УКРАЇНСЬКА АКАДЕМІЯ БАНКІВСЬКОЇ СПРАВИ

На правах рукопису

КОЛДОВСЬКИЙ В’ЯЧЕСЛАВ ВАСИЛЬОВИЧ

УДК 330.341.1:65.012.32:004.4

УПРАВЛІННЯ ІННОВАЦІЯМИ НА ЕТАПАХ ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Спеціальність 08.02.02 – економіка та управління науково-технічним прогресом

ДИСЕРТАЦІЯ
на здобуття наукового ступеня
кандидата економічних наук

Науковий керівник –
Козьменко Сергій Миколайович,
доктор економічних наук, професор

Суми – 2005

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ЕОМ	Електронно-обчислювальна машина
ІС	Інформаційна система
ПЗ	Програмне забезпечення
CMMI	Інтегрована модель характеристики зрілості (Capability Maturity Model Intergrated)
LOC	Рядки коду (Lines Of Code)
MSF	Інфраструктура рішень компанії Microsoft (Microsoft Solutions Framework)
SEI	Інститут інженерії програмного забезпечення (Software Engineering Institute)
SLC	Життєвий цикл програмного забезпечення (Software Life Cycle)
SLOC	Рядки вихідного коду (Source Lines Of Code)
SW-CMM	Модель характеристики зрілості програмного забезпечення (Software Capability Maturity Model)

ВСТУП

Актуальність теми. Починаючи з другої половини XX-го сторіччя спостерігається інтенсивний розвиток НТП, який значним чином був викликаний виникненням і широким розповсюдженням ЕОМ. Апаратне забезпечення ЕОМ вдосконалювалося до нинішнього часу і продовжує вдосконалюватися високими темпами, зокрема, продуктивність ЕОМ подвоюється приблизно кожні півтора-два роки. Щодо продуктивності розробки і показників ефективності програмного забезпечення (ПЗ), то їх ріст відбувається суттєво повільнішими темпами, ніж ріст показників апаратного забезпечення. Незважаючи на значний прогрес в сфері створення ПЗ, воно на даний час є і, ймовірно, залишиться у найближчому майбутньому результатом інтелектуальної праці людини, а тому в значній мірі залежить від здатності людей у обмежений строк створювати якісне ПЗ, яка розвивається порівняно повільними темпами. Згідно з різними статистичними оцінками, не більше третини проектів з розробки ПЗ можна вважати повністю успішними, інші закінчуються повним провалом, чи суттєво виходять за рамки встановлених бюджетних і часових обмежень.

Актуальність дисертаційного дослідження визначається необхідністю вирішення задачі забезпечення ефективного управління розробкою ПЗ, орієнтованого на створення конкурентноздатного продукту, за рахунок побудови дієвого механізму управління інноваціями на етапах життєвого циклу програмного забезпечення.

Дослідження проблем управління інноваціями проводили такі вітчизняні і закордонні вчені, як: А. Алімов, В. Божкова, Н. Гончарова, Л. Гохберг, С. Ільєнкова, С. Ілляшенко, Н. Кондратьєв, Л. Мельник, М. Портер, Д. Сахал, Й. Шумпетер, Ф. Янсен та ін. Питання управління розробкою ПЗ досліджували А. Альбрехт, Б. Боем, Ф. Брукс, Г. Буч, А. Джейкобсон, К. Джонс, Е. Йордон, А. Кокбюрн, Ф. Крачтен, С. Макконелл, Д. Парнас, Дж. Румбах, В. Хамфрі, С. Чулані та ін.

Курс на постійне поліпшення процесу створення ПЗ на всіх етапах життєвого циклу був визнаний як один з найважливіших, однак існуючі результати досліджень не дають цілісної картини для компанії-розробника ПЗ, яким саме чином його можна досягти в галузі управління інноваціями на етапах життєвого циклу ПЗ.

Недостатньо розробленими залишаються питання використання кількісних показників процесу розробки ПЗ як основи для прийняття управлінських рішень, існують протиріччя при використанні показників для визначення розміру ПЗ та моделей життєвого циклу, відсутні єдині погляди на проблему управління інноваціями для досягнення постійного поліпшення показників якості і вартості ПЗ.

Ставлячи за мету науково-методичне забезпечення управління інноваціями на етапах життєвого циклу ПЗ, автор виходив, насамперед, із необхідності створення комплексної системи управління інноваціями, інтегрованої із сучасними моделями життєвого циклу ПЗ, основаної на кількісних показниках прийняття рішень, з урахуванням сучасних тенденцій у галузі розробки ПЗ.

Усе вищевикладене й обумовило вибір об'єкта, теми, мети дослідження, її актуальність, сформувало структуру дисертаційної роботи.

Зв'язок роботи з науковими програмами, планами, темами Наукові результати, теоретичні положення і висновки дисертаційного дослідження були використані при розробці наукових тем і програм. Дисертант брав участь як співавтор у виконанні наступних тем і програм: «Сучасні технології фінансово-банківської діяльності в Україні» (№ ДР 0102U006965), де автором були запропоновані рекомендації щодо управління впровадженням нових технологій з використанням програмного забезпечення у банківській сфері; результати дослідження, теоретичні положення та висновки було використано при виконанні структурного розділу «Впровадження системи, побудованої з використанням інтернет-технологій для публічного моніторингу бюджетного процесу в Україні» науково-дослідної теми на замовлення Кабінету Міністрів України «Удосконалення використання програмно-цільового методу в процесі управління видатковою частиною бюджету» (затверджена розпорядженням КМУ № 729-р від 27.11.2003 р.), де автором запропоновано конкретний варіант впровадження системи публічного моніторингу бюджетного процесу в Україні, побудованої з використанням інтернет-технологій.

