Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Порівняльний аналіз гнучких та традиційних підходів до розробки програмного забезпечення для розподілених команд

Виконав(ла): студент(ка) 6 курсу, групи СНм-61
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

		Базюк Н.М.
	(підпис)	(прізвище та ініціали)
Керівник		Боднарчук І.О.
	(підпис)	(прізвище та ініціали)
Нормоконтроль		Мацюк О.В.
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		Боднарчук І.О.
	(підпис)	(прізвище та ініціали)
Рецензент		Карпінський М.П.
	(підпис)	(прізвище та ініціали)

Тернопіль
2021

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«21» вересня 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

студенту Базюк Назарій Миколайович
(прізвище, ім'я, по батькові)

1. Тема роботи **Порівняльний аналіз гнучких та традиційних підходів до розробки програмного забезпечення для розподілених команд**

Керівник роботи **к.т.н., доц. Боднарчук І.О.**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*28*» *жовтня* 2021 року № *4/7-908*

2. Термін подання студентом завершеної роботи 20 грудня 2021 р.

3. Вихідні дані до роботи Літературні джерела з тематики роботи

4. Зміст роботи (перелік питань, які потрібно розробити)
ВСТУП; 1 ТЕОРЕТИЧНІ ОСНОВИ МЕТОДИКИ ПОРІВНЯННЯ ПІДХОДІВ ДО УПРАВЛІННЯ ПРОЕКТАМИ; 1.1 Традиційне управління проектами; 1.2 Огляд водоспадних моделей розробки і швидке створення прототипів; 1.3 Agile Project Management; 1.4 Scrum; 1.5 Діяльність Scrum; 1.6 Управління проектами Kanban; 1.7 Бережливе управління проектами (Lean); 1.8 Теорія обмежень (Theory of Constraints – TOC); 2 ГНУЧКІСТЬ У ПОРІВНЯННІ З ТРАДИЦІЙНИМ УПРАВЛІННЯМ ПРОЕКТАМИ; 3 ТЕОРЕТИЧНЕ ОБґРУНТУВАННЯ ДОСЛІДЖЕННЯ; 3.1 Результати дослідження застосування гібридних підходів в софтверних компаніях; 3.2 Архітектурне проектування, як процес покращення координації команд, та забезпечення якості програмної продукції в глобальних проектах; 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ; 4.1 Покращення безпеки праці в українських IT-компаніях, як результат їх взаємодії з західними компаніями; 4.2 Вплив електромагнітного імпульсу (ЕМІ) ядерного вибуху на елементи виробництва та заходи захисту; ВИСНОВКИ; ПИСОК ВИКОРСИТАНИХ ДЖЕРЕЛ; ДОДАТКИ

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Дмитроца Л.П., доц.		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викл.		

7. Дата видачі завдання 21 вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	21.09.21-27.09.21	*Виконано*
2.	Підбір наукових джерел по темі роботи	28.09.21-04.10.21	*Виконано*
3.	Переклад та опрацювання наукових джерел	05.10.21-11.10.21	*Виконано*
	по темі кваліфікаційної роботи		
4.	Виконання дослідження щодо огляду атак на	12.10.21-18.10.21	*Виконано*
	комп'ютерні системи		
5.	Оформлення першого розділу	19.10.21-25.10.21	*Виконано*
6.	Оформлення другого розділу	26.10.21-01.11.21	*Виконано*
7.	Оформлення третого розділу	02.11.21-08.11.21	*Виконано*
8.	Виконання завдання до підрозділу «Охорона праці»	09.11.21-15.11.21	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в	16.11.21-22.11.21	*Виконано*
	надзвичайних ситуаціях»		
10.	Оформлення кваліфікаційної роботи	23.11.21-29.11.21	*Виконано*
11.	Нормоконтроль	30.11.21-05.12.21	*Виконано*
12.	Перевірка на плагіат	05.12.21	*Виконано*
13.	Попередній захист кваліфікаційної роботи	14.12.21	*Виконано*
14.	Захист кваліфікаційної роботи	20.12.2121	

Студент _____ Базюк Н.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

"Порівняльний аналіз гнучких та традиційних підходів до розробки програмного забезпечення для розподілених команд" // Базюк Назарій Миколайович // Тернопільський національний технічний університет ім. Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНм-61 // Тернопіль, 2021 // с. – 68, рис. – 13, табл. – 5, джерел – 33.

Ключові слова: РОЗРОБКА ПЗ, AGILE, УПРАВЛІННЯ ПРОЄКТОМ, ГНУЧКА РОЗРОБКА, ПОРІВНЯННЯ.

Це дослідження показало, що гібридне управління проектами не зовсім схоже ні на традиційне управління проектами, ні на гнучке (швидке, Agile) управління проектами. Як результат дослідження отримано керівні принципи для вибору ситуацій, які заслуговують гібридного управління проектами. Дослідження також приносить користь практикам, оскільки воно орієнтує керівників проектів, які не впевнені, коли і де використовувати якусь парадигму управління. Одним із результатів є можливість для керівників проектів краще зрозуміти, чому і коли слід вибирати традиційні підходи, а також чому і коли слід віддавати перевагу гнучким або гібридним підходам до управління проектами.

ANNOTATION

"Comparative analysis of agile and traditional approaches to software development for distributed teams " // Nazarii Baziuk // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Computer Science Department, group СНмз-61 // Ternopil, 2021 // p. – 68, fig. – 13, tables. – 5, ref. – 33.

Keywords: SOFTWARE DEVELOPMENT, AGILE, PROJECT MANAGEMENT, FLEXIBLE DEVELOPMENT, COMPARISON

This study showed that hybrid project management is not quite similar to traditional project management or flexible (fast, Agile) project management. The study resulted in guidelines for selecting situations that deserve hybrid project management. The study also benefits practitioners because it guides project managers who are unsure of when and where to use a management paradigm. One of the results is an opportunity for project managers to better understand why and when traditional approaches should be chosen, as well as why and when flexible or hybrid approaches to project management should be preferred.

ЗМІСТ

ВСТУП

Актуальність теми.

Гнучкі технології мають ряд переваг перед традиційними у забезпеченні постійного зв'язку з замовником для врахування змін вимог, прискорення поставок готового продукту і успішно використовуються при розробці невеликих проектів 1-2 командами. Але в останні роки вони стали широко використовуватись для розробки великих програмних проектів багатьма командами які розташовуються в різних країнах. Ця технологія отримала назву глобальної розробки, і при її застосуванні виявились серйозні проблеми, пов'язані з координацією територіально розподілених команд розробників, та забезпеченням якості програмних продуктів [1]. Для вирішення виникаючих проблем були розроблені спеціальні платформи такі як SAFE, SofS, IGM,LESS та інші [2], а також були адаптовані самі гнучкі технології. Ця адаптація в основному полягала у введенні нових ролей в командах, таких як архітектор бізнес процесів та архітектор системи, а також була створена координуюча команда. Це фактично привело до включення елементів традиційних планових підходів в гнучкі методи. Подальша еволюція цих змін привела до створення підходів до розробки великих програмних проектів, які отримали назву гібридні.

Гібридний підхід до розробки ПЗ - це будь-яке поєднання гнучких та традиційних (орієнтованих на план) підходів, які організаційний підрозділ використовує та підлаштовує під свої власні контекстні потреби [3]. Іншими словами, гібридні підходи до розробки ПЗ охоплюють різні елементи керованих планами методів (наприклад, водоспад або V-модель) та гнучких методів (наприклад, Scrum або Kanban).

Таким чином, організації реалізують специфічний гібридний підхід вибираючи як гнучкі, так і керовані планами елементи для поєднання корисних елементів.

Проведені дослідження показують, що поєднання традиційних і гнучких практик є найкращим способом управління проектом, оскільки методи повинні бути адаптовані до потреб робочого контексту протягом життєвого циклу проекту, і хоча розроблено багато платформ для застосування гнучких технологій, вони мають ряд суттєвих обмежень [2].

Мета роботи. Основне питання дослідження, яке буде розглянуто в цій роботі, стосується виявлення відмінностей в критичних факторах успіху, пов'язаних із традиційним, гнучким та гібридним управлінням проектами.

Тому були поставлені такі задачі:

– Якою мірою використовуються традиційні та гнучкі методи управління проектами.

– Як ці підходи поєднуються.

– Які існують проблеми в управлінні проектами за допомогою гнучких і традиційних підходів до управління проектами.

Проведене дослідження показує, чи є суттєва різниця у критичних факторах успіху (Critical Success Factors – CSF) між традиційним, гнучким та гібридним підходами до управління проектами.

Використовуючи визначений список CSF, виконується порівняння традиційних, гнучких та гібридних підходів до управління проектами.

Об'єкт дослідження: процеси управління проектів з розробки програмного забезпечення

Предмет дослідження: моделі управління проєктами з розробки програмного забезпеченя.

Практична цінність роботи.

Для практичного застосування це актуально, оскільки в нинішньому управлінні проектами підходи традиційного та гнучкого управління хоча самі по собі – хороші, але часто використовуються ізольовано, а ідеологічні суперечки між двома таборами здаються непримиренними. Для науковців важливо визнати та включити підвищену потребу в належній практиці управління та пов'язаній

теорії як у гнучкості заради зміни моделей управління проектами, так і в плануванні безпеки.

Апробація результатів та особистий внесок здобувача. Основні положення роботи доповідались, розглядались та обговорювались на науковій конференції Тернопільського національного технічного університету. Результати кваліфікаційної роботи опубліковані у тезах студентської наукової конференції, яка проводились у ТНТУ.

1 ТЕОРЕТИЧНІ ОСНОВИ МЕТОДИКИ ПОРІВНЯННЯ ПІДХОДІВ ДО УПРАВЛІННЯ ПРОЕКТАМИ

Робота пропонує наукову основу для критеріїв успіху проекту (Project Successful Criteria – PSC) та критичних факторів успіху (Critical Successful Factors – CSF) в управлінні проектами. Основна увага в цьому дослідженні зосереджена на критичних факторах успіху у проектах; вони відображають відмінності між трьома підходами традиційного, швидкого (Agile) та гібридного управління проектами. Два найбільш чітко протилежних підходи до управління проектами передують опису того, що називається "магічним трикутником" (він же "залізний трикутник"); його основні особливості та його припущення з'являються відповідно до усталеної моделі обмежень в управлінні проектом.

1.1 Традиційне управління проектами

1.1.1 Традиційна водоспадна модель

У моделях водоспаду кожна фаза має заздалегідь визначену початкову та кінцеву точки та точно визначені результати. На певних етапах та в кінці кожного етапу документи планування проекту затверджуються, як правило, Офісом управління проектами (PMO).

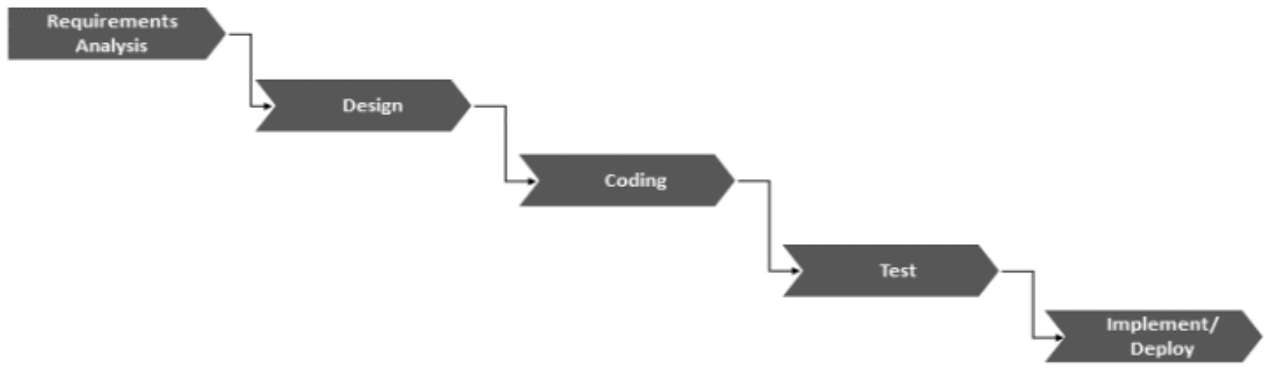

Рисунок 1.1 – Базова водоспадна модель розробки ПЗ

У класичній моделі водоспаду кожна фаза повинна точно визначити, що має бути досягнуто (вимога), іноді доповнюючи описом того, як цього можна досягти (дизайн). Щоб проект забезпечував кожне з досягнень (у термінах проекту, які часто називають дієвими результатами чи делівері), необхідно створити документи, які точно описують цей результат. У технічних проектах це називається специфікацією. В принципі, наступну фазу не можна розпочати, доки не буде завершено попередню фазу. Результати етапу завжди служать обов'язковими вказівками для наступного етапу, що робить планування менш гнучким.

Наприклад, на етапі формування вимог створюється документ, в якому перераховані всі вимоги замовника. Назва «Водоспад» походить від часто використовуваного графічного представлення етапів проекту, організованих у послідовності сходинок.

Якщо етап не може бути успішно завершений, наприклад, тому що на етапі впровадження виявляється, що деякі пункти специфікації не можуть бути реалізовані, як було заплановано, проект повинен повернутися до однієї з попередніх фаз. Завдяки простій і зрозумілій структурі прогресу проекту від фази до фази, модель водоспаду до сих пір користується великою популярністю сьогодні. Це зменшує небезпеку неможливості прогресу через занадто багато паралельних дій у неправильному порядку. У зв'язку з обов'язковим виконанням етапу, перш ніж почати наступний етап, забезпечується якість мети проекту. Однак важливо, щоб якість була чітко визначена та перевірена наприкінці кожної фази, оскільки вона визначає, чого можна досягти на наступному.

У цьому випадку для кожної фази точно визначені види діяльності та результати, яких необхідно досягти. Наприкінці фази, тобто на «воротах» до наступної фази, проект повинен зупинитися, доки не буде перевірено завдання проекту та результати цієї фази. Тільки якщо результат проектної роботи достатній, проект може перейти до наступної фази. Цей контроль якості після кожної фази проекту також називають воротами якості. У класичній моделі

Waterfall ворота якості не повторюються. Це означає, що ігнорування проблем якості на будь-якому рівні робить дуже ймовірним, що вони залишаться в готовому продукті. Хоча класична модель водоспаду все ще дуже популярна, через її послідовну обробку, одним з її найбільших недоліків є те, що проблема якості стає очевидною лише тоді, коли досягнуто воріт. Крім того, вимоги встановлюються на початку проекту. На даний момент проект не завжди цілком керований. Якщо потрібні зміни, вони повинні бути включені за допомогою так званих запитів на зміни – чейндж-реквести. Однак запити на зміни можуть збільшити час і бюджет проекту. Якщо є багато запитів на зміни, фокус початкової мети може бути втрачено.

У сучасному світі, керованому технологіями, доступністю даних і, як наслідок, швидшим прийняттям рішень, швидка адаптація до змін потрібна набагато частіше, ніж 20 років тому. Тому традиційна модель водоспаду вважається більш негнучкою в адаптації до змін. Зосереджено повністю на процесі і ігнорує людей, що в сучасному світі також стає все більш важливим фактором для успішного управління проектами.

Популярною водоспадну модель робить її контрольний аспект, чіткий зв’язок між продуктивністю та витратами, що створює «магічний трикутник», який створює враження у керівництва про визначення та управління ризиками проекту шляхом введення формалізму у процес управління проектом.

Підводячи підсумок, традиційна модель водоспаду використовує найчистішу форму лінійно-послідовної моделі процесу. Він виник у виробництві, де потрібні високоформалізовані процеси, оскільки зміни дизайну стають непомірно дорогими. З цієї причини підхід Waterfall вимагає детальних вимог і специфікацій перед початком будь-якої реалізації.

1.1.2 V-подібна модель

V-Model також використовує послідовні фази проекту. Вона отримала назву на честь розташування окремих етапів проекту, як показано на рисунку 1.2. На лівій гілці V вказуються об’єкт проекту, який буде реалізований, від

чорнового до детального проекту. Після реалізації зазначеного пункту проекту, права гілка V проходить знизу вгору. Спочатку тестуються всі компоненти, потім вони інтегруються в всю систему або підсистему і, нарешті, тестуються в цій новій інтеграції. Зрештою, вимоги замовника перевіряються до тих пір, поки замовник погоджується зі специфікаціями.

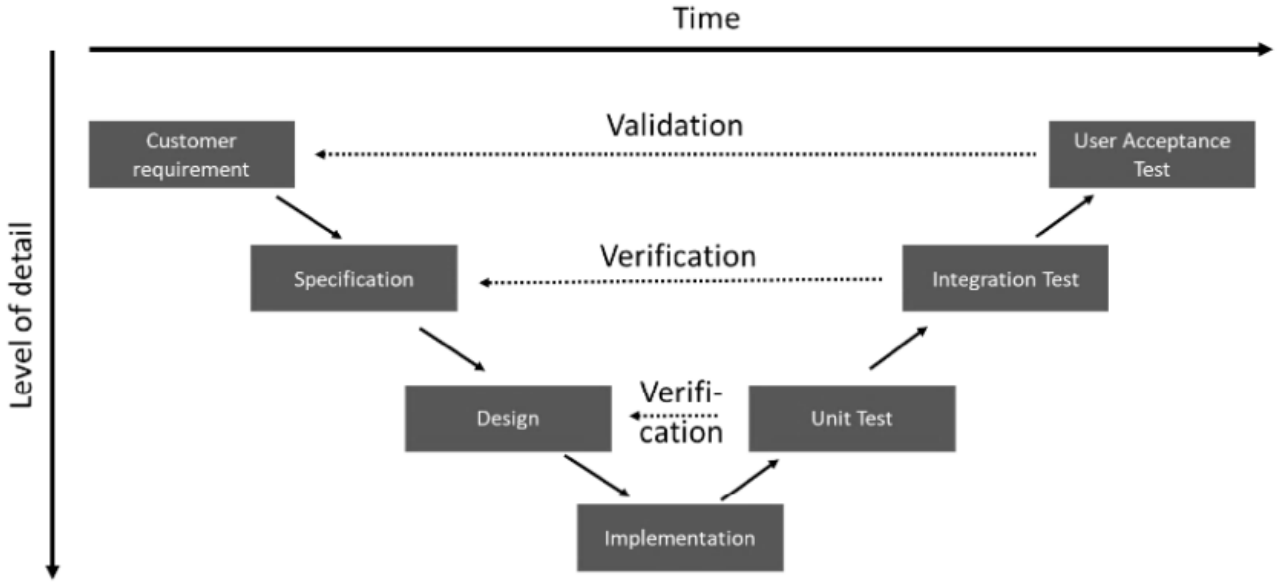

Рисунок 1.2 – V-подібна модель розробки

Кожна специфікація та етап проектування лівої гілки перевіряється у вигляді відповідних тестів. Ця структура просуває ідею якості та міцно інтегрує її в модель. Все, що було зазначено і для чого створено проект, згодом має бути перевірено в рамках верифікації та валідації. Верифікація – це чітка демонстрація (зазвичай шляхом вимірювання), що конкретна вимога (специфікація) виконана. Верифікація відповідає на питання, чи правильно вона була реалізована (з точки зору вимоги). Валідація підтверджує, що запит клієнта, задокументований у специфікаціях, виконано: вона відповідає на питання, чи була розроблена правильна річ для замовника. У V-моделі окремі фази розташовані в послідовності.

