

БАКАЛАВРСЬКА РОБОТА

БР. ІІ - 61.00.00.000 ІЗ

Група ІІ-22-3

Білоус Богдан

2026

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Білоус Богдан Васильович

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

БАКАЛАВРСЬКА РОБОТА

Розробка ПЗ з фокусом на автономію команд у процесі створення

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

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

Здобувач освітнього ступеня Білоус Б.В.
(підпис, ініціали та прізвище здобувача)

Науковий керівник Храбатин Роман Ігорович, к.ф-м.н, доцент
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту

Завідувач кафедри

доц. Бандура В.В.
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківський національний технічний університет нафти і газу
Інститут, факультет інформаційних технологій
Кафедра інженерії програмного забезпечення
Ступінь вищої освіти бакалавр
Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедрою ІПЗ
доцент. В.В. Бандура

“ ” 2026 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Білоус Богдану Васильовичу

(прізвище, ім'я, по-батькові)

1. Тема проекту (роботи) "Розробка ПЗ з фокусом на автономію команд у процесі створення"

керівник проекту (роботи) Храбатин Роман Ігорович, к.ф.-м.н, доцент т

затверджені наказом вищого навчального закладу від “ 21 ” квітня 2026 р. № 179/7

2. Строк подання студентом проекту (роботи) 10 червня 2026 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження перелічених практик

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

1. Аналіз вимог до програмного забезпечення

2. Аналіз вимог і постановка задачі

3. Проектування системи

4. Реалізація проекту

5. Тестування та впровадження

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

1 Матриця авторитету (рис. 1.1, ст. 21)

2 Етапи розробки програмного забезпечення (рис. 2.2, ст. 27)

3. Організація «Project Cloud (рис. 2.3, ст. 31)

4 Приклад процесу кодування (рис. 2.4, ст. 34)

5 Перешкоди для автономних команд (рис. 3.3, ст. 65)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

2. Дата видачі завдання 2026 р.

Керівник

_____ (підпис)

Завдання прийняв до виконання _____

(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Визначення та обґрунтування теми роботи	17.01.2026	виконано
2	Огляд існуючих концепцій, рішень та сервісів в даній області	21.02.2026	виконано
3	Побудова моделі або алгоритму власного рішення	01.03.2026	виконано
4	Документування реалізації власного оригінального рішення вибраними засобами	21.04.2026	виконано
5	Оформлення пояснювальної записки бакалаврської роботи	02.05.2026	виконано

Студент

_____ Білоус Б.В..

(підпис)

Керівник роботи

_____ Юрчишин В.М.

(підпис)

АНОТАЦІЯ

Бакалаврська робота містить 70 сторінки, 8 рисунків, 1 таблиця, список використаних джерел із 40 найменувань.

Метою роботи є дослідження принципів розробки програмного забезпечення з фокусом на автономію команд та аналіз підходів до підвищення ефективності, гнучкості й швидкості розробки в сучасних ІТ-проєктах.

Об'єкт дослідження: процеси розробки програмного забезпечення в умовах командної взаємодії.

Предмет дослідження: методи, підходи та інструменти забезпечення автономії команд у процесі створення програмних продуктів, включаючи розподіл ролей, механізми прийняття рішень та організаційні практики.

Результати дослідження: проведено аналіз концепції автономних команд, визначено ключові фактори, що впливають на рівень їх самостійності, а також проаналізовано практичні кейси впровадження автономних команд.

У першому розділі розглянуто теоретичні основи автономії команд, визначено її сутність, роль у процесі розробки ПЗ, а також досліджено моделі організації команд і механізми прийняття рішень.

У другому розділі досліджено методи та інструменти забезпечення автономії команд, зокрема підходи до збору та аналізу даних, а також сучасні практики управління розробкою програмного забезпечення.

У третьому розділі розглянуто реалізацію принципів автономії команд у практичних проєктах, проведено оцінку ефективності командної роботи та проаналізовано вплив рівня автономності на результати розробки.

Висновок: впровадження автономії команд у процесі розробки програмного забезпечення сприяє підвищенню продуктивності, покращенню якості продукту та зменшенню часу розробки.

КЛЮЧОВІ СЛОВА: АВТОНОМІЯ КОМАНД, РОЗРОБКА ПЗ, AGILE, DEVOPS, КОМАНДНА ВЗАЄМОДІЯ, УПРАВЛІННЯ ПРОЄКТАМИ, ЕФЕКТИВНІСТЬ РОЗРОБКИ.

ANNOTATION

The thesis consists of 70 pages, 8 figures, 1 tables, and a reference list with 40 sources.

The aim of the work is to study the principles of software development with a focus on team autonomy and to analyze approaches to improving efficiency, flexibility, and development speed in modern IT projects.

Research object: software development processes in the context of team collaboration.

Research subject: methods, approaches, and tools for ensuring team autonomy in the software development process, including role distribution, decision-making mechanisms, and organizational practices.

Research results: the concept of autonomous teams was analyzed, key factors influencing their independence were identified and practical cases of implementing autonomous teams in projects were examined.

The first section describes the theoretical foundations of team autonomy, its essence, role in software development, and models of team organization and decision-making mechanisms.

The second section analyzes methods and tools for ensuring team autonomy, including approaches to data collection and analysis, as well as modern software development management practices.

The third section covers the implementation of team autonomy principles in real projects, evaluates team performance, and analyzes the impact of autonomy levels on development outcomes.

Conclusion: implementing team autonomy in software development enhances productivity, improves product quality, and reduces development time; however, it requires proper process organization, and appropriate management tools.

KEY WORDS: TEAM AUTONOMY, SOFTWARE DEVELOPMENT, AGILE, DEVOPS, TEAM COLLABORATION, PROJECT MANAGEMENT, DEVELOPMENT EFFICIENCY.

ЗМІСТ

ВСТУП.....	8
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ АВТОНОМІЇ КОМАНД У РОЗРОБЦІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	11
1.1 Сутність та принципи автономії команд у розробці програмного забезпечення	11
1.2 Організаційні моделі, ролі та механізми прийняття рішень у команді	15
1.3 Роль автономії у підвищенні ефективності процесу розробки ПЗ	18
1.4 Висновок по розділу	25
РОЗДІЛ 2. МЕТОДИ ТА ІНСТРУМЕНТИ ЗАБЕЗПЕЧЕННЯ АВТОНОМІЇ КОМАНД	26
2.1 Методи дослідження автономності команд у розробці ПЗ	26
2.2 Методи збору та аналізу даних щодо ефективності командної роботи	28
2.3 Інструменти підтримки автономії команд (Agile, DevOps, CI/CD)	33
2.4 Висновок по розділу	36
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПІДХОДІВ ТА ОЦІНКА АВТОНОМІЇ КОМАНД	37
3. Розподіл повноважень і відповідальності в автономних командах	39
3.2 Аналіз рівня автономності команд на прикладі кейс-проектів	47
3.3 Оцінка ефективності та обговорення результатів впровадження	56
3.4 Висновок по розділу	69
ВИСНОВКИ	70
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	72

					БР.ІІ –61.00.00.000 ІІЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Розробка ПЗ з фокусом на автономію команд у процесі створення Пояснювальна записка	Літ.	Арк.	Акрушіє
Розроб.		Білоус Б.В.					7	
Перевір.		Храбатин Р.М.						
Реценз.		Юрчишин В.М.						
Н. Контр.		Піх М.М.						
Затверд.		Бандура В. В.				ІФНТУНГ ІІ-22-3		

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій та зростання складності програмних систем особливого значення набуває ефективна організація процесу розробки програмного забезпечення. Традиційні підходи, що базуються на жорсткій ієрархії управління та централізованому прийнятті рішень, дедалі частіше виявляються недостатньо гнучкими для швидкої адаптації до змін вимог, ринку та технологічного середовища. У зв'язку з цим зростає інтерес до підходів, орієнтованих на підвищення автономії команд розробників, що дозволяє прискорити процес створення програмних продуктів, покращити їх якість та забезпечити більш ефективну взаємодію між учасниками проєкту.

Актуальність теми зумовлена необхідністю підвищення продуктивності команд розробки, зменшення залежності від централізованих рішень та забезпечення швидкої реакції на зміни вимог. У сучасних умовах використання підходів, таких як Agile та DevOps, автономні команди стають ключовим фактором успіху проєктів, оскільки вони здатні самостійно приймати рішення, швидко впроваджувати зміни та ефективно взаємодіяти з іншими командами. Проте недостатня формалізація механізмів автономії, нечіткий розподіл ролей і відповідальності можуть призводити до зниження ефективності, що обумовлює необхідність ґрунтовного дослідження даної теми.

Метою роботи є дослідження підходів до розробки програмного забезпечення з фокусом на автономію команд, а також аналіз методів і інструментів, що забезпечують ефективну організацію командної роботи та підвищення якості програмних продуктів.

Завданнями дослідження є аналіз теоретичних основ автономії команд у процесі розробки програмного забезпечення, дослідження моделей організації команд і механізмів прийняття рішень, вивчення сучасних методів та інструментів підтримки автономії, а також аналіз практичних аспектів впровадження автономних команд у проєктах. Крім того, передбачається оцінка ефективності командної роботи залежно від рівня автономності та формулювання рекомендацій щодо її підвищення.

					БР.ПІ - 61.00.00.000 ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

Об’єктом дослідження є процеси розробки програмного забезпечення в умовах командної взаємодії.

Предметом дослідження є методи, моделі та інструменти забезпечення автономії команд, включаючи організаційні структури, розподіл відповідальності, підходи до управління та використання сучасних практик розробки.

Методи дослідження включають аналіз наукових і технічних джерел, порівняльний аналіз існуючих підходів, моделювання процесів командної взаємодії, а також експериментальне дослідження ефективності впровадження автономних команд у програмних проєктах.

У роботі передбачається дослідити особливості впровадження автономії команд, визначити її вплив на продуктивність, якість програмного забезпечення та швидкість розробки, а також сформулювати практичні рекомендації щодо ефективної організації командної роботи. Очікується, що результати дослідження сприятимуть покращенню процесів розробки програмного забезпечення та підвищенню конкурентоспроможності ІТ-продуктів.

Бакалаврська робота містить 70 сторінки, 8 рисунків, 1 таблиця, 3 розділи, список використаних джерел із 40 найменувань.

					БР.ПІ - 61.00.00.000 ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ АВТОНОМІЇ КОМАНД У РОЗРОБЦІ ПЗ

1.1 Сутність та принципи автономії команд у розробці програмного забезпечення

Плинність кадрів спричиняє різні зміни у відділах кадрів та розробках програмного забезпечення компанії. Дослідження вказують на те, що існують різні фактори, негативно або позитивно пов'язані з плинністю кадрів. Створюються моделі, що намагаються пов'язати визначальні фактори з плинністю кадрів, наприклад, яка розробила модель мотиваційних дій на індивідуальному рівні поведінки користувача.

Компанії-розробники програмного забезпечення демонструють високий рівень плинності кадрів, і ці постійні зміни та плинність кадрів породжують деякі проблеми, такі як збільшення витрат компаній, складність управління командою, зниження гармонії на робочому місці, вплив на успіх проектів та інші. Наприклад, 90% проектів страждають від плинності кадрів.

Високий рівень плинності кадрів стає проблемою, оскільки він свідчить про те, що фахівці не залишаються в компанії тривалий час, що призводить до низького рівня утримання кадрів у компаніях та проектах. Крім того, сама плинність кадрів спричиняє значні витрати, незалежно від того, чи розглядаються вони як грошова класифікація, чи також як рівень якості роботи та індивідуальна робота [1]. У проектах DSD рівень плинності кадрів, як повідомляється, навіть вищий, ніж у локалізованих проектах, що може бути пов'язано зі специфікою роботи, пов'язаною з географічним розташуванням, комунікацією тощо.

Моделі показують позитивні та негативні фактори, що призводять до плинності кадрів, розрізняючи плинність кадрів, коли фахівець вже залишив проект та компанію, а також наміри щодо плинності. Професійні наміри можна помітити у певний момент, коли він жив та діяв у проекті. Деякі фактори згадуються в літературі через їхній вплив на плинність кадрів та наміри щодо

					БР.ПІ - 61.00.00.000 ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

плинності кадрів. Автономія є важливим фактором та мотивацією для програмної інженерії в цьому контексті [2]. Одним із недоліків потребує походження автономії від інженерів-програмістів, а ступінь автономії полягає в тому, що він/вона може впливати на продуктивність та мотивацію. Тому основною метою в цій роботі є представлення взаємозв'язку між автономією та плинністю кадрів інженерів-програмістів, які залучені до проектів розподіленої розробки програмного забезпечення (DSD).

Таким чином, щоб відповісти на дослідницьке питання «Який зв'язок між автономністю та плинністю кадрів у розподілених програмних проектах?», ми розробили два підпитання:

- RQ1.1 – Які основні виміри автономії впливають на плинність кадрів? розподілені програмні проекти?
- RQ1.2 – Які основні фактори, пов'язані з автономією та плинністю кадрів у розподілених проектах розробки програмного забезпечення?

Автономія відома як одна з найбільш вивчених характеристик роботи, і вона відіграє центральну роль у дизайні мотивації роботи. На ранній стадії автономія розглядалася як синонім того, наскільки свободою та незалежністю з індивідуальної точки зору мав член команди під час виконання проекту або робочого завдання.

Після цього дослідження щодо цієї теми чітко зазначили, що концепція автономії на робочих місцях згадується, коли існували такі характеристики, як свобода, незалежність, прийняття рішень, вибір робочого розпорядку та методів виконання завдань.

Тим не менш, також вказали на три аспекти, взаємопов'язані з автономією, зосереджені на свободі, такі як робочий розпорядок, прийняття рішень та методи роботи.

Автономія тісно пов'язана з теорією самовизначення, яка має на меті пояснити фактори, що можуть опосередковувати мотивацію. У цьому контексті компетентність, ідентифікація з роботою та автономія – це три психологічні потреби, вроджені людині для того, щоб відчувати себе задоволеною на робочому

					БР.ІІІ - 61.00.00.000 ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

місці, що виражені цією теорією. Крім того, Райан та Десі зазначили, що чим вища автономія, яку відчуває людина, тим більш мотивована вона.

Автономія на роботі була визнана центральним фактором у проектуванні роботи, що призвело до багатьох позитивних результатів. Незважаючи на це, справедливо поставили під сумнів її прогностичну роль для кількох показників, включаючи намір щодо звільнення, а також взаємозв'язок між ними.

У програмній інженерії фактори самовизначення можна описати як бар'єри для мотивації та задоволення роботою [3]. Професійна демотивація вважається прогностичним фактором плинності кадрів, що розглядається в специфічному та відмінному контексті під час роботи над розподіленими проектами.

Автономія була визначена як важливий мотиваційний фактор для розробників програмного забезпечення, які працюють у розподілених моделях. Ці фахівці також більш толерантні до змін мотивації, оскільки вони природно піддаються годинам антисоціальної роботи та необхідності подорожувати.

У контексті розподіленого проекту сама автономія не вважається фактором, який може мотивувати члена команди. Однак, коли її немає, потреба людини в автономії може сприяти підвищенню незадоволеності роботою та призводити до плинності кадрів у цій команді.

Плинність кадрів негативно впливає на розробку програмного забезпечення [4]. Хоча багато авторів вважають плинність кадрів природною частиною організації, високий рівень плинності кадрів зазвичай призводить до негативного впливу на ефективність процесів.

Плинність кадрів може бути класифікована як зовнішня, коли члени команди залишають організацію, або внутрішня, коли вони залишаються в компанії, але змінюють свою попередню роботу. Цей процес може відбуватися добровільно, коли сам працівник вирішує залишити компанію та свою роль, або вимушено, що відбувається, коли організація вирішує припинити свої відносини з працівником.

Крім того, коефіцієнти плинності кадрів можна розрахувати, поділивши кількість працівників, які залишили організацію за певний час, на загальну

					БР.ІІІ - 61.00.00.000 ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

кількість працівників протягом цього ж періоду [5]. Однак у щоденній організаційній практиці дані потребують більш детальної обробки, щоб визначити причини, що призвели до плинності кадрів. Це розуміння є однією з основних перешкод. Такі дані є дуже важливими для створення стратегій, які можна ефективно застосовувати для належного подолання плинності кадрів та її наслідків.

Статті показують, що сприйнятий намір щодо автономності роботи негативно пов'язаний зі звітністю про плинність кадрів лише для працівників з високим рівнем сприйнятої підтримки [6]. Однак ці результати необхідно інтегрувати з новими методами роботи, такими як Розподілена розробка програмного забезпечення (DSD).

DSD стало поширеною практикою в усьому світі, що посилюється фінансовими, структурними та глобальними факторами.

Серед кількох характеристик, пов'язаних з DSD, є потенціал організації залучати кваліфікованих фахівців, навіть перебуваючи в місцевому контексті з дефіцитом фахівців.

Ця неймовірна здатність залучати та наймати спеціалізованих фахівців позитивно впливає на продуктивність праці та якість послуг, а також продукції.

