1-8 SP). Занадто великі PBI повинні бути декомпоновані (розділені). | Development Team |

Практичне застосування та роль QA як Gatekeeper: QA-інженери, як ключові учасники процесу Refinement, повинні мати повноваження вето на PBI, що не відповідає DoR. Це перетворює QA з кінцевого тестувальника на превентивний фільтр якості на етапі планування. Якщо PBI не пройшов DoR, він не потрапляє в спринт, що захищає Velocity команди та запобігає втраті часу, спричиненій роботою над невизначеними вимогами.

Хоча ***** вже має DoD, ризик Q1 (Проблеми з якістю через поспіх)

та W3 (Розбіжність очікувань клієнта) вимагає інтегрувати елементи, які менеджмент повинен контролювати такі як Exit criteria проєкту (а не лише спринта). Це забезпечує, що готовий до випуску продукт відповідає не тільки технічним стандартам, але й стратегічним бізнес-цілям.

Таблиця 3.2

Опис DoD

| **Елемент розширеного DoD** | **Вимога/Стандарт** | **Мінімізований ризик** |
|---|---|---|
| **Покриття бізнес-логіки автотестами** | Забезпечення покриття ключових сценаріїв (Acceptance criteria) автотестами. Мінімальний поріг покриття критичного функціоналу має бути встановлений на рівні 90%. | Регресія при майбутніх змінах, низька технічна якість (Q1) |
| **Делегування (Delegation) документації** | Призначення відповідального (Technical writer / розробник/QA) за оновлення технічної та користувацької документації до завершення спринта. | Технічний борг документації, ризик W3 (Розбіжність очікувань) |
| **Менеджерські Exit criteria (Реліз)** | Застосування кількісних та якісних критеріїв для релізу: Defect density не перевищує 0.1 на 1000 рядків коду, проведено A/B тестування для 80% нових функцій, відсутність критичних/високих багів на Production. | Ризик Q1 (Проблеми з якістю), репутаційні та фінансові ризики |

Ця пропозиція прямо відповідає баченню, що менеджмент повинен розуміти та застосовувати Exit criteria як фінальний, некомпромісний фільтр для захисту репутації компанії та фінансових інтересів клієнта.

Людський фактор (вигорання, нерозуміння ролей, внутрішні конфлікти, залежність від ключових гравців) є одним із найбільш непередбачуваних та руйнівних ризиків. Проєктні пропозиції спрямовані на створення стійкої та самоорганізованої команди, яка керується знанням бізнес-контексту.

Ризик Р2 (Недостатній розвиток компетенцій) та Р3 (Незрозумілість ролей у гібридній моделі) мінімізується через індивідуалізовані навчальні програми, які посилюють крос-функціональність та бізнес-орієнтованість.

Таблиця 3.3

Коучинг працівників компанії

| **Цільова група** | **Фокус коучингу** | **Ключові активності** |
|---|---|---|
| **QA-інженери** | Пріоритезація Бізнес-Цінності | Навчання оцінці впливу (Impact assessment) дефектів на дохід/репутацію. Симуляційні вправи для самостійної пріоритезації дефектів (Business-Critical, Feature-Critical, Aesthetic). |
| **Менеджмент (PM, Delivery Lead)** | Практика Exit Criteria | Регулярні симуляційні ігри та воркшопи для прийняття рішень про випуск на основі кількісних Exit Criteria. Створення ментального бар'єру проти випуску недосконалого продукту. |

Наведена структура коучингу сфокусована на двох ключових групах, які мають прямий вплив на якість кінцевого продукту та процеси прийняття рішень. Для QA-інженерів головний акцент робиться на навчанні пріоритезації бізнес-цінності. Це передбачає освоєння методик оцінки впливу дефектів на дохід і репутацію, а також практику розподілу дефектів за критичністю (Business-Critical, Feature-Critical, Aesthetic) через симуляційні вправи. Для менеджменту (PM, Delivery Lead) ключовою активністю є

практика використання Exit criteria, керівники регулярно беруть участь у симуляційних воркшопах, щоб навчитися приймати кількісно обґрунтовані рішення про випуск продукту та свідомо формувати "ментальний бар'єр" проти релізу недосконалого рішення.

Ризик P1 (Перевантаження та вигорання ключових членів команди) є критичним і виникає через залежність від 1-2 фахівців. Ефективне делегування задач запобігає вузьким місцям та підвищує стійкість команди. Пропонується створити та підтримувати матрицю компетенцій (Skill Matrix) як інструмент для візуалізації крос-функціональності та планування навчання.

Таблиця 3.4

Рівні делегування

| **Рівень делегування** | **Опис та відповідальність** | **Мінімізація ризику** |
|---|---|---|
| **Рівень 5 (Повне делегування)** | Команда самостійно визначає, як, коли і хто робить. Лідер лише затверджує цілі (наприклад, рефакторинг допоміжних модулів). | Знижує ризик вигорання лідера, підвищує швидкість прийняття рішень та довіру. |
| **Рівень 3 (Обговорення та спільне рішення)** | Лідер пропонує рішення, команда обговорює, пропонує альтернативи. Рішення приймається спільно (наприклад, Вибір нової бібліотеки для тестування). | Впроваджує принцип колективного володіння (Shared Ownership), запобігаючи залежності від однієї особи. |
| **Рівень 1 (Консультативне делегування)** | Лідер приймає рішення, але зобов'язаний консультуватися з виконавцями (розробниками, QA), чи можуть вони виконати завдання без перевантаження (наприклад, Визначення цільового результату спринта). | Боротьба з ризиком P1 оскільки графік розробника є частиною рішення, запобігаючи вигоранню. |

Практичне застосування: На плануванні спринта, делегування завдань

має відбуватися з урахуванням цілей розвитку, стимулюючи розробників із нижчим рівнем компетенції в певній області брати завдання під менторством Senior-фахівця (парне виконання/шедовінг).