Мета і задачі дослідження. Метою дисертаційної роботи є вдосконалення науково-методичних підходів та розробка практичних рекомендацій щодо управління інноваціями на етапах життєвого циклу програмного забезпечення.

Відповідно до поставленої мети в дисертаційній роботі було визначено такі основні задачі:

- визначити сутність і місце економічного поняття «інновації на етапах життєвого циклу ПЗ»;
- проаналізувати моделі і стандарти життєвого циклу ПЗ на відповідність основним вимогам до ефективного управління інноваціями;
- дослідити науково-методичні підходи до визначення економічних параметрів процесу розробки ПЗ;
- вдосконалити науково-методичні підходи до розробки економічного механізму управління інноваціями на етапах життєвого циклу ПЗ;
- розробити науково-методичні підходи до управління впливом сторонніх компонентів на процес розробки ПЗ;
- розробити систему економічних показників для оцінки управління інноваціями на етапах життєвого циклу ПЗ;
- вдосконалити науково-методичні підходи до управління персоналом в процесі інноваційного менеджменту програмних проектів;
- вдосконалити науково-методичні підходи до оцінки економічної доцільності впровадження інновацій на етапах життєвого циклу ПЗ;
- запропонувати комплексний механізм управління інноваціями на етапах життєвого циклу ПЗ, побудований на основі системи економіко-технічних показників і інструментарію, призначеного для управління інноваціями при реалізації програмних проектів;
- розробити рекомендації щодо інтеграції системи управління інноваціями в загальну систему управління життєвим циклом ПЗ.

Об'єктом дослідження є економічні відносини, що виникають у процесі управління інноваціями при реалізації програмних проектів.

Предметом дослідження є організація управління інноваціями на етапах життєвого циклу програмного забезпечення.

Методи дослідження. Методологічну основу дисертаційного дослідження складають фундаментальні положення інноватики, сучасні концепції управління,

інноваційного менеджменту, окремі аспекти програмної інженерії, наукові положення, присвячені визначенню основних параметрів життєвого циклу ПЗ. Були використані науково-методичні розробки відомих вчених у галузі інноваційного менеджменту, а також у галузі управління програмними проектами.

У процесі дослідження були використані такі сучасні методи дослідження: порівняльний та системно-структурний аналіз (при здійсненні аналізу інновацій на етапах життєвого циклу ПЗ та моделей і стандартів життєвого циклу ПЗ), економіко-математичні методи (при розробці системи економіко-технічних показників для оцінки управління інноваціями на етапах життєвого циклу ПЗ), статистичні методи (при здійсненні аналізу використання сторонніх компонент та розробці підходів до управління впливом сторонніх компонентів на процес розробки ПЗ).

Інформаційно-фактологічну базу дослідження склали: зібрані, опрацьовані й узагальнені особисто автором первинні матеріали проектів з розробки ПЗ, статистичні звіти окремих інформаційно-аналітичних компаній світу, опубліковані в монографічній літературі та періодичних наукових виданнях результати досліджень фахівців з управління програмними проектами, матеріали внутрішньої документації і проведеного анкетування фахівців АТ «Датекс Україна».

Наукова новизна одержаних результатів полягає в розвитку відомих та розробці і обґрунтуванні нових науково-методичних підходів до управління інноваціями на етапах життєвого циклу ПЗ.

Найбільш значними науковими результатами дисертаційного дослідження є такі:

вперше:

- запропоновано комплексний механізм управління інноваціями на різних етапах життєвого циклу програмного забезпечення, побудований на основі розроблених автором системи економіко-технічних показників і науково-методичних рекомендацій, призначених для управління інноваціями при реалізації програмних проектів;

- запропоновано підхід до управління використанням сторонніх компонентів (складових програмних систем, які створюються і підтримуються зовнішніми поста-

чальниками) при розробці програмного забезпечення на основі формування і прийняття рішень, що базуються на економічних і технічних показниках діяльності компанії-розробника програмного забезпечення;

удосконалено:

- економічний зміст поняття «інновації на етапах життєвого циклу програмного забезпечення» за рахунок визначення сутності і особливостей процесу розробки програмного забезпечення;
- визначення ролі і місця, особливостей та меж використання моделей і стандартів життєвого циклу в процесі управління інноваціями при реалізації програмних проектів;
- науково-методичні підходи до управління персоналом в процесі інноваційного менеджменту програмних проектів за рахунок розвитку організаційно-управлінської структури і системи мотивації;
- науково-методичні підходи до оцінки економічної доцільності впровадження інновацій на етапах життєвого циклу програмного забезпечення за рахунок формування конкретних рекомендацій стосовно особливостей використання традиційних показників оцінки інвестиційних проектів по відношенню до програмних проектів і використання показників, які мають враховувати специфіку галузі програмного забезпечення;

дістали подальшого розвитку:

- науково-методичні підходи до визначення економічних показників процесу розробки програмного забезпечення на основі запропонованих автором показників: коефіцієнта успішності нової версії програмного продукту, коефіцієнта перспективності грошових надходжень, показника «відставання» середовища розробки, коефіцієнта залежності продукту від сторонніх компонентів, коефіцієнта коригуючого супроводу;
- процедура використання ітераційних підходів до управління життєвим циклом програмного забезпечення на основі безперервного поліпшення процесу розробки і впровадження програмних систем.

Практичне значення одержаних результатів полягає в тому, що представлені в роботі теоретичні положення, висновки і методичні рекомендації доведені до рівня конкретних і методик і пропозицій, які можуть бути використані з метою покращення показників програмних проектів, за рахунок орієнтації на постійне поліпшення процесу на основі управління інноваціями на етапах життєвого циклу ПЗ.