У проектах DSD рівень плинності кадрів може бути підвищеним через фізичну відстань, а в дослідженні плинність кадрів згадується як один із п'яти найважливіших ризиків для проектів DSD. Попередні короткі дослідження стосувалися актуальності мотиваційних факторів для запобігання цьому явищу в розподілених проектах.

Розподілена розробка програмного забезпечення

Розподілений проект – це група людей у різних місцях, які працюють об'єднані в одному проекті протягом тривалого періоду. Такий тип програмного проекту, в якому залучені людські ресурси розподілені на відстані, регіонально, національно або глобально, визначається як розподілена розробка програмного забезпечення (DSD).

Крім того, DSD можна класифікувати за двома факторами: відстань між робочими командами, яка проявляється як оншорна (команди, розташовані в

					БР.ІІІ - 61.00.00.000 ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

одній країні) та офшорна (команди, розташовані в різних країнах). З іншого боку, контрольні відносини, які центральна організація має над віддаленими командами, проявляються як аутсорсинг (найм сторонньої компанії) або інсорсинг (створення віддаленого підрозділу компанії).

Дослідження показують, що постачальники послуг офшорного аутсорсингу можуть зменшити плинність кадрів, покращуючи баланс між роботою та особистим життям і впроваджуючи більш сприятливу для сім'ї політику зайнятості, таку як підвищена автономія. Крім того, постачальники послуг аутсорсингу можуть ефективніше винагороджувати інновації та структурувати контракти, щоб забезпечити володіння програмним продуктом для покращення утримання персоналу.

1.2 Організаційні моделі, ролі та механізми прийняття рішень у команді

Масштабна розробка програмного забезпечення традиційно визнається ініціативою, до якої найкраще підходить з використанням традиційного управління «зверху вниз» з чіткою структурою управління, попередньо визначеними механізмами прийняття рішень та ролями з різними рівнями відповідальності [7]. Однак розробку масштабних програмних систем ніколи не можна повністю передбачити заздалегідь, оскільки система постійно зростає у складності та часто потребує відхилення від початкових специфікацій. Крім того, коли масштабні проекти стають більш складними, критичними та терміновими, їх стає важче контролювати. Властива напруга між стабільністю та змінами у масштабних програмних проектах вимагає стратегії управління, яка підтримує обидві перспективи. Як наслідок, багато великих організацій з розробки програмного забезпечення переходять від підходу «зверху вниз» до більш збалансованого підходу до управління, який поєднує прийняття рішень «знизу вгору» та «зверху вниз», і все частіше застосовує альтернативні організаційні моделі для посилення здатності компанії до змін та уникнення викликів традиційних ієрархій. На практиці багато

					БР.ПІ - 61.00.00.000 ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

великих проектів розробки потім використовуватимуть гнучкі методи на рівні команди в поєднанні з фреймворком управління проектами, таким як Project Management Body of Knowledge або PRINCE2.

Зі зростанням популярності гнучких методів розробки програмного забезпечення та пов'язаних з ними концепцій, таких як автономні та уповноважені команди, компанії-розробники програмного забезпечення знайшли нові способи розподілу повноважень між тими, хто виконує роботу, та тими, хто обіймає керівні посади. У гнучкій розробці центр прийняття рішень переміщується від керівника проекту до команди розробників програмного забезпечення, а процес прийняття рішень змінюється з індивідуального та централізованого на спільний та децентралізований [8]. Однак, коли багато автономних гнучких команд разом працюють над досягненням однієї мети, потрібно багато додаткових зусиль щодо координації та управління, що в попередніх дослідженнях великомасштабних гнучких проектів виявилось складним завданням. Аналогічно, дослідження з управлінської науки показують, що міжгрупова координація є серйозною проблемою, коли групи мають повноваження. Для координації різних команд можна використовувати різноманітні механізми, наприклад, описують 14 механізмів міжкомандної координації у великомасштабному програмному проекті та виявили, що така координація впливає на внутрішні процеси команд та прийняття рішень. Крім того, під час впровадження спільного децентралізованого процесу прийняття рішень у великих проектах з багатьма командами необхідно вирішувати проблеми узгодження. Попередні дослідження розглядали проблеми, пов'язані з відсутністю узгодженості між командами [9]. Якщо кожна команда працює незалежно, процеси розробки команди та технічні рішення зрештою відрізнятимуться та можуть бути дуже не пов'язаними один з одним. Крім того, відсутність обміну знаннями між командами може призвести до дублювання роботи, непорозумінь та проблем інтеграції. Однак, коли практика в командах стає залежною від практики інших команд, команди навряд чи можна назвати автономними.

Підсумовуючи, балансування необхідності узгодження та досягнення

					БР.ПІ - 61.00.00.000 ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

переваг децентралізованої структури прийняття рішень, що включає багато автономних команд у масштабних програмних проектах, залишається складним завданням. Залишаються відкритими кілька важливих питань: якщо кожній команді нададуть повноваження та відповідальність за організацію своєї роботи, як організація може забезпечити загальну координацію та успіх масштабного проекту? Хто відповідає за загальну відповідність цілям проекту? Які ефективні структури прийняття рішень для координації між командами? Зрештою, оскільки гнучкі методи є емерджентними, процеси, принципи, прийняття рішень та структури роботи рідко бувають повністю визначеними заздалегідь. Отже, повноваження автономної команди та те, як така команда організовує роботу у великомасштабному контексті, потребують адаптації до проекту та його контексту. Тому, щоб зрозуміти автономію команди у великомасштабних гнучких середовищах, необхідно вивчити кілька проектів у різних контекстах.

Мета цієї роботи - поглибити наше розуміння того, як збалансувати автономію команди та децентралізоване прийняття рішень з необхідністю організаційного контролю та узгодження у масштабних зусиллях з гнучкої розробки, тобто які повноваження повинні мати команди. Крім того, ми прагнемо дослідити, де може бути ідеальне співвідношення автономії команди та організаційного контролю. Наше дослідження зумовлене наступним дослідницьким питанням:

RQ: Як розподіляються організаційні повноваження у масштабній гнучкій розробці програмного забезпечення?

Щоб відповісти на наше дослідницьке питання, ми вивчили дві великомасштабні Agile-організації, які перебувають на різних етапах впровадження Agile-технологій у різних філіях великої міжнародної телекомунікаційної компанії Ericsson [10]. У цій роботі ми використовуємо визначення масштабної розробки програмного забезпечення, тобто як розробку, що виконується в:

«організації з розробки програмного забезпечення, що налічують 50 або більше осіб або щонайменше шість команд. Усі люди не обов'язково повинні бути розробниками, але повинні належати до однієї організації з розробки

					БР.ІІІ - 61.00.00.000 ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

програмного забезпечення, яка розробляє спільний продукт або проект, і, таким чином, мати потребу у співпраці».

1.3 Роль автономії у підвищенні ефективності процесу розробки ПЗ

Автономія є центральним принципом гнучких методів. Однак поняття автономних команд не є новим і вивчалось та описувалося з різних точок зору в минулому; дослідження в цій галузі проводяться з давніх часів, які вивчали саморегульованих шахтарів у 1950-х роках. Такі команди описувалися з соціотехнічної точки зору як такі, що складаються з 10-15 членів, які пройшли перехресне навчання, керуються баченням, мотивовані тиском однолітків та беруть на себе обов'язки своїх колишніх керівників. Члени таких команд повинні активно шукати дані та зворотний зв'язок, щоб дізнатися, наскільки добре вони виконують свої завдання, та вживати коригувальних заходів з власної ініціативи для покращення своєї продуктивності [11]. У цій роботі ми спираємося на широко вживане визначення автономних команд, у огляді ефективності команд: «працівники, які зазвичай виконують тісно пов'язані або взаємозалежні роботи, яких ідентифікують та можна ідентифікувати як соціальну одиницю в організації, і яким надано значні повноваження та відповідальність за багато аспектів їхньої роботи, такі як планування, складання графіків та призначення завдань членам, а також прийняття рішень з економічними наслідками».

Автономні команди були описані з точки зору організаційної теорії. «Образи організацій» описує чотири принципи для створення таких команд:

- (a) навчання навчанню, що підкреслює важливість здатності команди долучатися до подвійного циклу навчання та забезпечувати постійне вдосконалення;
- (b) мінімальна критична специфікація, в якій керівництво описує лише критичні аспекти, необхідні для ефективного функціонування команд;
- (c) необхідна різноманітність, яка полягає у відповідності складності та різноманітності навикишнього середовища; та

					БР.ІІ - 61.00.00.000 ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

(d) надмірність функцій.

Три останні принципи передбачають, що команди повинні складатися з членів з різноманітними навичками, щоб справлятися з різноманітністю зовнішнього середовища, і що окремі члени команди повинні бути в змозі значною мірою допомагати та замінювати один одного за потреби. Перевага міжфункціональних команд полягає в їхній здатності черпати знання з кількох джерел та виконувати кілька видів діяльності одночасно, а не послідовно, що економить час. Міжфункціональна команда розробників програмного забезпечення часто складається з членів з різними ролями, наприклад, тестувальників, розробників, архітекторів, дизайнерів та аналітиків. Однак обмін знаннями в міжфункціональних гнучких командах є складним завданням через складність взаємодії між різними спеціалістами та зацікавленими сторонами. Члени таких команд часто розподіляють завдання відповідно до своєї ролі чи відповідальності, що призводить до зосередження на їхній індивідуальній продуктивності та меншій допомоги один одному. Низька надмірність та висока індивідуальна автономія порушують четвертий принцип.

Хоча автономні команди повинні нести відповідальність за багато аспектів своєї роботи, у таких командах все ще існує вимога до лідерства. Лідерство має бути розсіяним, а не централізованим, що означає, що лідерство передається особі з ключовими знаннями, навичками та здібностями, пов'язаними з конкретним завданням, з яким стикається команда в будь-який момент [12]. Хоча менеджер зберігає лідерство в управлінні проектами, члени команди керують рішеннями та несуть за них відповідальність, коли вони володіють знаннями, якими необхідно ділитися на різних етапах проекту. Основні проблеми спільного прийняття рішень у гнучких середовищах пов'язані з різними конфліктами, включаючи потребу в короткостроковому прогресі проти довгострокової якості, членів, що працюють повний робочий день проти тих, що працюють неповний робочий день, та нових розробників проти робіт з обслуговування. Іншими словами, таким командам потрібен чіткий напрямок та спільні цілі.

					БР.ІІІ - 61.00.00.000 ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

Рівні автономії

Хоча було виявлено, що автономія та самоврядування позитивно впливають на команди, рівень автономії, який є доцільним за певних умов, залишається важливим, але невирішеним питанням. Автономія має низку наслідків і залежить від низки факторів, включаючи взаємозалежність завдань, складність завдань та структуру команди [13]. Позитивний вплив автономії був знижений у короткострокових командах і був зумовлений складністю завдань. Крім того, гнучкість та адаптивність, які традиційно вважаються перевагами автономних команд, також можуть бути пов'язані з дисфункціональними результатами, припускаючи, що організації повинні ретельно враховувати належний рівень автономії та свободи дій, що надаються командам. Описано автономні команди як команди, яким надано значні повноваження та відповідальність за багато аспектів їхньої роботи. Тому ми будемо використовувати вираз «рівень автономії» для позначення обсягу повноважень, якими володіє команда.

У роботі, присвяченій авіаційній галузі та галузі охорони здоров'я, досліджувалося корисність механізмів, відмінних від ієрархічної влади зазначено, що в міжфункціональних робочих процесах (між відділами та ієрархіями) компанії, які працювали найкраще, мали вищий рівень координації між ролями. Однак у цих контекстах на ієрархічну владу все ще можна покладатися за потреби, наприклад, у випадках конфлікту чи розбіжностей. Ґрунтуючись на дослідженнях авіакомпаній, обговорювалося взаємозв'язок між різними типами виконавчих підрозділів та обсяг повноважень. Хекман звернула особливу увагу на те, як повноваження розподіляються між тими, хто виконує роботу, і тими, чії обов'язки є переважно управлінськими. Самокерований виконавчий підрозділ має певний, але обмежений обсяг повноважень, тоді як самопроектуючий та самоврядний підрозділ має більше відповідальності (див. рис. 1.1). У гнучкій розробці програмного забезпечення команді надаються повні повноваження робити все, що вона вважає за необхідне для досягнення «мети».

					БР.ІІІ - 61.00.00.000 ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

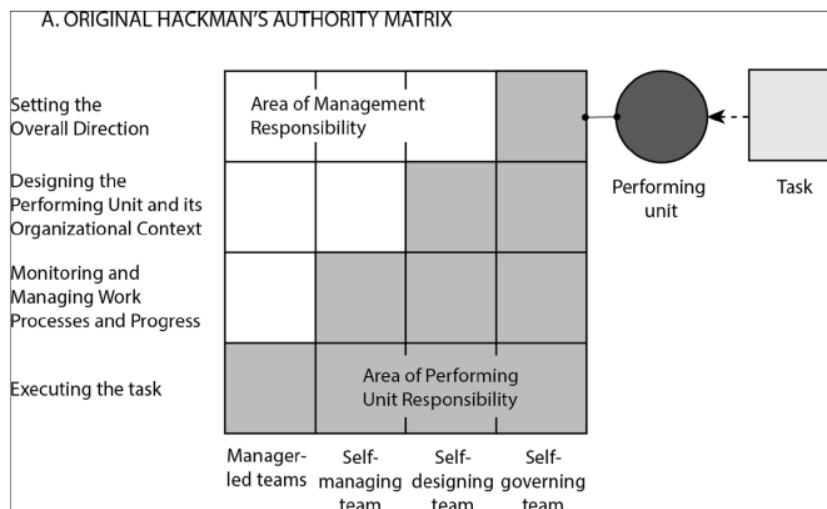


Рисунок 1.1 - Матриця авторитету

Однак, що це означає у великомасштабному середовищі, незрозуміло. У дослідженні двох великих програмних компаній та великої компанії з обробки даних наводять приклади того, як можна застосувати недооцінку повноважень. Повноваження щодо виконання роботи були повністю децентралізованими в їхніх випадках, але не для інших областей прийняття рішень.

Оскільки метою нашого дослідження є розуміння того, як розподіляються повноваження у масштабному гнучкому середовищі з багатьма командами, ми використовуємо роботу як основу для нашого аналізу. Його робота виникла з вивчення команд у масштабному контексті, і частина його матриці повноважень раніше була протестована на організаційному рівні.

Виконувальна одиниця – це група, що складається з кількох осіб, які працюють взаємозалежно над спільним завданням. Виконувальною одиницею в нашому дослідженні є гнучка команда розробників програмного забезпечення. За Гекманом, у команді, очолюваній менеджером, члени мають повноваження лише виконувати завдання, тоді як самокерована команда несе відповідальність за виконання своїх завдань, а також за моніторинг та управління власними робочими процесами та прогресом [14]. Зі збільшенням обсягу повноважень команда отримує відповідальність як за проектування команди та її контексту, так і за встановлення загального напрямку, і таким чином команди переходять від самостійно-

го проектування до самоврядування. Зведений огляд різних рівнів відповідальності або повноважень команди наведено в таблиці разом з нашими конкретними прикладами. Команда може бути уповноважена виконувати функції, що виходять за межі загального рівня наданих їй повноважень, або нести відповідальність лише за частину певних функцій.

Автономія у розробці програмного забезпечення великого масштабу

Дослідження ефективності команд показують, що продуктивність автономних робочих груп дуже залежить від ситуації, а вплив практик, які використовують ці групи, залежить від таких факторів, як характер робочої сили та характер організації. Таким чином, справедливо поставити питання, чи отримують команди у контексті масштабної розробки програмного забезпечення користь від збільшення обсягу повноважень, і чи призводить зменшення повноважень до небажаного негативного впливу на продуктивність команди.

Більшість досліджень автономії проводяться в рамках невеликих проектів з однією командою. Аналогічно, пов'язані теорії, зосереджуються на автономії на груповому рівні, де група є виконавчою одиницею, яка працює над завданням в організаційному контексті. Незважаючи на те, що дослідження Хекмана проводилися у великомасштабному середовищі (авіалінії), інтерпретація цієї теорії в контексті великомасштабних гнучких проектів програмного забезпечення не є простою. У таких проектах багато команд працюють над завданнями, які разом сприяють розробці системи, що часто випускається для клієнтів, що забезпечує постійний зворотний зв'язок. Завдання можуть відрізнятися за деталями; наприклад, завдання може бути функцією або компонентом. Таким чином, систему можна розглядати як комбінацію кількох незалежних або взаємозалежних завдань, які мають бути інтегровані [15]. Тому рівень повноважень команд, що працюють у великомасштабних проектах з багатьма командами, може бути обмеженим через необхідність координації з іншими командами та зацікавленими сторонами та необхідність узгодження вхідних, вихідних даних та процесів у процесі розробки системи. Крім того, оскільки команди залежать одна від одної, їм часто доводиться чекати, поки інші команди завершать роботу, що може вплинути на

					БР.ІІІ - 61.00.00.000 ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

авторитет команди. У дослідженні координації у великомасштабних розподілених гнучких проектах виявили, що доступ до хороших інструментів координації та комунікації сприяє автономії команди.