Технічний борг (Technical Debt) – це відкладене рішення щодо якості, яке неминуче призводить до ризику Q2 (Проблеми з інтеграцією) та Q3 (Нестабільність системи). У ***** ця проблема вимагає постійної уваги та фінансового планування.[11, с. 11-12]

Недостатньо просто "пам'ятати" про технічний борг, його потрібно активно фінансувати та планувати. Пропонується внести системні зміни у планування спринта.

Таблиця 3.5

Способи зменшення технічного боргу

| **Механізм управління боргом** | **Опис / Умови застосування** | **Мета мінімізації ризику** |
|---|---|---|
| **Обов'язкова квота на технічний борг** | Мінімум 15% від загальної Velocity Спринта має бути зарезервовано під рефакторинг, оновлення бібліотек, оптимізацію продуктивності та покращення покриття автотестами. | Гарантія постійного погашення боргу, запобігання переростанню боргу вкритичний (Q2, Q3). |
| **Hardening sprints (Спринти зміцнення)** | Проводяться після кожного 5-го Спринта або перед великим релізом (1-тижневий фокус). Включає: закриття Open Bugs, масштабний рефакторинг критичних модулів, Penetration/Load Testing. | Підтвердження масштабованості та безпеки, створення "чистого"середовища перед випуском у продакшн |

Продовження табл 3.5

| | | |
|---|---|---|
| **Вимірювання боргу (Debt Measurement)** | Використання інструментів статичного аналізу коду (SonarQube, CodeClimate) для кількісної оцінки боргу (у людино-годинах або фінансовому еквіваленті). | Дозволяє Product Owner'у включати погашення боргу в Product Backlog. |

Якість має бути вбудована в процес від початку (принцип Shift-Left), що вимагає стратегічного зсуву в ролі QA.

Таблиця 3.6

Стратегічний зсув QA

| **Стратегічний зсув** | **Ключова діяльність** | **Бізнес-результат** |
|---|---|---|
| **Перехід до Shift-Left та BDD** | Участь QA в Refinement як партнера Product owner'a. Написання детальних Acceptance Criteria та сценаріїв BDD ("Як Користувач, Я Можу...") ще до написання коду. | Гарантує, що розробник отримує чітке, бізнес-орієнтоване розуміння перевірки PBI, зменшуючи потребу в переробках. |
| **Автоматизація як ключовий пріоритет** | Показник ефективності QA включає "відсоток збільшення покриття критичного функціоналу автотестами". Мета: досягти 100% автоматизації критичного шляху користувача (Happy Path). | Знижує ризик Q3 (Нестабільність системи) та пришвидшує майбутні релізи. |

Усі вищезазначені пропозиції створюють інтегровану трирівневу

систему захисту від типових ризиків IT-проєктів у *****. Ця модель забезпечує не просто "реакцію" на проблеми, а створення прогнозованого, стійкого та орієнтованого на якість середовища розробки.

Є три рівні захисту від типових ризиків розробки проєктів, цей захист допоможе менеджерам покращити якісь роботи команди а також підніме довіру клієнтів, цей захист можна застосовувати до будь-яких проєктів.(Див. Додаток А)

Реалізація цих ідей дасть змогу ***** дотримуватись Agile - одночасно будуючи переваги через стабільність усередині, якість продукту та відкриті процеси, саме вони найчастіше впливають на успіх.

### 3.2. Рекомендації щодо оптимізації процесів Scrum (спринти, ретроспективи, залучення стейкхолдерів)

У разі аутсорсингу IT-проектів стабільність і якість забезпечуються не просто дотриманням Scrum, а й постійним удосконаленням його елементів. Налаштування процесів має спрямовуватися на підвищення прозорості, контролю та гнучкості, водночас зменшуючи бюрократію й загрози від коливань вимог (W1) чи перевантаження команди (P1).

Ефективність спринту як основи циклу залежить не стільки від терміну, скільки від попередньої роботи (Refinement). Тому важливо уточнювати завдання. Це допомагає краще прогнозувати обсяг за тиждень і менше губитись у процесі.

Часто під час спринту виникають труднощі через слабке обговорення умов на початковому етапі - це трапляється, коли Refinement проводиться поверхнево й до роботи потрапляють PBI без DoR, що призводить до затримок і переробок. Натомість масштабного сесійного Refinement краще запровадити регулярну практику невеликих дозавантажень з чітко виділеними часовими рамками: Розробники мають приділяти 10 % свого тижневого навантаження саме на пропрацювання завдань. Такий ліміт стає захищеною частиною графіку, яка не передбачена для вирішення оперативних задач, однак може

бути задіяна лише для конкретизації потреб, складання первинних Acceptance Criteria або оцінювання часу на задачі. Головним напрямком Refinement є не просто комунікація, а доведення беклогу до стану готовності (DoR) принаймні на два майбутні спринти. Це забезпечує наявність у продукт овнера завжди підготовленого й структурованого беклогу. Під час цього процесу QA разом з розробниками мають проводити BDD-зустрічі - для формування точних сценаріїв типу "Given-When-Then". Такі сценарії потім використовуються як основа автоматизованих тестів, одночасно скорочуючи непевності. Відповідальність за регулярність Refinement тепер лежить на Scrum Master'і. Процедура має бути обмежена за часом та охоплювати задачі з достатньою деталізацією. У результаті - менший ризик W1 (Мінливі вимоги). Рання проробка дозволяє команді в спринті займатися кодингом, а не уточнюванням завдань. Отже, прогнозування швидкості роботи стає значно надійнішим.