Отримані результати дисертаційного дослідження можуть бути використані в компаніях, які займаються розробкою ПЗ, з метою підвищення показників конкурентноздатності їх продуктів як у короткостроковому, так і у довгостроковому періодах. Розроблені автором пропозиції враховують сучасні підходи до управління життєвим циклом ПЗ і можуть бути інтегровані у процес розробки ПЗ компанії, незалежно від методологій, моделей життєвого циклу і стандартів, які використовуються.

Наукові і методичні положення роботи, зокрема комплексний підхід до управління використанням сторонніх компонентів при розробці ПЗ, комплекс інструментів управління інноваційними процесами на етапах життєвого циклу ПЗ, система економіко-технічних показників для оцінки управління інноваціями на етапах життєвого циклу ПЗ та підходи до управління персоналом при здійсненні інноваційних процесів використовуються відділом розробки ПЗ компанії АТ «Датекс Україна» (довідка № 253 від 25.06.2005 р.) Також основні положення дисертаційного дослідження використано як основу для методичної бази навчальних дисциплін «Інформаційні системи в менеджменті» та «Управління проектами інформатизації», які викладаються в Українській академії банківської справи НБУ (акт від 27 червня 2005 року).

Особистий внесок здобувача. Дисертаційна робота є самостійно виконаною науковою працею, в якій автором сформульовано і науково обґрунтовано підходи до управління інноваціями на етапах життєвого циклу ПЗ. Наукові положення, висновки і рекомендації, які виносяться на захист, одержані автором самостійно. З наукових праць, опублікованих у співавторстві, у дисертаційній роботі використано лише ті положення, які розроблені автором особисто.

У роботі [36] визначено перспективи та можливості використання альтернативних бізнес-моделей функціонування розробників програмного забезпечення на вітчизняному ринку. Особистий внесок здобувача полягає у визначенні основних характеристик і принципів використання альтернативних бізнес-моделей.

У роботі [7] аналізується вплив процесу розробки програмного забезпечення на систему автоматизованого управління бізнес-процесами компанії. Особистий внесок здобувача полягає у систематизації підходів і визначенні основних рекомендацій, які допомагають підвищити ефективність автоматизованого управління бізнес-процесами компанії.

У роботі [39] сформульовано рекомендації по створенню ефективної системи автоматизованого управління бізнес-процесами підприємства.

У роботі [30] визначено основні проблеми використання новітніх технологій у сфері розробки програмного забезпечення. Особистий внесок здобувача полягає у визначенні поняття «технологія» по відношенню до процесу розробки програмного забезпечення та формуванні конкретних рекомендацій з метою їх вдосконалення.

У роботі [37] запропоновано загальні рекомендації щодо використання універсальної мови BPML для опису бізнес-процесів компанії.

У роботі [78] визначено головні умови та методи, що мають бути використані під час реалізації проектів з автоматизації промислових підприємств. Особистий внесок здобувача полягає у запропонованих методах реалізації проектів з автоматизації, зокрема використанні спірального підходу та технології MSF.

У роботі [32] здійснено аналіз сучасних моделей розробки програмного забезпечення. Особистий внесок здобувача полягає у систематизації та характеристиці моделей.

У роботі [38] розглянуто актуальні питання управління інноваційними процесами при розробці програмного забезпечення, подані конкретні рекомендації та запропоновано інструменти управління інноваціями.

У роботі [40] запропоновано здійснювати інтенсифікацію інноваційних процесів за рахунок забезпечення безперервності інноваційного процесу та орієнтації на управління персоналом. Особистий внесок здобувача полягає у інтеграції системи

управління інноваціями у бізнес-процеси компанії з метою забезпечення безперервності інноваційного процесу, а також формуванні конкретних рекомендацій стосовно побудови ефективної системи управління персоналом при здійсненні інноваційних процесів.

У роботі [34] здійснено аналіз моделей життєвого циклу програмного забезпечення.

У роботі [41] розглянуті підходи визначення економічних параметрів інноваційних процесів на етапах життєвого циклу програмного забезпечення.

Апробація результатів дисертації. Основні положення і результати виконаного наукового дослідження доповідались, обговорювались та одержали позитивну оцінку на наукових та науково-практичних конференціях і семінарах: Першій міжнародній науково-практичній конференції «Сучасні проблеми управління» (м. Київ, 2001 р.), науково-технічній конференції викладачів, співробітників і студентів Сумського державного університету (м. Суми, 2002 р.), п’ятій міжнародній науково-практичній конференції «Мотивація інноваційно-інвестиційної діяльності підприємств та ринку праці в контексті інтеграції України до ЄС» (м. Хмельницький, 2005 р.)

Публікації. Результати дослідження опубліковано в 13 наукових працях загальним обсягом 7,60 друк. арк., з яких особисто дисертанту належить 5,28 друк. арк., у тому числі 7 статей у фахових виданнях (з них 5 – в співавторстві), 3 публікації в збірниках матеріалів конференцій (з них 1 – у співавторстві).

Структура та обсяг дисертації. Дисертація складається з вступу, трьох розділів, висновків, списку використаних джерел з 170 найменувань та додатків. Загальний обсяг дисертації 184 сторінки, у т.ч. містить 14 таблиць на 14 сторінках, 25 рисунків на 24 сторінках, список використаних джерел на 16 сторінках, 2 додатки на 7 сторінках.

РОЗДІЛ 1
АНАЛІЗ НАУКОВО-МЕТОДИЧНИХ ПІДХОДІВ ДО УПРАВЛІННЯ ІННОВАЦІЯМИ В СФЕРІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Сутність і місце інновацій на етапах життєвого циклу програмного забезпечення

Стрімкий розвиток НТП, починаючи з другої половини XX сторіччя, значним чином був викликаний виникненням і широким розповсюдженням ЕОМ. Однак самостійно без відповідного ПЗ ЕОМ не здатні виконувати будь-які корисні операції. Саме ПЗ надає ЕОМ можливості для здійснення корисних розрахунків, управління виконавчими пристроями, представлення інформації у будь-якій відомій формі і т.д.