Ця модель проекту використовується в галузях, де безпека продукції має особливе значення, наприклад, фармацевтична або аерокосмічна промисловість.

V-модель широко використовується, оскільки вбудована верифікація та валідація моделі може надати повний доказ того, що вимоги були виконані. З іншого боку, результати проекту оцінюються лише в кінці; ранній зворотній зв'язок з розробниками та клієнтами не надається.

Критики також висловили заперечення, що V-подібна модель “ зображує лінійний і контрольований процес і неявно припускає, що замовник зрозуміє зміст і діяльність усіх завдань і підзадач розробки програмного забезпечення (навіть найбільш детальних). Таким чином, V-подібна модель не дасть точний опис практики розробки програмного забезпечення. Крім того, критики звинувачують цю модель у складності документації та, можливо, надмірній бюрократизації невеликих проектів.

1.1.3 Спіральна модель

Спіральна модель покращує тестування, надаючи багато тестових ітерацій, зменшуючи ризик створення невалідного продукту. Спіральна модель відрізняється від моделі водоспаду тим, що вона вводить ітераційний підхід, який працює в серії невеликих підпроектів або ітерацій за допомогою прототипування, що відрізняє спіральну модель від традиційного методу та V-моделі.

Найвідомішим прикладом моделі повторюваного процесу є спіральна модель, показана на рисунку 1.3. Кожна ітерація (повторення) використовується для визначення й уточнення вимог, уточнення цілей та покращення дизайну від вимог від грубого проектування до детального проектування. У спіральній моделі початок проекту показано в центрі. У кількох повторюваних ітераціях визначаються цілі та вимоги, оцінюються альтернативні рішення, визначаються ризики, які необхідно мінімізувати, а вимоги впроваджуються та перевіряються, перш ніж остаточно планується наступна ітерація.

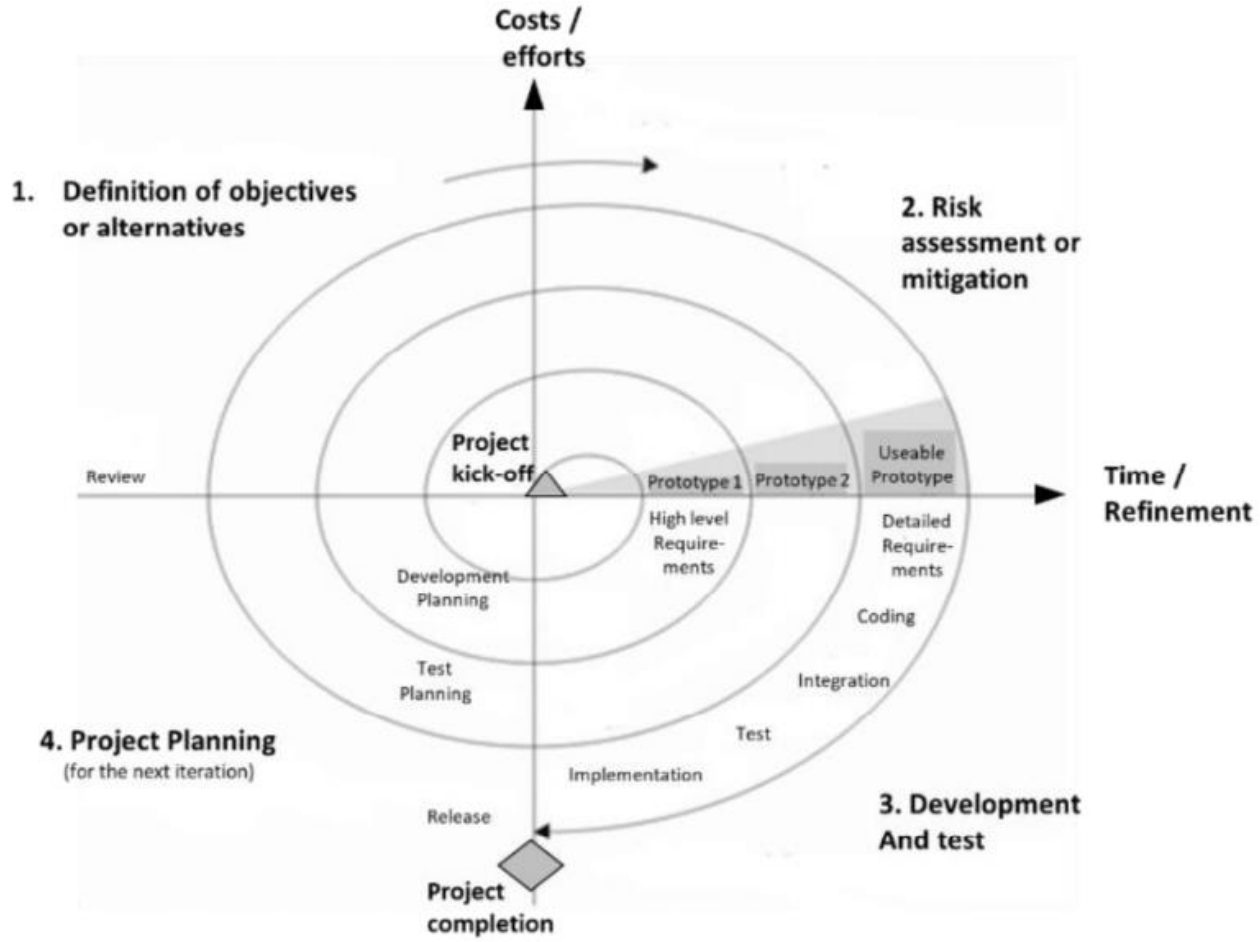

Рисунок 1.3 – Спіральна модель

Від ітерації до ітерації елемент проекту стає більш конкретним щоразу, коли виконується ітерація. Проект може бути завершений, коли остання ітерація зробить готовий продукт доступним. Плануючи цикли повторення, слід подбати про те, щоб у кінці кожного циклу був життєздатний результат. Кожен неповний результат повинен давати відповідну інформацію для планування та виконання наступного проходу циклу. Інакше загальний результат проекту не матиме конкретизації та стабільності.

Перевагами структурованої обробки нестабільних вимог за допомогою цих ітерацій є свідома покрокова конкретизація результатів проекту та виконання його цілей, а також той факт, що програмне забезпечення вперше створюється на початку життєвого циклу програмного забезпечення, тобто створюється новий прототип. Недоліком є ризик додаткових зусиль щодо проекту, якщо будуть потрібні виправлення пізніших етапів. Усвідомлена поетапна процедура також

може призвести до змін, які затягнуть завершення проекту. Крім того аналіз ризиків вимагає спеціального досвіду і, як і V-модель, вона не підходить для невеликих проектів.

Завдяки своєму повторюваному характеру та акценту на поступовій деталізації результатів проекту, спіральна модель нагадує добре відому модель Agile процедури Scrum, яка розглядається далі. Однак тут слід зазначити, що гнучкість досягається не шляхом проходження процесів, а шляхом відмінного ставлення в багатьох сферах управління від традиційних моделей мислення.

1.2 Огляд водоспадних моделей розробки і швидке створення прототипів

При швидкому створенні прототипу споживач ознайомлюється з усіма розробками, що є явною перевагою, якої не має традиційна модель водоспаду, яка часто не дозволяє споживачеві продукту бути достатньо залученим на етапі розробки продукту; йому часто надають кінцевий продукт лише в кінці традиційного циклу. Це призводить до ситуацій, коли продукт може бути розроблений не відповідно до вимог замовника. Швидке створення прототипів вирішує цю проблему систематично, залучаючи замовника як інтегральну частину моделі.

Однак, оскільки складні вимоги до розробки продукту, наприклад, або розробки програмного забезпечення не можна повністю визначити заздалегідь, потреба в гнучкості зростає. Щоб задовольнити цю потребу, були розроблені Agile методи. Поступові підходи призводять до кращих результатів у випадках, коли повну функціональну специфікацію неможливо визначити заздалегідь.

Короткий опис різних моделей водоспаду та їх розвиток з часом можна побачити на рисунку 1.4.

1.3 Agile Project Management

Гнучка (Agile) розробка програмного забезпечення – це загальний термін для використання гнучкості в розробці програмного забезпечення, а також у розробці продуктів та інших не IT-проектах. Зараз він широко використовується, і дослідження показують його позитивний вплив, що дозволило більш успішно завершувати проекти. Agile характеризується самоорганізацією команд, а також ітеративним і поетапним підходом до включення нових вимог.

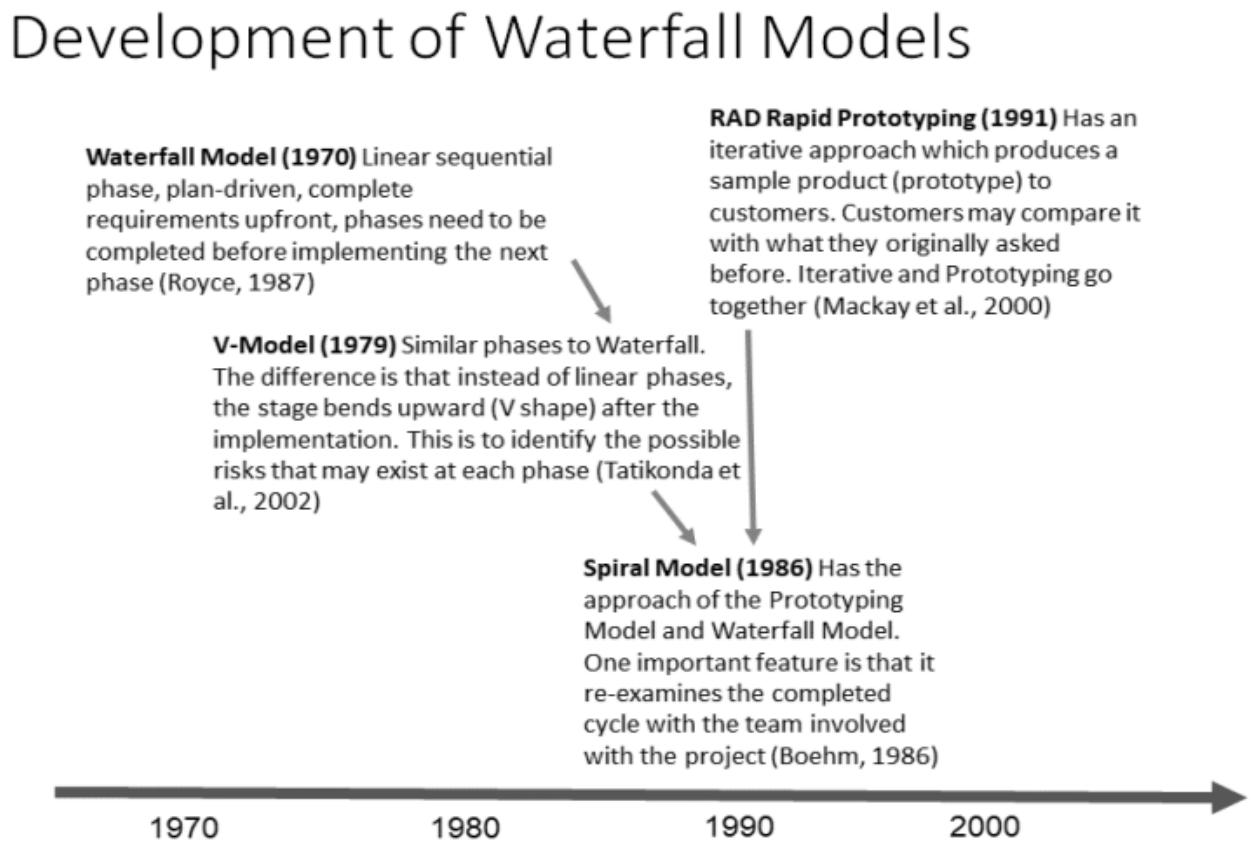

Рисунок 1.4 – Водоспадна модель розробки

Методика передбачає управління проектом з невеликою бюрократією та нечисленними правилами і, таким чином, може швидко адаптуватися до змін, мінімізуючи ризик помилок. Отже, гнучкість управління проектами є протидією традиційним процесам управління проектами, які часто вважаються важкими та бюрократичними.

Виникла чітка тенденція до використання Agile методів: VersionOne провела щорічне опитування Agile і виявила у 2013 році, що аж 84% провідних організацій сьогодні використовують Agile та Scrum процеси; у чотирнадцятому щорічному звіті (2020 р.) ця частка становила 95% (VersionOne, 2020 р.).

Однак ця тенденція не означає, що хтось вже вирішив проблеми інтеграції гнучких методів у загальні процеси довгострокового планування великих компаній чи груп компаній, описані вище. Загальні формулювання маніфесту Agile з його орієнтацією на клієнтів, співробітників і результати пропонують перенесення на інші сфери, що не є розробкою програмного забезпечення, у великих організаціях. Також в наукових колах нещодавно було опубліковано потік більш критичних статей [16], [17], [18]. Крім того, в роботі [18] стверджується, що гнучкість управління проектами набула популярності «на основі активної пропозиції з боку таких учасників, як консультанти, тренери та коучі", і, отже, має ознаки примхи керівництва. Навпаки, деякі дослідження залишаються оптимістичними щодо життєздатності та майбутнього розвитку гнучкого управління проектами і стверджують, що це більш тривала тенденція. Кілька звітів, опублікованих за останні роки, підтверджують цю точку зору, наприклад [19], [20]. У цих звітах робиться висновок, що гнучкість управління проектами помітно зросла у прийнятті та застосуванні в останні роки. Проте, чи можна це зробити успішно, значною мірою залежить від застосування та правильного використання ефективного підходу, що відповідає відповідній організаційній ситуації. Тому це дає можливість для проведення академічних досліджень.

1.4 Scrum

Scrum – це процедурна модель гнучкого управління розробкою програмних проектів. Вперше він був розроблений в рамках інженерії програмного забезпечення, але також може використовуватися в інших областях практики, наприклад, у розробці товарів.

За словами автора [21], Scrum передбачає лише кілька правил: фреймворку Scrum потрібно лише кілька правил і прийомів для реалізації в діяльності, артефактах і ролях. Він заснований на нещодавно розширеному розумінні того, що багато проектів програмної інженерії часто є занадто складними, щоб бути організованими у формі традиційного управління проектами (див. рис. 1.5).

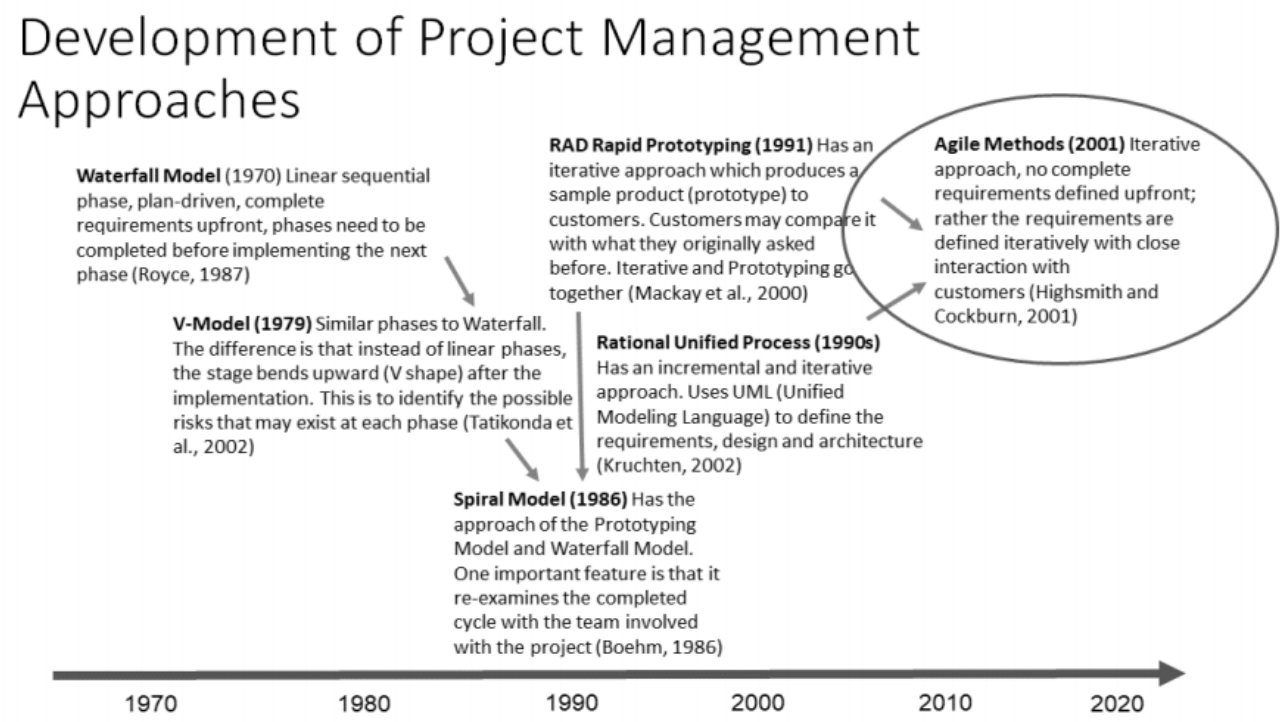

Рисунок 1.5 – Agile підхід до управління проектами

Припущення Scrum полягає в тому, що вимоги до цілей проекту можуть змінюватися в ході проекту, оскільки виникають нові виклики або цілі, які не були відомі на початку або підлягають динаміці, яка з’являється лише в ході проекту. Щоб забезпечити бажану гнучкість, команди зазвичай працюють у коротких циклах – так званих «спринтах» – для досягнення постійних покращень із чітко визначеним змістом. Швидке виконання підзадач, які роблять видимими перші успіхи, є суттєвими ознаками цього гнучкого підходу. Принцип «швидко зазнайте невдач – швидко навчіться» використовується свідомо і є бажаним для забезпечення стабільних результатів за допомогою досвіду та ітерацій. У

порівнянні з часом ініціювання проекту зміни та коригування в ході проекту вважаються досить поширеними та корисними [22].