Хоча Scrum не має ієрархії, у складних довгострокових проєктах варто щоспринта призначати Спринт-власника - ним стає досвідчений розробник чи QA. Такий підхід ґрунтується на постійній зміні ролей, через що знання краще розподіляються, а команда не залежить від одного фахівця. Під час спринта ця особа слідкує за тим, чи виконуються внутрішні Exit criteria для PBI, замість формального DoD. На Щоденному скоупі саме він координує обговорення перешкод, виступаючи як організатор процесу - подібно до фасилітатора, хоча це не скрам мастер. Важливо також забезпечити взаємодію між командою та PM/Scrum Master'ом. Наприклад, Спринт-власник перевіряє: чи пройшло Code Review старшим колегою, чи оновлено документацію в Confluence, і чи створено Pull Request у головну гілку. Це допомагає команді більше покладатись на себе, менше звертатись до Scrum Master'а з дрібницями, паралельно розвиваючи лідерські навички. Як результат - нижчий ризик критичної залежності від окремого спеціаліста.

Ретроспектива - це основа для покращень, проте нерідко перетворюється на звичайну розмову без чітких кроків, її варто зробити джерелом реальних змін. Аналіз KPI під час такої зустрічі допомагає командам

дивитись не на почуття, а на цифри, використовуючи прийоми типу «5 Чому», щоб знайти справжні причини труднощів. Кожна проблема за результатами показників потребує конкретного завдання у форматі SMART із вказаним відповідальним та очікуваними термінами. Такий елемент обов'язково потрапляє до майбутнього спринта як частина технічного боргу або операційного поліпшення. Лише тоди години на збиранні стануть реальною користю для організації.

Застосування цієї KPI-моделі допоможе ***** не лише проголосити себе Agile, а й показати реальну ефективність - завдяки конкретним бізнесовим підсумкам та відкритості перед керівництвом і замовниками.[35]

Кожна домовленість після ретро - має дотримуватись SMART-підходу. Завдання треба занести в беклог окремо. Пріоритет - «Вдосконалення процесу». Контроль за реалізацією покладено на продукт овнера.

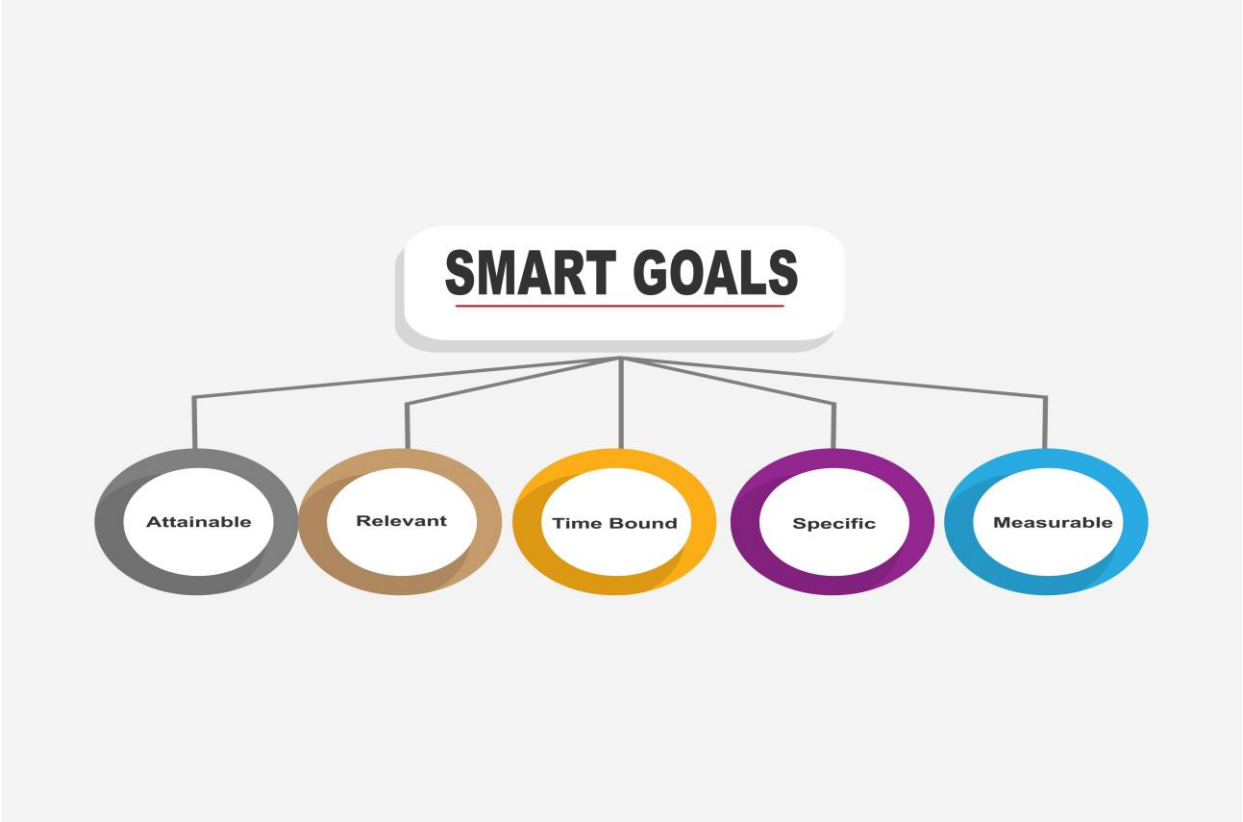


*Рис 3. Smart goals*

Оскільки потрібно відстежувати дії у роботі, кожне завдання типу PBI має мати чіткого власника - людину, яка за нього відповідає, і конкретний

строк закінчення, крім того, такі пункти слід позначати окремим тегом або виділяти в спеціальний блок на дошці (JIRA, Azure DevOps), наприклад «постійне вдосконалення», щоб важливі справи не загубилися серед решти функціоналу. Замість простого переліку, Scrum master повинен починати Ретроспективу з невеликого оновлення статусу минулих двох спринтів - це допомагає тримати всіх у курсі й підвищує особисту відповідальність. Щоб побачити реальні приклади, можна взяти SMART-завдання такого роду: запустити коротке 5-хвилинне демо кожного дня, починаючи з нового спринта (строковий елемент), де кожен показує свою частину результату, мета - скоротити баги через спільні правки мінімум на 10% (вимірювана частина).