Апаратне забезпечення, тобто продуктивність і ефективність самих ЕОМ, вдосконалювалося і продовжує вдосконалюватися стрімкими темпами, зокрема, у відповідності з емпіричним Законом Мура, який був виведений у 1965 р. та підтвердив свою правильність протягом сорока років, продуктивність ЕОМ подвоюється приблизно кожні півтора-два роки [13].

Щодо продуктивності розробки і ефективності ПЗ, то їх ріст відбувається суттєво повільнішими темпами, ніж зростання апаратного забезпечення. Незважаючи на значний прогрес в сфері створення ПЗ, воно на даний час є і, ймовірно, залишиться у найближчому майбутньому результатом інтелектуальної праці людини, а тому в значній мірі залежить від здатності людей у обмежений строк створювати якісне ПЗ, яка розвивається порівняно повільними темпами.

Перші ґрунтовні наукові праці з проблем ефективності процесу розробки ПЗ виникли ще у 1960-1970-х рр. і головні висновки, отримані дослідниками в той час істотно не змінилися і за сучасних умов. Зокрема, у 1975 р. Ф. Брукс, який на той час мав багатий досвід управління розробкою крупних програмних проектів, випустив перше видання своєї відомої праці «Міфічний людино-місяць чи як створюються програмні системи», в якій були отримані важливі висновки стосовно обмежень

продуктивності праці команд програмістів при розробці ПЗ, лише з незначними змінами і уточненнями ця робота перевидається протягом понад двох десятиліть [8].

У своїй роботі Ф. Брукс зазначив, що діяльність з розробки ПЗ є дуже специфічною, відрізняється особливою складністю і для неї не існує об'єктивних причин такого швидкого зростання продуктивності і скорочення вартості розробок, яке можна спостерігати у сфері мікроелектроніки [8].

Наука управління розробкою ПЗ має значну кількість питань, які досліджувалися вченими у різний час, зокрема, найбільш відомими є дослідження Б. Боема, Ф. Брукса, Е. Йордона, Д. Парнаса, С. Макконела, однак одним з найменш розкритих залишається питання управління інноваціями в даній сфері діяльності.

Дослідження проблем управління інноваціями проводили такі відомі закордонні і вітчизняні вчені як В. Божкова, Л. Гохберг, С. Ільєнкова, С. Ілляшенко, Н. Кондратьєв, Л. Мельник, М. Портер, Д. Сахал, О. Теліженко, Й. Шумпетер, Ф. Янсен та ін.

Основоположником теорії дослідження інновацій вважається відомий економіст Й. Шумпетер, який першим у економічній літературі провів межу між термінами «нововведення» та «інновації». Він стверджував, що «задача підприємців – реформувати і революціонізувати спосіб виробництва шляхом впровадження нових винаходів, а в більш загальному розумінні – за рахунок використання нових технологій для виробництва нових товарів, проте новим методом, завдяки відкриттю нового джерела сировини чи нового ринку готової продукції – в тому числі й до реорганізації існуючої та створення нової галузі промисловості» [75, С. 72].

Й. Шумпетер відзначав, що виробництво передбачає комбінування існуючих речей і сил, а виробництво чогось іншого чи іншим чином означає створення нових комбінацій із речей та сил і, відповідно, виділив п'ять випадків здійснення нових комбінацій:

- виробництво нового, тобто ще невідомого споживачам, блага чи створення нової якості того чи іншого блага;
- впровадження нового, тобто ще практично невідомого методу (способу) виробництва, в основі якого не обов'язково лежить нове наукове відкриття

і який може полягати у новому способі комерційного використання існуючого товару;

- отримання нового джерела сировини чи полуфабрикатів, незалежно від того, чи існувало це джерело раніше, чи просто не приймалося до уваги, не було доступним, чи його ще необхідно було створити;
- проведення відповідної реорганізації, наприклад, створення монопольного становища (шляхом створення тресту) чи підрив монопольного становища іншого підприємства;
- освоєння нового ринку збуту, тобто такого ринку, на якому дана галузь промисловості даної країни ще не була представлена, незалежно від того, існував цей ринок чи ні [75].

В цілому, розвиваючи ідеї Й. Шумпетера, різні вчені по-різному трактували термін «інновації» залежно від об'єкта, предмета та напрямку свого дослідження. Відмітимо ті трактування, які мають найбільш важливе значення по відношенню до нашого дослідження.

У науковій літературі поширеним є визначення інновації як «нововведення, що «революціоналізує», покращує, розвиває, вдосконалює вже існуюче в тій чи іншій сфері, причому під нововведенням слід розуміти новий вид продукції, технології, методу, послуги і т.д.» [42, С. 4]. Фактично подібне визначення узагальнює раніше наведення визначення Й. Шумпетера.

Інноваційний процес розглядається як «комплексний процес створення, розповсюдження і використання нового практичного засобу (нововведення) для нової (чи для кращого задоволення вже відомої) суспільної потреби, одночасно цей процес пов'язаний з впровадженням нових змін в тій соціальній і суспільній сфері, в якій здійснюється його життєвий цикл» [42, С. 4].

Подібне визначення інновацій і інноваційного процесу, на наш погляд, є занадто широким і фактично охоплює будь-які зміни, спрямовані на покращення способу здійснення діяльності з задоволення потреб споживачів, що ускладнює уточнення предмету вивчення саме економічної науки.