Планування в Scrum розробляється ітераційно для просування продукту і складається з наступних артефактів (див. рис. 1.6):

– довгостроковий план (= беклог продукту), який постійно розробляється та вдосконалюється;

– детальний план (беклог спринту), який створюється лише для наступного циклу діяльності (= спринт).

Наприкінці спринту команда Scrum відповідає за доставку готового часткового продукту (= ціннісний продукт). Цінний продукт (= «в готовому до використання виді) має бути в придатному стані для доставки замовнику. Після циклу продукт, вимоги та процедури переглядаються та розвиваються в наступному спринті.

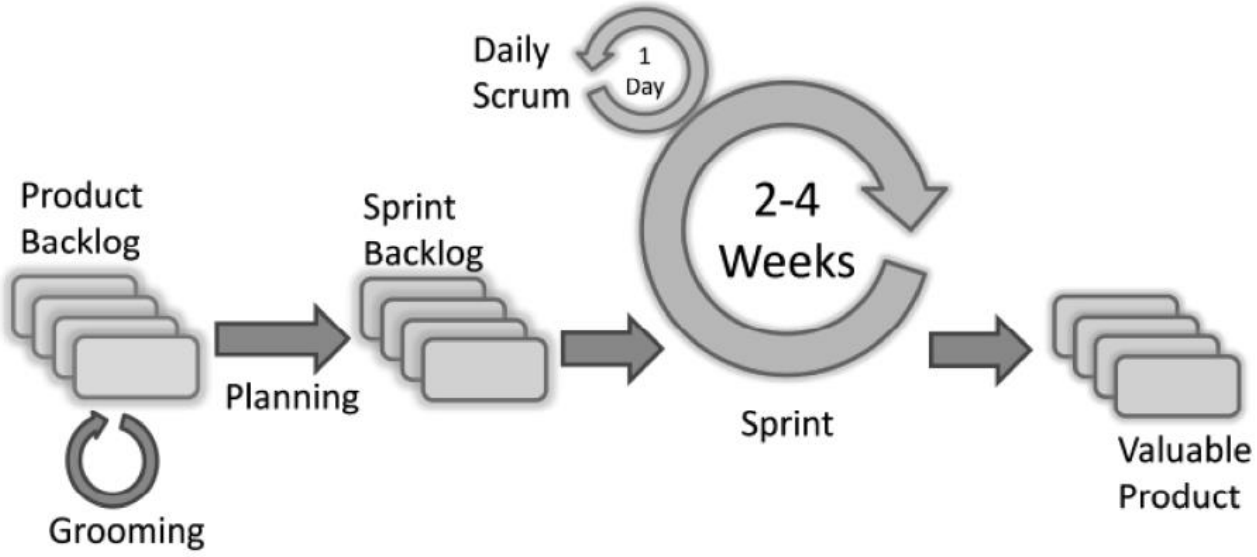

Рисунок 1.6 – Модель Scrum та її компоненти

Scrum має такі компоненти:

– Ролі в Scrum: Власник продукту, Scrum Master і Scrum-команда.

– Scrum-заходи: спринт, планування спринту, щоденний Scrum, огляд спринту та ретроспектива спринту.

– Артефакти Scrum: беклог продукту, беклог спринту, цінний приріст продукту, план випуску та беклог перешкод.

1.4.2 Ролі Scrum

Три ключові ролі в Scrum - це власник продукту, Scrum-майстер і команда розробників. Кожному з них закріплена своя функція управління з конкретними обов’язками та завданнями. Тільки якщо вони в повній гармонії, проекти можуть бути успішними. Незважаючи на повну гнучкість в управлінні проектами, суворе дотримання правил Scrum має фундаментальне значення. Звичайно, це завжди включає чітке розмежування ролей у Scrum.

Крім того, існують інші ролі, такі як користувачі, зацікавлені сторони та керівництво, які зазвичай не є невід’ємною частиною команди Scrum.

Власник продукту та його роль у Scrum.

Завдання власника продукту – точно знати інтереси користувачів і зацікавлених сторін (історії користувачів) і послідовно представляти їх. Для цього, проект строго прагне бачити всі вимоги користувача. Зв'язок з ринком і клієнтом є основним принципом управління проектами Scrum, оскільки він фокусує свою увагу виключно на вимогах, які мають бути реалізовані при розробці, вирішуючи, які властивості або функціональні можливості продукту слід розробити та чи задовольняються вимоги компанії. Замовник не бере участь у процесі розробки під час окремих спринтів, які в Scrum реалізуються через роль Власника продукту, але клієнт повинен бути доступним протягом усього проекту, щоб переконатися, що його вимоги включені в процес. Завдяки цьому менеджмент проекту Scrum отримує набагато краще представлення клієнтів під час виконання проекту.

Скрам-майстер та їх роль у Скрам.

Скрам-майстер виступає як модератор і постачальник послуг у процесі Scrum і організовує спілкування команди розробників із «зовнішнім світом». Скрам-майстер відповідає за контроль дотримання цінностей і правил проекту та створення відповідних умов для успішного проекту. Вони включають надання

ресурсів, усунення потенційних бар’єрів та посередництво між власником продукту та командою розробників, тому головна роль Scrum-майстра полягає в тому, щоб допомогти команді працювати безперебійно в рамках вимог Беклогу продукту та Спринту. Як нейтральний постачальник послуг, Scrum-майстер ні до кого не «завчасно призначений» і не приймає жодних рішень по суті.

Команда та її роль у Scrum.

Як правило, команда в Scrum складається від чотирьох до дванадцяти співробітників проекту і має міждисциплінарну структуру. Деякі автори, такі як [23], віддають перевагу меншим командам; вони рекомендують від п'яти до дев'яти членів. На даний момент конкретні ієрархії, що пов’язують окремі сфери компетенції в команді, не передбачені. Під час спринту команда самоорганізується та самостійно впроваджує необхідні інкременти нового продукту. Мета полягає в тому, щоб створити сприятливу атмосферу, яка в значній мірі вільна від руйнівних зовнішніх впливів, що сприяє розвитку навичок кожного члена команди і, таким чином, допомагає досягти найкращих результатів у спільній роботі.

Порівняно з традиційним управлінням проектами, роль команди розробників вимагає більшої автономії з боку команди та меншого контролю за прогресом і звітами. Багато рішень приймаються командою незалежно, що також несе наслідки (наприклад, через невідповідність іншим рішенням). Для цього потрібна інша культура в організації, але яка також починається з вищого керівництвя.

Проекти Scrum включають додаткові ролі, такі як клієнт, менеджер, постачальник і, можливо, команди інших проектів або робочих потоків. Однак ці ролі часто, на відміну від багатьох традиційних проектних установок, не є частиною основної команди проекту. Тому одним із основних обов’язків Власника продукту є представлення цих зацікавлених сторін та їхніх відповідних потреб у розробці продукту. Скрам-майстер також виступає як представник; він виступає від зацікавлених сторін з організаційної точки зору (посередництво або посилення розвитку). Власники продуктів також потребують хороших відносин

із зацікавленими сторонами; їхня роль полягає в тому, щоб мати глибокі знання про ринок, щоб представляти клієнтів та їхні потреби в проекті, коли це потрібно.

1.5 Діяльність Scrum

Ключовою характеристикою Scrum є безперервний прогрес протягом двох-чотирьох тижнів за допомогою самостійних циклів, які називаються спринтами. Метою кожного спринту в Scrum є впровадження командою розробників нової функції продукту або компонента продукту (у гнучкій розробці програмного забезпечення це називають функціональними можливостями або інкрементами).

Ця нова функція має бути достатньо зрілою наприкінці кожного робочого сеансу, щоб власник продукту вважав її завершеною та потенційно доставленою. Таким чином поступово підвищується якість або ступінь зрілості продукту.

Обов’язкові критерії завершення.

Основою для оцінки є так зване «визначення готового». Це також характерно для гнучких методів управління проектами, таких як Scrum, і діє як контрольний список конкретних вимог, про які власник продукту та команда розробників узгодили перед початком спринту [23]. Ці вимоги зазвичай формулюються у вигляді так званих історій користувачів. Лише тоді, коли все, що зазначено в них, буде виконано, і обидві сторони домовляться про обов’язкове виконання або 100-відсоткове виконання критеріїв якості, робочий розділ буде успішно завершено. Під час спринту команда розробників самоорганізується і не стикається з новими вимогами. Таким чином, вона працює на власну відповідальність, що в той же час гарантує гнучкість проектної роботи, що є метою гнучкого управління проектом.

1.5.1 Зустрічі, огляд та ретроспектива спринту

Scrum-спринти мають фіксовану хронологію: спочатку Власник продукту, команда розробників і Scrum-майстер зустрічаються на зустрічі з планування спринту.

На основі бачення продукту, що зберігається в Бэклозі продукту (динамічний план, спочатку викладений у його базовій розбивці), координуються кроки впровадження наступного етапу роботи. Не рідко керівництво та користувачі також беруть участь у цих «початках». Учасники проекту обговорюють, яку характеристику продукту може реалізувати команда в наступному спринті. Після того, як завдання, які необхідно виконати для досягнення мети спринту, узгоджені, вони записуються в беклог спринта.

Як зазначалося раніше, спринт – це фіксований період, протягом якого реалізуються вимоги до продукту. Спринт починається з планування спринту. Після цього йде реалізація запланованих історій користувача (див. артефакти спринта). В кінці спринту перевіряється досягнутий приріст продукту (результат спринту) (Sprint Review) і оптимізуються внутрішні процеси (Sprint Retrospective). Потім відразу ж наступає наступний спринт, який знову починається з планування. Огляд окремих кроків спринту наведено на рисунку 1.6.

Після цього починається етап реалізації. У 15-хвилинних Daily Scrums статус-кво повідомляється на початку або в кінці кожного дня. Як правило, в цій процедурі також беруть участь Скрам-майстер і Product Owner. Таким чином, будь-які вузькі місця чи проблеми можна виявити на ранніх стадіях та усунути їх [24].

У літературі зазвичай передбачається фіксована тривалість спринту в 14-30 днів. Scrum-команди часто працюють разом у довгостроковій перспективі. Таким чином, склад команди та тривалість спринту постійні. Це полегшує команді оцінку того, що можна зробити в спринті, ніж в інших проектах, у яких немає цієї фіксованої структури. У будь-якому випадку, тривалість має залишатися постійною протягом періоду проекту. В ідеалі в порівнянних типах

проектів він ідентичний і постійний для всіх проектів. Наприкінці спринту всі учасники збираються для остаточного огляду спринту. Результат представляється та перевіряється Власником продукту, щоб побачити, чи було задоволено «Визначення готового продукту», і чи була досягнута мета Спринту. Ретроспектива «Спринта», що описана нижче, також служить для своєчасного обміну досвідом та обговорення пропозицій щодо покращення.

1.5.2 Артефакти Scrum

Беклог продукту (довгостроковий план).

Бэклог продукту містить описи різних вимог до продукту [21]. Він ніколи не вважається завершеним, оскільки він може змінюватися в ході проекту, щоб не відставати від мінливих потреб клієнтів. Різні вимоги в Беклозі продукту описуються по-різному, залежно від того, наскільки добре вони були зрозумілі. Вимоги, які описані досить детально, щоб бути реалізовані, стають так званими «отовими», що означає, що елемент виконано. Опис запису в Бэклозі продукту завжди включає оцінку зусиль щодо впровадження. На відміну від специфікацій у традиційному управлінні проектами, оцінка завжди виконується особою, відповідальною за реалізацію, тобто особою, яка в кінці виконає необхідну роботу. Це важливо, оскільки в традиційному управлінні проектами керівник проекту або PMO можуть призначати та оцінювати роботу без достатньої консультації з особами або командою, які фактично виконуватимуть роботу. Досить часто це призводить до розбіжностей між керівником проекту та особою чи командою, яка повинна виконувати роботу. Більше того, Беклог продукту не є вичерпним. Це жива колекція, яка оновлюється та деталізується під час реалізації проекту, і тому її обсяг може змінюватися протягом проекту. Це значний зсув у мисленні від традиційного управління проектами. Оскільки беклог продукту постійно зберігається, це означає, що існує необхідність узгодити зміни з побажаннями зацікавлених сторін. Він повинен містити достатньо вимог, щоб досягти стану «готовності», особливо якщо ці вимоги необхідні для наступного спринту.

Відповідальним за підтримку Бэклогів продукту є особа, яка виконує роль Власника продукту, як описано вище. Автори в [24] також визначили покращення та виправлення помилок для продукту, наприклад, у результаті огляду спринта, які мають бути додані до Беклогу продукту, як основний принцип. Записи (тобто вимоги до продукту) у Беклозі продукту пронумеровані та впорядковані за важливістю. Таким чином визначається важливість кожної вимоги по відношенню до інших. Таким чином, підхід Scrum впроваджує неявне управління важливістю вимог, що, отже, допомагає зосередитися на діях, які є актуальними для клієнта та результату. Навіть якщо одна або дві вимоги є однаково важливими, їх порядок має бути визначений, перш ніж його можна буде ввести в Беклог продукту. Оскільки замовлення сформульовано з точки зору клієнта, досягається краща зосередженість на діях і завданнях, які також важливі для клієнта [25].

Наприклад, запис для програми-календаря може виглядати як у таблиці нижче. Усі основні функції мобільного додатка перераховані та розбиті в резерві, як-от «Щотижневий перегляд календаря», «Створити нову зустріч» або «Перемістити наявні зустрічі».

Вимоги сформульовані в Scrum за допомогою так званих історій та епосів користувачів (Pries and Quigley, 2010). Історія користувача (юзер сторі) – це вимога, яка написана з точки зору певної ролі користувача з використанням системи або продукту, часто з точки зору користувача системи. Типовою формулюванням може бути «Як користувач, я хотів би мати можливість створити нову зустріч у своєму календарі одним клацанням миші» (Timinger, 2017). Епік є великим невизначеним чи погано визначеним юзер сторі. На даний момент вимоги визначені лише приблизно і ще не можуть бути оцінені з точки зору зусиль. Особливо на початку проекту деякі вимоги все ще незрозумілі. Якщо вимоги невідомі або неповні, плани, засновані на них, піддаються невизначеності. Часто в цей момент робляться припущення, які згодом виявляються помилковими і призводять до відхилень від плану (Курнія та ін., 2018).

Таблиця 1.1 – Приклад беклогу продукту

ID	Історія користувача	Тип	Розмір, Сторі поінтів	Пріоритет	Класифікація
1	Тижневий перегляд календаря	Юзер сторі	40	150	Обов'язково
2	Створити нову зустріч (з датою, часом, описом)	Юзер сторі	100	250	Обов'язково
3	Перемістити наявні зустрічі	Юзер сторі	25	150	Обов'язково
4	Показати день	Юзер сторі	30	200	Обов'язково
5	Показати перегляд місяця	Юзер сторі	30	100	Опція
6	Видалити зустріч	Юзер сторі	50	250	Обов'язково
7	Імпорт відпусток з інших систем	Епік		100	Опція
8	Синхронізація з мобільним телефоном	Епік		150	Обов'язково

У Agile-проектах невизначеності приймаються та чітко документуються. Замість повністю сформульованої історії користувача, так званий Епіс включений в Беклог продукту. Іншими словами, Беклог продукту складається з User Stories та Epics.

Історії користувача з таблиці 1.1 можна було б просто назвати керуванням календарем як Епіс. Епіки допомагають нічого не забути і можуть бути трактовані в Бэклозі продукту як заповнювачі, які пізніше необхідно детально опрацювати. Приклад, як показано в Таблиці 1.2, може бути додатково підключений до інших систем або мобільних телефонів.

Scrum свідомо утримується від оцінки (впровадження) зусиль користувача історії в особистих днях. Натомість оцінюється розмір історій користувачів. Це показник складності історії користувача. Одиниця такого розміру називається

Story Points. Якщо історія користувача має вдвічі більше очок історії, ніж інша історія користувача, вона вдвічі складніша.