У робочих системах, де команди користуються високим рівнем влади, координація між командами виявилася серйозною проблемою. У великих Agile-проектах існує низка проблем, які охоплюють команди розробників, а також рішення, які одночасно впливають на багато команд [16]. Координація між командами є особливо складною, коли команди працювали незалежно для різних клієнтів, хоча створювані додатки були взаємозалежними. Коли цілі команди суперечать цілям організації, багато команд вирішують діяти у власних інтересах та не поважають ширший контекст. Дослідження масштабного Agile-середовища показало, що здатність збалансувати децентралізований процес прийняття рішень та централізовану структуру була важливим фактором успіху. Однак, вони стверджують, що необхідно краще зрозуміти, як багатокомандні програми або проекти розвитку балансують міжкомандну координацію та командні повноваження.

В огляді літератури про виклики та фактори успіху масштабних гнучких трансформацій автономія команди згадувалася як виклик, так і фактор успіху. З одного боку, розширення можливостей на низовому рівні було надзвичайно важливим для успіху трансформації. З іншого боку, автономію команди було складно організувати у масштабних умовах. Зокрема, проблеми виникали, якщо організація розробила свою модель розробки програмного забезпечення відповідно до гнучких принципів, і таким чином дозволила командам працювати незалежно, в той час як між командами існували залежності, що вимагали постійної співпраці [17]. Крім того, було складно збалансувати цілі команди та ширші цілі організації. Команда могла легко приділяти більше уваги власним цілям, ніж цілям усієї організації. Таким чином, перебування у складі великомасштабної організації може обмежити обсяг повноважень, які можна надати кожній команді.

Більше того, обговорювали, чи повинна вся організація дотримуватися єдиного гнучкого підходу, чи дозволити командам самим обирати свій гнучкий

					БР.ІІІ - 61.00.00.000 ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

підхід. Єдиний підхід допомагає організації прийняти узгоджений словник, що полегшує співпрацю між командами, порівняння роботи між командами та навіть легше переміщує членів до інших команд. Надання командам повноважень обирати власний підхід може бути ефективнішим для команди, але може ускладнити співпрацю між командами, наприклад, якщо різні команди використовують різні гнучкі методи та практики або навіть різні інтерпретації гнучкого методу. Одним із чудових прикладів, що згадувався, було те, що команди з різною тривалістю спринтів можуть створювати проблеми під час реалізації. Нарешті, дійшли висновку, що між проектами в одній великій організації можуть знадобитися різні типи гнучких процесів для різних типів розробки, залежно від типу програмного забезпечення, що розробляється, та розміру проекту. Таким чином, виходячи з систематичної літератури, важливо, щоб у масштабному гнучкому середовищі організація знайшла баланс між обсягом повноважень команди щодо процесу розробки та необхідністю дотримуватися того ж гнучкого підходу, а також дотримуватися центральних рішень щодо залежностей та координації між командами.

Збільшення авторитету команди – це трансформаційний процес. Досліджували розширення прав і можливостей у семи традиційних компаніях, що працюють з масштабною розробкою програмного забезпечення, в яких обрані команди отримали підвищену автономію та змогу обійти формальні ієрархії. Автори припускають, що ініціативи, які кидають виклик традиційним організаційним структурам, все ще повинні узгоджуватися з бізнес-цілями, існуючими формальними та затвердженими структурами, а також балансувати рішення «знизу вгору» (часто обмежені короткостроковими тактичними рішеннями) зі стратегіями «зверху вниз» та централізованим прийняттям рішень. Важливо, що однією з головних причин балансування є необхідність узгодження роботи кількох команд, залучених до однієї програми.

Незважаючи на брак досліджень щодо автономності команд у масштабній гнучкій розробці програмного забезпечення, звіти практиків та гнучкі фреймворки торкаються цієї теми [18]. Мабуть, найголовніше, що підхід, який

					БР.ІІ - 61.00.00.000 ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

використовує Spotify, побудований навколо ідеї автономних гнучких команд, об'єднаних спільними цілями та низкою механізмів координації «знизу вгору», включаючи спільноти практиків, які називаються гільдіями. Agile-коучі у Spotify намагаються збалансувати узгодження автономії між командами. Основні фреймворки для масштабування мають більш обмежувальний підхід до автономності команд. У Scaled Agile Framework, SAFe, команди повинні працювати відповідно до загального ритму високого рівня, який називається Program Increment, і який зазвичай триває близько 12 тижнів. Команди можуть працювати або за ScrumXP, що є поєднанням Scrum та XP, або за Kanban на рівні команди. У рамках LeSS усі команди дотримуються процесу Scrum із синхронізованим ритмом. В обговоренні ми детальніше порівнюємо ці підходи з нашими висновками.

1.4 Висновок по розділу

У першому розділі було розглянуто теоретичні основи автономії команд у процесі розробки програмного забезпечення. Проаналізовано сутність та основні принципи автономії команд, що забезпечують самостійність у прийнятті рішень і виконанні завдань. Досліджено організаційні моделі, ролі учасників та механізми прийняття рішень у командах розробки. Також визначено роль автономії у підвищенні ефективності процесу створення програмного забезпечення, зокрема у покращенні комунікації, швидкості розробки та адаптивності до змін. Отримані результати формують теоретичну основу для подальшого дослідження методів і практичних підходів до забезпечення автономності команд.

					БР.ІІІ - 61.00.00.000 ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. МЕТОДИ ТА ІНСТРУМЕНТИ ЗАБЕЗПЕЧЕННЯ АВТОНОМІЇ КОМАНД

2.1 Методи дослідження автономності команд у розробці ПЗ

У цій роботі ми розглядаємо питання, як знайти правильний баланс між автономією команди та організаційною узгодженістю у масштабній гнучкій розробці програмного забезпечення [19]. Розуміння того, скільки повноважень надати командам, не ставлячи під загрозу загальну продуктивність, є критично важливим під час впровадження організаційних змін для підвищення гнучкості та продуктивності масштабної розробки. Ми розглядаємо це питання, досліджуючи способи розподілу організаційних повноважень у масштабних гнучких проектах програмного забезпечення, спираючись на матрицю розподілу повноважень Хекмана, та досліджуючи практичні способи прийняття рішень щодо виконання завдань команди, управління способами роботи, проектування команди та визначення загального напрямку. Крім того, ми прагнемо зрозуміти, як різні рівні повноважень команди сприймаються командами, причини централізації певних рішень та необхідність узгодження між командами.

У цьому підрозділі ми представляємо наші методи дослідження, огляд наших проектів з розгляду конкретних випадків, а також методи збору та аналізу даних.

Дизайн тематичного дослідження

Оскільки метою цього дослідження є вивчення та надання розуміння того, як розподіляються організаційні повноваження у масштабній гнучкій розробці програмного забезпечення, важливо вивчити команди розробників програмного забезпечення на практиці. Дослідження, про яке йдеться в цій роботі, є дослідницьким дослідженням кількох випадків. Під час проведення дослідження кількох випадків ми дотримувалися п'ятиетапного процесу:

1. Дизайн тематичного дослідження: цілі визначені, а тематичне дослідження сплановано.
2. Підготовка до збору даних: визначено процедури та протоколи збору

					БР.ІП – 61.00.00.000 ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

даних.

3. Збір доказів: виконання збору даних для досліджуваної справи.
4. Аналіз зібраних даних.
5. Звітність.

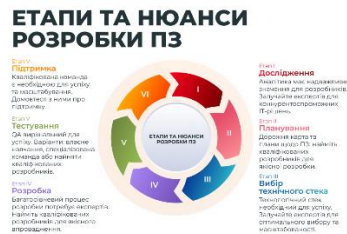


Рисунок 2.1 – Етапи розробки програмного забезпечення

Щоб зрозуміти, як розподіляються повноваження між командами та рештою організації, ми розробили багатоетапне тематичне дослідження двох масштабних проектів гнучкої розробки програмного забезпечення в Ericsson, світовому лідері в галузі телекомунікаційних систем. Ми обрали Еріка Ссонна як суб'єкта дослідження на основі нашої тривалої співпраці з компанією та наших знань про їхні існуючі масштабні програми розробки. У нашому дослідженні ми досліджуємо обсяг повноважень гнучких команд у порівнянні з організаційним контролем. Повноваження команди досліджуються шляхом дослідження того, хто бере участь у прийнятті різних типів рішень, скільки свободи чи повноважень надається гнучким командам, та які рішення узгоджуються на рівні проекту. Як основу для опису обсягу повноважень, виділених командам, ми використовували чотирирівневу матрицю повноважень. Ми структурували наш збір даних відповідно до областей повноважень:

- 1) визначення загального напрямку,
- 2) проектування виконавчої одиниці та її контексту,
- 3) моніторинг та управління робочими процесами
- 4) виконання завдання.

Ми виявили точки прийняття рішень у межах чотирьох областей повноважень. На основі зібраних даних ми спробували зрозуміти, наскільки необхідний організаційний контроль та узгодженість, щоб самокеровані Agile-команди

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		27

могли співпрацювати для досягнення спільної мети. На відміну від, які вивчали неординарні команди, яким через особливі обставини надано вищий рівень повноважень, такі як команди, що працюють у нових стратегічних напрямках та над інноваціями, або пілотні команди, що впроваджують нові практики розробки, ми вивчаємо команди розробників, які працюють над добре зарекомендували себе продуктами [20]. Тому ми очікуємо, що жодна окрема команда не може мати повних повноважень у досліджуваних середовищах. Отже, ми прагнемо обговорити, наскільки повноваженнями можуть і повинні мати команди та щодо яких аспектів, а також скільки і який вид організаційного контролю потрібен.

2.2 Методи збору та аналізу даних щодо ефективності командної роботи

Ericsson є провідним постачальником телекомунікаційних систем та обладнання для операторів мобільного та фіксованого зв'язку. Компанія розробляє універсальні програмні продукти, що пропонуються відкритому ринку, та складні складні системи з версіями, орієнтованими на клієнтів та сегменти ринку. Багато з цих систем складаються з багатьох мільйонів рядків коду та кількох технологій, а їх розробка та обслуговування залучають багатьох розробників та допоміжний персонал. Приблизно з 2007 по 2009 рік багато підрозділів Ericsson вдалися до гнучкої трансформації, яка фундаментально змінила їхні методи роботи. Для нашого дослідження ми обрали два випадки, які ми називаємо Project Mobility та Project Cloud. Проекти були відібрані на основі вибірки зручності. Обидва проекти були нам знайомі, оскільки ми вивчали їх у зв'язку з іншими масштабними гнучкими темами та вважали їх цікавими та придатними для цього дослідження. Project Mobility був частиною дослідницького проекту, спрямованого на визначення того, як збалансувати потреби в управлінні та контролі якості у великих розподілених організаціях та потреби команди в автономії. Проект Cloud став наступником проекту, який організація вважала зразковим з точки зору впровадження та використання гнучких методів, і який використовувався як бенчмарк для міжнародного впровадження гнучких методів в Ericsson. Хоча

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		28

обидва вибрані проекти були великими та дотримувалися методології гнучкої розробки програмного забезпечення, способи їх впровадження відрізнялися [21]. Наприклад, у проекті Cloud були впроваджені спільноти практиків (CoP) як підтримку масштабування гнучкої розробки, тоді як такі паралельні структури були відсутні в проекті Mobility. Крім того, проекти надходили від різних підрозділів Ericsson та мали різний досвід та історію гнучкої трансформації. Тому, вибираючи ці випадки, ми очікували знайти відмінності в тому, як розподілялися повноваження та скільки свободи та повноважень було надано гнучким командам програмного забезпечення в кожному з цих випадків.

Випадок 1 - Мобільність проекту

Мобільність проекту, див. рис. 2.2, це масштабна розробка програмно-інтенсивної системи, що відповідає за мережу мобільного широкосмугового доступу. На момент нашого дослідження, у квітні та травні 2016 року, система розвивалася майже 20 років і складалася з 30 підсистем, що містять понад п'ять мільйонів рядків коду, написаних різними мовами програмування, а також інтегрованих зі сторонніми продуктами. Організація дотримувалася традиційного процесу розробки програмного забезпечення, керованого планом, до 2008 року, коли розпочалася гнучка трансформація, і з того часу впровадила багато змін у процесі та управлінні розробкою. Мобільність проектів спиралася на підхід на основі Scrum.

На момент нашого дослідження систему розробляли близько 30 команд з трьох місць: Швеції, Китаю та Росії. Наш аналіз зосередився на шведській частині організації проекту [22]. Шведські команди в Project Mobility були міжфункціональними командами, що склалися з розробників, системних менеджерів, тестувальників, scrum-майстра та автора документації (останні два були розподілені між кількома командами). Хоча метою компанії було надати командам можливість робити що завгодно в будь-якому місці системи, на практиці команди спеціалізувалися на розробці додатків, платформ або віртуалізації, а також могли спеціалізуватися на низці конкретних підсистем.

Через масштабність, складність та кількість змін в організації та процесі (наприклад, перехід від дисциплінованих команд компонентів до

					БР.ІП – 61.00.00.000 ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

міжфункціональних команд функцій), організація запровадила суворий моніторинг та контроль ефективності. Це здійснювалося різними керівними ролями.

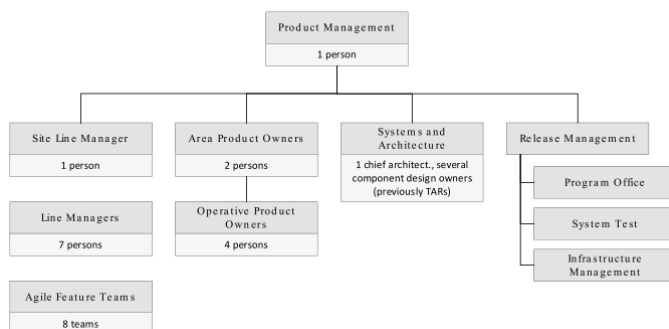


Рисунок 2.2 - Організація мобільності в рамках проекту на шведському майданчику

Технічні координатори ретельно перевіряють протягом усього життєвого циклу продукту та наявність ключової компетенції в цій галузі (тобто ключова компетенція не повинна належати одній особі). Раніше технічну якість результатів команди перевіряли експерти, відповідальні за технічні області (TAR), які розглядали пропозиції щодо дизайну функцій, код та остаточні рішення [23]. Однак, оскільки команди ставали більш досвідченими, наглядові функції були мінімізовані, щоб сприяти відповідальності команди за результати своєї роботи, а роль TAR була перетворена на більш проактивну роль, яка називається архітектором дизайну (DA). DA об'єднують власників продукту, системних архітекторів та команди та виступають зацікавленими сторонами на ранніх етапах розробки функцій для нетривіальних функцій та функцій з архітектурним впливом. Деякі з цих ролей вимагають повної зайнятості, тоді як інші є роботами на умовах неповного робочого дня. Наприклад, роль власника дизайну зазвичай призначається лінійному керівнику, тоді як TAR/DA зазвичай працюють старшими розробниками 50% свого часу. На момент нашого розслідування очікувалося, що команди будуть відносно незалежними, хоча організація підтримки навколо команд все ще була досить великою.

Випадок 2 - Проект Хмара

Проект Cloud стосується міграції основного вузла телекомунікаційної мер

					БР.ІП – 61.00.00.000 ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

ежі з виділеного обладнання на віртуалізоване обладнання в хмарі. Початковий вузол, на момент нашого дослідження, використовувався понад 300 телекомунікаційними операторами по всьому світу. Система є великою, складається з понад 30 мільйонів рядків коду, а її розробка розпочалася понад 15 років тому. До 2009 року система розроблялася з використанням каскадного програмного процесу. Починаючи з весни 2009 року, організація перейшла до «легкого та гнучкого» способу роботи, за їхніми власними словами, натхненна фреймворком масштабування Large-Scale Scrum (LeSS). Основною причиною трансформації була необхідність скоротити цикл розробки та наростити потенціал для швидшого реагування на потреби ринку та клієнтів. Процес розробки був заснований на Scrum, з тривалістю ітерації два тижні.

У 2014 році було прийнято рішення відмовитися від кастомного обладнання та перейти на віртуальну платформу, що працює в хмарі. Ця міграція була організована як проект з окремими командами, тоді як інші команди продовжували паралельно розробляти та підтримувати стару систему. У міру того, як проект міграції набирив обертів, а потреба в підтримці старого продукту зменшувалась, команди поступово переходили від початкової системи до Project Cloud [24]. Під час збору даних близько 30 клієнтів тестували хмарний продукт.

Організація розробки Project Cloud складалася з десяти команд на двох майданчиках, розташованих у двох країнах: Фінляндії та іншій європейській країні. Тут ми зосередимося на фінському майданчику, на якому було розташовано вісім команд, як показано на рис. 2.3.