Відповідно до міжнародних стандартів інновація визначається як «кінцевий результат інноваційної діяльності, що отримав втілення у вигляді нового чи вдосконаленого технологічного процесу, що використовується в практичній діяльності, чи у новому підході до соціальних послуг» [65]. Важлива риса цього підходу до визначення інновацій полягає в тому, що акцент зроблено на фразі «кінцевий результат інноваційної діяльності», тобто зазначається, що впровадженню інновацій передує цілеспрямована діяльність, яка може охоплювати питання розробки чи пошуку нововведення і визначення економічної доцільності його впровадження у практику.

Інші підходи до визначення інновацій звертають увагу саме на основне джерело їх походження – науково-технічні розробки, зокрема інновація визначається як «використання результатів наукових досліджень і розробок, спрямованих на вдосконалення процесу діяльності виробництва, економічних, правових та соціальних відносин в області науки, культури, освіти і в інших сферах діяльності суспільства» [20, С. 4] чи «нововведення, пов'язане з науково-технічним прогресом (НТП) і, що полягає у відновленні основних фондів і технологій, в удосконаленні управління й економіки підприємства» [11, С. 9].

Ф. Янсен, аналізуючи ідеї Й. Шумпетера, запропонував визначення, згідно з яким інноваціями є «одночасний прояв двох світів, зокрема світу техніки і світу бізнесу. Коли зміни відбуваються лише на рівні технології, Шумпетер називає їх нововведенням. И лише тоді, коли до змін підключається бізнес, вони стають інноваціями» [77, С. 4].

Важливість такого визначення інновацій полягає у чіткому відокремленні тих змін, які неминуче і постійно відбуваються в межах будь-якого технологічного процесу, від тих, що можуть розглядатися як економічна категорія «інновації». На наш погляд, дана категорія набуває економічного забарвлення за тих умов, коли вона стає об'єктом впливу економічних процесів, зокрема об'єктом управління під час прийняття економічних рішень, тобто таких, що оперують поняттями економічної доцільності, вартості, окупності і т.д.

Подібний смисл вкладено в даний термін, коли інновації трактуються як «прибуткове використання новацій у вигляді нових технологій, видів продукції і послуг,

організаційно-технічних і соціально-економічних рішень виробничого, фінансового, комерційного, адміністративного чи іншого характеру» [11, С. 10]. В подібному випадку слід звернути увагу на словосполучення «прибуткове використання», що передбачає необхідність визначення економічної доцільності впровадження інновацій, що є, безумовно, сферою застосування знань економічної науки.

Слід звернути увагу, що існують підходи до визначення інновацій, у яких головна увага сфокусована на процесному підході до питання реалізації на практиці нововведень [50, 69]. Зокрема, у [49, С. 429] інновації трактуються як «процес розробки, освоєння, експлуатації та вичерпання виробничо-економічного та соціально-організаційного потенціалу, який лежить в новації».

Існують підходи, згідно з якими інновація може розглядатися одночасно як процес і як кінцевий результат цього процесу, необхідний зміст терміну визначається контекстом, у якому він використовується: «Інновація відноситься одночасно до результату і до процесу застосування технологічно допустимого рішення проблеми, викликаної технологічними можливостями чи клієнтськими потребами» [144, С. 68].

Сутнісне визначення інновацій представлено Д. Кокуріним. Зокрема, стверджується, що інновації є результатом діяльності по перетворенню та оновленню попередньої діяльності, що призводить до заміни чи доповнення одних елементів іншими. Перехід у новий стан чи набуття нової якості є наслідком об’єктивного сутнісного протиріччя між дійсністю та можливим станом. Цілеспрямована суб’єктивна діяльність соціальних суб’єктів по вирішенню даного протиріччя і являє собою інноваційну діяльність [33].

Також інноваційна діяльність визначається як види діяльності, які безпосередньо пов’язані з отриманням, відтворенням нових наукових, науково-технічних знань та їх реалізацією в матеріальній сфері економіки. В більшій мірі інноваційна діяльність пов’язується з доведенням наукових, технічних ідей, розробок до конкретної продукції і технології, що користується попитом на ринку [46].

Управління інноваціями (інноваційний менеджмент) визначається як комплекс формальних та неформальних правил, принципів, норм, установок та ціннісних орієнтацій, що регулюють різні сфери інноваційної діяльності [22].

Необхідно відзначити, що від розробки і впровадження інновацій безпосередньо залежить конкурентноздатність підприємства, зокрема, шістьома головними факторами конкурентноздатності згідно з думкою М. Портера у порядку зменшення їх значимості є наука, технологія, інформація, капітал, кваліфікована робоча сила, інфраструктура [56]. Причому слід підкреслити, що фактори, котрі формують інновації (наука та технологія) М. Портером визначені як найважливіші.

Крім того, М. Портер вказав на важливе джерело походження інновацій, він стверджував, що появу інновацій стимулює саме жорстка конкурентна боротьба [56].

Відносно класифікації інновацій, слід зазначити, що існує багато критеріїв, за якими інновації поділяються на певні групи. Для даного дослідження важливою є класифікація залежно від технологічних параметрів, згідно з якою інновації поділяються на продуктові (передбачають створення нових продуктів) та процесні (передбачають нові методи організації виробництва, тобто нові технології) [21].

Діяльність з розробки ПЗ за кількістю винаходів та інновацій не поступається будь-якій іншій високотехнологічній галузі, що означає необхідність проведення відповідних досліджень з метою підвищення ефективності інноваційного менеджменту. У табл. 1.1 представлено кількість патентів на захист винаходів у сфері ПЗ, поданих для реєстрації у країнах Європи за період з 1976 по 2003 р.

На нашу думку, що стосується сфери розробки програмного забезпечення, то доцільно більш детально розглянути сутність, структурні складові і особливості даного виду діяльності, перед тим, як визначити роль і місце інновацій по відношенню до нього.