Кожні три місяці - чи одразу після важливого етапу проєкту, наприклад перед бета-версією - варто провести аналіз самої ретроспективи. Мета - перевірити, наскільки корисним був процес покращень. Замість цього команда дивиться: чи справді минулі заходи поліпшили швидкість роботи, якість коду або загальний клімат у групі. На такому зборі оцінюють середню продуктивність до й після реалізації спринта. Також порівнюють час, витрачений на правку помилок, особливо після нових правил тестування. Додатково враховують частоту успішного закриття спринтів. До обговорення ставлять питання: «Які три ініціативи принесли найбільший результат останнім часом?»? Що ми зробили, але це не допомогло - і з якої причини? Такий аналіз дає змогу уникнути сліпої віри в процеси й забезпечує те, щоб команда не гаяла час на марні дії, замість цього концентруючись на тих кроках, що реально працюють.

Участь зацікавлених осіб - зокрема в аутсорсингу - потребує поєднання чіткої комунікації й обережності, щоб не завадити роботі команди.

Sprint Review має бути не просто показом результатів - це час для аналізу та коригування напряму разом із замовниками. Замість звичайної демонстрації, команді варто фокусуватися на конкретному запиті: чи спринт справді досягає мети? Керівник продукту має бути присутнім, адже без нього обговорення марне. У процесі слід чітко прописати, як знайдені дані змінять

план на майбутнє - наприклад, зміна напрямку розробки з «X» на «Y», бо нові результати підтвердили перевагу останнього. Важливо також продемонструвати кілька ключових показників стану команди: стабільні темпи, спад технічного боргу або частота серйозних помилок. Їх треба пояснювати простими словами: "Робочий ритм тримається - терміни реальні", щоб усі розуміли значення.

Щоб зменшити ризики W3 (неочікувані вимоги клієнта) та P3 (невизначені ролі), потрібно чітко прописати - хто, коли й як отримує дані. Замість постійного дублювання інформації, Product owner має регулярно узгоджувати деталі під час щоденного або щотижневого синхронізаційного заходу(мітингу), там же обговорюються перешкоди й критерії приймання задач - це частина Daily Standup чи Refinement, де головну роль грає PM або Product Owner. Клієнтський менеджмент (Sponsor) бере участь раз на спринт у огляді результатів - Sprint Review, мета якого: демонстрація прогресу, коригування стратегічних напрямів беклогу, затвердження виділення коштів на технічний борг. У свою чергу, внутрішній менеджмент ***** отримує оновлення стану проєкту кожні два спринти через Status Report від Project Manager’а для контролю завантаженості команди, фінансових показників і аналізу актуальних ризиків за допомогою теплової карти. Наразі команди, що взаємодіють між собою, збираються раз на тиждень - це Coordination meeting. Під час таких сесій Scrum Master чи Tech Lead узгоджують технічні питання, наприклад, оновлення API, а також готують інтеграційні спринти. Такий підхід допомагає тримати всі обговорення конкретними й доцільними. В результаті команда не витрачає час марно через зайві зустрічі і завжди має актуальні дані про процес.

### 3.3. Розробка моделі показників ефективності (KPI) IT-проєктів в Agile-середовищі

Добре KPI у ***** на базі Agile має бути не лише про час. Важливо дивитись на продуктивність та цінність окремо. Один погляд - для

внутрішнього контролю роботи команд. Інший - щоб клієнти чітко бачили, як команда впливає на їхні цілі.

Ці метрики використовуються Project Manager (PM), Scrum Master (SM) та командою для інспекції та адаптації внутрішніх процесів (розділ 3.2). Вони відображають прогнозованість, якість виконання та ефективність розробки.

Таблиця 3.7

Метрики прогнозованості та якості

| № | KPI (Метрика) | Бізнес-Ціль та Порогове значення | Відповідальний |
|---|---|---|---|
| 1 | **Стабільність Velocity** | Підвищення прогнозованості доставки (Predictability) для клієнта. **Цільовий діапазон: 0.0 - 0.4** (нижче – краще). | Scrum Master |
| 2 | **Успішність Спринт-цілі** | Гарантія фокусу команди на ключовій цінності спринта та захист від переривань. | Product Owner/Scrum Master |
| 3 | **Час, витрачений на переробку (Rework)** | Відображає якість коду та стабільність вимог. **Високий показник (>15%)** сигналізує про проблеми з DoD. | Tech Lead / QA Lead |
| 4 | **Загальний час виконання (Lead Time)** | Оцінка загальної гнучкості та пропускної спроможності всього процесу доставки (від ідеї до Production). | Project Manager |

На цій таблиці ми бачимо що які метрики повинні використовувати проджект менеджер та скрам мастер щоб якісно керувати проєктом та тримати все під контролем. Часто що відповідальний по деяким з цих показників змінюється через те що не кожна команда має такі ролі як Scrum Master або

інші, в цьому випадку за метрики скрам мастера відповідає хтось інший, зазвичай це падає на плечі проджект менеджера.

Ці метрики використовуються для зовнішньої звітності клієнту та стратегічного планування в *****. Вони відповідають на питання: "Чи приносить наша робота бізнес-цінність?"