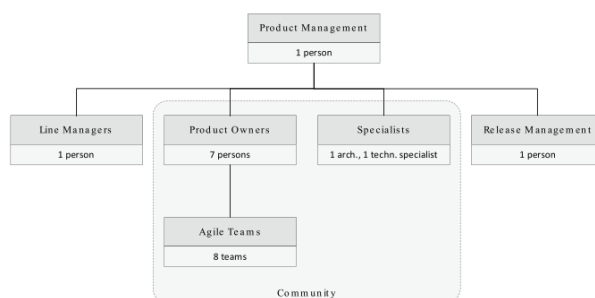


Рисунок 2.3 - Організація «Project Cloud»

Більшість команд у Project Cloud були міжфункціональними, тобто вони

					БР.ІП – 61.00.00.000 ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

були здатні виконувати архітектурну роботу, розробку, тестування та інтеграцію. Організація прагнула до гнучкого ідеалу, за якого будь-яка команда могла б працювати над будь-якою функцією від початку до кінця. Крім того, певні типи спеціалізованих завдань тестування, які не могли бути виконані в командах з розробки функцій, виконувалися кількома окремими командами тестування.

Міграція в хмару була технічно складною та являла собою новий етап для всіх учасників проєкту, зменшуючи розрив у знаннях між технічними спеціалістами та розробниками. Незважаючи на те, що розробку здійснювали міжфункціональні команди.

Організація старого продукту складалася з кількох ролей поза командами. У Project Cloud, навпаки, якомога більше з них було інтегровано в команди, щоб створити команди, що працюють повноцінно. Поза командами залишилися лише два експерти: архітектор і технічний спеціаліст, який вивчає майбутні технології.

В рамках впровадження гнучких методів розробки оригінального продукту, спільноти практиків (CoPs) були запроваджені та отримали центральну роль [25]. CoPs спочатку використовувалися для підтримки трансформації, а пізніше – для допомоги в координації роботи між Agile-командами. В організації розробки старого продукту було сформовано кілька CoP, включаючи CoP коучингу, CoP Scrum-майстрів та CoP безперервної інтеграції. Будь-хто в організації міг брати участь у CoP, і будь-хто в спільноті, хто хотів обговорити тему, яку він чи вона вважав важливою для продукту чи організації, міг запланувати зустріч CoP, запросити учасників та розпочати створення спільноти навколо неї. CoP використовувалися для обговорення тем, спільних для кількох команд, координації роботи та прийняття рішень, пов'язаних з продуктом та розробкою. Частота та регулярність зустрічей CoP варіювалися. Деякі CoP зустрічалися регулярно на постійній основі, наприклад, CoP займалися координацією роботи між командами, які створюють спільну функцію. Інші організовувалися ad hoc, наприклад, для вирішення потреби спільного архітектурного планування або спільного проектування функцій.

Використовуючи цю культуру спільноти учасників (CoP) як основу, Project Cloud розвинув її в напрямку того, що вони назвали *прийняттям рішень*

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		32

на основі спільноти. У такому способі роботи, як пояснили наші респонденти, окремі CoP були об'єднані в єдину уповноважену спільноту. Усі члени команди та всі власники продуктів були членами цієї спільноти, але ні менеджери. Рішення з широким впливом приймає спільнота в цілому, а не більш спеціалізована CoP чи кілька осіб, які приймають рішення. Спільнота мала кілька фіксованих термінів для зустрічей щотижня, і вони резервувалися та використовувалися за потреби.

2.3 Інструменти підтримки автономії команд (Agile, DevOps, CI/CD)

Ми збирали дані за допомогою індивідуальних та парних інтерв'ю. Респондентів номінували, на наше прохання, представники компаній, залучених до наших дослідницьких проектів, та відбирали з метою досягнення максимальної варіабельності щодо ролі, досвіду та ставлення, а також охоплення всього життєвого циклу розробки [26]. Ми проводили напівструктуровані інтерв'ю з людьми, які обіймають різні ролі, як на рівні команди, так і на керівних та допоміжних посадах. Ми збирали інформацію щодо участі респондента в прийнятті рішень, його поглядів та досвіду. По можливості ми спеціально запитували про повноваження команди та неучасть/неучасть у прийнятті рішень.

У проекті «Мобільність» ми мали обмежені можливості відвідати місце проведення проекту, яке знаходилося в іншому місті, тому вирішили провести парні інтерв'ю, оскільки вони можуть генерувати велику кількість даних і включати широкий спектр відповідей протягом обмеженого часу. Ще однією перевагою парних інтерв'ю є те, що учасники можуть спиратися на відповіді іншого, що призводить до ідей та спостережень, які можуть не виникнути під час індивідуальних інтерв'ю. Взаємодія між учасниками в групі підкреслює їхнє бачення світу, мову, яку вони використовують щодо різних тем, а також їхні цінності та переконання щодо конкретних ситуацій. Групові процеси, за її словами, можуть допомогти людям виявити та пояснити свої погляди способами, які неможливо зробити в індивідуальних інтерв'ю. Щоб досягти цього ефекту, наші пари склалися з людей з однаковою роллю. Однак парні інтерв'ю не позбавлені

					БР.ІП – 61.00.00.000 ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

недоліків. Перш за все, обговорення вимагають вмілої модерації від дослідника з точки зору забезпечення плавності обговорення, але водночас зосередження обговорення на темі. Ще одне завдання фасилітатора - надати простір для обох учасників, щоб вони могли зробити свій внесок. Оскільки в парних інтерв'ю брали участь двоє дослідників, одна людина вела інтерв'ю, а інша робила нотатки. Та, яка робила нотатки, також підтримувала основного інтерв'юера та стежила за тим, щоб усі важливі аспекти були охоплені.

У проекті «Мобільність» ми повідомляємо про наші висновки, засновані на восьми інтерв'ю з 14 респондентами, присвячених обговоренню процесу прийняття рішень. Ці інтерв'ю були частиною більшого поздовжнього дослідження, в якому було проведено загалом 28 інтерв'ю та сім семінарів з командами, а також проаналізовано організаційні схеми та описи ролей.

У Project Cloud ми провели чотири напівструктуровані інтерв'ю. Подібно до Project Mobility, ці інтерв'ю були частиною більшого поздовжнього дослідження гнучкої трансформації в цій організації, під час якого ми провели 78 інтерв'ю. Порівнюючи дані Project Mobility та Project Cloud на зустрічі всіх чотирьох дослідників, ми виявили потребу в додатковому зборі даних у Project Cloud для отримання відповідних даних з обох проектів щодо особливостей повноважень щодо прийняття рішень [27]. З цією метою ми провели чотири додаткові інтерв'ю з членом команди, власником продукту, лінійним менеджером та коучем. Таким чином, загалом ми провели 82 інтерв'ю з Project Cloud. Для цієї роботи початкові 78 інтерв'ю надали довідкову інформацію про історію, структуру організації, процеси та способи роботи. Аналіз прийняття рішень та автономії в розділі «Результати» базується на чотирьох нових інтерв'ю; тому ми включаємо лише ці чотири інтерв'ю, представляючи збір даних далі.

Усі інтерв'ю були записані та транскрибовані. Середня тривалість дванадцяти основних інтерв'ю, восьми з Project Mobility та чотирьох з Project Cloud, включених до цього дослідження, становила 73 хвилини. Усі інтерв'ю в Project Mobility проводилися англійською мовою, тоді як у Project Cloud три інтерв'ю проводилися фінською мовою, а одне – англійською. Цитати, використані в статті з Project Cloud, були перекладені з фінської мови дослідниками. Короткий

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		34

виклад інтерв'ю доступний у таблиці 2. Якщо в інтерв'ю брали участь більше однієї особи, це вказується у колонці « роль інтерв'юера».

Аналіз даних

Транскрибовані інтерв'ю були імпортовані в програмний інструмент для якісного аналізу даних NVivo. Ми кодували, використовуючи заздалегідь визначену схему кодування, засновану на концепціях влади. Згідно з схемою, під час виконання роботи в організації мають бути виконані чотири різні функції. Ми навели приклади того, як ми розуміємо ці функції в середовищі розробки програмного забезпечення. По-перше, хтось повинен задати напрямок для команди. По-друге, хтось повинен спроектувати команду, організувати необхідну організаційну підтримку для завдань структурування роботи, вирішити, хто їх виконуватиме, встановити основні норми поведінки в робочому середовищі та переконатися, що люди мають необхідні ресурси та підтримку [28]. По-третє, хтось повинен контролювати та керувати робочим процесом - збирати та інтерпретувати дані про те, як виконується робота, та ініціювати коригувальні дії за потреби. І, нарешті, хтось повинен виконувати роботу. Всі ці функції повинні бути присутніми в організації. Кодування для аналізу даних про Project Mobility та Project Cloud було виконано цієї роботи відповідно.

Ми дотримувалися цілісного процесу кодування, щоб зрозуміти чотири сфери повноважень, що обговорювалися в інтерв'ю, шляхом присвоєння фрагментів тексту категоріям ("node" в NVivo). Ми класифікували фрагменти інтерв'ю, пов'язані з прийняттям рішень, з точки зору особи, яка приймає рішення (команда, а не команда) або ролі (лінійний менеджер, власник продукту тощо). Читаючи стенограми, ми виявили різні типи рішень, які можна об'єднати в рамках чотирьох сфер повноважень. Наприклад, один з опитаних розповів про свою участь в обов'язкових курсах, які ми додали до нової категорії «Хто вирішує, що покращить практику» в розділі «Моніторинг та управління робочим процесом і прогресом».

У рамках відкритого кодування ми також закодували цікаві висловлювання думок, проблем, події, успішні масштабні практики, випадки або дії/взаємодії, а також інформацію, пов'язану з контекстом.

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		35

Створення спільного розуміння того, як кодувати чотири рівні повноважень, було ітеративним процесом. Частина інтерв'ю в рамках проекту «Мобільність» була спочатку закодована двома першими авторами, а потім усі автори зустрілися, щоб обговорити процес кодування, приклади кодів та способи тлумачення даних. Після цієї зустрічі два перші автори завершили кодування проекту «Мобільність», а два останні автори – кодування проекту «Хмара» [29]. Після завершення кодування ми синтезували наші висновки. Спочатку ми переглянули коди, пов'язані з чотирма сферами повноважень, і підсумували, як приймалися певні рішення в кожному конкретному проекті. Під час синтезу ми не лише виділили, хто мав повноваження приймати рішення з певної теми, але й чи було рішення прийнято в консультації з іншими або на основі їхніх внесків. Під час процесу синтезу ми провели кілька зустрічей з усіма авторами. Для збагачення резюме було додано ілюстративні цитати. Приклад процесу кодування можна знайти на рис. 2.4.

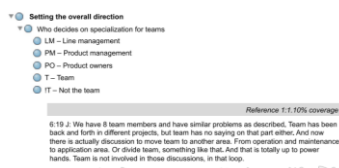


Рисунок 2.4 - Приклад процесу кодування

2.4 Висновок по розділу

У другому розділі було досліджено методи та інструменти забезпечення автономії команд у розробці програмного забезпечення. Розглянуто методи дослідження автономності команд, що дозволяють оцінювати рівень самостійності та ефективності роботи колективу. Проаналізовано підходи до збору та аналізу даних щодо ефективності командної взаємодії та продуктивності розробки. Також досліджено інструменти підтримки автономії команд, зокрема Agile-підходи, DevOps-практики та CI/CD-конвеєри, які сприяють оптимізації процесів розробки та швидкому реагуванню на зміни. У результаті визначено ефективні засоби підтримки автономної роботи команд у сучасних програмних проектах.

					БР.ІП – 61.00.00.000 ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. . РЕАЛІЗАЦІЯ ПІДХОДІВ ТА ОЦІНКА АВТОНОМІЇ КОМАНД

3.1 Розподіл повноважень і відповідальності в автономних командах

Під час виконання роботи в організації мають бути виконані чотири різні функції. По-перше, хтось повинен встановити напрямок для виконання організаційної роботи [30]. По-друге, хтось повинен розробити виконавчий підрозділ та організувати організаційну підтримку, необхідну для роботи. По-третє, хтось повинен окреслити, контролювати та керувати робочим процесом. Нарешті, хтось повинен виконувати роботу. Ми класифікували спостережувані явища за цими чотирма типами організаційної влади.

Повноваження команди в Project Mobility

У проекті «Мобільність» ми проаналізували міжфункціональні команди зі Швеції, які працювали над завданнями. Кілька ролей регулярно взаємодіяли з міжфункціональними командами. Це були оперативний власник продукту (зазвичай один на команду), лінійний менеджер (один на команду), Scrum-майстер (який може бути в команді або поза нею, або спільно з кількома командами), а також власник дизайну та відповідальні за технічні аспекти (TAR), які підтримують команди під час виконання завдань. Іншими важливими ролями, що з'явилися в Project Mobility, були власник системи та місцевий власник продукту (по одному для кожної області продукту); команда архітекторів, що складається з головного архітектора, архітекторів програмного забезпечення та архітекторів дизайну; та керівна група, яка в основному складається з лінійних менеджерів.

Визначення загального напрямку

Загальний напрямок розвитку продукту в Project Mobility визначався регіональними власниками продукту та власником системи, які приймали остаточне рішення щодо того, що має бути в системі. Разом вони визначали та пріоритезували відставання та призначали завдання щодо функцій командам.

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		37

Архітектура також визначає напрямок розвитку продукту . Архітектуру продукту визначала команда архітекторів, що складалася з експертів, що займали ролі, пов'язані з архітектурою, на рівні продукту та команди, тобто архітекторів дизайну, які були частиною звичайних команд і працювали на 50% як старші розробники . Команди повинні були консультуватися з архітекторами дизайну щоразу, коли функція, яку вони впроваджували, впливала на архітектуру. Іншими словами, командам дозволялося змінювати архітектуру за умови, що зміни були схвалені архітекторами.

Функції, що призначалися командам, тобто спеціалізація команд, також впливали на напрямок розвитку організації та команд [31]. Спеціалізація визначалася ринковим попитом і приймалася власниками продуктів у відповідній області. Відсутність участі в тому, над чим працюватиме команда, розглядалася як обмеження та те, що знижувало рівень її можливостей, як скаржився один Scrum-майстер:

«Команда постійно обмінювалася досвідом щодо різних проектів, але в цьому питанні команда не має права голосу. А тепер фактично обговорюється питання про переведення команди в іншу область, з експлуатації та технічного обслуговування в область застосування, або про розділення команди. І це повністю залежить від «керівних кадрів». Команда не бере участі в цих обговореннях, у цьому циклі. Ми не знаємо, що принесе день за днем. Це погано впливає на моральний дух».

Хоча власник продукту та власник системи визначали, над якими функціями працюватимуть команди, команди могли робити внесок у бачення та контент системи під час хакатонів, на яких від команд очікувалося, що вони запропонують власні ідеї, окрім тих, що були в беклозі. Хакатони – це щомісячні ініціативи, спрямовані на конкретний продукт, які запроваджує керівна команда, щоб надати командам платформу для впливу на напрямок розвитку продукту та виявлення інноваційних ідей. Як описав менеджер з розвитку:

«Я також вирішив, що один день на місяць кожен відділ повинен проводи-

					БР.ІП – 61.00.00.000 ПЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

ти день інновацій. І вони можуть займатися будь-якими інноваціями чи вдосконаленнями, які їм подобаються. Це має зрештою призвести до кращих результатів для Ericsson, але це не повинно бути відкладеним завданням у порядку АРО».

Хакатони були досить новими для організації, і не кожна команда чи член команди брали участь у хакатонах, тому їхній ефект ще належить визначити, як описав Scrum-майстер:

«Так (я беру участь), у нас був один минулого тижня для секції. Мені дуже подобається ідея, але важко знайти час. Але тепер це річ згори, ти маєш робити це один день на місяць. Тепер ми намагаємося частіше відвідувати це та планувати. Я спробую. Але для першого я був досить стресований, тому що намагався щось зробити в кінці дня».

Проектування команди та її організаційного контексту

Рішення щодо того, хто належить до міжфункціональної команди в Project Mobility, приймалися керівною групою [32]. Більшість цих рішень приймалися без консультацій з командами. Безпосередній керівник пояснив, що хоча рішення щодо розміру та профілю команди могли обговорюватися з командами, з командами не консультувалися щодо певних кадрових рішень, особливо щодо осіб, які повертаються з батьківської відпустки, коли команди змінювали спеціалізацію, об'єднували існуючі команди або створювали нові. Однак, як повідомлялося, це впливало на моральний дух команди, як пояснив Scrum-майстер:

«У нас з'явилися нові учасники. Не те щоб ми мали цього робити, ми їх просто отримали. Тож це дуже позбавляє нас сили. Але чудові хлопці, і вони нам подобаються. Але нам не було чого сказати з цього приводу. Це було справді погано».