Застарілий підхід, зокрема той, що зустрічається в радянській літературі, визначає програмне забезпечення як набір алгоритмів, викладених алгоритмічною мовою програмування чи у машинних кодах, і призначений для розв’язання певних задач з застосуванням ЕОМ. Зокрема, замість терміну «програмне забезпечення» дуже часто використовувався термін «математичне забезпечення» [18].

Таблиця 1.1

Кількість патентів на винаходи у сфері ПЗ, поданих у країнах Європи за період з 1976 по 2003 р. [100]

Країна	Кількість патентів	Частка у загальній кількості, %
США	32873	44,1
Японія	21709	29,1
Європейський Союз	16338	21,9
Канада	1219	1,6
Інші країни світу	2378	3,2
Всього	74517	100,0

Подібне досить вузьке визначення ПЗ є наслідком основної сфери застосування ПЗ у СРСР, якою була, насамперед, сфера наукових розрахунків, відповідно, цікавість для замовника розрахунків становив їх кінцевий результат, а не засіб у вигляді ПЗ, за допомогою якого він досягався. Відповідно ПЗ не розглядалося як самостійний ринковий продукт, підхід до створення якого має бути таким же, як до будь-якого товару чи послуги на ринку. Слід зазначити, що і в СРСР були окремі наукові дослідження з метою уточнення визначення ПЗ, наприклад, дослідженнями у цьому напрямку займався А. Бабій [4], проте, подібні публікації не стали основою більш серйозних наукових досліджень і практичних розробок.

Однак сучасний підхід розглядає ПЗ, насамперед, як товар, у якому зацікавлений кінцевий споживач [110, 129, 148, 134].

Сучасні підходи до визначення діяльності з розробки ПЗ розглядають її як таку, що містить три складові: процеси, продукти та ресурси [3]. Власне ПЗ слід розглядати невідривно від вказаних складових, оскільки взаємодія покупця і продавця не обмежується лише здійсненням операції купівлі-продажу, звичайно вона починається з детального аналізу вимог клієнта (чи ринку, коли мається на увазі масове ПЗ), включає взаємодію на етапі розробки і впровадження та продовжується на етапі супроводження продукту.

Діяльність з розробки ПЗ почала розглядатися як будь-яка інша самостійна діяльність з розробки ринкового продукту в зв’язку зі створенням концепції життєво-

го циклу ПЗ (SLC, Software Life Cycle), що стала наслідком проведеної у жовтні 1968 р. в німецькому м. Гармиш конференції підкомітету НАТО з науки і техніки [74]. Модель SLC отримала назву «каскадної» чи «водоспадної», її графічне представлення SLC зображено на рис. 1.1.

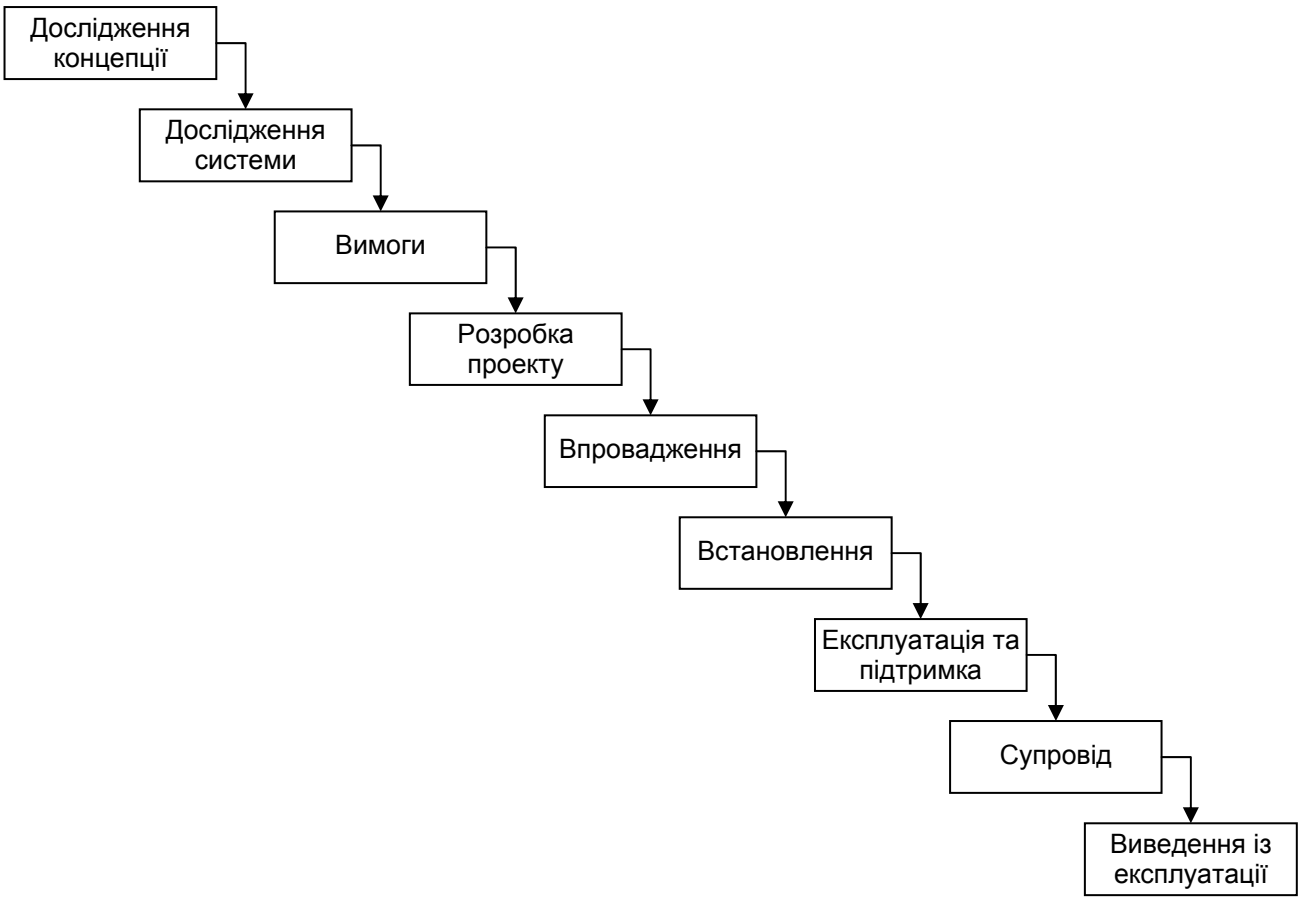


Рис. 1.1. Каскадна модель життєвого циклу розробки ПЗ SLC [74]