Таблиця 3.8

Метрики цінності та задоволеності клієнта

| № | Метрика (KPI) | Основна мета (Ризик) | Відповідальний |
|---|---|---|---|
| **1** | **Показник окупності цінності** | Оцінка цінності функціоналу (Ризик W2: Низька бізнес-цінність). | Product Owner |
| **2** | **Рівень задоволеності клієнта** | Управління очікуваннями та співпрацею (Ризик W3: Розбіжність очікувань). | Project/Account Manager |
| **3** | **Коефіцієнт технічного боргу** | Забезпечення стійкості та швидкості розробки (Ризик P2: Накопичення тех. боргу). | Tech Lead / PM |
| **4** | **Середній час відновлення** | Відображення стійкості системи після збою. | DevOps Team / Tech Lead |

Щоб працювало добре - не вистачить просто збирати метрики (KPI). Треба їх досліджувати, а потім діяти на основі цих даних. Інакше ефекту майже не буде. Насправді важливо, щоб команда навчилася робити висновки з інформації. Це значно підвищує шанси на успішний Agile.

Потрібен єдиний автоматичний дашборд - наприклад, у Power BI чи JIRA Advanced Roadmaps. Він має показувати KPI на двох рівнях доступу. Перший варіант - «Внутрішній» - для менеджерів, скрам-мастерів та команди. Тут видно детальні дані: швидкість роботи, повторну працю, час обробки завдань. Цей погляд допомагає готувати конкретні кроки під час ретроспектив.

Учасники бачать неочищені цифри. Другий тип - «Клієнтський» - створено для замовників і спонсорів. Показує довгострокові результати: користь від продукту, задоволеність клієнта, терміни реалізації. Таку версію використовують на оглядах спринтів, аби зрозуміти майбутній курс. Інформація подана загальна, без зайвих подробиць.

Пропонують запустити систему сигналізації відхилень - якщо є проблема, про це одразу видно. Так аналіз даних займає менше часу

- Зелений - це коли метрика потрапляє в очікувані межі.
- Жовтий колір означає невелике відхилення - наприклад, Velocity Consistency більше за 0.3, це слід врахувати під час контролю й додати до розділу «Дії» на Ретроспективі.
- Червоний означає велике відхилення - наприклад, швидкість скоротилася більше ніж на 20%, або обсяг переділки перевищив позначку у 15%. У такому разі потрібні термінові дії: автоматичне повідомлення PM чи керівникам та негайна увага до проблеми.

Аналіз KPI має щотижнево й щомісячно входити до управління проєктами - так дані перетворюються на дії. Під час Sprint review (3.2.3) Product owner разом з PM показують Стратегічні KPI, аби узгодити беклог із цінністю для бізнесу та пояснити клієнту пріоритет PBI. На ретроспективі (3.2.2), завдяки Операційним KPI, скрам мастер і команда шукають кореневі причини проблем і створюють конкретні кроки. Наприклад, якщо KPI "Rework" у "червоному" стані - виникає запитання: «Чому довелося переробляти роботу?», далі аналізується причина - слабкий DoR, поспіх або поверховий code review.

Застосування цієї KPI-моделі допоможе не лише оголосити себе Agile, а й показати реальну ефективність завдяки конкретним бізнесовим досягненням, що забезпечить чіткість перед керівництвом і замовниками.

## 3.4. Очікувані результати від впровадження запропонованих заходів та їх економічне обґрунтування

Застосування цілісного пакету кроків - з фокусом на стандартизації Scrum (DoD/DoR), поліпшенні його дії та введенні дворівневої KPI-моделі - стає стратегічним рухом із чіткою метою отримати економічний прибуток разом із замовниками. Ці дії мінятять основу управління: тепер важливими стають не емоції, а факти, що особливо потрібне для довготривалого росту у сфері аутсорсингу. Результати очікуються як числові - наприклад, економія коштів, так і нечислові - зокрема, посилення надійності бізнес-процесів.[17 с. 50-51]

Прямий економічний результат забезпечується кращою роботою команди, меншими втратами часу й точнішим плануванням - це допомагає утримувати клієнтів.

Найбільші втрати трапляються, коли доводиться усувати помилки після завершення етапу - або передивитися роботу через невизначені критерії. Застосування чітких правил готовності задач і їхнього закриття на стадіях перевірки коду зменшує такі накладки значно швидше.

Дуже важливо слідкувати за часом який йде на переробку завдань, адже якщо цей час великий це означає що проблема глибше ніж здається, можливо вимоги до завдань поставлені не коректно або часто міняються вимоги, тому час на переробку завдань повинен відповідати певним нормам.

Час на переробку одна з найважливіших метрик, взагалі переробка завдань не повинна відбуватись але якщо таке стається то це немає стати постійною частиною розорбки, якщо так стається значить РО можливо не надають коректні вимоги до спринтів або задач, також можливі інші причини такі як вигорання команди або просто кваліфікованість працівників.

Таблиця 3.9

Корегування часу на переробку

| **Показник** | **Початковий стан (Припущений)** | **Цільовий показник (Прогноз)** | **Економічний вплив** |
|---|---|---|---|
| **Час на переробку (Rework)** | 15% від загального часу розробки | Зниження до **< 8%** | **Звільнення 7%** часу команди для створення нової, оплачуваної бізнес-цінності. |
| Економічна вигода: При середній команді з 8 фахівців (завантаження 100%), зниження Rework на 7% вивільняє еквівалент понад 0.5 повних робочих одиниць (FTE). Це близько 11 робочих днів на місяць, які тепер можуть бути спрямовані на прискорення розробки нових функцій або інвестиції у технічний борг. Це безпосередньо підвищує маржинальність проєкту, оскільки клієнт отримує більше цінності за ту ж ціну. | | | |

Впровадження Definition of Ready (DoR) та практики Continuous Refinement гарантує, що команда бере в роботу лише готові завдання, а також мінімізує зовнішні переривання. Це прямо впливає на стабільність Velocity та здатність надавати клієнту точніші прогнози релізів.(Див. Додаток Б)

Постійне інвестування часу в технічний борг, приблизно на рівні 10–15% Backlog, разом із наголосом на MTTR (середній час відновлення), підвищують стабільність системи й прискорюють реакцію на неполадки. Такий підхід ефективно запобігає простоям: скорочення MTTR з 3 до 1 години за умови вартості простою $5000/год дає економію $10 000 на кожному серйозному збої. Крім фінансової вигоди, це також зберігає довіру клієнтів та

авторитет *****. У той самий час, сталий контроль над технічним боргом дозволяє уникнути масштабних, коштовних переробок старого коду, що особливо важливо для проєктів із довгою життєвою крапкою.