Керівництво постійно змінювало допоміжні ролі команд. Наприклад, роль TAR (Transfer Assurance Reporter) була запроваджена для забезпечення якості результатів команди, зосереджуючись на розгляді дизайнерських пропозицій, коду та рішень щодо функцій. Пізніше цю роль було замінено роллю власника дизайну, щоб надати командам більше відповідальності, зосередившись на консультаціях на вимогу щодо архітектури продукту на ранніх етапах розробки. Впровадження, перевизначення та видалення цих ролей були частиною великого

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		39

процесу вдосконалення, який проводила керівна група. Як і у випадку з рішеннями щодо штатного розстановлення команди, команди майже не брали участі.

Моніторинг та управління робочими процесами та прогресом

Команди в Project Mobility мали право на значний ступінь автономії щодо своїх внутрішніх методів роботи. Крім того, ретроспективи команд використовувалися для обговорення того, чи потрібні корективи для пошуку кращих методів роботи та підвищення продуктивності. Scrum-майстер описав свободу вибору способу роботи наступним чином:

«Ми починали зі Scrum, можливо, теоретично, але ми адаптували власні методи роботи, які підходили команді. І ми постійно змінюємо речі. ми можемо використовувати Scrum, ми можемо використовувати Kanban деякий час. Тож немає загального рішення щодо того, чого слід дотримуватися».

Продуктивність команди контролювалася та обговорювалася кожною міжфункціональною командою та її близькими співробітниками, тобто членами команди, оперативним власником продукту та лінійним менеджером. Моніторинг ефективності включав зворотний зв'язок щодо результатів та їхньої якості. Ми виявили, що оперативні власники продукту та TAR раніше займалися контролем якості за допомогою обов'язкових оглядів коду та демонстрацій [33]. Однак, оскільки команди отримували більше досвіду, необхідність «функцій поліції» обговорювалася та ставилася під сумнів, як описав оперативний власник продукту:

«Я бачу один ризик у ролі TAR як ролі «гейткепера»: команда може використовувати її як буфер безпеки. Якщо команди думають: «Гаразд, якщо щось не так, це виявить TAR», і я думаю, нам потрібно від цього відмовитися. Тож команди беруть на себе цю відповідальність, а не покладаються на когось іншого, щоб той перевіряв і переконався, що все справді добре».

Хоча команда мала свободу вибору робочого процесу, існувала велика потреба в узгодженні команд; багато обов'язкових процесів і навіть покращень були визначені лінійними менеджерами та групою керівництва. До них належали контрольний список якості перед задачею, тривалість спринту, обов'язкові звітні зустрічі та навчальні курси. Однак неможливість впливати на обов'язкові

					БР.ІП – 61.00.00.000 ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

процеси не завжди добре сприймалася командами, оскільки до команд різного рівня зрілості ставилися однаково. Як пояснив Scrum-майстер, цільові покращення іноді сприймалися як безглузді, якщо обов'язкові рішення вирішували проблему, з якою не стикалася конкретна команда:

«Тепер, чого я раніше не бачив, так це безліч директив. Кожна команда повинна робити те й те, замість того, щоб працювати з командами та йти вперед, щоб з'ясувати, які проблеми у вашій команді, що ви можете зробити для покращення команди...»

Бо хтось це зробив, і це було добре, тож усі повинні це робити... замість того, щоб намагатися зосередитися на тому, що можна покращити в різних сферах у різних командах. Цього б я й волів.

Кожна команда відповідала за модульне тестування, тоді як інтеграційне та системне тестування виконувала команда безперервної інтеграції (CI).

Виконання командного завдання

Команди отримували завдання з беклогу, причому функції були пріоритетні для місцевого власника продукту. Завдання зазвичай містили короткий опис, який команда згодом деталізувала та розробляла. Команди взаємодіяли з оперативними власниками продукту під час розробки функцій і могли консультуватися з власниками дизайну, якщо була потрібна думка технічного експерта або архітектурна експертиза та порада.

Коли робота над функціями була завершена та протестована, команда упакувала функцію, перевірила її повноту, заповнивши контрольний список готовності, та відправила її до відділу безперервної інтеграції. Остаточну доставку замовнику здійснили менеджери з релізів.

Повноваження команди в хмарі проектів

У Project Cloud усі команди були членами спільноти, яка була центральним органом прийняття рішень у проекті. Це надало командам повноваження як на рівні команди, так і на рівні проекту. Як зазначалося раніше, спільноту формували всі команди проекту та власники продуктів. За особливих обставин менеджери продуктів та лінійні менеджери могли брати участь у зустрічах спільноти, але вони не вважалися членами спільноти.

					БР.ІП – 61.00.00.000 ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

Громада приймала рішення, які раніше належали керівництву, включаючи стратегічні технічні питання, а також ті, що стосуються процесів, наприклад, визначення загального напрямку розвитку та визначення методів роботи. Ця передача повноважень щодо прийняття рішень від керівництва до громади стала значною культурною зміною в організації.

Визначення загального напрямку

З огляду на те, що продукт знаходився у зрілому сегменті ринку, управління продуктом зберегло основну відповідальність за загальний напрямок розвитку системи. Порівняно зі старим проектом, технологічна зміна від спеціалізованого обладнання до віртуалізованого хмарного рішення надала можливості для нових функцій. Це, у свою чергу, призвело до того, що організація-розробник надавала як зворотний зв'язок щодо існуючих вимог, так і ідеї для абсолютно нових [34]. Взаємодія між розробкою та управлінням продуктом змінилася від командно-контрольного стилю до стилю, заснованого на довірі та діалозі, а також з певним відчуттям бунту проти традиційного способу роботи, як описав власник продукту:

«На практиці в нас більше немає управління «зверху вниз», тому, якщо виробник або архітектор щось вирішує, ніхто не сприймає це серйозно, якщо за цим не стоїть громада. Рішення має бути прийняте громадою та схвалене нею».

«Загальний напрямок визначає громада. Але це дуже складно, наприклад, якщо у них [штаб-квартири] є архітектор, який має право вирішувати, і він приходить сюди і наказує нам щось зробити, а потім йому кажуть, що він не може вирішувати. що у вас немає влади, ви можете говорити все, що хочете, але громада все одно вирішить і здійснить впровадження. Тож у нас є репутація піратського корабля в тому сенсі, що ми сміливо приймали рішення поза межами основного потоку, які, озираючись назад, виявилися дуже хорошими».

Перехід від організації-виконавця до організації, яка тісно бере участь у вирішенні питань, ґрунтувався на тому факті, що спільнота Project Cloud отримала цінні знання про можливості та обмеження нової технології. Це дало їм впевненість у можливості висловлюватися щодо рішень, пов'язаних з вимогами, та вести переговори з керівництвом продукту. Менеджер продукту стверджував,

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		42

що це відкрите обговорення та переговори, засновані на глибокому розумінні продукту та клієнтів у спільноті, призвели до покращення вимог:

«Ми [спільнота] справді дивні істоти в тому сенсі, що якщо виникає вимога [яку ми вважаємо безглуздою], ми. кажемо, що це насправді так не працює, ми цього не робитимемо. І зокрема, якщо ця вимога виходить від людини, яка не мала практичного досвіду роботи з хмарою, і вона сперечається з двадцятьма хлопцями [які є технічними експертами], які виконують реальну роботу з розробки, можна здогадатися, чиї аргументи кращі. Або іноді ми ставимо під сумнів цінність для клієнта, яку забезпечує [вимога], і це часто призводить до цікавих дискусій [35]. Принаймні, з нашим керівництвом продукту. Якщо ми кажемо їм, що це не створює цінності для клієнта, і ми можемо добре це мотивувати, то вимога досить швидко забувається».

Окрім обговорення вимог, спільнота відповідала за архітектурні рішення. У той час як стара організація мала окремі ролі архітекторів, у Project Cloud технічні керівники та експерти були частиною міжфункціональних команд. Таким чином, вони були занурені в технологію, беручи участь у щоденній технічній роботі, а не жили в «архітектурній вежі зі слонової кістки» далеко від фактичної розробки. Це дозволило командам виконувати розробку від початку до кінця, як пояснив власник продукту:

«Ми [в CoP] обмінюємося інформацією, разом вирішуємо проблеми та приймаємо рішення. І всі на одній лінії, а рішення здебільшого приймаються командами та спільнотою. У минулому. у нас був системний відділ, який вивчав питання за кілька років до цього та писав специфікації, які потім були остаточно визначені та передані розробникам. Зараз все навпаки. По суті, найбільшою зміною було те, що старі системні експерти та системні тестувальники були інтегровані в команди... Команди дійсно стали повністю готовими до роботи».

У проєкті залишилася одна роль архітектора, хоча ця роль перетворилася з особи, яка приймає рішення, на помічника, який підтримує громаду та допомагає забезпечити відповідність архітектурно значущих рішень, прийнятих громадою, корпоративним рекомендаціям.

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		43

Завдяки спільноті команди відчували, що вони можуть впливати на загальний напрямок розвитку організації, і що рішення, прийняті спільнотою, базуються на найкращих знаннях організації. Член команди пояснив, наскільки вони можуть впливати на загальний напрямок:

«На мою думку, досить багато. Зокрема, тепер, коли ми запустили цю віртуалізовану версію нашого продукту, у спільноті з'явилося багато людей, які оцінювали варіанти та просували її у правильному напрямку. І це, мабуть, одна з сильних сторін Ericsson Finland, що ми прийняли правильні рішення, оскільки ми широко обговорили проблеми та спробували виявити всі наявні всередині компанії знання».

Хоча організація прагнула сформувати команди, які могли б працювати над будь-якою функцією, без будь-якої спеціалізації, на практиці це виявилось складним. Головне занепокоєння стосувалося продуктивності, і це означало, що власники продукту здебільшого вирішували, над чим кожна команда працюватиме далі. Член команди пояснив:

«Я думаю, що це, можливо, трохи погано. Звичайно, вони [власники продукту] точно б нас послухали, якби ми сказали їм, що хотіли б робити більше цього і менше цього, але вони насправді не вимагають від нас багато... І якщо якась команда має певний штамп, що вона зосереджена на певних типах завдань, то власники продукту, оскільки вони зосереджені на пропускній здатності, завжди дають цій команді однакові завдання, що може бути неоптимальним з точки зору розвитку компетенцій. Тому я думаю, що власники продукту та лінійне керівництво повинні трохи подумати про потреби та бажання людей у розвитку компетенцій».

Проектування команди та її організаційного контексту

Хоча лінійне керівництво відповідало за рішення щодо рекрутингу та комплектування персоналу, з командами активно консультувалися, і їхні рекомендації, як правило, виконувалися. Багато рішень приймалися на основі діалогу між командами та лінійними керівниками. Під час нещодавнього процесу зовнішнього рекрутингу вибрані члени команди брали участь у співбесідах та активно

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		44

брали участь у технічних частинах співбесід, що давало їм вплив на те, хто має бути частиною команди.

Рівень повноважень, наданих командам щодо їхнього формування та організаційного контексту, можна проілюструвати на прикладі нещодавньої значної організаційної зміни [36]. Оскільки деякі команди були занадто малими, половину команд було призначено для реструктуризації. Цього було досягнуто, пропонуючи командам вибір: або команди самі приймають ці рішення, або лінійні менеджери приймають їх у консультації з командами. Один коуч описав:

«.. Команди вважали, що було б краще, якби рішення приймало лінійне керівництво, а з командами лише консультувалися. Було б справедливо сказати, що лінійне керівництво було дещо розчароване повільністю, пов'язаною з необхідністю обговорювати з людьми, вислуховувати їх, а потім обговорювати більше з кимось іншим. Але кінцевий результат, заснований на відгуках команд, видається досить хорошим, оскільки самі команди відчують, що вони змогли вплинути на ситуацію, і це робить їх набагато комфортнішими [з рішеннями]».

Що стосується організаційного контексту, власники продуктів або інші члени спільноти допомагали командам зв'язуватися з потрібними експертами, коли була потрібна зовнішня експертиза. В результаті команди Project Cloud мали прямий контакт з іншими командами та допоміжними ролями як всередині, так і поза проектом.

Моніторинг та управління робочими процесами та прогресом

Рішення щодо методів роботи та інструментів, які використовувати, приймалися як на рівні команди, так і на рівні спільноти. Якщо рішення не суперечило тому, що було узгоджено на рівні спільноти, і не створювало проблем для інших команд, рішення могла прийняти одна команда. Команди мали право визначати власні методи роботи, якщо це не впливало на інші команди. Це керівництво в Project Cloud мало на меті максимізувати автономію команди, обмежуючи її лише для того, щоб уникнути конфлікту чи хаосу в організації. Якщо команда хотіла внести зміни, що впливають на інших, або змінити узгоджене рішення спільноти, вона мала скликати зустріч спільноти. Це нам пояснив член команди:

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		45

«Команда самостійно вирішує [свої робочі процеси]... Так, і, наприклад, коли близько півтора року тому [в команду] прийшли нові люди, ми провели цю сесію [в нашій команді], де обговорили, як працювати, як ми хочемо працювати, і створили певні інструкції... Коли щось вирішено в спільноті, команда не може це змінити, а повинна повернути це спільноті [і запропонувати змінити]. Наприклад, спільно узгоджені речі, як-от використання інструменту Gerrit тощо, ми не можемо самі вирішити, що нам це не подобається і що ми не будемо це використовувати».

Щоб краще зрозуміти типи проблем, пов'язаних із процесами, які команда може вирішувати самостійно, а що належить спільноті, ми запитали, чи може команда, наприклад, самостійно вирішити використовувати Scrum чи Kanban. Ми дізналися, що оскільки між членами команди існують певні точки узгодження, рішення, що впливають на цю узгодженість, вимагають змін від усіх, і тому повинні прийматися в спільноті. На момент інтерв'ю всі команди мали синхронізовані двотижневі спринти [37]. Однак усередині спринтів кожна команда могла працювати так, як їй хотілося. Наприклад, багато команд використовували дошки Kanban, деякі мали фізичні дошки, а деякі - електронні.

Кожна команда узгодила зі своїм власником продукту спосіб моніторингу прогресу та продуктивності команди. Команди повинні були відстежувати свій прогрес за допомогою дощок Kanban або Scrum та проводити щоденні стендап-наради, в яких також брав участь їхній власник продукту. Команди розглядали щоденні стендап-наради як звітні зустрічі, а не як внутрішні командні наради, що проводилися для координації, через присутність власників продукту. Постачальники вимагали від кожної команди вести певний вид дошки, а також проводити щоденні стендап-наради.

Якість забезпечувалася переважно за допомогою автоматизованого тестування, яке проводилося на кількох рівнях та в різних циклах. Працюючи над новими функціями, команди впроваджували нові автоматизовані тести як частину розробки функцій. Команди також відповідали за безперервну інтеграцію та автоматизовану систему тестування під назвою «пральні машини». Команди по

черзі кожні два тижні контролювали роботу пральних машин. Тому можна сказати, що команди та спільнота в цілому відповідали за якість, а також за її перевірку.

Виконання командного завдання

Коли власники продукту призначали командам функції для розробки, вони виконували роботу з розробки самостійно, беручи на себе повну відповідальність за проектування, впровадження та тестування цих функцій. У випадках, коли їхня робота впливала на інші команди, дизайн чи архітектуру продукту, відповідні рішення приймалися спільнотою. На зустрічах спільноти інші команди та експерти, такі як архітектори, висловлювали свою думку щодо рішення.

Остаточну доставку клієнту здійснювали менеджери з релізів у співпраці з власниками продукту, оскільки підготовка релізу вимагала «бюрократичної» роботи, від якої команди були захищені.

3.2 Аналіз рівня автономності команд на прикладі кейс-проектів

У таблиці 3.1 ми підсумовуємо розподіл повноважень у двох досліджуваних проектах. Білі клітинки представляють області та рішення, за які відповідають окремі команди. Світло-сірі клітинки представляють області, узгоджені між командами та зазнавали впливу команд або були спільно вирішені командами як спільнотою, тоді як темно-сірі клітинки представляють області та рішення, прийняті лінійним керівництвом, власниками продуктів або іншими управлінськими ролями без попередніх консультацій з командами. Зауважте, що власники продуктів не вважаються частиною команд, оскільки це було так в обох наших випадках. Причиною цього було те, що власники продуктів (РО) у Project Cloud та власники продуктів (ОРО) у Project Mobility обслуговували кілька команд. Це означає, що наші команди називають Командою розробників у сенсі посібника Scrum, а не Командою Scrum, яка включає РО. У найновішій версії посібника Scrum концепція Команди розробників відсутня, а натомість згадується лише Розробники.

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		47

Нашою метою було зрозуміти обсяг повноважень, наданих окремій команді, яка працює над масштабним проектом розробки програмного забезпечення, на які рішення може впливати одна команда, і які рішення приймаються без консультацій з командами. Далі ми коротко порівняємо ці випадки.

Визначення загального напрямку

У Project Mobility власники продукту регіону та власник системи вирішували, що має бути в системі, тоді як команди могли зробити свій внесок у бачення та контент системи, пропонуючи ідеї під час хакатонів. У Project Cloud відкрите обговорення зі спільнотою мало сильний вплив на загальний напрямок, надаючи як відгуки щодо існуючих вимог, так і ідеї для нових, тоді як управління продуктом залишало за собою остаточну відповідальність за загальний напрямок розвитку системи.