В даний час модель SLC вважається застарілою і у «чистому» вигляді майже не застосовується, однак вона стала основою практично всіх сучасних моделей вивчення життєвого циклу розробки ПЗ, термінологія та послідовність етапів SLC були запозичені більш досконалими моделями, тому їх сутність та значення для характеристики діяльності з розробки ПЗ не втратили своєї актуальності в наш час. Аналіз інноваційної складової моделей розробки ПЗ буде здійснено у параграфі 1.2.

Відповідно до рис. 1.1 діяльність з розробки ПЗ фактично включає 9 послідовних етапів. Узагальнено прийнято виділяти чотири етапи: проектування, розробка, впровадження та супровід. Проектування включає перші три етапи SLC (дослідже-

ня концепції, дослідження системи, вимоги), розробка включає четвертий етап SLC (розробка проекту), впровадження включає п’ятий та шостий етапи (впровадження та встановлення), супровід – останні три етапи (встановлення, експлуатація та підтримка, виведення із експлуатації). Для цілей даного дослідження етап «проектування» буде спрощено розглядатися як складова етапу «розробка», оскільки з економічного погляду результатом послідовного виконання цих двох етапів буде ПЗ як товар, який виступатиме об’єктом економічних відносин між розробником і клієнтом у наступних етапах впровадження та супроводу.

Іноді як окремий етап каскадної моделі виділяється тестування, наприклад у [3], однак цей етап доцільно виділяти лише для тих моделей, які передбачають ітераційне виконання чи повернення на попередній етап, у оригінальній каскадній моделі тестування є складовою частиною етапу розробки проекту і звичайно окремо не виділяється.

Даючи визначення сутності ПЗ, Ф. Брукс зазначає, що «сутністю програмного об’єкту є конструкція, що складається із поєднаних разом концепцій: наборів даних, взаємозв’язків між елементами даних, алгоритмів та викликів функцій» [8, С. 100].

Характеризуючи природу процесу розробки ПЗ, Ф. Брукс виділив дві групи труднощів, з якими зустрічаються розробники: труднощі, що витікають із сутності ПЗ, тобто такі, які внутрішньо властиві природі ПЗ, та труднощі, які супроводять розробку ПЗ на певному етапі розвитку науки і техніки, однак не є такими, що властиві його природі [8]. Труднощами, що витікають із сутності ПЗ були названі такі як складність, узгодженість, здатність до змін, незримість.

Складність ПЗ є характерною особливістю даного продукту цивілізації і проявляється у значно більш високій мірі, ніж у будь-якій іншій галузі людської діяльності. Ф. Брукс підкреслює, що складність є сутністю програмних об’єктів і нелінійно прискорюючими темпами зростає разом зі зростанням їх обсягів. На відміну від продуктів інших галузей людської діяльності, програмні об’єкти не складаються із повторювальних елементів, оскільки такі елементи в процесі розробки мають вилучатися і оформлюватися у вигляді допоміжних модулів. Складність ПЗ визначається великою кількістю можливих його станів, великим обсягом інформації, який містять

вихідні коди ПЗ, що в результаті призводить до не лише до технічних, а й до адміністративних проблем.

Г. Буч виділяє чотири головні причини, що викликають складність ПЗ:

- складність предметної області розробки ПЗ;
- складність управління процесом розробки ПЗ;
- необхідність забезпечення достатньої гнучкості ПЗ;
- незадовільні способи опису великих дискретних систем [9].

Узгодженість передбачає відсутність інформаційної асиметрії між усіма учасниками процесу розробки ПЗ. Неефективність комунікацій, різне сприйняття системи та припускання неперевірених допущень її розробниками призводять до того, що окремі складові єдиного програмного комплексу, а також взаємозв'язок ПЗ із зовнішніми вимогами стає неузгодженим, що вимагає суттєвих витрат ресурсів на усунення даної проблеми.

Здатність до змін. Характеризуючи здатність ПЗ до змін у порівнянні з іншими результатами людської праці, Ф. Брукс зазначає: «…програмне забезпечення легше змінювати: це чиста думка, нескінченно податлива» [8, С. 102]. Видима в теорії легкість змін ПЗ на практиці виливається у істотні витрати ресурсів, зростання складності та порушення концептуальної цілісності продукту. Об'єктивними факторами, що вимагають змін у ПЗ є необхідність усунення помилок, розширення функціональності та забезпечення сумісності із змінами середовища експлуатації.

Незримість. На відміну від карт, схем, рисунків та інших засобів візуального абстрагування, які можуть бути використані для представлення більшості об'єктів матеріального і нематеріального світу, програмні об'єкти дуже слабко піддаються наглядній візуалізації. Прийняті на практиці нотації блок-схем, моделей даних [15], мови графічного моделювання, такі як, наприклад, UML [43, 10], не здатні достатньо ефективно вирішити проблему наглядного представлення структури ПЗ і принципів його функціонування. Незримість властива природі ПЗ і призводить до складнощів під час його розуміння і осмислення, а також ускладнює процеси обміну інформацією між учасниками проекту.