Таблиця 1.2– Приклад беклогу спринту

Scrum User Story 2: створіть нову зустріч (з датою, часом, описом) Сторі-поінтів: 100		
Оцінки завдань Scrum	**Сторі-поінти**	**Оцінка зусиль у людино-днях (MDs)**
Створення діалогового вікна для введення зустрічі	20	1 MD
Створення а бази даних схема для призначення	40	2 MD
Підключення діалогу до бази даних	30	1,5 MD
Тестування введення зустрічі	10	0,5 MD

Зазвичай до впровадження належать тестові та документаційні роботи. Визначення того, що саме потрібно для реалізації описаної функції, називається визначенням готового. Історія користувача повністю реалізована лише тоді, коли вся робота, пов’язана з історією користувача, була завершена та інтегрована в робочий приріст продукту. Отже, всі ці роботи слід враховувати при оцінці. Залежно від історії користувача, при їх реалізації необхідно враховувати різні аспекти. Оцінка розміру історій користувачів у Scrum часто виконується в контексті виконання так званого scrum-покеру планування (= семінар для оцінки трудомісткості). Усі члени команди розробників, Власник продукту та Scrum Master мають брати участь. Завдання має бути повністю описано у вигляді історій користувачів. Scrum Master знайомить розробників з історіями користувачів, але дозволені лише ті запитання, які допомагають зрозуміти роботу. Потім команда оцінює відносне навантаження для окремих історій користувачів. Кожен розробник отримує набір карток з оцінками. Оцінки не відповідають абсолютному виміру, а лише оцінюють відносне навантаження розповідей користувачів, які підлягають оцінці. Якщо необхідно, ці так звані

бали історії можна потім перетворити в абсолютні оцінки витрат, використовуючи значення людино-дня (MD) для беклогу спринта.

Беклог спринту (детальний план) описує роботу, заплановану на поточний спринт (так званий Scrum Sprint). Це робиться на спеціальній нараді (Планування спринту, див. нижче), де вибираються відповідні вимоги з продакт-беклогу.

У бэклозі спринту цей запис тепер можна розбити на такі завдання (таски) (див. таблицю 1.2). Історії користувачів Бэклогу продукту описують вимоги замовника або користувача. Виходячи з цього, мають бути виведені конкретні завдання, які повинна виконувати команда для виконання вимог. Це робиться на початку кожного спринту як частина планування спринту. Історії користувачів, які будуть реалізовані в наступному спринті, будуть вибрані з Беклогу продукту та сформульовані. Коли історія користувача розбивається на завдання, її загальний розмір, зазначений у сторі-поінтах, розподіляється у вигляді окремих завдань. Кожне завдання також отримує оцінку розміру в блоці Story Point і відповідну оцінку зусиль. Усі завдання на спринт збираються в Беклог спринту, як показано в таблиці 1.2.

Маніфест Agile визначає своїм найважливішим принципом: «Найвищим пріоритетом є задоволення клієнтів за допомогою ранньої та постійної доставки цінного програмного забезпечення». З цим принципом існує думка про ітераційний розвиток. За цією думкою також стоїть ідея про те, що клієнт повинен отримати придатний для використання продукт з чіткою негайною вартістю, що веде до концепції приростів: наприкінці кожного спринту новий приріст продукту або корисний продукт мав бути створений. Приріст продукту є результатом спринту. Це може бути, наприклад, нова функціональність програмного забезпечення або функція кінцевого продукту, наприклад, можливість програмного забезпечення створювати нову зустріч або перегляд календаря, як показано вище.

Важливо, щоб кожен приріст продукту був придатним для використання та/або презентабельним. Власник продукту або клієнт повинен мати можливість «випробувати» новий продукт. Лише після цього можна отримати зворотний

зв’язок, конкретизувати епоси та завершити історії користувачів. Цей проект зменшує ризик виконання «неправильної» функції або діяльності, що є основною перевагою підходу Scrum. Уявлення, отримані в результаті цих зустрічей і зустрічей, є головним внеском у планування наступного спринту.

Вимога, щоб інкремент продукту був виконуваним або презентабельним, впливає на вибір історій користувачів для наступного спринту. Історії користувачів завжди потрібно вибирати так, щоб разом вони створювали щось презентабельне.

Хоча центральні проблеми жорсткої моделі водоспаду вирішуються за допомогою підходу Agile Scrum, деякі недоліки підходу Scrum очевидні. Незалежні методи роботи команди-виконавця накладають певні обмеження на планування безпеки компанії. Досить важко оцінити, який результат очікується в кінці деяких спринтів і проекту в цілому. Вимірювання загального успіху є відповідно проблематичним, а також важким для великих компаній, якщо вони в основному звикли до середньо- та довгострокового планування.

Scrum має ряд переваг і може призвести до значного підвищення продуктивності в управлінні змінами в компаніях. З одного боку, клієнт, команда проекту та інші зацікавлені сторони працюють тісно разом, тому потреби клієнта знаходяться в центрі всіх заходів спринта, а його побажання та проблеми можуть бути включені в гнучкість процесу. Самоорганізація Scrum-команд не вимагає виснажливого контролю та важких адміністративних процесів. Крім того, оскільки відповідальність чітко визначена, канали та структури комунікації дозволяють виявляти проблеми та вирішувати їх під час виконання проекту. Спринт з обмеженим часом, щоденний Scrum та зосередженість на прозорості дозволяють командам Scrum швидко адаптуватися, якщо потрібно. Але є і явні недоліки. Одним із можливих недоліків є те, що масштабні проекти за допомогою Scrum можуть втратити уявлення про проект в цілому, і важко масштабувати підхід [27]. Підхід Scrum може вимагати від команди зосередитися на основній діяльності окремого клієнта, але це може втратити загальну картину або мету проекту. Наприклад, програмна програма може бути розроблена точно

відповідно до потреб клієнта, але операційному відділу може бути важко забезпечити ефективне обслуговування такого програмного забезпечення. Підходи до масштабування Scrum намагаються усунути цей очевидний недолік, але, схоже, виникають труднощі з координацією необхідних заходів та ресурсів, а також відсутність візуалізації вмісту продукту та беклогу спринта може бути проблемою в процесі Scrum.

1.6 Управління проектами Kanban

Іншим проектним підходом, який виник у виробничому середовищі і зараз частіше використовується в розробці програмного забезпечення, є Kanban. Спочатку Канбан використовувався в Toyota для забезпечення виробництва «точно вчасно». Замість того, щоб виробляти продукцію способом, який спричиняв високі витрати на зберігання, виробництво було змінено, щоб мінімізувати час зберігання товарів. За допомогою системи Kanban можна реалізувати систему витягування, що робить виробництво більш ефективним.

Термін походить від японців і заснований на процедурі, яка використовувалася в той час. «Канбан» означає «знак» або «карта» у спрощеному перекладі; під час впровадження системи знаки передавались по ланцюжку виробництва для управління ресурсами та машинами. За допомогою знаків можна було безпосередньо побачити, де виникають вузькі місця та які заходи необхідно провести далі, щоб згладити ці вузькі місця.

Kanban вже використовувався в Toyota в 1940-х роках. Основні принципи, які були сформульовані в той час, існують і досі. Він досяг повної зрілості в 1970-х і 80-х роках як метод підвищення гнучкості та ефективності виробництва фірми. Лише набагато пізніше Девід Дж. Андерсон передав ідею Kanban управлінню проектами. Андерсон був натхненний принципами Lean, формулюванням Theory of Constraint і картками Kanban з виробничої системи Toyota [27]. Теоретичні основи Канбан описані в чотирьох основних принципах і шести практиках. Це показує, що Kanban дотримується еволюційного підходу

до управління змінами та прагне постійного вдосконалення методу роботи. Чотири основні принципи та шість практик важливо зрозуміти також тому, що ці концепції використовуються в управлінні проектами.

Дошка Kanban – це інструмент для відображення та візуалізації робочого процесу, і як такий є ключовим компонентом методу Kanban. Дошка поділена на колонки та доріжки для плавання. Кожна колонка представляє фазу процесу, а доріжки для плавання представляють різні сфери завдань діяльності. Коли завдання входить у робочий процес, воно розміщується на картці Kanban, яка проходить через кожен стовпець дошки. Тому вона називається «Дошка завдань Канбан». Кожна дошка Kanban розділена на три основні розділи, які показують статус завдань. Статус може бути «Зроблено», «В процесі» (з підстанами: наприклад, розробка, тестування, розгортання) або «Готово».

Щоб деталізувати дії процесу, за потреби можна створити додаткові підрозділи (щоб вказати підстани, наприклад, розробка, тестування, розгортання). На практиці використовуються інші варіації, наприклад, інкорпорація, що є кроком забезпечення якості, таким як ревю або затвердження власником продукту. Мета полягає в тому, щоб точно візуалізувати робочий процес. Наприклад, команда розробників може використовувати дошку завдань Kanban із багатьма колонками та доріжки для плавання, показані на рисунку 1.7.

На відміну від Scrum, Kanban не прописує процеси та структури в деталях і не визначає конкретних ролей, таких як Scrum Master, власник продукту або щоденний Scrum. Він також не має фіксованого часу розробки (це не обмежений час), як у спринту. Але це не означає, що такі процеси та елементи не можуть бути інтегровані в управління проектами Kanban. Багато команд, які працюють з управлінням проектами Kanban, справді щодня зустрічаються на правлінні Kanban, щоб обговорити поточну ситуацію з робочими пакетами та подібні запитання до тих, що містяться в Daily Scrum.

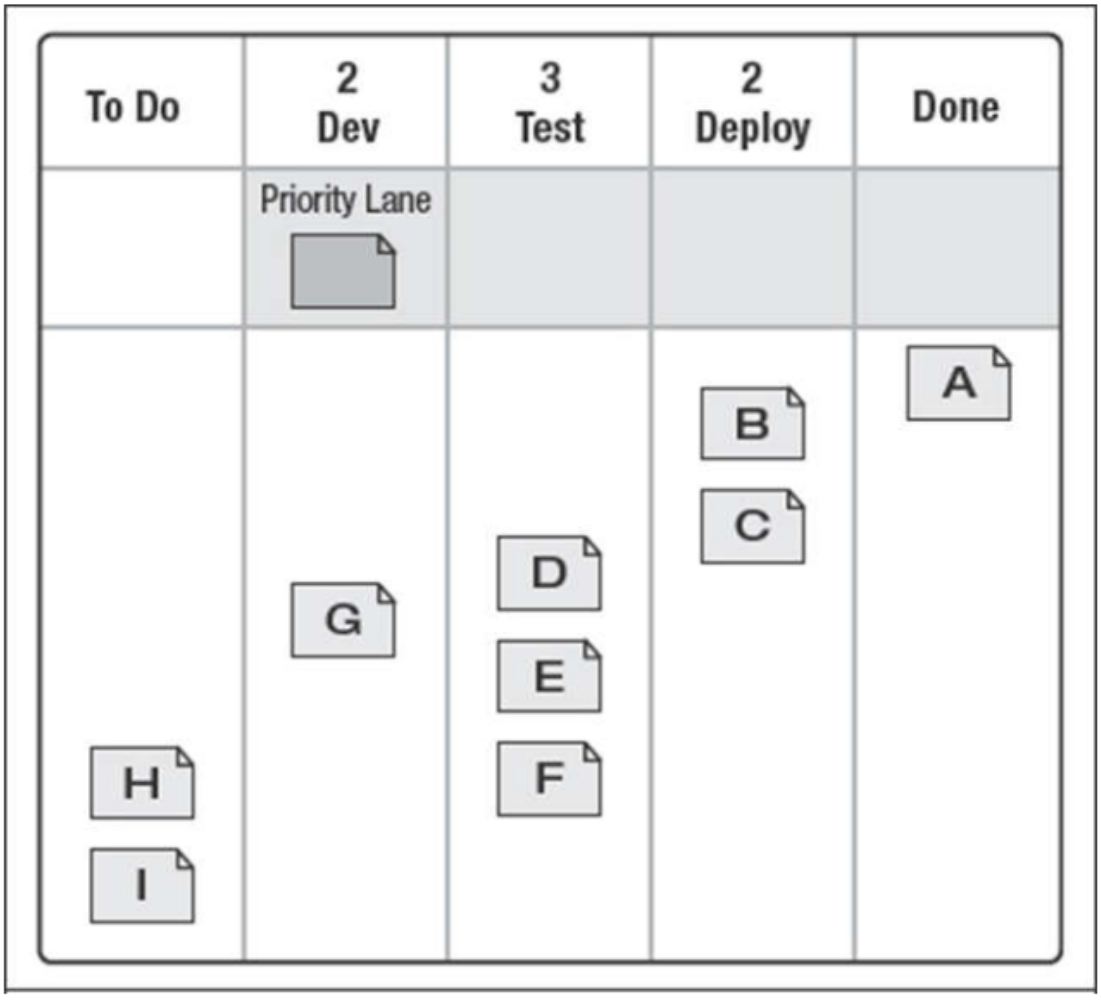

Рисунок 1.7 – Канбан-дошка в управлінні проектами розробки програмного забезпечення

Канбан, що лежить в основі принципів виробничої системи Toyota, сприяє самоорганізації. У ньому зазначається, що співробітники самостійно виконують завдання (= принцип витягування), замість того, щоб їх призначав керівник проекту або PMO. Це означає, що менеджери проектів виконують іншу роль, ніж у традиційному управлінні проектами. Однак, на відміну від Scrum, у багатьох проектах Kanban координує керівник проекту. На практиці для цього досить часто потрібна роль, яка включає елементи традиційного менеджера проекту, Scrum Master і власника продукту.

Власник. Керівник проекту може задавати критичні запитання на дошці Kanban, виявляти вузькі місця і з цією критичністю позбавляти команду проблем і підводних каменів. Менеджер проекту також є інтерфейсом з керівництвом поза

межами основного проекту і діє як «екземпляр ескалації» для команди. На відміну від традиційних менеджерів проектів, менеджер проекту Kanban, який призначає та редагує робочі пакети, надає команді такі ж свободи, як і в командах Scrum.

Андерсон виділяє чотири основні принципи Канбан [27]:

а) Почніть з того, що ви робите зараз, оскільки занадто багато часу часто витрачається на обширне планування, яке в будь-якому випадку незабаром потребує перегляду чи коригування.

б) Прагнення до поетапних, еволюційних змін, маючи на увазі, що зміни не повинні представляти єдиного серйозного порушення, а скоріше проявлятися в поступових покращеннях;

в) Поважайте поточні процеси, ролі, обов'язки та посади, що означає, що зміни, щоб бути успішними, не можуть бути здійснені всупереч зацікавленим сторонам, а радше разом з ними.

г) Стимулювання лідерства та підзвітності на всіх рівнях організації, що вказує на те, що зміни повинні бути ініційовані керівництвом.

Перші три принципи тісно пов'язані. Вони підкреслюють практичність і простоту введення та використання Канбан. Філософія полягає в тому, що завжди можна почати відразу з того місця, де він є. На цій основі принцип постійного вдосконалення невеликими кроками покращить процес і організаційну структуру відповідно до основних принципів і практик.

Канбан можна запустити відразу, без особливої підготовки. Це полегшує його впровадження, але спочатку керівник проекту та команда повинні знати поточний стан усіх видів діяльності та заходів. Відправною точкою у впровадженні Kanban є вивчення того, хто інтенсивно бере участь у проекті і хто має які відповідальності і повноваження. Це визначає, хто і як бере участь, а також яке завдання виконує проект і яким чином.

Щоб система Kanban працювала краще в сучасному середовищі, кількість процесів, що запускаються одночасно, має бути обмежена. Це робить

непотрібним постійне перемикання між процесами, що забезпечує кращий робочий процес і не відволікає співробітників від поточної діяльності проекту.

Під час аналізу ефективності Kanban було виявлено, що система Kanban є дуже хорошим варіантом для моніторингу роботи міжнародних команд та найкращого розподілу їх потенціалу. Принцип витягування забезпечує виконання тільки тих завдань, які дійсно необхідні. В управлінні проектами завдання досить часто виконуються, коли їх взаємозалежність складна, але не велика.

Кожен проект повинен використовувати структуру розбивки робіт, щоб мати уявлення про взаємозалежності. В залежності від проекту, по складності, пріоритету і типу, комбінований підхід в гібридному управлінні проектом добре підходять для системи Канбан. Методи також можна комбінувати, залежно від реалізації, але іноді методи можуть суперечити один одному. Аналіз важливих тенденцій можна дуже добре використовувати як доповнення до плану мережі. Однак «основні принципи мережі та системи Kanban значно суперечать один одному» … [оскільки] «в мережі дії визначені заздалегідь, тоді як у Kanban вони виконуються гнучко за принципом витягування». Перевага підходу до управління проектами Kanban полягає в тому, що, хоча цей метод дає лише приблизні рекомендації, його можна адаптувати до будь-якого проекту, а отже, він дуже добре підходить для гібридного управління проектами.

Керівництво проектами Kanban прагне до поступальних, еволюційних змін. Якщо поточний стан відомий, можна внести покращення. Пропускний час роботи над проектом є дуже важливим критерієм, за яким вимірюється успішність Kanban. Необхідно вжити заходів, які дозволять персоналу періодично записувати час виконання, що робить процеси управління проектами видимими.

Канбан наполягає на тому, щоб зміни вносилися поступово: він заснований на еволюції, а не на революції. Коли мета полягає в тому, щоб не втратити важливих зацікавлених сторін і взяти всіх на шлях змін, незначні зміни краще,

ніж великі, оскільки вони пов’язані з меншим ризиком того, що їхні наслідки не будуть повністю охоплені і очікуване покращення не здійсниться.

Критики еволюційних змін побоюються, що невеликі зміни унеможливлять будь-які серйозні інноваційні зміни. Успішні менеджери знають про цю слабкість еволюційних змін і регулярно розмірковують про статус своєї компанії або проекту. На додаток до еволюційних удосконалень вони досліджують, чи взагалі можливо поступово наблизитися до оптимуму, чи для цього потрібні великі революційні зміни. Використовуючи принцип поступальних еволюційних змін, Канбан дотримується принципу «Плануй-роби-перевіряй-дій», посилаючись на безперервне планування вдосконалень, їх впровадження, аналіз ефективності та управління. Однак покращення також потребують більш радикальних кроків і суттєвих змін. Покращення є і буде лише вдосконаленням, якщо воно встановлене та застосоване протягом тривалого періоду. Деякі автори суперечать аргументу Андерсона [27], що всі відповідні зацікавлені сторони мають бути залучені до створення покращень, що вимагає інших методів та інструментів управління проектами, ніж дошка Канбан. Це пов’язано з тим, що Канбан є досить директивним, недостатнім підходом, який потребує доповнення.