Крім того, у Project Mobility архітектуру продукту розробляла команда архітекторів, що складалася з архітекторів-дизайнерів команд, тоді як у Project Cloud спільнота відповідала за архітектурні рішення [38]. Зрештою, у Project Mobility спеціалізацію команди та розроблені функції визначали власники продуктів регіонів, тоді як у Project Cloud власники продуктів визначали спеціалізації команд, але прислухалися до порад команд щодо того, над чим вони хотіли б працювати.

Проектування команди та її організаційного контексту

Керівна група в проекті «Мобільність» вирішувала, хто входить до складу кожної команди та які допоміжні ролі виконують, здебільшого без консультацій з командами, що негативно вплинуло на моральний дух команди.

У Project Cloud лінійне керівництво взяло на себе остаточну відповідальність за набір нових членів команди, а також за переміщення людей між командами, активно залучаючи команди до цього процесу. Спільнота допомагала командам у зв'язку з ключовими експертами та іншими командами.

Зведена інформація про розподіл повноважень.

Сфера повноважень	Тип рішень	Project Mobility	Project Cloud
Встановлення загального напрямку	Хто вирішує щодо завдань команди та спеціалізації?	APOs	POs
	Хто розробляє архітектуру продукту?	Команда архітекторів	Спільнота (Community)
	Хто вирішує, що має бути в системі?	POs та SO (вхідні дані з хакатонів)	PMs та POs разом зі спільнотою
Формування команди та контексту	Хто вирішує, хто входить до команди?	LM	LM після консультації з командою
	Хто може виключати людей із команди?	LM	LM після консультації з командою
	Хто визначає необхідні допоміжні ролі?	LM	Команда
Моніторинг та управління процесами	Хто визначає та покращує процеси проекту?	LG	Спільнота
	Хто контролює прогрес команди?	Команда, PO та LM	Команда та PO
	Хто покращує внутрішні процеси команди?	Команда або Команда, PO та LM	Команда або Команда та спільнота
	Хто перевіряє якість?	Команда, PO та LM	Команда та спільнота
Виконання завдань	Хто проєктує функції (features)?	Команда та OPOs	Команда
	Хто розробляє функції?	Команда	Команда
	Хто постачає (доставляє) функції?	RM	RM та PO

Білі клітинки позначають сфери відповідальності команд, світло-сірі — сфери, в яких команди можуть впливати на прийняття рішень або з приводу яких з ними проводяться консультації. Темно-сірі клітинки позначають сфери, в яких рішення приймаються без консультацій з командами/

Моніторинг та управління робочими процесами та прогресом

Кожна команда мала основні повноваження вирішувати свої внутрішні процеси в Project Mobility. Однак, для узгодження між командами, багато обов'язкових процесів, таких як контрольний список якості перед здачею та тривалість спринту, були визначені лінійними менеджерами та керівною групою.

Прогрес та якість команди контролювалися командою, її ОРО та лінійним менеджером [39]. Інтеграцію та системні тести виконувала окрема команда СІ. У Project Cloud загальні процеси проекту та зміни до них були узгоджені в спільноті. Внутрішні процеси команди та способи роботи визначалися кожною командою самостійно, якщо вони не суперечили угодам на рівні спільноти. Прогрес команди контролювався командою та її РО, а команди та спільнота несли відповідальність за якість своєї роботи.

Виконання командного завдання

У Project Mobility команди несли повну відповідальність за проектування, впровадження та модульне тестування функцій, а також отримували підтримку від своїх регіональних та оперативних власників продукту, а також від власників дизайну. Так само, у Project Cloud команди несли повну відповідальність за проектування, впровадження та тестування функцій, а також отримували підтримку від своїх власників продукту. В обох проєктах менеджери з релізів займалися остаточною доставкою клієнтам.

Порівняння випадків

Результати перехресного аналізу показують, що команди в Project Mobility мали аналогічний рівень повноважень, як і в Project Cloud, коли йшлося про виконання командних завдань та прийняття рішень щодо внутрішніх процесів команди. Однак рішення на рівні проекту в Project Mobility значною мірою приймалися без консультацій з командами, тоді як команди в Project Cloud або спільно приймали рішення, або принаймні впливали на них щодо управління способами роботи в рамках проекту, проектування команд та контексту проекту, а також визначення загального напрямку для розробленої ними системи. Це було зроблено через спільноту, сформовану командами, до якої входили технічні експерти та експерти з продуктів, а також власники продуктів. Через високий рівень технічної невизначеності на початку проекту виникла потреба об'єднати компетенції та разом знайти найкращі можливі рішення для виниклих проблем. Через цю технічну невизначеність усі були на одній стартовій лінії щодо компетенцій [40]. Успіх ранніх рішень довів, що рішення, прийняті спільнотою, мали вищу ймовірність успіху, ніж індивідуальні рішення, що стало причиною для

					БР.ІП – 61.00.00.000 ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

закріплення ролі спільноти як центрального форуму для прийняття рішень, особливо щодо технічних рішень. Єдині рішення, які були прийняті без консультації з командами Project Cloud, стосувалися розподілу завдань та спеціалізацій між різними командами. Тут команди традиційно не мали великого впливу на рішення, оскільки рішення приймали власники продукту. Однак опитані члени команди вважали це проблематичним і сподівалися, що команди також матимуть більший вплив у цій галузі в майбутньому.

RQ1.1 – Які основні виміри автономії впливають на плинність кадрів у проектах DSD? На етапі вилучення матеріалів було визначено деякі виміри автономії, пов'язані з концепціями, що визначали та мали зв'язок з видом автономії, наданої працівнику. Було знайдено вісім вимірів автономії. Нижче ми описуємо знайдені виміри.

Пов'язаність та компетентність були пов'язані з професійною потребою відчувати себе частиною команди та ідентифікувати себе з проектом, а також мати можливість навчатися новим навичкам та розвивати існуючі. Тому автономія повинна бути збалансована між цими двома вимірами, оскільки почуття стресу та демотивації можуть процвітати без компетентності для виконання завдань з вищим рівнем автономії. У цьому сенсі також зазначається, що фахівці, які не відчують зв'язку з роботою, можуть почуватися ізольованими, коли їм надається високий рівень автономії.

Контекстний вимір пов'язаний з контекстом, у якому ця автономія застосовується, незалежно від того, чи йдеться про продукт, особисті чинники, планування чи прийняття рішень. Контекст можна узагальнити як індивідуальну, внутрішню та зовнішню автономію.

Індивідуальна автономія стосується рівня свободи та свободи дій, які має людина у виконанні доручених завдань. З іншого боку, зовнішня автономія описується як вплив, який менеджери здійснюють на інших осіб, незалежно від того, чи беруть вони участь у діяльності команди, чи ні. Внутрішня автономія стосується того, як усі члени команди спільно розподіляють повноваження щодо прийняття рішень. Крім того, ці виміри автономії повинні бути збалансовані, щоб уникнути плинності кадрів у розподілених проектах.

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		51

Вимір *адаптації* пов'язаний з характеристиками роботи, які можуть призвести до плинності кадрів. Власні характеристики цих розподілених проектів можуть бути пов'язані з розподілом команди, оскільки офшорні проекти демонструють вищий рівень плинності кадрів порівняно з оншорними.

Офшорні розподілені проекти мають антисоціальний робочий час як сильну характеристику, що знижує автономію, пов'язану з робочим розпорядком, а також знижує професійну мотивацію. З іншого боку, розподілені проекти (оншорні та офшорні) позитивно пов'язані з різноманітністю роботи та зв'язком команди завдяки гнучким методологіям, що визначається як позитивний фактор для мотивації інженерів-програмістів.

Різнорманітність – це вимір, пов'язаний з досвідом фахівця. Тож робота, яка пропонує *індивідуальну* автономію, що може пробудити творчий потенціал працівника та забезпечити його знаннями, як правило, є кар'єрним якорем і вважається охоронцем передвісників намірів щодо зміни кадрового потенціалу.

Які основні фактори, пов'язані з автономією та плинністю кадрів у проектах DSD?

Ми визначили основні фактори, пов'язані з автономією та плинністю кадрів у проектах DSD. У літературі широко розглядається важливість врахування *інтересів працівників*.

Комунікація також була зазначена як важливий фактор, пов'язаний з автономією, головним чином у розподілених проектах. Коли компанія не може налагодити належний комунікаційний потік, поширеними є неефективні робочі процеси та працівники з низьким рівнем відданості організації та, як наслідок, високим рівнем плинності кадрів.

Інші фактори, пов'язані з *окремими особами*, також є важливими, такі як розумове перевантаження, конфлікти, пов'язані з ролями, соціальна підтримка, підтримка з боку організації, вимоги до роботи, алуценість та вигорання. Індивідуальні фактори безпосередньо пов'язані з автономією, оскільки вони можуть сильно впливати на мотивацію та задоволення від роботи, і все ще можуть сприяти більш значним намірам щодо звільнення, якщо вони не збалансовані.

Можливості працевлаштування та задоволення від кар'єри

					БР.ІП – 61.00.00.000 ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

безпосередньо впливають на наміри щодо звільнення. Автономія пов'язана з цими факторами, оскільки вона може значно покращити професійну задоволеність, дозволяючи працівнику розробляти план, пріоритети та практики в роботі. Автономія також може покращити якість завдань, розвиваючи відчуття більш спеціалізованої роботи у тих, хто бере участь у її виконанні, коли вона застосовується в мотивованій команді з високим рівнем задоволення від своєї кар'єри.

Режим роботи виявляється значним фактором, пов'язаним з автономією, оскільки навіть у незначній мірі автономія зазвичай сприймається як позитивний та превентивний фактор плинності кадрів. Однак, щоб це сталося, менеджери повинні шукати способи забезпечення належної комунікації між членами команди та гарантувати, що моніторинг команди не сприймається як авторитетний, оскільки така практика, як правило, знижує це відчуття індивідуальної автономії.

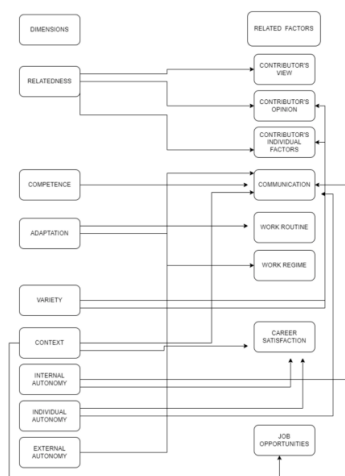


Рисунок 3.1 - Виміри автономії та пов'язані з ними фактори

Крім того, *робочий розпорядок* також є важливим фактором, пов'язаним з автономією. Певний рівень автономії сприймається як корисний для виконання завдань, оскільки він дозволяє працівнику розставляти пріоритети в призначеному йому проекті та встановлювати практики, які вважаються необхідними для його виконання. Ця більша свобода у визначенні пріоритетів дає працівнику відчуття вищого контролю, що стає превентивним фактором для намірів щодо звільнення.

Після проведення вилучення даних та визначення вимірів автономії, що

впливають на плинність кадрів у DSD, а також факторів, пов'язаних з автономією та плинністю кадрів, ми проаналізували виміри та фактори. Таким чином, ми зробили репрезентативну картину того, як виміри можуть впливати на фактори. Додамо, що Рисунок 2.4 представляє знайдені докази у SLR та що необхідні подальші дослідження, щоб показати кореляцію цього набору змінних.

Виміри автономії вказували на те, що тема має різні підрозділи, і що вони можуть по-різному проявлятися в робочому процесі інженера-програміста. Фактори, пов'язані з автономією, стосуються способів, якими автономія може модулюватися сприйняттям інженером-програмістом свого середовища та робочого процесу.

У цьому сенсі деякі виміри можуть бути пов'язані з пов'язаними факторами, оскільки вони стосуються факторів робочого середовища та їх вимірів. Наприклад, вимір пов'язаності пов'язує пов'язаний фактор Погляд, думку автора та індивідуальні фактори. Для того, щоб відчувати пов'язаність з роботою, важливо знати точку зору, думку та різні індивідуальні фактори працівника, аналізуючи ці фактори для кожного з них. Ці фактори можуть впливати на мотиваційні фактори працівника, як показано в: «Невідповідність між потребою людини в автономії та ступенем автономії, який вона насправді має, може впливати на рівень мотивації».

Вимір компетентності також може модулюватися фактором комунікації, оскільки автономія розглядається як мотивуючий фактор, коли фахівець компетентний виконувати завдання, інакше його мотивація може бути знижена, що спричиняє стрес. Комунікація є частиною цього процесу, оскільки важливо, щоб працівник міг спілкуватися зі своєю командою та керівництвом, навіть у процесах стресу та демотивації.

У розподілених проектах комунікація є одним з головних викликів для керівництва, оскільки багато з цих співробітників мають різні графіки та види діяльності. Як показано в «Командам потрібна спільна мета, необхідні навички, компетентність та взаємна довіра між членами команди для розвитку внутрішньої автономії. Протягом періоду проекту команди отримали повний набір навичок для розробки, обслуговування, розгортання, моніторингу та підтримки

					БР.ІП – 61.00.00.000 ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

програм, за які вони відповідають. Крім того, підтримка була надана експертами з права та комунікацій для забезпечення захисту даних та використання чіткої мови в продуктах». Також, «Робота окремих осіб може бути підірвана, якщо головний офіс діє жорстко та втручається в комунікацію, або якщо їхня робота контролюється занадто суворо. Для розробників, які працюють під пильним наглядом замовника, автономія може бути проблематичною».

Вимір адаптації, який пов'язаний з характеристиками роботи проекту, є прикладом розподілу команд. У цьому сенсі фактори режиму роботи та індивідуальні фактори працівника можуть бути пов'язані з цим виміром. Індивідуальні фактори впливатимуть на те, як працівник сприймає робочий розпорядок, визначений для команди, а робочий режим буде пов'язаний зі способом адаптації працівника до управління своїм робочим навантаженням та його конкретними характеристиками. Як показано в «Автономія має деякі передумови для реалізації, серед яких надмірність навичок (оскільки це впливає на здатність команди адаптуватися до змінних ситуацій)».

Вимір різноманітності тісно пов'язаний з креативністю працівника в робочому середовищі, де автономія може стати справжнім кар'єрним якорем. Пов'язані фактори думки та окремих працівників пов'язані з ідеєю, що ви повинні знати свого працівника, щоб мати змогу визначити його досвід. У цьому сенсі фактор якості завдання також пов'язаний з робочим процесом та творчою й управлінською роллю, яку виконує працівник.

Виміри контексту, в якому пов'язані внутрішня та індивідуальна автономія, можуть бути пов'язані з факторами, пов'язаними з комунікацією та задоволенням від кар'єри, як зазначено: «Внутрішня автономія стосується ступеня, до якого всі члени команди спільно розподіляють повноваження щодо прийняття рішень, тоді як індивідуальна автономія стосується обсягу свободи та свободи дій, які має людина у виконанні доручених завдань та їхньому задоволенні».

Зовнішня автономія також може бути пов'язана з відповідним фактором комунікації, як зазначено: «В ідеалі, в організаціях з високим рівнем довіри, команді слід дати проблему для вирішення, а потім рішення має бути виключно відповідальністю команди». Контекстний вимір може бути пов'язаний з

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		55

фактором, пов'язаним з можливостями працевлаштування, як досліджують: «Можливість працювати з дому була вигідною не лише для працівників, але й для фірм – надання працівникам можливості працювати з дому або з міст на їхній вибір призвело до дуже низької плинності кадрів». Крім того, життєво важливо знати кар'єрну орієнтацію працівників, оскільки на основі таких знань організації можуть знизити високі показники плинності кадрів, надаючи можливості працевлаштування та стимули, що відповідають кар'єрній орієнтації своїх працівників.

3.3 Оцінка ефективності та обговорення результатів впровадження

Великі та складні програмні системи можуть характеризуватися безпрецедентним масштабом з точки зору рядків коду, обсягу даних, що зберігаються, до яких здійснюється доступ, які маніпулюють та уточнюються, а також кількості з'єднань, взаємозалежностей, апаратного забезпечення, обчислювальних елементів та клієнтів. Такі системи розробляються в складних багатокомандних контекстах, де команді необхідно взаємодіяти з експертами та узгоджувати багато рішень щодо завдань і процесів з іншими командами та організацією. Як результат, автономія команди зменшується. З цих причин великомасштабні програмні проекти часто мають складну структуру управління з централізованими повноваженнями. Однак, гнучка розробка програмного забезпечення спирається на ідею автономних команд із широкими повноваженнями приймати рішення щодо власної роботи.