Сильні результати допомагають ***** рости й залишатися на крок попереду, бо впливають на команду та здатність швидко адаптуватись.

Покращення задоволеності клієнтів (CSAT/NPS): дворівневий KPI-дашборд дає прозорість - замість Story Points клієнт бачить реальні бізнес-показники, що разом із стабільною швидкістю роботи команди зменшує невідповідності між очікуваннями та результатами. Цей ефект безпосередньо підвищує довіру клієнтів, адже вони частіше купують додаткові послуги й рекомендують компанію далі, при цьому обслуговування наявних партнерів коштує значно менше, ніж пошук нових.

Кращий настрій у колективі та менше звільнень: ліквідація нерозберихи й перевантаження (ризик P1) через чіткі процедури - наприклад, стандартизований Refinement або передача обов’язків (наявність ролі "Спринт-Власника") - призводить до зниження стресу під час роботи. Це веде до більшого задоволення працею, про що свідчать результати внутрішнього опитування eNPS. Коли співробітники залишаються довше, компанія економить на пошуку нових кадрів і адаптації, не втрачаючи важливі знання. Команди без частих змін у складі працюють продуктивніше, особливо якщо між ними високий рівень взаєморозуміння.

Масштабування та стандарти: оформлення процесів через єдині DoD і DoR, шаблони Ретроспектив, а також SMART дії - це прискорює створення нових команд. Завдяки уніфікованим і чітко описаним процедурам, новачки швидше включаються до роботи. Гільдії всередині компанії забезпечують постійний обмін знаннями й допомагають швидко запускати нововведення. Завдяки цьому організація гнучко реагує на зміни технологічного середовища, не переробляючи підходи окремо для кожного проєкту.

Отже, реалізація запропонованих кроків робить невидиму працю видимою цінністю - економічний прибуток зростає за рахунок внутрішнього

підвищення продуктивності, водночас ***** посилює свої позиції на ринку через більшу довіру клієнтів, відкритість і задоволеність їх потреб.

# ВИСНОВКИ

Вивчення використання Scrum у IT-компанії ***** показало що успішне керування проєктами залежить від чітких правил та єдиних підходів. Замість хаотичного, спонтанного застосування методу потрібен системний і послідовний підхід - це важливо для стабільного росту бізнесу й покращення якості продукту. Дослідження довело: основні проблеми - стрибки швидкості розробки команди, невизначені строки завершення задач і надмірна кількість перероблених елементів - не через слабку компетентність фахівців, а тому що User story потрапляють до спринта неготовими, мають сумнівні формулювання чи безліч прихованих технічних обмежень. Основним рішенням, що запобігає безладу, стало суворе дотримання стандартизованого Definition of ready (DoR). Впровадження чітких критеріїв DoR - наприклад, підтвердженої вигоди для бізнесу (з цільовими показниками та очікуваним прибутком), повного технічного опису (угодженими рішеннями, ясною складністю, без блокувальних факторів) і готовності до перевірок (наявність деталізованих умов приймання, що враховують всі граничні ситуації) - перетворює сесію Backlog refinement не на просте обговорення, а на реальний етап контролю якості. Цей метод не лише економить час команди, витрачений через зриви й правки помилок по ходу спринта, натомість повертає його до систематичної передачі корисних функцій. DoR діє як сортувальник: стримує потік зайвих завдань у беклог, стабілізує швидкість роботи, також полегшує прогнозування термінів. Саме така стабільність допомагає забезпечити фінансову надійність проєкту.

Логіка кроків показує зріст прибутковості - не тільки через економію всередині, а завдяки стратегічній цінності для замовника. Жорсткі стандарти на початку (DoR) і в кінці (Done/Exit) майже ліквідують переробки, адже помилки знаходять раніше, ще до старту робіт. Кращі оцінки разом з чіткою роботою команд дозволять ***** точніше обчислювати строки й скорочувати збитки від затримок. Показник MTTR падає з 3 до 1 години; якщо простій коштує 5000$/год, то кожен випадок дає економію у 10 000$, що виділяє

компанію серед конкурентів. Швидший запуск продукту зменшує фінансові ризики клієнта й одночасно підвищує довіру до неї як до надійного партнера. Така стійка репутація дасть змогу укладати контракти з кращою маржею й тримати клієнтів довше. З іншого боку, запуск дворівневої системи KPI - де окремо оцінюються стратегічні показники (типу Value Realization Rate, Project CSAT чи MTTR) та ефективність команд у повсякденних процесах (наприклад, Rework rate, час виконання задач, сталість продуктивності) - забезпечить ясний погляд на дані без зайвої складності. Завдяки цьому менеджмент матиме реальну основу для рішень замість припущень або суперечок на емоціях; ще й спілкування з замовниками стане точнішим через прозорі метрики прогресу, пов’язані з їхніми справжніми потребами. Надалі, коли внутрішні процеси будуть надійними, прогнозованими, а продукт - якісним, загальна атмосфера в компанії поліпшиться: команда почуватиметься самостійнішою, навантаження скоротиться, результативність зросте - все це підвищить задоволеність серед персоналу (eNPS), знизить текучість кадрів і допоможе заощадити на рекрутингу, адаптації нових співробітників та передачі знань всередині. Стабільні команди, своєю чергою, автоматично працюють продуктивніше.