Одним із наслідків двох основних принципів, викладених вище, є те, що поточні процеси, ролі, обов’язки та звання мають поважатися. Коли зміни ініційовані погано і погано повідомляються, працівники, яких вони стосуються, займають позицію відмови і блокують заплановані зміни всюди, де вони можуть.

Завжди знайдуться співробітники, яких задовольняє лише статус-кво і не упустять жодного шансу «торпедувати» зусилля, щоб його змінити. Проте може бути корисно або необхідно змінити процеси, ролі, обов’язки та посади для покращення організації та її діяльності. Але важливо дотримуватися другого основного принципу і нехай еволюція змін буде поступовою. Зміна має бути детально роз’яснена всім особам, яких вона торкається, і бажано мати невеликий розмір команди.

Agile означає, серед іншого, відповідальність окремих керівників. Трансферні команди дають їм свободу самоорганізації. Канбан наголошує на цьому аспекті. На відміну від Scrum, Kanban залишає конкретний дизайн керівнику проекту, що є важливою відмінністю, оскільки дозволяє йому краще адаптуватися до нових ситуацій управління проектом. Таким чином, Kanban є свідомо окресленою формою Scrum і може бути описаний, як незалежна модель процесу. На практиці Kanban може використовуватися і використовується окремо, а також у поєднанні зі Scrum або традиційними процедурними моделями для гібридного управління проектами. Тому він зазначений в опитуванні як незалежний підхід, незважаючи на критику з боку інших авторів.

1.7 Бережливе управління проектами (Lean)

Бережливе управління (Lean) поєднує принципи та методи для ефективного безвідходного проектування ланцюга створення вартості. Багато ідей виникли з виробничої системи японського автовиробника Toyota, яка розпочала свою тріумфальну ходу в середині 20 століття. Тим часом Lean-менеджмент також використовується в невиробничих сферах, таких як закупівля або управління проектами. Німецька асоціація управління проектами (GPM) визнала та додала Lean-менеджмент проектів у спеціальний розділ управління проектами [28].

Це дуже хороший приклад системи управління виробництвом, а її філософія також може внести свій внесок в управління проектами.

Представлені наразі процедурні моделі Scrum і Kanban не лише реалізують цінності маніфесту Agile, але й поділяють деякі принципи Lean виробництва. Вони зосереджуються на діяльності команди, що створює додаткову вартість, і уникає непотрібних планів і млявої структури проектів. Однак, незважаючи на багато схожості, керування проектами Lean не слід «прирівнювати до гнучкого управління проектами».

Розгляд теорії обмежень, описаний у [29], показує, як ці принципи можна застосувати до управління проектами Lean. Навіть якщо це не те саме, що і Lean-менеджмент, концепції допомагають нам сформулювати методи, які запобігають марнотратству в управлінні проектом, і показують, як це приносить користь загальному підходу до управління проектом.

У літературі про Lean-менеджмент (докладно знайдені різні принципи, що втілюють філософію Lean. У контексті управління проектами актуальними є наступні сфери, і з них можна вивести принципи Lean для управління проектами [30]. Ідея полягає в тому, що вся основна діяльність зосереджена на клієнтах та їхніх потребах. Після цього слідує ідентифікація потоку цінності. Ця ідея використовує підхід, який також широко використовується в Lean Management для управління проектами, щоб оптимізувати створення вартості для клієнтів і мінімізувати відходи. Цей перший крок в управлінні потоком створення цінності полягає в картографуванні потоків ресурсів та інформації через компанію. Це робиться з точки зору клієнтів під час виробництва або обробки замовлення, оскільки клієнти визначають виробничі вимоги та всі залучені процеси. Цей потік, потік цінностей, записується, оцінюється з часом і відображається у вигляді блок-схеми з простими стандартизованими символами.

Потік створення цінності включає всі етапи процесу та дії, які відбуваються під час кожної діяльності, як основні, так і допоміжні заходи, що не створюють додавання вартості. Критеріями для оцінки процесу та відправними точками для вдосконалення процесу є частка (що має бути зменшена) заходів, що не додають вартості, час виконання замовлення (що потрібно скоротити, якщо можливо) і частка часу обробки в часі виконання. Принцип витягування встановлюється, коли ресурс або діяльність є «на складі» або простою, що означає, що завдання перетягуються від одного кроку до іншого в процесі додавання вартості. При цьому наступний крок отримує завдання/матеріали/інформацію з попереднього кроку, який сигналізує про дії, які необхідно виконати, щоб уникнути простою чи очікування.

Необхідно проводити поточну оцінку, щоб дозволити безперервно вдосконалювати етапи процесу. Ці принципи Lean також можна застосувати до управління проектами. В основі їх лежить прагнення уникнути будь-яких відходів (час простою, час очікування, продукція на складі). Наприклад, звіт про стан проекту, який не використовується керівництвом, можна розглядати як типовий приклад марнотратних ресурсів у сфері управління проектами. Використовуючи ці 5 принципів (визначте цінність для клієнта, визначте діяльність із додавання вартості, забезпечте безперервний потік, встановіть принцип притягнення та постійно вдосконалюйте) створено бережливе управління проектом.

Перш ніж ближче розглянути принципи Lean і розглянути їх конкретні наслідки для управління проектами, може бути корисно коротко обговорити теорію обмежень Голдрата [29], щоб краще зрозуміти наслідки для управління проектом Lean.

1.8 Теорія обмежень (Theory of Constraints – TOC)

Ключове повідомлення теорії обмежень полягає в тому, що пропускна здатність системи визначається її заданими вузькими місцями. В управлінні проектом це означає, наприклад, що якщо всі завдання, виконані проектом, мають бути перевірені та задокументовані однією особою, це обмежує пропускну здатність функцій у всій системі (тут: проект). У цьому випадку немає сенсу вказувати чи розробляти результати швидше через вузьке місце на етапі тестування. Поки це вузьке місце залишається, виконання всієї системи (проекту) не може бути прискорене.

Для керівника проекту вже можна отримати конкретну інструкцію з цього спостереження: завжди необхідно спочатку виявити будь-яке вузьке місце в проекті, реформувати спосіб його функціонування, вжити подальших заходів, якщо необхідно, щоб забезпечити функціонування проекту. На рисунку 1.8 наведено деякі вказівки щодо того, як теорія обмежень може бути використана в

управлінні проектами, щоб гарантувати, що принципи Lean використовуються для покращення управління проектами.

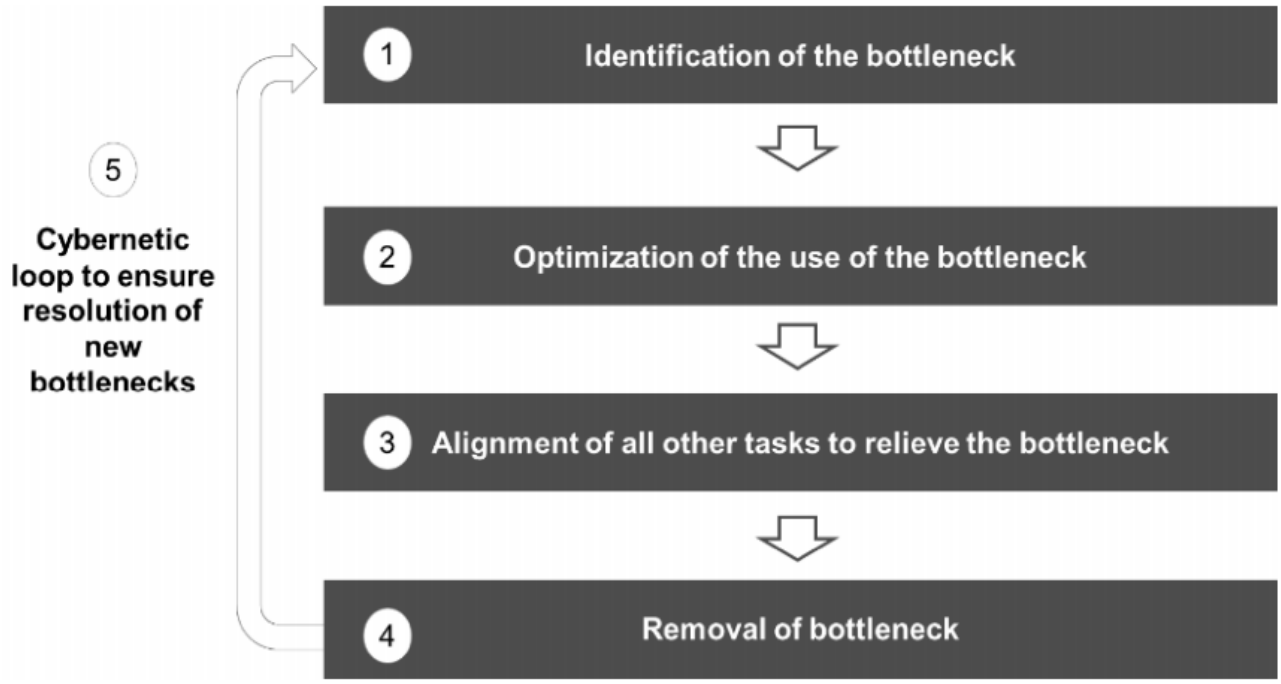

Рисунок 1.8 – Процес оптимізації вузьких місць для збільшення пропускної здатності в проектах

1. Виявлення вузького місця.

Якщо проект взагалі складний, керівнику проекту важко мати на увазі всі пакети робіт і працівників одночасно. Тому важливо зосередитися та втрутитися там, де пропускна здатність обмежена вузьким місцем проекту (див. пункт 1 на рисунку 2.8).

В управлінні проектами можна зробити спостереження, подібні до спостережень виробничої лінії, якщо старший керівник проекту може відповісти на такі запитання:

– Персонал чекає, тому що потрібно виконати інші завдання, чи субпідрядники все ще працюють? Чи означає це час очікування інших працівників?

– Чи скаржаться працівники на перевантаження, яке можна підтвердити доказами?

Хорошим підходом є порівняння запланованих витрат і термінів виконання з фактичними витратами та часом виконання. Наприклад, щоб візуалізувати хід і час виконання, можна використовувати дошку Канбан або традиційні діаграми Ганта або стовпчасті діаграми для візуалізації вузького місця.

2. Оптимізація розташування вузького місця.

Після визначення вузького місця наступним кроком є оптимізація його роботи в проекті. Знову ж таки, корисно перекласти виробничі принципи Lean Management та налаштувати їх на управління проектами:

– Заходи, які не сприяють успіху проекту, слід виключити (наприклад, діяльність лінійної функції – це може заплутатися в проектах через нечіткі ролі та відповідальність.

– Важливо також враховувати баланс поточної роботи. При цьому завдання, які можуть виконувати інші співробітники, якщо вони самі не є вузькими місцями, повинні бути передані їм.

3. Узгодження всіх інших завдань для усунення вузького місця

Ділянка, яка була визначена як вузьке місце, обмежує швидкість успішного завершення проекту. Тому немає сенсу інвестувати в оптимізацію інших завдань або співробітників – за двома важливими винятками:

– Визначте безкоштовні ресурси, які можуть усунути проблеми, беручи до себе деякі завдання на цьому етапі, які слід перепризначити для цієї мети.

– Визначте завдання перепризначення, які є вхідними для завдання вузького місця, щоб вузьке місце не чекало цих введень (тобто вузьке місце не повинно бути переміщено з однієї діяльності на іншу в проекті, де вона може знову заблокувати потік система). Вузьке місце обмежує загальну швидкість обробки проекту.

4. Усунення вузького місця.

З точки зору загального проекту чи програми важливо вживати коротко-, середньо- та довгострокових заходів для мінімізації вузьких місць. Приклади цього:

– Звільнення працівника від вузького місця шляхом доручення завдань іншим працівникам. Забезпечення того, що навички часто поширюються в організації проекту, наприклад, шляхом ротації посад.

– Покращення використовуваних процесів або інструментів, щоб працівники з вузьким місцем могли працювати більш ефективно. Наприклад, типовою проблемою проекту є необхідність чекати рішення керівництва через такі органи, як Керівний комітет. Необхідно знайти інші способи, які не забивають систему управління проектом, наприклад, процедури циркуляції цих вхідних даних.

– Додавання співробітників так що хтось може взяти на себе завдання, які тільки працівник вузького місця може колись виконати.

– Створення кібернетичного циклу для забезпечення стійкості.

Повторюючи ці кроки, нові вузькі місця проекту або програми ототожнювати і можуть бути вирішені, наприклад, в одному випадку машини може призвести до вузького місця, яке обмежує проект прогресу. Якщо з’являється нове вузьке місце, наприклад хвора людина з критичним завданням проекту, процес слід повторити, щоб переконатися, що це нове вузьке місце не сповільнює проект. Таким чином, регулярна перевірка (кібернетичний цикл, яка є наукою про комунікацію та теорію контролю) нових вузьких місць є важливою для того, щоб зробити управління проектом Lean стійким і вбудованим в організацію проекту.

Застосовуючи принципи Lean і принципи TOC, управління проектами Lean може закріпитися. Він може зробити це в контексті існуючих практик та інструментів (наприклад, дошка Kanban), щоб збагатити існуючі практики управління проектами, залежно від контексту, у формі гібридного підходу до управління проектами.

2 ГНУЧКІСТЬ У ПОРІВНЯННІ З ТРАДИЦІЙНИМ УПРАВЛІННЯМ ПРОЕКТАМИ

Протягом багатьох років методи водоспаду були повсюдним підходом до управління проектами. Вони характеризуються послідовним підходом і використовуються в різних галузях, особливо в розробці програмного забезпечення. Як ми бачили, назва походить від часто використовуваного графічного представлення фаз проекту, організованого у вигляді каскаду. Проекти, побудовані на моделі Waterfall, складаються зі статичних етапів, які обробляються у заздалегідь визначеній послідовності (аналіз вимог, проектування, тестування, впровадження, обслуговування).

Однією з сильних сторін методу водоспаду є широкі можливості контролю на кожній фазі проекту. Високоформалізований процес також підвищує ймовірність того, що всі вимоги будуть розглянуті заздалегідь. Однак ці переваги компенсуються серйозними недоліками з точки зору гнучкості, якщо, наприклад, загальні умови, припущення або ключові дані проекту змінюються під час його виконання. Особливо в проектах оцифрування такі зміни є скоріше правилом, ніж винятком. Отже, це одна з головних причин, чому модель водоспаду зазнала жорсткої критики в останні роки.

Гнучкі методи використовують принципово інший підхід до управління проектами. Вони були розроблені в першу чергу для проектів, де важливі гнучкість і швидкість. Agile проекти характеризуються короткими циклами виконання, які в основному організовані в так званих спринтах. Agile методи найкраще підходять для проектів, які вимагають відносно невеликого контролю, але потребують спілкування в реальному часі в невеликих, мотивованих командах. Agile управління проектами покладається на високий рівень інтерактивності між усіма залученими сторонами, що дозволяє швидко коригувати під час виконання проекту. Особливо в проектах розробки програмного забезпечення, Agile методи стали широко прийнятними і стали «новою нормою».

Таблиця 2.1 – Відмінності в традиційному та Agile підходах до управління проектами

Характеристика	Управління проектом водоспаду	Agile управління проектами
1.Філософія	Мінімізація ризику, орієнтація на процес	Швидкість доставки, орієнтація на людей
2.Плановий підхід	Лінійний і прогнозний, попереднє планування, поняття передбачуваність планування, Зосередженість на поданні цілісної картини	Ітераційний і адаптивний, мінімальне попереднє планування, поняття непередбачуваності планування і, отже, потреба в дослідницькому підході, більша зосередженість на баченні
3. Інструмент планування	Аналіз критичного шляху	Спринти та відставання
4. Фокус	Досягнення плану, повнота за обсягом	Цінність бізнесу, зосередьтеся на кількох видах діяльності, які мають найбільшу цінність
5. Значення Вимірювання	Контроль прогресу за допомогою управління одержаною вартістю (EVM)	Контроль прогресу за допомогою діаграми «вигорання».
6.Вимоги і Змінити Менеджмент	Фіксовані, детальні вимоги, визначені заздалегідь Зміни потрібно керувати та контролювати	гнучкий, вимоги, детально визначені під час роботи над проектом Зміни вітаються протягом усього проекту
7. Залучення користувачів і зацікавлених сторін	Керується планом управління зацікавленими сторонами на «як необхідна» основа	Рання та постійна участь, Спільне управління зосереджено на замовнику та постачальнику співробітництво
8. Прийняття рішень та структури проекту	Автократична, ієрархічна організаційна структура, більш схильна до «командування та контролю»	Децентралізоване, швидке та гнучке реагування на зміни до співпраці лідерів
9. Управління проектами циклів	Послідовний процес, зосередження на етапах проекту (наприклад,	Ітерації спринту, зосереджені на інкрементах продукту, які можуть бути доставлені як

Характеристика	Управління проектом водоспаду	Agile управління проектами
	просування по одній стадії, без перекриття)	корисний продукт клієнту або користувачу
10. Контроль проекту/Розмір проекту	Зустрічі проекту та керівний комітет, уроки, отримані наприкінці проекту	Щоденні стендап-зустрічі та поточні ретроспективи спринту
11. Складність	Великий і складний	Маленький і складний
12. Команда	Великі та спеціалізовані команди, Команди, керовані PMO	Невеликі мультидисциплінарні команди, команди, що самоорганізуються
13. Результати	Доставка продукції в кінці або на окремих етапах	Продукт працює з першого дня (мінімально життєздатний продукт), постійні вдосконалення
14. Документація	Високий наголос на документації	Низький акцент на документації; використання негласних знань
15. Спілкування	Офіційне спілкування	Неформальне спілкування

Проблеми можна виявити легше, а команди можуть вносити зміни на початку процесу розробки, не чекаючи завершення тривалих тестових запусків. Прихильники гнучких методів обіцяють, серед іншого, зниження ризиків вимог, миттєвий зворотний зв’язок, меншу складність і, зрештою, швидші результати.