У цій роботі ми розглянули, як збалансувати автономію команди з необхідністю організаційного контролю та узгодженості у масштабних зусиллях з гнучкої розробки. З цією метою ми дослідили, як розподілялися організаційні повноваження у масштабній гнучкій розробці програмного забезпечення у двох емпіричних випадках в Ericsson - одному на фінському майданчику та одному на шведському. Це питання цікаве, оскільки в гнучкому середовищі очікується, що фокус прийняття рішень переміститься з керівника проекту на команду розробників програмного забезпечення, а процес прийняття рішень змінюється з індивідуального та централізованого на спільний та децентралізований. Водночас

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		56

організація повинна забезпечити, щоб усі гнучкі команди у масштабному проекті працювали над досягненням однієї мети, а робота була скоординована між командами, що традиційно досягається за допомогою формального організаційного контролю. Далі ми обговоримо, як розподілялися організаційні повноваження у досліджуваних проектах, і що ми можемо з цього дізнатися щодо балансування автономії команди та організаційного контролю.

Авторитет у масштабних Agile-проектах

В обох випадках проектів команди мали повний контроль лише над виконанням завдань. Для інших трьох рівнів повноваження або належали керівництву, або розподілялися між командами та керівництвом та/або допоміжними ролями, або приймалися на колективному форумі. Остання форма прийняття рішень представляла особливий інтерес, оскільки вважалося, що повноваження належать або робочому підрозділу, або керівництву (яке може консультиватися з робочими підрозділами або дозволяти їм впливати). Наші результати показують, що повноваження також могли формально належати спільноті розробників (розподілятися між командами).

Крім того, ми дізналися, що повноваження команди з часом мали тенденцію до зростання. По-перше, у Project Cloud впровадження спільнот зменшило управлінські повноваження, оскільки багато рішень (наприклад, щодо функцій, процесів, продукту та команд) тепер були передані командам. По-друге, у Project Mobility рівень функцій моніторингу (зовнішні експерти перевіряють роботу команди) був знижений, і команди стали відповідальними за управління якістю та відповідністю. Наші результати узгоджуються, що організації повинні ретельно розглядати належний рівень автономії та свободи дій, наданих командам. Хоча ми виявили докази того, що команди стверджують, що мають занадто мало повноважень, ми не знайшли доказів того, що команди отримують їх занадто багато.

Під час класифікації команд у Project Mobility та Project Cloud відповідно до матриці повноважень, справедливо буде сказати, що команди в обох великомасштабних проектах мають більше повноважень, ніж типовий підрозділ, очолюваний менеджером, але рівень повноважень все ще недостатній для того, щоб кваліфікувати їх категоризацію як самокерованих, самопроектуючих або

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		57

самокерованих підрозділів. Самокерування відбувається, коли підрозділ контролює рішення щодо виконання завдань, а також управління та моніторингу роботи та прогресу. Таким чином, ми розглядаємо команди в обох проектах як частково самокеровані, оскільки повноваження часто розподіляються з керівництвом та/або спільнотою. Крім того, вплив окремих команд на рішення вищого рівня різниться. У Project Mobility повноваження команди щодо організаційного контексту та дизайну підрозділу, а також загального напрямку, нижчі, ніж у Project Cloud, де команди залучені до прийняття рішень спільнотою.

Через популярність різних комерційних та некомерційних прескриптивних підходів та фреймворків до масштабування, ми додали три найпопулярніші з них до нашого порівняння, щоб побачити, як їхні рекомендації відповідають нашим результатам та який рівень автономії вони передбачають. Результат видно, куди ми додали фреймворк SAFe - Scaled Agile, фреймворк LeSS - Large-Scale Scrum та адаптацію Scrum, популярну завдяки Spotify, яку часто називають «моделлю Spotify». Спільнота в моделі Spotify стосується гільдій та інших міжкомпанійних підрозділів з представниками команд. Записи в таблиці базуються на нашій інтерпретації прескриптивних рекомендацій фреймворків, а не на окремих випадках їх впровадження.

Розглядаючи автономію команди в рамках масштабування, SAFe являє собою найтрадиційнішу модель з найвищим рівнем стандартизації та управлінської відповідальності, а отже, і найнижчим ступенем автономії команди. Команди зосереджені на розробці функцій у синхронізованих спринтах як частина 12-тижневих релізних поїздів, використовуючи процеси Scrum або Kanban. Хоча Project Mobility не спирався на модель SAFe, рівень автономії команди був близьким до того, що передбачає модель SAFe.

Модель LeSS має інший підхід до управління, наголошуючи на необхідності того, щоб менеджери сприяли роботі команд розробників та брали на себе відповідальність за системне вдосконалення організації розробки, але дозволяючи командам та власникам продуктів піклуватися про всі рішення щодо продукту, а також про вдосконалення процесів. Цей погляд видно в наданні командам вищого ступеня автономії. Як згадувалося раніше, модель розробки, що

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		58

використовується в Project Cloud, була натхненна LeSS, що також проявляється в дуже схожому підході до автономії команди, з тією основною відмінністю, що Project Cloud чітко визначив спільноту як орган, що приймає рішення.

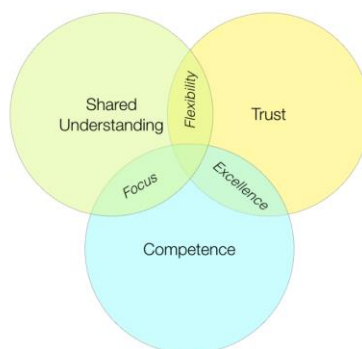


Рисунок 3.2 - Фактори взаємодоповнюють і підтримують один одного на шляху до створення високоефективних автономних команд.

Модель Spotify, також заснована на Scrum, має високий рівень автономії команди, оскільки команди беруть участь у прийнятті будь-яких рішень або можуть впливати на них, від виконання завдання до визначення загального напрямку. Особливе значення, як спільнота в Project Cloud, так і спільнота команд у моделі LeSS, мають також спільноти практиків у моделі Spotify, які називаються гільдіями. Керівництво, здається, має дещо більшу участь у прийнятті рішень у моделі Spotify, ніж у Project Cloud або моделі LeSS, оскільки воно залучене через лінійне керівництво або керівну групу як до рішень щодо проектування команди та її організаційного контексту, так і до визначення загального напрямку.

Узгодження роботи команди у великомасштабних Agile-проектах

Причини обмеження повноважень команди у досліджуваних випадках полягали не в тому, що організація використовувала авторитарний стиль управління (елементи якого ми також виявили), а в тому, щоб забезпечити, щоб усі команди, залучені до масштабних зусиль з розвитку, досягли очікуваних результатів щодо спільних цілей. Організації можуть досягти узгодженості та забезпечити досягнення цілей за допомогою двох основних режимів контролю: формального (поведінка та результат) та неформального (кланового та самоконтролю) режимів контролю. Формальний контроль «зверху вниз» здійснюється,

спираючись на документацію, письмові правила та процедури, які вимагають дотримання певної поведінки для досягнення бажаних результатів і можуть бути пов'язані з управлінським контролем. Неформальний або висхідний контроль заохочує групи та окремих осіб контролювати власну роботу, що вони можуть робити, якщо групові та особисті цілі значною мірою сумісні з цілями організації в цілому. Такий контроль може здійснюватися як колективні дії в так званому клані, члени якого спілкуються та погоджуються щодо набору норм і цінностей, що надає членам клану можливості.

Ми виявили, що в Project Mobility команди зазнавали впливу директив узгодження «зверху вниз», тоді як команди в Project Cloud використовували механізми узгодження «знизу вгору», в яких спільноту команд можна розуміти як клан. Як і в попередніх дослідженнях, які показали, що команди, які не беруть участі в розробці загальних політик та ініціатив узгодження, можуть не повністю дотримуватися цих ініціатив, ми виявили, що низхідний підхід у Project Mobility не підходив для всіх команд, а особливо не для досвідчених команд, які постійно розмірковували над власними процесами та вдосконалювали їх. Досвідчені команди сприймали запропоновані найкращі практики (запропоновані керівництвом) як загрозу їхній здатності приймати правильні рішення щодо своєї роботи. Наші висновки узгоджуються, що низхідний підхід до впровадження процесів у масштабних гнучких середовищах досяг неоднозначного успіху, оскільки багато членів вважали, що процеси були нав'язані тими, хто не розумів наслідків або проблем, які потрібно вирішити. Узгодження, що застосовується зовнішніми зацікавленими сторонами, знижує авторитет команди і, отже, є перешкодою для автономії команди у масштабних гнучких середовищах.

На відміну від переважно формальних механізмів контролю в Project Mobility, підхід «знизу вгору» в Project Cloud з часом виник, оскільки керівництво дозволяло та заохочувало команди брати на себе більше відповідальності. Підхід «знизу вгору» до узгодження спільних зусиль надає більший потенціал для розширення можливостей команди та підвищення її автономії, оскільки члени команди беруть участь у прийнятті рішень. Таким чином, результати цих ініціатив можуть мати не лише форму «універсального рішення», як це часто

					БР.ІП – 61.00.00.000 ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

буває у формальному контролі, орієнтованому на правила та процедури, але й враховувати ініціативи, адаптовані до конкретних потреб команд. Отже, такий підхід до узгодження може краще підтримувати команди у масштабних гнучких середовищах, де потреби кожної команди можуть суттєво відрізнятись. Розширення можливостей у Project Cloud було встановлено шляхом взаємних коригувань представниками команд у спільноті, новому неформальному підрозділі управління, що складається з усіх команд, що працюють у проекті, власників продукту та технічних експертів. Примітно, що лінійне керівництво не було частиною спільноти. Хоча окремі команди в цьому проекті не мали повних повноважень на всіх рівнях, ми вважаємо, що наявність сильного впливу на дизайн команди та загальний напрямок системи як частини спільноти задовольняє потреби команд і, водночас, потреби в узгодженні, які має організація. Важливо також зазначити, що спільнота мала право приймати рішення, а не просто рекомендувати рішення для затвердження керівництвом. Водночас ми виявили, що ні команди, ні спільнота в Project Cloud не мали значного впливу на спеціалізацію команди та управління компетенціями. Тому, хоча рівень повноважень, наданий спільноті як одиниці, перевищує рівень самоврядування, він не відповідає критеріям самопроектуючої або самоврядної одиниці. Однак результати нашого аналізу можуть допомогти проекту проаналізувати сфери, де повноваження спільноти обмежені, та вирішити, чи слід надати більше повноважень для підвищення рівня автономії спільноти, щоб задовольнити вимоги команди до подальшої автономії.

Визначення ідеального місця для командної автономії та організаційного контролю

Здатність збалансувати децентралізований процес прийняття рішень та централізовану структуру була важливим фактором успіху для великомасштабних проектів. Аналогічно, наші дані вказують на те, що ні повна автономія окремих команд, ні повна автократія управління не є дієвим підходом до розподілу повноважень у масштабній гнучкій розробці. Повна автономія окремих команд є нереалістичною, оскільки команди у великих організаціях не досягають результатів своєї роботи самостійно, і для забезпечення якості та уникнення загрози

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		61

взаємозалежності необхідна узгодженість. Рішення, прийняті керівництвом у великих та складних контекстах, можуть легко не враховувати потреби та складність роботи, що виконується окремими командами, через що принаймні деякі зусилля керівництва щодо узгодження сприймаються командами як непотрібні та такі, що відлякують їх.

Отже, яка альтернатива? Наші результати демонструють, як децентралізація влади може бути досягнута шляхом сприяння формуванню спільнот для спільного прийняття рішень «знизу вгору», що водночас задовольняє потребу в централізованій структурі та узгодженості. Це узгоджується, що в наукоємній роботі, де завдання менш повторювані та більш неоднозначні, або в середовищі, де потрібні інновації та креативність, клановий контроль є доречним та сильним механізмом контролю для організацій. У нашому випадку спільноту в Project Cloud можна розглядати як клан, де уповноважені члени разом приймають широкий спектр рішень, а потім контролюють команди та окремих осіб у цій спільноті. Оскільки всі команди залучені до клану, команди значною мірою контролюють себе, за винятком рішень, які впливають на всю спільноту. Подібні механізми кланового або спільнотного контролю були виявлені в трьох організаціях, які сприяють самоврядуванню. Примітно, що рівень децентралізації у трьох випадках, досліджених, відрізняється, причому рівень повноважень «знизу вгору» зростає пропорційно розміру організацій (найбільша організація мала найнижчий рівень децентралізації). Таким чином, стратегія балансування децентралізованих повноважень із централізованими структурами, здається, забезпечує узгодженість, необхідну для вирішення складності та залежностей, притаманних масштабній розробці програмного забезпечення, одночасно використовуючи переваги автономії команди. Хоча було б передчасно стверджувати, що це являє собою ідеальну «золоту середину» між двома протилежними силами, ми вважаємо, що цей баланс, досягнутий у Project Cloud та підтриманий моделями LeSS та Spotify, може наближатися до неї. Дійсно, можливо, що єдиної «золотої середини» не існує, але правильний баланс між формальним контролем та автономією команди залежить від контексту та може змінюватися, наприклад, залежно від організаційної культури, зрілості гнучкої реалізації, рівня кваліфікації

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		62

розробників, атрибутів продукту та його ринків тощо. Подальше дослідження цього питання за допомогою додаткових емпіричних досліджень може бути плідним.

Наслідки для практики та теорії

Наше дослідження дало низку висновків, які мають практичне значення. Щодо формального контролю, ми виявили, що:

- Забезпечення формального контролю може бути проблематичним, особливо для добре функціонуючих та досвідчених команд, і призвести до знеохочення та відсутності відданості таким ініціативам;
- Потреба у формальному контролі змінюється з часом, залежно, наприклад, від зрілості команд.

Для менеджерів масштабних Agile-проектів, які шукають способи підвищення рівня авторитету команди без шкоди для потреб узгодження, ми рекомендуємо знайти способи розвитку механізмів кланового контролю, які пропонуються як відповідний механізм контролю для наукоємної, менш повторюваної та більш неоднозначної роботи, а також коли потрібні інновації та креативність:

- Сприяти формуванню сильних організаційних спільнот.
- Навчати членів спільноти у створенні спільних ментальних моделей, розвитку довіри та зв'язків.
- Сприяти розвитку громад, наділених повноваженнями ухвалювати рішення.

Окрім практичних наслідків, наша робота має значення для теорії. Ми використовували модель для вивчення розподілу повноважень у контексті масштабних розробок. Модель пояснює більшість наших спостережень, але недостатньо моделює ситуацію, коли повноваження є спільними. Результати показують, що повноваження щодо прийняття рішень можуть бути частково децентралізованими або спільними, і ми додатково уточнили, чи приймаються рішення командами, чи спільно з керівними або допоміжними ролями, чи приймаються в умовах спільноти. Крім того, не описано певні важливі фактори, такі як зміни з часом або ситуативний характер розподілу повноважень. Таким чином, це дослідження

					БР.ІП – 61.00.00.000 ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

показує, що модель, хоча й корисна, потребує внесення змін, щоб краще відобразити складність розподілу повноважень у масштабних гнучких умовах.

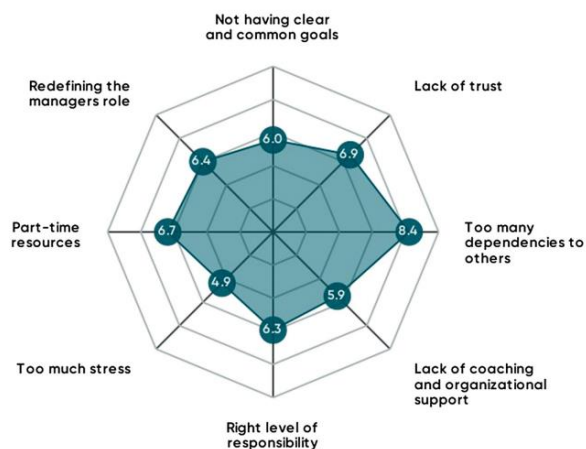


Рисунок 3.3 - Перешкоди для автономних команд.

Обмеження та загрози валідності

У цій роботі ми прагнули зрозуміти, як конфлікт між прагненням команд до автономії, як це розуміється в контексті гнучкої розробки програмного забезпечення, та прагненням організації до узгодженості та контролю, пов'язаним з великомасштабними проектами, проявляється в промислових умовах. Це було зроблено шляхом проведення дослідження кількох випадків двох масштабних проектів розробки програмного забезпечення в Ericsson, в яких кілька команд розробляли один і той самий продукт.

Наше емпіричне дослідження має описовий характер і призвело до детального аналізу розподілу авторитетності. Далі ми обговорюємо обмеження та загрози валідності наших висновків відповідно до визначень валідності та надійності.

Внутрішня *валідність* стосується валідності причинно-наслідкових зв'язків і має відношення лише до пояснювальних або причинно-наслідкових результатів, тому не є актуальною в нашому випадку, оскільки це дослідження має дослідницький характер.