Отже, запропонований план змін охоплює не лише структурні, а й культурні сторони роботи за Scrum - це крок у майбутню сталість організації. Він передбачає покращення ключових механізмів: DoR для Refinement та чіткі Exit criteria на завершення етапів; паралельно вводиться підтримка постійного прогресу через SMART-ретроспективи. Ці сесії спираються на KPI, щоб знаходити причини проблем, а не реагувати на їх прояви. Ретроспектива перестає бути просто обговоренням - тепер це формальне прийняття конкретних дій. Контроль технічного боргу забезпечує виділення фіксованої частини часу (10–15%) кожного спринта на технічні задачі. Це допоможе уникнути збільшення вартості підтримки продукту і скорочує потребу в масштабних, дорогих переробках. Застосування цього підходу разом із коучингом, спрямованим на аналіз впливу дефектів (для QA-інженерів) та

чітких кількісних критеріїв завершення (для керівництва), стає основою змін. Коли такі дії будуть системними й увійдуть до повсякденних процесів, ***** перетвориться на організацію з високим рівнем операційної зрілості, де управління проєктами ґрунтується на реальних даних. Це дасть змогу не просто формально слідувати Scrum’у, а використовувати його для отримання конкретної користі, яку можна виміряти, забезпечуючи стале покращення результатів і сильнішу позицію на конкурентному IT-ринку. Головне це успішність проєкту оцінюватиметься не за обсягом закритих задач, а за тим, наскільки скорочено ризики та збільшено цінність для клієнта.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Agile Manifesto. Manifesto for Agile Software Development. URL: https://agilemanifesto.org / (дата звернення: 05.07.2025).
2. Beck K. Extreme Programming Explained: Embrace Change. 2nd ed. Boston, MA : Addison-Wesley Professional, 2004. 256 p.
3. Березін О. В., Дуда С. Т., Міценко Н. Г. Управління потенціалом підприємства : навч. посіб. Львів : Магнолія, 2011. 205 с.
4. Бурєннікова Н. В. Економіка підприємства – оновлені підходи до бенчмаркінгу в контексті моделювання результативності діяльності та багатовимірного ранжування. Бізнес Інформ. 2022. № 3. URL: http://nbuv.gov.ua/UJRN/binf_2022_3_12 (дата звернення: 21.07.2025).
5. Бутенко І. А., Курносова А. В. Методичні підходи до оцінки ефективності системи управління персоналом підприємства. Економічні інновації. 2015. С. 56–61.
6. Гордієнко Н. І., Ілляшенко О. В., Литовченко О. Ю. Організація та обліково-аналітичне забезпечення управління результативністю діяльності підприємства. Підприємництво та інновації. 2020. № 13. С. 14–18. URL: http://nbuv.gov.ua/UJRN/pidinnov_2020_13_6 (дата звернення: 30.07.2025).
7. Григорук П. М., Завгородня Т. П., Григорук С. С. Оцінювання результативності діяльності компанії. Моделювання та інформаційні системи в економіці. 2023. № 96. С. 56–66.
8. Drucker P. The Practice of Management. New York, NY : Harper Business, 2023. 416 p.
9. Друкер П. Ф. Практика менеджменту : монографія. Москва : Вільямс, 2023. 400 с.
10. Дудар Т. Г., Волошин Р. В. Основи менеджменту : навч. посіб. для студ. вищ. навч. закл. Тернопіль : Економічна думка, 2013. 350 с.
11. Kerzner H. Project Management: A Systems Approach to Planning, Scheduling, and Controlling. Hoboken, NJ : Wiley, 2022. 1250 p.

12. Cohn M. Succeeding with Agile: Software Development Using Scrum. Boston, MA : Addison-Wesley Professional, 2020. 250 p.
13. Cohn M. User Stories Applied: For Agile Software Development. Boston, MA : Addison-Wesley Professional, 2024. 300 p.
14. Матвіїшин Є. Г., Миляник Р. В. Оцінювання розвиненості управлінських компетентностей через призму результативності діяльності організації. Ефективність державного управління. 2022. № 1–2. С. 73–75. URL: http://nbuv.gov.ua/UJRN/efdu_2022_1-2_1 2 (дата звернення: 21.08.2025).
15. Назарчук Т. В., Косіюк О. М. Менеджмент організацій : навч. посіб. Київ : Центр учбової літератури, 2022. 560 с. URL: https://pdf.lib.vntu.edu.ua/books/2017/menedzhment_org.pdf (дата звернення: 09.08.2025).
16. Офіційний сайт компанії *****. URL: https://www.*****.dk/en (дата звернення: 05.08.2025).
17. Палеха Ю. І., Мошек Г. Основи менеджменту теорія і практика : навч. посіб. Київ : Ліра-К, 2023. 528 с. URL: https://lira-k.com.ua/preview/12290.pdf?srsltid=AfmBOoot6aqF_l4ezpqNh1-hbMRGTS_pwCeWUSudqdhA93BcgqbWGFtI (дата звернення: 01.08.2025).
18. Пол Дж. Філдінг. Як керувати проєктами. Харків : Фабула, 2023. 280 с.
19. Robbins S. P., Coulter M. Management. 15th edition. Hoboken, NJ : Pearson, 2021. 600 p.
20. Schwaber K. Agile Software Development with Scrum. Upper Saddle River, NJ : Prentice Hall, 2002. 180 p.
21. Schwaber K., Sutherland J. The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game. Scrum.org & Scrum Inc., 2020. URL: https://scrumguides.org/ (дата звернення: 25.07.2025).
22. Sutherland J. Scrum: The Art of Doing Twice the Work in Half the Time. New York, NY : Currency, 2024. 288 p.