Оскільки підходи Waterfall і Agile управління проектами настільки різні за філософією, мисленням і налаштуванням, можна очікувати, що підходи Waterfall добре вписуються в заздалегідь визначені та стабільні середовища (складні середовища), тоді як Agile, здається, більш пристосований до ситуацій, які є більш нестабільними. і менш передбачувані (складні середовища).

ЗТЕОРЕТИЧНЕ ОБҐРУНТУВАННЯ ДОСЛІДЖЕННЯ

3.1 Результати дослідження застосування гібридних підходів в софтверних компаніях

Для отримання емпірично обгрунтованої оцінки сучасного стану використання традиційної та гнучкої технології в глобальних проектах було виконано міжнародний дослідницький проект HELENA [4]. До його виконання було залучено 60 команд з 30 країн що охоплювали 5 континентів. В результаті було отримано та оброблено 263 відповіді на питання спеціально розроблених анкет

Для подальшого аналізу набору методів та можливостей їхнього комбінування було використано три категорії - традиційний, гнучкий, і загальний, де :

- Agile + Загальний = Agile;
- Традиційний + Загальний = Традиційний;
- Agile + Традиційний = Гібрид.

При цьому, як гнучкі було розглянуто та класифіковано наступні підходи: Scrum, Safe, Lean, LESS, Nexus, XP, Kanban, DevOps, ScrumBam, Crystal, DSDM та FDD. Аналогічно, як традиційні: водоспад, спіральна модель, V-модель, RUP, PRINCE2 та SSADM. Інші підходи класифікуються як загальні це: ітеративна розробка, розробка на основі предметної області (DDD), архітектура, керована моделями (MDA), командний процес ПЗ (TSP), [5]. Це обумовлено тим, що "загальні" процеси можуть бути або гнучкими, або традиційними - залежно від контексту. Виходячи із результатів кількісних та якісних опублікованих досліджень, можна зробити наступний розподіл проектів за категоріями (табл.3.1).

Таблиця 3.1 – Відсоток використання категорій методів в проектах

Клас категорії	%
Agile	25,0
Традиційний	2,0
Гібридний	72,0
Інший	1,0

Як видно із результатів, на практиці гібридні підходи прийняті в більшості (72%) проектів глобальної розробки, тоді як 25% використовують лише гнучкі методи, а 5% - лише традиційні.

Досліджувалась також ступінь використання гнучких технологій в гібридні категорії при розробці глобальних проектів (рис 3.1).

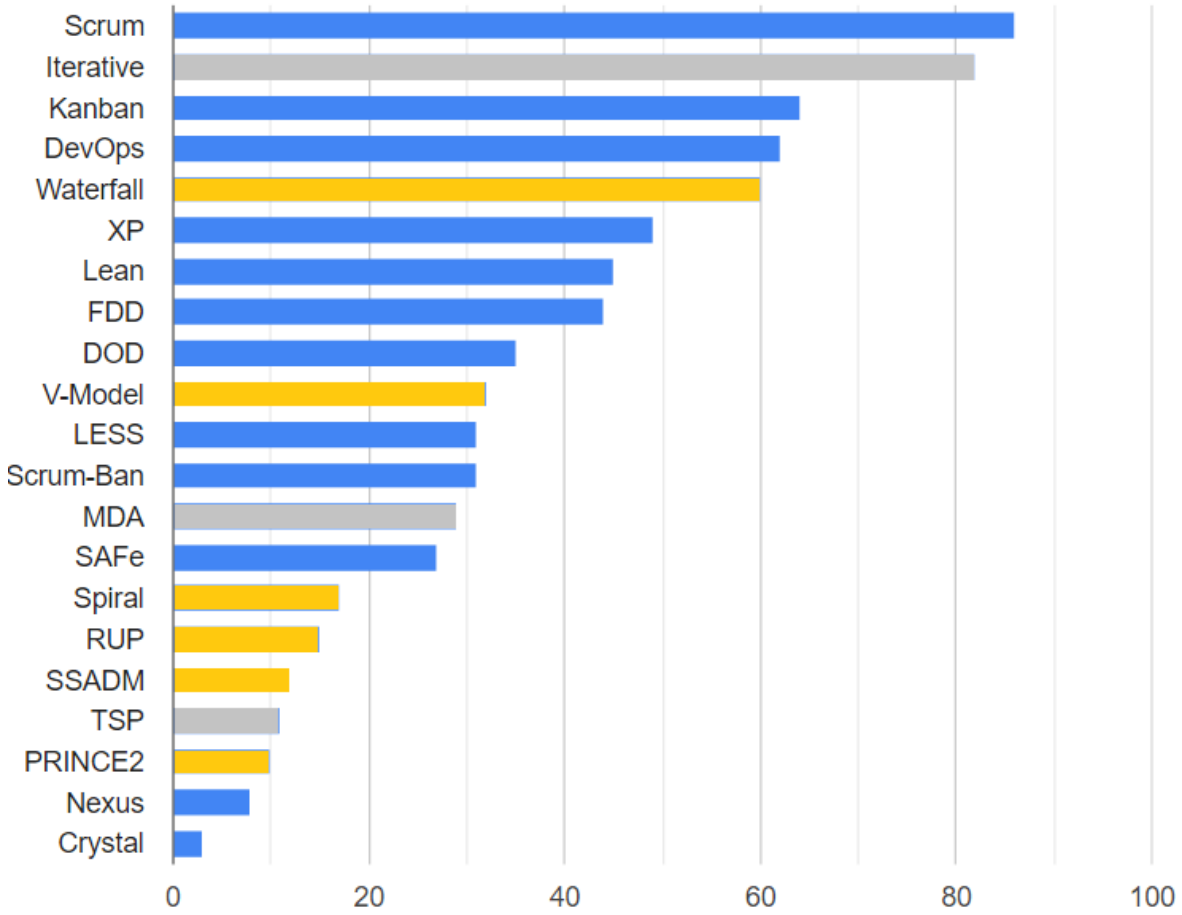

Рисунок 3.1 – Частота використання підходів

(гнучкий – синій, традиційний - помаранчевий, загальний - сірий)

За отриманими даними, можна зробити висновок, що серед гнучких підходів найчастіше використовуються Scrum та Kanban, а найрозповсюдженішим традиційним підходом є водоспад (waterfall). Також, глобальні проекти використовують більше підходів ітеративних розробок та підходів на основі Канбану. Серед фреймворків, які були створені для масштабування гнучких методів, переважають LESS та Safe, які розроблені на базі Nexus [2], [6].

На основі аналізу результатів опитування команд розробників, можна зробити висновок що вибір конкретного методу розробки базувався на досягненні наступних цілей:

- підвищення продуктивності;
- покращення процесу планування та оцінки;
- підвищення якості продукції;
- збільшення частоти поставок клієнтові.

Отже, в залежності від очікувань та вимог, кожна організація може обрати для себе найкращий варіант гібриду.

Таблиця 3.2 – Частота комбінацій гнучких та традиційних методів (де 1 – поширено, 2 – помірно)

	PRINCE2		RUP		Spiral		Waterfall		V-Model	
	1	2	1	2	1	2	1	2	1	2
Scrum (86%)	3%	8%	4%	13%	4%	14%	18%	41%	14%	19%
Kanban (64%)	3%	7%	3%	13%	3%	12%	19%	40%	17%	20%
Dev Ops (62%)	3%	9%	4%	11%	1%	14%	18%	38%	9%	19%
XP (49%)	2%	8%	6%	18%	2%	21%	17%	46%	9%	25%
Lean (45%)	4%	9%	6%	15%	6%	18%	17%	45%	20%	22%
FDD (44%)	3%	10%	3%	14%	7%	16%	22%	43%	20%	21%
LESS (31%)	4%	10%	7%	18%	4%	17%	18%	48%	13%	26%
Scrumban(31%)	4%	10%	6%	17%	4%	19%	22%	41%	15%	28%
SAFe (27%)	4%	15%	10%	22%	6%	21%	22%	51%	18%	25%
Nexus (8%)	10%	30%	10%	40%	5%	30%	15%	40%	10%	45%
Crystal (3%)	14%	43%	29%	43%	14%	43%	14%	43%	29%	43%

Для того, аби порівняти як часто традиційні та гнучкі методи поєднуються в проектах великого масштабу, результати було зведено до таблиці 3.2, де рядки показують конкретні гнучкі підходи, а стовпці показують частоту кожного традиційного підходу.

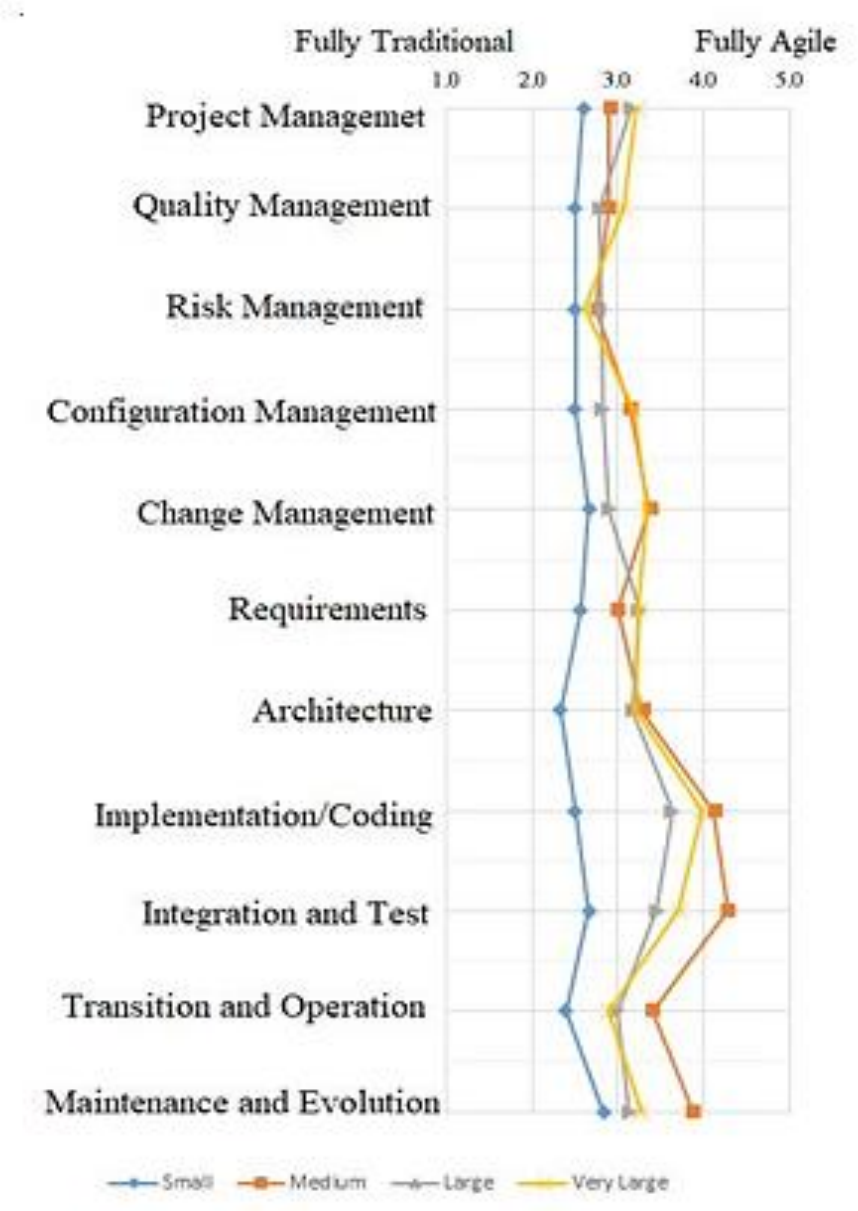

Рисунок 3.2 – Застосування підходів до виконання процесів при розробці проектів різного масштабу

Згідно результатів, гібридні проекти виконуються частіше, ніж суто гнучкі. Зрозуміло, що можливості адаптації та налаштування кожного з цих підходів повинні відповідати конкретним потребам поточної ситуації [6].

Частота використання певних підходів, та методів розробки для виконання стандартних процесів, зображена на графіку (рис. 3.2), де :

10 – повністю традиційний підхід;

20 – в більшості своїй традиційний;

30 – збалансоване використання традиційного та гнучкого підходу;

40 – переважно гнучкий підхід;

50 – повністю гнучкий.

Як видно із наведеного графіку, в більшості випадків вибір проводиться збалансовано між гнучкими та традиційними методами. Однак для таких процесів, як управління ризиками та управління якістю, частіше використовується традиційний підхід, тоді як заходи з розвитку, такі як впровадження/кодування та інтеграція/тестування, як правило, проводяться більш дотримуючись гнучких підходів [4].

Однак, для вирішення датування релізу, тестування та впровадження архітектури продукту, прийнята методологія у більшості випадків відповідає плановому підходу.

3.2 Архітектурне проектування, як процес покращення координації команд, та забезпечення якості програмної продукції в глобальних проектах

Базуючись на принципах Agile Маніфесту [7] практики Agile розглядають первинну розробку архітектури, як витрати часу та зусилля, які можуть не окупитись. Існувало переконання, що архітектуру можна створювати в ітераціях, шляхом простого рефакторінгу. В Agile приділяли основну увагу при плануванні ітерацій потребам замовника і користувацьким історіям, залишаючи поза увагою архітектурне проектування. Однак при застосуванні Agile до великих проектів ці погляди змінились. Так провідні фахівці в застосуванні

гнучких технік [8] показали, що ігнорування архітектурного проектування може спричинити неоптимальні рішення,а автор [9] стверджував, шо невідповідність розробки великих проектів архітектурному проекту, може привести до їх провалу. В роботах [10],[11] підкреслена ключова роль архітектурного проекту при застосуванні agile до розробки великих і складних проектів, а також фактичну неможливість їх виконання без використання архітектури. Це відбувається через втрату зв'язку беклогів з усією системою, що робить неможливим передбачити реальні наслідки своїх рішень при плануванні кожної ітерації. Для вирішення цих проблем були впроваджені в ітерації гнучких методів елементи архітектурного проектування та почали реалізовувати зв’язок архітектурного проекту всієї програмної системи з проектами локальних команд [3]. При реалізації великих проектів в рамках гнучких методів створювалась «координуюча» команда, яка вела архітектурний проект всієї програмної системи і здійснювала координацію внесення змін в архітектури компонентів, які розробляли локальні команди [3]. Для цього були введені нові ролі в координуючій команді, такі як архітектор системи, архітектор бізнес процесів та інші. Процеси архітектурного проектування в локальній команді може вести або архітектор, або один з членів команди. суміщати ці обов’язки [12].

Для забезпечення ефективності реалізації взаємодії процесів на локальному і глобальному рівнях необхідно було створити CASE засіб, який би автоматизував процеси створення та обміну архітектурної інформації. В [13] було запропоновано двохетапну схему, зображену на рис. 3.3. На першому етапі з врахуванням вимог створюються альтернативні варіанти архітектури за технологією, яка залежить від прийнятого стилю документування

Для оцінювання альтернатив обчислюються їх відносні оцінки по кожному з критеріїв якості з використанням модифікованого методу аналізу ієрархій (MAI). Вибір кращого варіанта архітектури виконується методом аналізу компромісів, або по значенню інтегрального показника якості.

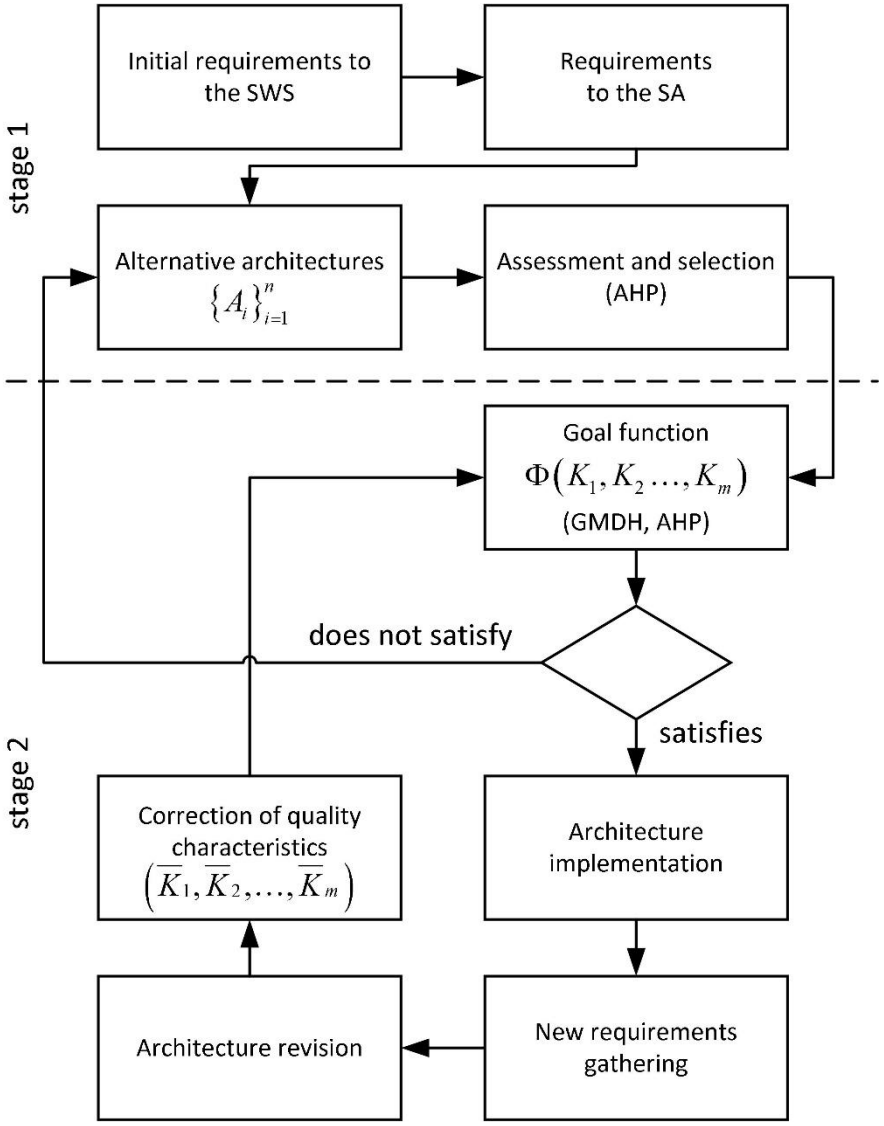

Рисунок 3.3 – Схема процесу впровадження архітектурного проектування з гнучкими методами

На другому етапі вибраний варіант архітектури впроваджується в гнучкому методі розробки. В разі виявлення нових вимог до ПС, або зміни існуючих, вносяться відповідні зміни в архітектуру. Зміни виконуються шляхом корегування коду відповідного патерну, або його заміною на альтернативний. Для дослідження, та оптимізації впливу цих змін на якість проводиться корекція значень критеріїв якості. Процедура корекції розроблена на основі застосування методу «заміщення – компенсації», в якому збільшення деяких критеріїв якості відбувається за рахунок зменшення інших, що дозволяє оптимізувати ці зміни і не виходити за межі бюджету проекту.