Конструктивна *валідність* тематичного дослідження стосується визначення правильних операційних заходів для досліджуваних концепцій. Основною

загрозою конструктивній валідності цього дослідження було визначення того, що являє собою автономію команди та організаційний контроль у масштабних проектах розробки програмного забезпечення. Для цього ми використовували матрицю повноважень як теоретичну основу, яка розрізняє сферу відповідальності робочого підрозділу (автономію) та сферу відповідальності керівництва (контроль), використовуючи чотири різні сфери відповідальності. Одна з потенційних загроз конструктивній валідності в нашій роботі пов'язана з точністю нашої інтерпретації матриці повноважень Хекмана в контексті програмної інженерії, тобто типів рішень, що входять до кожної сфери відповідальності, та контексту багатокомандного проекту, тобто додаткових горизонтальних зв'язків між підрозділами на додаток до вертикального виміру між підрозділами та керівництвом, включеного до початкової матриці повноважень. Запропоновано конкретні типи рішень, базуючись на визначеннях сфер відповідальності та прикладах. Наші пропозиції узгоджуються з роботою, які вивчали децентралізацію повноважень у трьох компаніях (дві з яких були компаніями-розробниками програмного забезпечення) та базували свою емпіричну роботу на працях. Автори стверджують, що повноваження щодо прийняття рішень можуть бути централізовано зосереджені у менеджерів або децентралізовані в таких сферах:

- (1) управління персоналом та ефективністю;
- (2) управління та моніторинг виконання роботи;
- (3) виконання роботи;
- (4) розподіл роботи та ресурсів;
- (5) організація та проектування роботи;
- (6) стратегія фірми.

Наша інтерпретація моделі перетинається (щодо сфер відповідальності) та розширює (щодо типів рішень), а відмінності в інтерпретації концепцій зумовлені різними об'єктами дослідження (командами в нашому дослідженні та організаціями в їхньому). Щоб підвищити валідність наших спостережень, ми особливо спиралися на тріангуляцію даних та дослідників. Ми провели спільні семінари з чотирма авторами, щоб обговорити інтерпретацію моделі та створити набір визначень, які були використані для збору та аналізу даних інтерв'ю. На

					БР.ІП – 61.00.00.000 ПЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

конструктивну валідність також могло впливати те, як респонденти розуміли досліджуване явище. Щоб вирішити цю проблему, ми явно не включали питання щодо рівнів автономії, а натомість ставили питання щодо різних ситуацій прийняття рішень, що стосуються щоденних обов'язків респондентів. Примітно, що наші висновки обмежуються типами рішень, включеними до дослідження, і повторення дослідження в нових контекстах може вимагати додавання нових або зміни типів рішень, що стосуються чотирьох сфер відповідальності, або додаткових нових рішень чи зміни самих сфер відповідальності. Щоб забезпечити конструктивну валідність наших власних висновків, ми повідомили про результати на рівні окремих типів рішень та уникнули агрегування даних на основі рівнів повноважень або сфер відповідальності.

Зовнішня валідність тематичного дослідження стосується області, на яку можна узагальнити результати, що відомо як аналітичне узагальнення тематичного дослідження. По-перше, ми хотіли б зазначити, що ми вивчали два окремі випадки і таким чином узагальнюємо за подібністю та різноманітністю, на відміну від статистичних властивостей в узагальненнях на основі вибірки. У контексті нашого дослідження зовнішня валідність стосується ступеня підтримки узагальнення наших висновків щодо розподілу повноважень у великомасштабних проектах гнучкої розробки програмного забезпечення. Наші висновки свідчать про те, що повноваження у великомасштабних проектах у низці сфер відповідальності можуть належати командам або керівництву або спільноті. Примітно, що ми не стверджуємо, що повноваження будуть або будуть розподілені певним чином. Мінливість реального світу, головна проблема зовнішньої валідності, означає, що в інших контекстах фактичний спосіб розподілу повноважень може відрізнятись, як і у двох досліджуваних випадках. Однак ми вважаємо, що спостережуваний спільнотний спосіб прийняття рішень як альтернатива командним або управлінським рішенням може мати актуальність та практичність для подібних проектів. На основі нашого дослідження ми можемо висунути гіпотезу, що теоретичну популяцію, релевантну в нашому контексті, можна охарактеризувати наступними трьома аспектами. По-перше, в обох досліджуваних випадках системи, що розроблялися, були великими, багатогранними та технічно

					БР.ІП – 61.00.00.000 ПЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

складними. По-друге, організації в обох випадках трансформували свою розробку програмного забезпечення з каскадного процесу на гнучку розробку. По-третє, кількість команд розробників була відносно великою. Іншими словами, спільнотний повноваження можуть бути особливо цікавими для організацій, що розробляють великомасштабні складні програмні системи за участю багатьох команд розробників, які перейшли від традиційної каскадної розробки до гнучких методів роботи. Варто зазначити, що хоча обидва досліджені випадки були розподілені по кількох місцях, ми не вивчали розподіл повноважень у всіх місцях і не можемо робити конкретних заяв щодо того, чи розподілені повноваження однаково чи по-різному. Потрібні подальші дослідження, щоб зрозуміти природу розподілу повноважень у кількох місцях. Нарешті, щоб підвищити аналітичну узагальнюваність наших описових висновків, ми надали розширені контекстуальні описи кожного з досліджених випадків, щоб дозволити аналітичні порівняння та синтез з іншими дослідженнями на подібні теми.

Надійність дослідження стосується повторюваності дослідження. Якби інші дослідники провели таке саме дослідження, чи дійшли б вони тих самих висновків? Основною загрозою для надійності цього дослідження є мінливість у зборі даних. Збір даних проводився з використанням загального підходу до інтерв'ю, який вніс мінливість до тем, що обговорювалися в інтерв'ю. Наші результати також можуть бути піддані подальшим загрозам надійності з точки зору достовірності даних, зібраних під час інтерв'ю. Щоб вирішити цю загрозу, ми тріангулювали джерела даних, залучивши достатньо «широкий та різноманітний» набір респондентів, тобто (великий) набір, що містить респондентів, які виконують різні ролі в обох випадках. Ми також тріангулювали дослідників, проводячи всі інтерв'ю з двома інтерв'юерами спільно. Загальна кількість інтерв'ю для Project Cloud становила 82, чотири з яких були зосереджені саме на розподілі повноважень у проекті. Невелику кількість опитаних, лише з одним представником на кожну роль, можна вважати недоліком, оскільки ми не змогли триангулювати результати для кожної конкретної ролі. Однак усі відповіді опитаних на різних ролях узгоджувалися щодо розподілу повноважень у проекті, що змушує нас вважати, що додавання більшої кількості опитаних мало б обмежену

					БР.ІП – 61.00.00.000 ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

цінність. У проекті «Мобільність» кількість інтерв'ю та опитаних була більшою: вісім інтерв'ю з 14 опитаними. У проекті «Мобільність» інтерв'ю були частиною більшого поздовжнього дослідження, в якому було проведено загалом 28 інтерв'ю та сім семінарів з командами, і таким чином ми отримали хороші базові знання про цей випадок. Крім того, ми перевірили наш опис випадку з представниками проекту, які визнали, що наша інтерпретація щодо розподілу повноважень є точною. Ґрунтуючись на вищезгаданій триангуляції, ми впевнені, що наші висновки, особливо щодо проекту «Мобільність», точно відображають сприйняття членів організацій.

Нещодавня тенденція до гнучких трансформацій як на рівні розробки, так і на корпоративному рівні поставила під сумнів загальні принципи розподілу повноважень, що дозволило б приймати більше рішень «знизу вгору». Однак, як найкраще досягти балансу між необхідністю узгодження та пов'язаним з цим контролем якості та процесів, а також необхідністю автономності команд, залишається відкритим питанням. У нашому дослідженні кількох випадків ми підтвердили, що масштабні зусилля з розробки програмного забезпечення вимагають як узгодження, так і автономії, і що автономія змінюється з часом. Методи реалізації узгодження є широкими та різноманітними. Ініціативи узгодження «зверху вниз» включають централізацію повноважень щодо прийняття рішень у руках керівництва, впровадження керівних ролей та функцій, а також обов'язкових процесів та контрольних списків. Ініціативи узгодження «знизу вгору» включають надання повноважень щодо прийняття рішень командам, процеси прийняття рішень на основі консенсусу та делегування певних функцій від експертів командам. Водночас свобода окремих команд, що працюють у середовищі масштабної розробки програмного забезпечення, не може бути безмежною через складні залежності роботи з іншими командами та зацікавленими сторонами. Таким чином, структури, подібні до спільноти, подібні до кланових структур, з'явилися як альтернативні механізми контролю. Тому, хоча чистий контроль «зверху вниз» може знизити авторитет та моральний дух окремих команд, клановий контроль задовольняє потребу окремих команд впливати на рішення та брати участь у

					БР.ІП – 61.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		68

процесі прийняття рішень, водночас забезпечуючи неавторитарне узгодження та контроль між командами. Дійсно, може не існувати єдиної оптимальної зони для командної автономії та організаційного контролю. Правильний баланс між формальним контролем та командною автономією може залежати від контексту та змінюватися, наприклад, залежно від організаційної культури, зрілості гнучкої реалізації, рівня кваліфікації розробників та атрибутів продукту та його ринків. Слід провести подальші дослідження, щоб краще зрозуміти різні способи балансування потреби в узгодженні та автономії, а також вплив контексту на те, як це зробити успішно.

3.4 Висновок по розділу

У третьому розділі було розглянуто практичну реалізацію підходів до забезпечення автономії команд та оцінку їх ефективності. Проаналізовано принципи розподілу повноважень і відповідальності в автономних командах, що дозволяє підвищити рівень самоорганізації та продуктивності. Досліджено рівень автономності команд на прикладі кейс-проектів та визначено фактори, які впливають на успішність їх діяльності. Також проведено оцінку ефективності впроваджених підходів та аналіз отриманих результатів. Отримані результати підтверджують доцільність використання автономних моделей організації команд для підвищення якості та ефективності процесів розробки програмного забезпечення.

					БР.ІП – 61.00.00.000 ПЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У ході виконання бакалаврської роботи було досліджено підходи до розробки програмного забезпечення з фокусом на автономію команд, що є комплексним процесом, спрямованим на підвищення ефективності, гнучкості та якості створення програмних продуктів. У роботі проаналізовано існуючі моделі організації командної роботи, визначено їхні переваги та недоліки, а також досліджено сучасні методології розробки, які сприяють підвищенню рівня автономності команд. Охоплено повний цикл дослідження - від аналізу теоретичних основ до практичної оцінки ефективності впровадження автономних підходів у проєктах.

На початковому етапі було проведено аналіз предметної області, визначено ключові поняття автономії команд, ролей та механізмів прийняття рішень. Розглянуто існуючі підходи до організації команд у розробці ПЗ, що дозволило виявити основні проблеми, зокрема залежність від централізованого управління, затримки у прийнятті рішень та обмежену гнучкість процесів. Моделювання командної взаємодії та аналіз організаційних структур дали змогу сформулювати уявлення про ефективні механізми розподілу відповідальності та делегування повноважень.

У межах другого розділу було досліджено методи та інструменти забезпечення автономії команд, включаючи підходи до збору та аналізу даних, а також сучасні практики управління розробкою. Особливу увагу приділено використанню Agile-методологій, DevOps-практик та інструментів автоматизації, що сприяють скороченню часу розробки та підвищенню якості продукту. Визначено, що ефективне впровадження автономії потребує не лише технологічної підтримки, але й змін у культурі організації, зокрема довіри до команди та прозорості процесів.

У третьому розділі проведено аналіз практичного впровадження автономії команд та оцінено її вплив на результати розробки. Досліджено рівень самостійності команд у різних проєктах, їхню здатність приймати рішення,

					БР.ІП – 61.00.00.000 ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

відповідальність за результати та ефективність взаємодії. Отримані результати показали, що підвищення рівня автономності сприяє зростанню продуктивності, покращенню якості програмного забезпечення та зменшенню часу реагування на зміни вимог.

У процесі дослідження було також виявлено певні обмеження впровадження автономних підходів, зокрема ризики недостатньої координації між командами, можливі труднощі у масштабуванні та потребу в чітко визначених стандартах і процесах. Це підкреслює необхідність збалансованого підходу до впровадження автономії, який поєднує самостійність команд із ефективними механізмами контролю та взаємодії.

Загалом, результати роботи підтверджують, що орієнтація на автономію команд є перспективним напрямом розвитку процесів розробки програмного забезпечення. Вона дозволяє підвищити адаптивність організації, покращити якість продукту та скоротити час його виведення на ринок. У подальшому доцільним є дослідження підходів до масштабування автономних команд, інтеграції їх у великі організації та використання сучасних інструментів аналітики для підвищення ефективності управління процесами розробки.

					БР.ІІ – 61.00.00.000 ІЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА

1. Agile Manifesto. (2001). agilemanifesto.org. <https://agilemanifesto.org/>

2. Scrum Guide. (2020). scrumguides.org. <https://scrumguides.org/>

3. Microsoft Azure DevOps Documentation. (2025). learn.microsoft.com. <https://learn.microsoft.com/en-us/azure/devops/>

4. DevOps Resource Center. (2025). aws.amazon.com. <https://aws.amazon.com/devops/>

5. Kim, G., Humble, J., Debois, P., & Willis, J. (2021). *The DevOps Handbook*. IT Revolution.

6. Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate*. IT Revolution.

7. Bass, L., Weber, I., & Zhu, L. (2015). *DevOps: A Software Architect's Perspective*. Addison-Wesley.

8. Martin, R. C. (2017). *Clean Architecture*. Pearson.

9. Fowler, M. (2018). Microservices Resource Guide. martinowler.com. <https://martinfowler.com/microservices/>

10. Newman, S. (2021). *Building Microservices*. O'Reilly.

11. Team Topologies. (2019). teamtopologies.com. <https://teamtopologies.com/>

12. Skelton, M., & Pais, M. (2019). *Team Topologies*. IT Revolution.

13. Atlassian. (2025). Agile Coach. atlassian.com. <https://www.atlassian.com/agile>

14. Google Cloud DevOps Practices. (2025). cloud.google.com. <https://cloud.google.com/devops>

15. Humble, J., & Farley, D. (2010). *Continuous Delivery*. Addison-Wesley.

16. Continuous Integration. (2025). martinFowler.com. <https://martinfowler.com/articles/continuousIntegration.html>

17. DORA State of DevOps Report. (2024). cloud.google.com. <https://cloud.google.com/devops/state-of-devops>

18. Spotify Engineering Culture. (2023). engineering.atspotify.com.
<https://engineering.atspotify.com/>
19. SAE Framework. (2025). scaledagileframework.com.
<https://scaledagileframework.com/>
20. LeSS Framework. (2025). less.works. <https://less.works/>
21. GitHub Actions Documentation. (2025). [docs.github.com](https://docs.github.com/actions).
<https://docs.github.com/actions>
22. GitLab CI/CD Documentation. (2025). [docs.gitlab.com](https://docs.gitlab.com/ee/ci/).
<https://docs.gitlab.com/ee/ci/>
23. Docker Documentation. (2025). docs.docker.com.
<https://docs.docker.com/>
24. Kubernetes Documentation. (2025). [kubernetes.io](https://kubernetes.io/docs/).
<https://kubernetes.io/docs/>
25. ThoughtWorks Technology Radar. (2025). [thoughtworks.com](https://www.thoughtworks.com/radar).
<https://www.thoughtworks.com/radar>
26. Beck, K. (2003). *Test Driven Development*. Addison-Wesley.
27. Continuous Feedback in Agile Teams. (2024). [scrum.org](https://www.scrum.org/resources).
<https://www.scrum.org/resources>
28. Highsmith, J. (2009). *Agile Project Management*. Addison-Wesley.
29. Larman, C. (2004). *Agile and Iterative Development*. Addison-Wesley.
30. Cockburn, A. (2006). *Agile Software Development*. Addison-Wesley.
31. DevSecOps Guide. (2025). [owasp.org](https://owasp.org/www-project-devsecops/). <https://owasp.org/www-project-devsecops/>
32. Google SRE Book. (2016). [sre.google](https://sre.google/books/). <https://sre.google/books/>
33. Accelerating Software Delivery with Autonomous Teams. (2024). [mckinsey.com](https://www.mckinsey.com/). <https://www.mckinsey.com/>
34. Deloitte Insights on Agile Teams. (2025). [deloitte.com](https://www2.deloitte.com/).
<https://www2.deloitte.com/>
35. Managing Team Autonomy in Software Engineering. (2023). IEEE Xplore. <https://ieeexplore.ieee.org/>

36. Software Engineering Institute. (2024). sei.cmu.edu.
<https://www.sei.cmu.edu/>
37. ISO/IEC 12207 Software Life Cycle Processes. (2017). iso.org.
<https://www.iso.org/standard/63712.html>
38. ISO/IEC 27001 Information Security. (2022). iso.org.
<https://www.iso.org/isoiec-27001-information-security.html>
39. State of Agile Report. (2025). digital.ai. <https://digital.ai/resource-center/analyst-reports/state-of-agile-report>
40. Modern Software Engineering. (2019). Farley, D. Addison-Wesley

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: " Розробка ПЗ з фокусом на автономію команд у процесі створення "

Обсяг пояснювальної записки: 70 аркушів

Дата закінчення бакалаврської роботи 10 червня 2026р.

Підпис студента _____