23. Про невідкладні заходи щодо забезпечення функціонування та розвитку освіти в Україні : Указ Президента України від 04.07.2005 р. № 1013/2005. Офіційний вісник України. 2005. № 27. С. 1–5.
24. Харченко С. В. Управлінські аспекти забезпечення результативності використання потенціалу підприємства. Актуальні проблеми економіки. 2009. № 1 (89). С. 136–141.
25. Highsmith J. Agile Software Development Ecosystems. Boston, MA : Addison-Wesley Professional, 2022. 240 p.
26. Шершньова З. Є. Стратегічне управління : підручник. Київ : КНЕУ, 2004. 699 с.
27. Антонюк І. В., Мороз О. І. Вплив цифрової трансформації на систему управління інноваційним розвитком підприємства. Економіка та суспільство. 2023. № 51. URL: https://economyandsociety.in.ua/index.php/journal/article/view/2800 (дата звернення: 18.08.2025).
28. Ващенко В. В., Шматков Д. В. Роль бізнес-аналітики в управлінні проєктами: сучасні тенденції. Науковий вісник Ужгородського університету. Серія: Економіка. 2022. Вип. 1 (59). С. 138–144.
29. Гаврилко П. П., Олексієнко Л. А., Приходько В. С. Сучасні моделі та методи оцінювання ефективності управління ІТ-проєктами. Вісник Хмельницького національного університету. Економічні науки. 2022. № 4. С. 182–187.
30. Крикавський Є. В., Нагірна О. М. Управління ланцюгами постачання в умовах невизначеності: підхід на основі Agile-методологій. Логістика. 2023. № 1 (13). С. 15–20.
31. Лисенко М. В., Коваленко О. І. Цифровізація управління людськими ресурсами як фактор підвищення результативності організації. Вісник Київського національного університету імені Тараса Шевченка. Економіка. 2024. № 1 (226). С. 45–51.

32. Маслова О. В., Петренко І. С. Особливості застосування Kanban-методу в не-IT сферах діяльності. Економічний вісник Запорізької політехніки. 2022. № 4 (18). С. 60–67.
33. Музиченко О. Г., Ткаченко Р. Д. Впровадження OKR (Objectives and Key Results) як інструменту стратегічного управління ефективністю. Проблеми економіки. 2023. № 1 (55). С. 220–225. URL: http://nbuv.gov.ua/UJRN/Pekon_2023_1_32 (дата звернення: 10.08.2025).
34. Скопенко О. В., Даниленко С. В. Управління ризиками в Agile-проєктах: світовий досвід та українські реалії. Вісник економічної науки України. 2022. № 1 (42). С. 136–141.
35. Смішко О. В., Полякова І. М. Методологія Lean у підвищенні операційної ефективності підприємств. Ефективна економіка. 2023. № 11. URL: http://www.economy.nayka.com.ua/?op=1&z=10672 (дата звернення: 11.08.2025).
36. Черній В. А., Семенов Г. А. Гнучкість організаційної структури підприємства в умовах швидких ринкових змін. Економіка: реалії часу. 2024. № 2 (72). С. 58–64.
37. Shneiderman B., Plaisant C., Cohen M., Jacobs S., Elmqvist N., Diakopoulos N. Designing the User Interface: Strategies for Effective Human-Computer Interaction. 6th ed. Boston, MA : Pearson, 2023. 800 p.
38. Sutherland J., Schwaber K. The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game. Scrum.org & Scrum Inc., 2020.
39. Larman C., Basili V. R. Agile and Iterative Development: A Manager’s Guide. Boston, MA : Addison-Wesley Professional, 2023. 250 p.
40. Kotter J. P. Leading Change. Boston, MA : Harvard Business Review Press, 2022. 208 p.
41. Mintzberg H. Managing. Oakland, CA : Berrett-Koehler Publishers, 2023. 288 p.

# ДОДАТКИ

Додаток А

Рівні захисту проєктів

| **Рівень захисту** | **Основний ризик, що мінімізується** | **Інструмент Scrum/Agile** | **Як імплементуються пропозиції ***** (Конкретні дії)** |
|---|---|---|---|
| **1. Превентивний (Планування)** | Ризики скоупу та визначення (W1, W3) | Definition of Ready (DoR) | Суворе включення бізнес-цінності та Exit Criteria у DoR. QA-інженер, навчений бізнес-пріоритетам, має право вето. |
| **2. Внутрішній (Виконання)** | Ризики людські та якості (P1, Q1) | Делегування, DoD | Матриця Компетенцій для збалансованого делегування. 15% квота на Технічний борг у кожному Спринті. |
| **3. Контрольний (Реліз)** | Ризики технічного боргу та бізнес-цінності (Q2, W2) | Exit criteria | Менеджерські Exit criteria (наприклад, Defect density < 0.1) використовуються як обов'язкові умови для випуску продукту. Hardening Sprints перед релізом. |

Додаток Б

Прогнозування економічного впливу

| **Показник** | **Початковий стан (Припущений)** | **Цільовий показник (Прогноз)** | **Економічний Вплив** |
|---|---|---|---|
| **Стабільність Velocity (Consistency)** | Коефіцієнт варіації > 0.4 (висока нестабільність) | Зниження до **0.20–0.25** | Покращення прогнозованості термінів доставки. |
| **Lead time** (Час від ідеї до доставки) | Високий, через простої та блокери | Зниження на **~20%** | **Зниження Cost of Delay (Ціна зволікання).** Прискорення виходу продукту на ринок (Time to Market) генерує дохід для клієнта раніше. Якщо продукт починає генерувати прибуток на один тиждень раніше, фінансовий ефект для клієнта може вимірюватися десятками тисяч доларів, що зміцнює репутацію ***** як ефективного та надійного партнера. |