Перший і другий етапи об’єднуються процедурою обчислення цільової функції якості архітектури. Цільова функція визначається методом групового урахування аргументів в поєднанні з МАІ [13]. Запропонована процедура дозволить крім координації процесів також забезпечити якість проєкту.

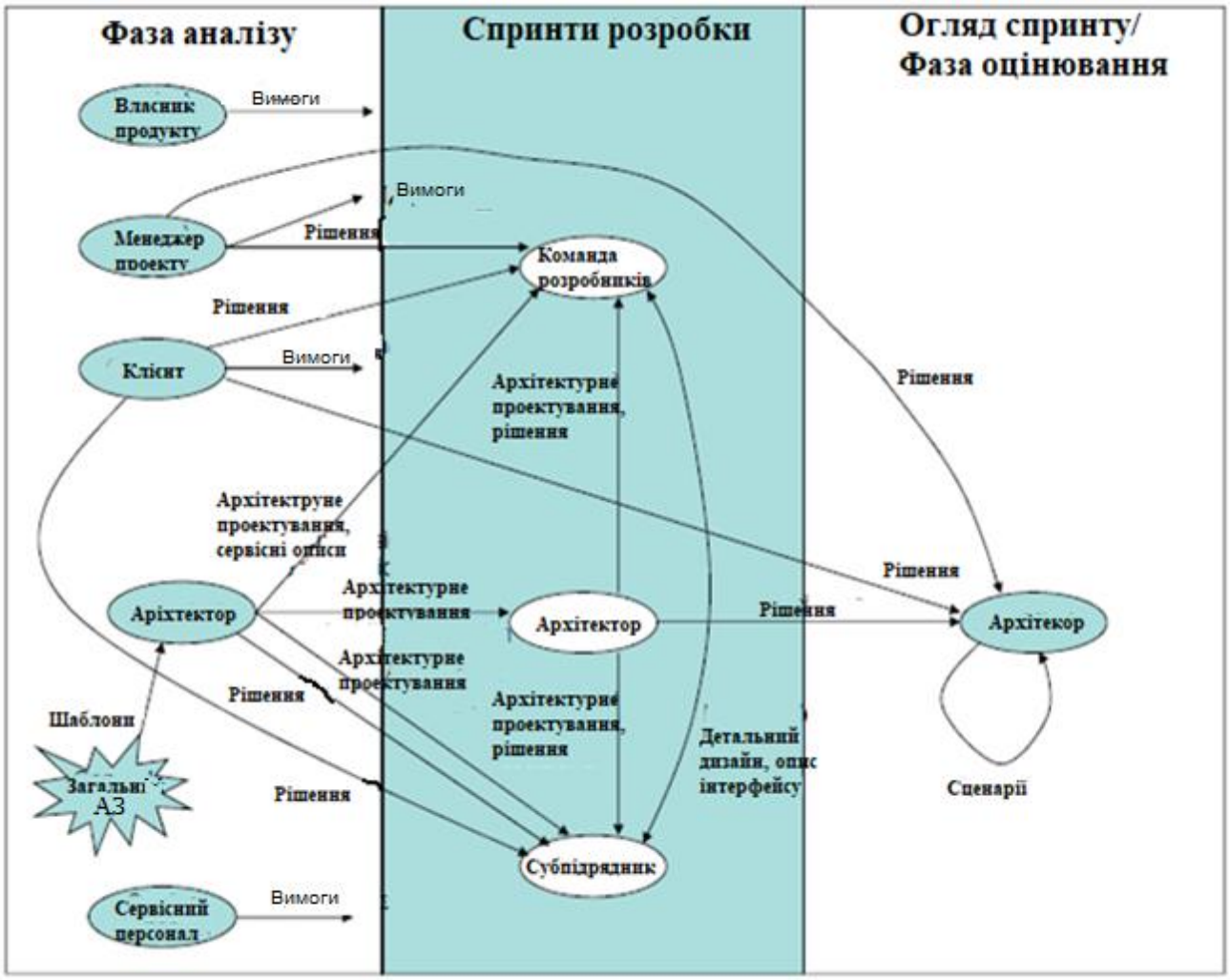

Рисунок 3.4 – Архітектурний інформаційний потік в спринтах Scrum

Для адаптації процесів архітектурного проектування до вимог Agile необхідно максимально зменшити час, який команди витрачатимуть на процеси архітектурного проектування. Для цього пропонується створити систему управління знаннями з архітектури, яка б могла підтримувати спільне використання командами архітектурної інформації. В ній повинні реалізовуватись всі процеси, передбачені в схемі рис. 3.4. Ці процедури повинні

виконуватись командами в ітераціях Agile, а також координуючою командою для архітектури всієї системи.

Для розробки структури такої системи необхідно розглянути інформаційні потоки які будуть реалізуватись в ітераціях гнучких технологій. На рис. 3.5 представлено схему потоків для випадку, коли архітектурне проектування виконується в нульовому спринті методу Scrum [16]. Тобто тут розробляється розширений архітектурний проект архітектором групи, або членом команди.

Тоді можлива архітектура системи управління архітектурними знаннями буде мати наступну структуру.

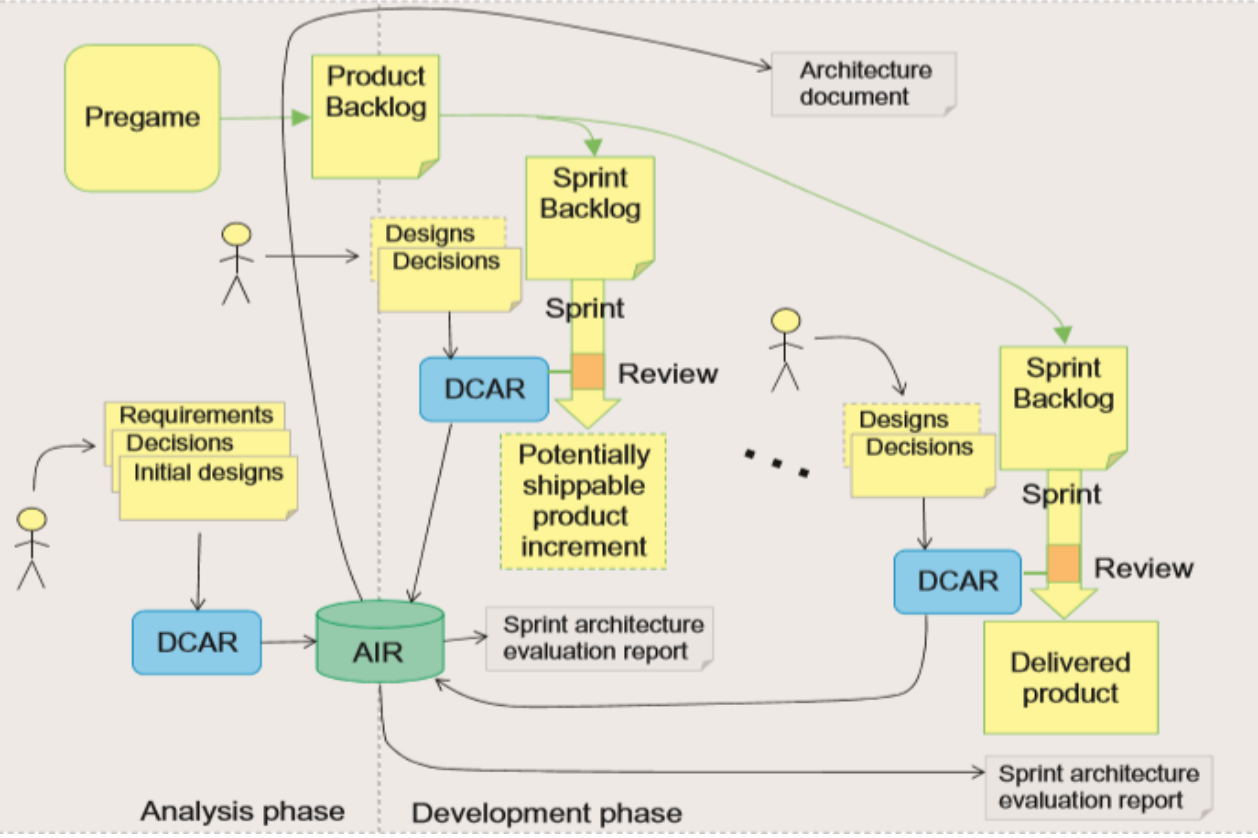

Рисунок 3.5 – Структура системи управління архітектурними знаннями

У підході з розширеною первинної архітектурою більшість архітектурних рішень, а також комплексний архітектурний проект приймається і створюється на фазі аналізу (або, в разі нульового спринту, на першому спринті), перед

спринтом розробки (рис. 3.4). Проте, цілком реалістично припустити, що під час розробки необхідно змінювати архітектуру, і деякі нові рішення приймаються командою Scrum або архітектором.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Покращення безпеки праці в українських ІТ-компаніях, як результат їх взаємодії з західними компаніями

Поява та впровадження нових інформаційно-комунікаційних технологій зумовлює необхідність подальшого вдосконалення охорони праці фахівців ІТ-індустрії. З метою належного правового забезпечення необхідно розширити ти доповнити перелік основних професій комп'ютерної галузі у національному класифікаторі ДК-003-2010, а також підготувати відповідний випуск у кваліфікаційному довіднику посад фахівців ІТ-індустрії, що сприятиме вирішенню питань їх соціального захисту, пенсійного забезпечення, атестації робочих місць основних професій за умовами праці на предмет подальших певних видів пільг та компенсацій за важкі шкідливі і небезпечні умови праці. Важливим напрямом стосовно визначення професійної придатності фахівців з інформаційних технологій є проведення психофізіологічної експертизи відповідно до 5 статті Закону України «Про охорону праці».

Робота з комп'ютерами нового покоління характеризується певним психофізіологічними перенавантаженнями, втомою зорового аналізатора, гіпокінезією, відсутність диференційованих норм праці при роботі з новою комп'ютерною технікою в залежності від віку, статі, категорії зорової роботи, режимів праці і відпочинку (протягом робочого дня, тижня, щорічного режиму відпусток).Все це потребує розробки нових нормативно-правових актів з регламентації праці та відпочинку фахівців ІТ-індустрії і стандартів підприємств, центрів комп'ютерної техніки, центрів інформаційних технологій, сучасних комп'ютерних класів.

Особлива роль з точки зору збереження та відновлення здоров'я працюючих в комп'ютерні галузі належить попереднім та періодичним

наглядам з подальшої психофізіологічної експертизи і встановленням професійної придатності при роботі з комп'ютерами нового покоління, який супроводжується виникненням певних факторів професійного ризику електро-травматизму при їх ремонті та обслуговуванні. В цьому зв'язку необхідне запровадження експертизи на предмет безпечної експлуатації ПЕОМ, тобто офіційне підтвердження фактичних параметрів електробезпеки, їх відповідності вимогам нормативної документації фахівців, які проводять таку експертизу повинні пройти навчання і перевірку знань відповідно до вимог ДНАОП 0.00-8.20-99.

За результатами експертизи повинні прийматися рішення про відповідність ПЕОМ нормам безпеки, терміни чергової експертизи, оформлюються протоколи вимірювань і випробувань, проведені у разі потреби розрахунки та експертний висновок. Для підвищення розумової працездатності то зорової роботи повинна здійснюватися ергономічна оптимізація в рамках системи «оператор-термінал», яка сприятиме результативній фізичній та інтелектуальний працездатності і відновленню психосоматичного здоров'я фахівців ІТ-індустрії [31].

Заслуговує на увагу зарубіжний досвід створення у приміщеннях та в зоні їх розміщення на територіях підприємств спеціальних візуальних комфортних умов та забезпечення вимог виробничої естетики, дотримання норм рівнів виробничого шуму та акустичної тиші за межами офісу. Також дуже важливим є використання в офісних приміщеннях та кабінетах психофізіологічного розвантаження функціональної музики, яка сприяє попередженню перевтоми і підтриманню необхідного рівня розумової працездатності фахівців комп'ютерної галузі [32], [33].

В цьому напрямі заслуговує на увагу створення при великих центрах інформаційних технологій кімнат (кабінетів) психофізіологічного розвантаження працівників галузі (на 5 місць).Зарубіжний досвід охорони праці при використанні новітніх інформаційних технологій та сучасного комп'ютерного обладнання передбачає з метою попередження наслідків

монотонної праці, підвищення рівня рухової активності і покращення розумової працездатності фахівців ІТ-індустрії під час технологічних перерв участь у спеціальних облаштованих приміщеннях необхідним спортивним інвентарем та різними тренажерами відповідних фізичних вправ, індивідуальних тренінгових завдань відповідно до віку, статі та категорії зорової роботи.

Такий підхід дозволяє зняти надлишкове психофізіологічне перевантаження, підвищити працездатність центральної нервової системи, попередити перевтому зорового аналізатора. Показана ефективність проведення різноманітних за своєю спрямованістю вправ робітників цієї галузі (приблизно на 5-30%).

Всі наведені заходи щодо вдосконалення охорони праці фахівців ІТ-індустрії повинні контролюватися службою охорони праці та комісією з охорони праці підприємства. Особливе значення у соціальному захисті цієї категорії працівників належить прийняття комплексного договору, який може забезпечити фахівців додатковими пільгами та компенсаціями.

4.2 Вплив електромагнітного імпульсу (ЕМІ) ядерного вибуху на елементи виробництва та заходи захисту

Ядерні вибухи в атмосфері й більш високих шарах призводять до виникнення потужних електромагнітних полів з довжиною хвиль від 1 до 1000 м і більше. Ці поля через короткочасне існування називають електромагнітним імпульсом (ЕМІ). ЕМІ виникає при ядерному вибусі у воєнний час, у мирний час – при випробуванні ядерної зброї або ядерних аваріях і катастрофах в атмосфері й космосі.

Основною причиною виникнення ЕМІ тривалістю менше 1 с вважають взаємодію гамма-променів і нейтронів ядерного вибуху з атомами газів повітря, внаслідок чого з них вибиваються електрони (ефект Комнтона) і хаотично розлітаються в середовищі позитивно заряджених атомів газів. Важливе значення має також виникнення асиметрії в розподілі просторових електричних

зарядів, пов'язаних з особливостями поширення гамма-променів і утворення електронів.

Гамма-промені, які випускаються із зони вибуху в напрямі поверхні землі, поглинаються в більш щільних шарах атмосфери, вибиваючи з атомів повітря швидкі електрони, які летять у напрямку гамма-променів зі швидкістю світла, а позитивні іони (залишки атомів) залишаються на місці. У результаті поділу і переміщення позитивних і негативних зарядів у цій області й у зоні вибуху, а також при взаємодії зарядів з геомагнітним полем Землі утворюються елементарні й результуючі електричні та магнітні поля ЕМІ, які досягають поверхні землі в зоні радіусом кількох сотень кілометрів. Виникають сильні поперечні токи і утворюється подібність великої "плоскої антени", яка випромінює потужний ЕМІ з часом наростання порядна 10 не і тривалістю більше 230 не; зі смугою частот від 10 кГц до 100 МГц. Залежно від висоти ядерного вибуху за інших однакових умов змінюються характер, інтенсивність ЕМІ і дальність його поширення.

При наземному і низькому повітряному вибуху уражаюча дія НМІ спостерігається на відстані кількох кілометрів від центру вибуху. Під час ядерного вибуху на висотах від 3 до 25 км утворюється симетричне джерело генерації, але радіус поширення ЕМІ залишається обмеженим внаслідок сильного поглинання гамма-випромінювання в щільних шарах атмосфери.

Найбільшу уражаючу дію має ЕМІ, що виникає при екзоатмосферному вибуху (більше 40 км). Зі збільшенням висоти вибуху збільшується і район джерела генерації ЕМІ, досягаючи в діаметрі тисячі кілометрів і товщини 20–40 км. Так, під час вибуху на висоті 80 км.

ЕМІ буде поширюватися на площі радіусом 960 км, а під час вибуху на висоті 160 км – на площі радіусом 1400 км. Екзоатмосферний ЕМІ характеризується дуже малим часом наростання (декілька сотень наносекунд), високою інтенсивністю електричного поля (більше 50 кВ/хв) і магнітного поля (близько 130 А/хв). Розряд блискавки порівняно з ЕМІ має значно більшу тривалість зростання і спаду (5–300 мКе), створює дуже потужні поля (близько

100 кВ/хв), несе значно більшу енергію, але спектр частот становить близько 10 МГц, тоді як для ЕМІ він більше 100 МГц. Пікове значення ЕМІ може досягти 50 000 В/хв, що дорівнює всій енергії яка випромінюється в радіочастотній частині спектра. Уражаюча дія ЕМІ обумовлена виникненням напруги і струмів у провідниках різної довжини, розміщених у повітрі, землі.

ЕМІ захвачують спектр частот від десятків до кількох сотень мегагерц, тобто діапазон, в якому працюють установки електропостачання, зв'язку і радіолокації. Напруженість електромагнітного поля, створюваного ЕМІ, досягає 50 000 В/м, тоді як у радіолокації вона не перевищує 200 В/м, а у зв'язку – 10 В/м.

У серпні 1958 р. у момент заатмосферного термоядерного вибуху, проведеного США над островом Джонсон, на Гавайях, які знаходяться за 1000 км від епіцентру вибуху, погасло освітлення на вулицях. Це сталося в результаті дії ЕМІ на повітряні лінії електропередач, які відіграли роль протяжних антен.

Величина ЕМІ залежно від ступеня асиметрії вибуху може бути різною – від десятків до сотень кіловольт на метр антени, тоді як чутливість звичайних УДК-приймачів становить кілька десятків або сотень мікровольт. Так, у разі наземного вибуху потужністю 1 Мт напруженість поля на відстані 3 км становить близько 50 кВ/м, а на відстані 16 км – 1 кВ/м. А у разі заатмосферного вибуху такої ж потужності напруженість поля становитиме тисячі кіловольт на метр площі в кілька тисяч квадратних кілометрів земної поверхні.

Уражаюча дія ЕМІ в приземній області й на землі пов'язана з акумулюванням його енергії довгими металевими предметами, рамними і каркасними конструкціями, антенами, лініями електропередачі та зв'язку, в них виникають сильні наведені струми, які руйнують підключене електронне та інше чутливе устаткування. У районі дії ЕМІ безпосередній контакт людини зі струмопровідними предметами небезпечний.

ЕМІ уражає радіоелектронну і радіотехнічну апаратуру. В провідниках індукуються високі напруги і струми, які можуть призвести до постійних або тимчасових пошкоджень ізоляції кабелів, відключення реле і переривників, пошкодження елементів зв'язку, магнітних запам'ятовуючих пристроїв у ЕОМ і

системах передачі даних тощо. Найбільш уразливими елементами обладнання є напівпровідникові прилади – транзистори, діоди, випрямлячі, інтегруючі кола, цифрові процесори, управляючі й контрольні прилади. Чутливі до пошкодження ЕМІ транзистори звукової частоти, перемикаючі транзистори, інтегруючі кола та ін.

Особливо чутливими до впливу ЕМІ є 6 основних груп об'єктів і систем:

1) системи передачі електроенергії: повітряні ЛЕП, кабельні лінії, різні види з'єднувальних ліній і повітряна електропроводка;

2) системи виробництва, перетворення і накопичення енергії: електростанції, генератори постійного і змінного струму, трансформатори, перетворювачі струмів і напруг, комутатори і розподільні пристрої, електричні батареї і акумулятори, паливні, сонячні й термоелементи;

3) системи регулювання і управління: електромеханічні й електронні датчики та інші елементи автоматики, комп'ютерні установки, м і к ро п роцесори;

4) системи споживання електроенергії: електродвигуни і електромагнітні, нагрівальні, холодильні, вентиляційні, освітлювальні установки та кондиціонери;

5) системи електротяги: електроприводи, напівпровідникові та інші типи перетворювачів;

6) системи радіозв'язку, передачі, зберігання і накопичення інформації: антени, хвилеводи, коаксильні кабелі, електронні прилади, радіопередавачі, радіоприймачі, установки автономного електропостачання, змішувачі, телефонні апарати, телеграфні установки, заземлені кабелі й проводи, АТС.

Найбільш стійкі до ЕМІ вакуумні електронні прилади, які виходять із ладу при енергії 1 Дж. Величина енергії ЕМІ залежить від ширини періоду частот антенних систем.

Більшість систем зв'язку працюють у діапазоні частот від середніх до ультрависоких і будуть пошкодженими залежно від робочого періоду частот. Радіолокаційні системи менше пошкоджуються від ЕМІ, тому що вони

працюють у періоді частот, де щільність енергії ЕМІ невелика. Іскріння, яке виникає під впливом високого електричного поля ЕМІ, може спричинити спалахування парів бензину та інших налив у сховищах.

Якщо ядерний вибух стався поблизу лінії електропостачання, зв'язку великої довжини, то наведені в них напруги можуть поширюватися по проводах на багато кілометрів, пошкоджувати апаратуру й уражати людей, які знаходяться на безпечній відстані відносно інших уражаючих факторів ядерного вибуху.

ЕМІ небезпечний і за наявності міцних споруд, розрахованих на стійкість проти ударної хвилі наземного ядерного вибуху, проведеного на відстані кількох сотень метрів.

Сучасний рівень знань про природу і властивості ЕМІ дає можливість розробити захист від нього і впровадити заходи захисту до яких входять схеми, стійкі до електромагнітної інтерференції, радіоелектронні елементи стійкі до ЕМІ, екранування окремих пристроїв або цілих електронних систем.

ВИСНОВКИ

Керівники проектів стикаються з тим фактом, що раніше підхід, що вважався єдино правильним, «один підходить для всіх» насправді не існує і що застосування традиційного підходу до всіх проблем управління проектами здається застарілим. Раніше керівники проектів дотримувались традиційного підходу Waterfall та його варіацій. Сьогодні в центрі більшості проектних зусиль є вимоги клієнтів, які часто не повністю зрозумілі.

Гнучкі підходи до управління проектами дозволяють компаніям швидше виходити на ринок і переносити функції, які цінуються клієнтом, у центр поставок. Однак гнучкий підхід не можна застосовувати до кожного результату проекту або підзавдання. Отже, в майбутньому керівнику проекту все більше доведеться використовувати гібридні підходи, вибираючи хорошу комбінацію існуючих методів та інструментів управління проектами. Такого роду налаштування управління проектами буде продовжувати рости та ставати все більш важливими з часом. Це, як наслідок, потребує вказівок, коли використовувати підхід до управління проектами, що лежить в основі цього дослідження.

Дослідження показало, що більшість з майже 80% керівників проектів використовують гібридне управління проектами, і тенденція відійшла від застосування лише одного стандарту найкращої практики, що можна підтвердити в цьому дослідженні. Незважаючи на те, що використовуються всі три гібридні моделі комбінацій (послідовне, паралельне, повністю інтегроване застосування іншої моделі процесу управління проектом), найбільш поширеною комбінацією є інтегроване застосування різних моделей процесу управління проектами.

Що стосується комбінаційної моделі, то дослідження показало, що більшість моделей комбінацій випливає з традиційної моделі управління проектами Waterfall, яка може бути збагачена іншими методами управління

проектами (у 53% випадків). Часто для збагачення моделі процесу Waterfall використовуються принципи та інструменти, що застосовуються до адаптації.

Дослідження показало, що збагачення за допомогою практик Scrum є дуже поширеним комбінованим шаблоном, який в основному використовується під час доставки або виконання проекту (також званий комбінаційний шаблон Water-Scrum-Fall). Причину цього можна побачити в договірних аспектах.

Далі слідує управління проектами Kanban, яке додається до традиційних практик управління проектами, щоб забезпечити оптимальний потік запланованих завдань. Крім того, Agile управління проектами використовує інші методи управління проектами. Вони в основному надходять від управління проектами Kanban і часто збагачуються практикою Design Thinking і DevOps. Таке мислення закладає основу гібридного підходу до управління проектами, орієнтованого на контекст, і є чітким доказом відходу від монолітного мислення асоціацій управління проектами єдиних стандартів «найкращої практики» в управлінні проектами («один стандарт підходить всім»).

Як ми бачимо з кроку PMI щодо впровадження Disciplined Agile (DA), асоціації з управління проектами, що рухаються до нового способу роботи, стануть чудовим джерелом для наукових кіл, щоб визначити, чи може цей новий підхід – від однієї найкращої практики – до умовного підходу призвести до кращої ефективності управління проектами та чіткого прогресу в практиці управління проектами.

СПИСОК ВИКОРСИТАНИХ ДЖЕРЕЛ

1. Torgeir Dingsøyr, Nils Brede Moe. Key Lessons from Tailoring Agile Methods for Large-Scale Software Development. Project Management Journal · January 2018

2. Philipp Diebold, Anna Schmitt, Sven Theobald. Scaling Agile – How to Select the Most Appropriate Framework. [Електронний ресурс] https://www.agilealliance.org/agile101/subway-map-to agile-practices

3. Torgeir Dingsøyr, Nils Brede Moe, and Eva Amdahl Seim Coordinating Knowledge Work in Multiteam Programs: Findings from a Large-Scale Agile Development Program. Project Management Journal Vol. 49(6) 64–77. 2018.

4. Marcelo Marinhoa,b, John Nollc,b, Ita Richardsonb, Sarah Beecham Plan-Driven Approaches Are Alive and Kicking in Agile Global Software Development. [cs.SE Jun 2019], [Електронний ресурс]. Режим доступу: https://www.researchgate.net/publication

5. Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. 2008. Selecting empirical methods for software engineering research. In Guide to advanced empirical software engineering. Springer, 2008, pp. 285–311

6. Torgeir Dingsøyr, Nils Brede Moe, Helena Holmström. Towards an +-Understanding of Scaling Frameworks and Business Agility. A Summary of the 6th International Workshop at XP2018. Електронний ресурс.: https://www.researchgate.net/publication/332854548.

7. Agile alliance: manifesto for agile software development. Available at: http:// agilemanifesto.org [retrieved 10.01.2012]

8. Dyba T, Dingsoyr T. Empirical studies of agile software development: a systematic review. Inform Softw Tech 2008;50(9–10):833–59

9. Bowers J, et al. Tailoring XP for large system mission critical software development. In: 2nd XP universe and first agile universe conference on extreme programming and agile methods XP/agile universe. Berlin: Springer-Verlag; 2012.

10. Ghanam Y. Andreichuk D., Maurer F. Reactive variability management in agile Software development. In: Agile conference. Orlando, FL: Computer Society; 2010, p. 27-34.

11. Abrahamsson P., Babar A.M., Krutchen P. Agile and Architecture- can the co- exist? IEEE Software 2010, 27(2): p.16-22.

12. Eloranta V-P, Koskimies K. Software architecture practices in agile enterprises. In: Mistrik I, Tang A, Bahsoon R, Stafford JA, editors. Aligning enterprise, system, and software architectures. Hershey, PA: IGI Global, 2012: pp. 312-344.

13. Olexandr Kharchenko Ihor Bodnarchuk, Volodymyr Lisovyi, Iryna Galay. Adaptive Method for Assessment and Selection of Software Architecture in Flexible Techniques of Design. Proceedings of the YVII International Scientific and Technical Conference CSIT 2018, -Lviv 2018–pp.76 -83.

14. Kharchenko A. Bodnarchuk I., Galay I. Multicriteria Architecture Choice of Software Under Design and Reengineering. Proceedings of the YV International Scientific and Technical Conference CSIT 2016, -Lviv 2016–pp.76 -83

15. Харченко О.Г. Боднарчук І.О. Галай І.О., Інструментальний засіб порівняльного оцінювання і багатокритеріального вибору архітектури програмних систем. Інженерія програмного забезпечення., 2015.-№1 Київ НАУ,с.10-24

16. Naslund, Dag, and Rahul Kale. "Is agile the latest management fad? A review of success factors of agile transformations." International Journal of Quality and Service Sciences (2020).

17. Piazza, Alessandro, and Eric Abrahamson. "Fads and fashions in management practices: Taking stock and looking forward." International Journal of Management Reviews 22.3 (2020): 264-286.

18. Madsen, Dag Øivind. "The evolutionary trajectory of the agile concept viewed from a management fashion perspective." Social Sciences 9.5 (2020): 69.

19. Kahan, Seth. "The Age of Agile: a guide to a revolution in innovation management." Strategy & Leadership (2018).

20. Kreslin, Damijan, et al. "Use of management tools and return on equity." International Journal of Management and Enterprise Development 20.4 (2021): 347-362.

21. Liu, Hsun-Wen Lisa. Balancing between agile and plan-driven software development methods to minimise project risk and improve quality. Diss. Glasgow Caledonian University, 2013.

22. Rubin, Kenneth S. Essential Scrum: A practical guide to the most popular Agile process. Addison-Wesley, 2012.

23. Pries, Kim H., and Jon M. Quigley. Scrum project management. CRC press, 2010.

24. Rothman, Johanna, and Mark Kilby. From Chaos to Successful Distributed Agile Teams: Collaborate to Deliver. Practical Ink, 2019.

25. Kurnia, Reni, Ridi Ferdiana, and Sunu Wibirama. "Software metrics classification for agile scrum process: a literature review." 2018 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI). IEEE, 2018.

26. Imani, Takeomi, Masaru Nakano, and Vittal Anantatmula. "Does a hybrid approach of agile and plan-driven methods work better for IT system development projects." International journal of engineering research and applications 1.2 (2017): 3.

27. Anderson, David J. Kanban: successful evolutionary change for your technology business. Blue Hole Press, 2010.

28. Erne, Rainer. Lean Project Management-wie man den Lean-Gedanken im Projektmanagement einsetzen kann. Wiesbaden: Springer Gabler, 2019.

29. Goldratt, Eliyahu M., and Jeff Cox. The goal: a process of ongoing improvement. Routledge, 2016.

30. Timinger, Holger. Modernes Projektmanagement: Mit traditionellem, agilem und hybridem Vorgehen zum Erfolg. John Wiley & Sons, 2017.

31. Вдосконалення охорони праці в IT-індустрії. // Харківський національний дорожний університет. [Електронний ресурс]. – Режим доступу: https://www.khadi.kharkov.ua/fileadmin/P_vcheniy_secretar/%D0%9E%D0%A5%D

0%9E%D0%A0%D0%9E%D0%9D%D0%90_%D0%9F%D0%A0%D0%90%D0%A6%D0%86/R_IT-INDUSTRIA.pdf

32. Сьогодні UA [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.segodnya.ua/lifestyle/fun/pochti-kak-u-google-chem-udivlyayut-ofisy-ukrainskih-it-kompaniy-764025.html

33. MRPL.CITY [Електронний ресурс]: [Веб-сайт]. – Електроні дані. – Режим доступу: https://mrpl.city/news/view/mariupolskaya-konditerka-stanet-biznes-tsentrom-foto-plusvideo

ДОДАТКИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

IX НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

8–9 грудня 2021 року

ТЕРНОПІЛЬ
2021

УДК 004.42
Н. Базюк, А. Флейтута
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ІНЖЕНЕРІЯ ВИМОГ ДО ПРОГРАМНОГО ПРОДУКТУ В ГНУЧКИХ ТЕХНОЛОГІЯХ РОЗРОБКИ

UDC 004.42
N. Baziuk, A. Fleituta

SOFTWARE REQUIREMENTS ENGINEERING IN AGILE DEVELOPMENT

Управління ризиками визнано однією з важливих галузей знань в управлінні проєктами. Підхід до управління ризиками відрізняється як для традиційної, так і для гнучкої методології управління проєктами. Попередні дослідження з управління ризиками підкреслюють, що традиційне управління ризиками є процесом, який керується важким планом, і передбачає, що більшість вимог проєкту наявні до початку проєкту. З іншого боку, Agile-менеджмент проєктів використовує методологію, за якою результати поставляються ітераційно. Таким чином, гнучка методологія спрямована на мінімізацію тривалості проєкту, визначення пріоритетності результатів з високою вартістю, а також покращення якості результатів проєкту та зниження ризику.

Цілі застосування традиційного та гнучкого підходу в управлінні проєктами різні і дають найкращий результат лише за певних обставин. Традиційний підхід до управління ризиками проєкту, управління ризиками виконується під час ініціації проєкту, і це підходить, оскільки проєкти з високими термінами вимагають належного планування через поєднання ризиків, розтягнутих вздовж терміну.

У гнучкому підході до управління ризиками проєкту, через ітераційний характер управління, очевидно, що процеси управління ризиками виконуються через часті проміжки часу, оскільки проєкт розділений на кілька фаз. Це допомагає детально визначити ризики, пов'язані з етапами проєкту, їх серйозність та вплив. Крім того, оскільки фази розділені, терміни, протягом яких необхідно виконувати управління ризиками, скорочуються, що допомагає краще зрозуміти природу ризику та зменшує його невизначеність. Також ймовірність пропуску будь-якого ризику мінімізується за допомогою гнучкого підходу до управління ризиками проєкту.

В Agile проєктах найкритичнішими є проблеми, пов'язані з людьми. Справді, одним з найважливіших факторів успіху в Agile проєкті є індивідуальна компетентність. Крім того, оцінка зусиль є постійною задачею для команди розробників Agile, особливо коли вона виконується вперше, і є проблеми з навичками Agile і плинністю кадрів. У Scrum індивідуальна мотивація дуже важлива і впливає на те, наскільки старанні члени команди.

Визнання недотримання усталеної практики може дати ранні ознаки ризиків, наприклад, низький моральний дух, виражений під час щоденної зустрічі, або уникнення обговорення проблем у разі відставання від графіка. Через виявлені проблеми існує сильна мотивація покращити управління ризиками в Agile проєктах, не зменшуючи їх гнучкість. Сучасне управління ризиками має бути в змозі інтегруватися в гнучкий процес для підтримки прийняття рішень.

Процес управління ризиками застосовується відповідно до типу, розміру та складності проєкту. Agile Risk Management здійснюється більше за допомогою практик, ніж уявлення. Багато практик Agile спрямовані на виявлення та пом'якшення ризиків протягом усього проєкту. Однак у реальному житті більшість проєктів вимагають поєднання обох підходів, і можна зробити висновок, що як гнучкі, так і традиційні методології управління ризиками доповнюють одна одну. Крім того, розвиток моделі ARM зменшує зусилля в управлінні ризиком.