Таким чином, природі ПЗ властиві об’єктивні складнощі, які, відповідно, відображаються на процес розробки ПЗ.

Слід відзначити, що характерна особливість діяльності з розробки ПЗ – велика доля витрат на робочу силу у загальній суми витрат на розробку. При розробці ПЗ доля витрат на оплату робочої сили за певних умов може досягати майже 100%, в той час коли інженерні компанії з інших сфер діяльності автоматично відносять проекти до проектів з високим ризиком, коли доля витрат на оплату робочої сили перевищує 50% [87].

Виробничі витрати, тобто витрати власне на відтворення копій розробленого ПЗ, можуть бути надзвичайно низькими у порівнянні з витратами на розробку [8, 74]. Це дозволяє використовувати такі форми розповсюдження програмних продуктів і побудови економічної моделі взаємовідносин з замовниками, які не прийнятні в інших сферах підприємницької діяльності, наприклад, безкоштовне розповсюдження програмного продукту і надання платних послуг впровадження і супроводу [36].

Що стосується раніше наведеного поділу інновацій на продуктові та процесні, то слід зазначити, що специфіка діяльності з розробки ПЗ передбачає створення нового продукту чи надання нових якостей існуючому продукту у кожному програмному проекті. Таким чином, створення продуктових інновацій є звичайною практикою діяльності компаній-розробників ПЗ, а саму діяльність, на наш погляд, можна охарактеризувати як інноваційну за своєю природою.

Далі розглянемо місце процесних інновацій у предметному виді діяльності.

В практиці управління проектами з розробки ПЗ використовується поняття «залізного трикутника», який визначає обмеження на найважливіші параметри програмного проекту: обсяг робіт, час та ресурси (рис. 1.2).

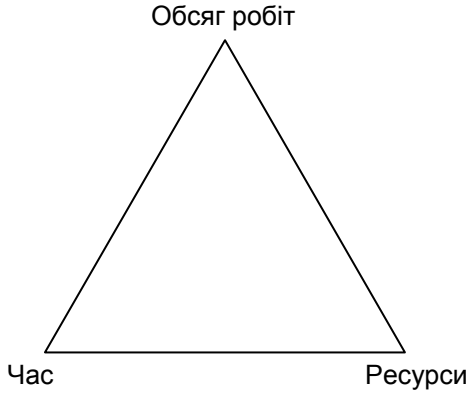


Рис. 1.2. «Залізний трикутник» – обмеження програмного проекту

Відповідно до правила «залізного трикутника» для забезпечення успіху програмного проекту необхідно, щоб фіксованими залишалися не більш, ніж дві його вершини, інакше проект буде перевантажений обмеженнями.

А. Коуберн пропонує додати до правила «залізного трикутника» нову вершину – процес і представити його таким чином, як зображено на рис. 1.3. [103]

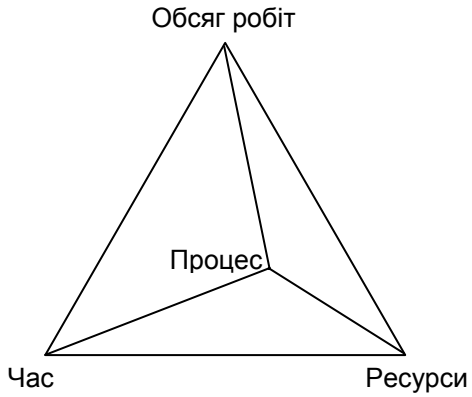


Рис. 1.3. Модифікований «залізний трикутник»

Необхідність введення додаткової вершини пояснюється наявністю на практиці проектів, у яких фіксованими є всі три вершини традиційного трикутнику, однак ці проекти залишаються успішними.

На наш погляд, процесні інновації дозволяють перевести трикутник із одного стану в інший, змінивши положення однієї чи одночасно двох його вершин: ресурсів та процесу. Графічно ефект інновацій можна представити наступним чином (рис. 1.4).

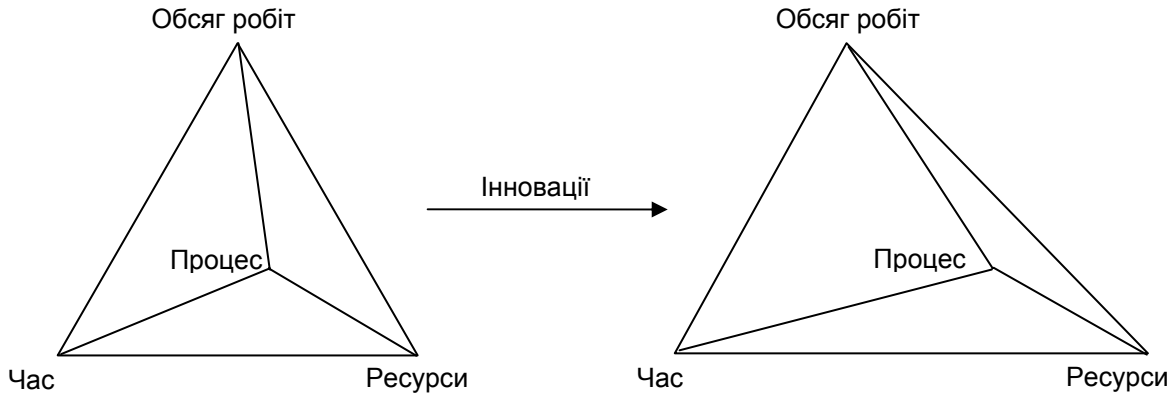


Рис. 1.4. Вплив процесних інновацій на зміну параметрів «залізного трикутника»

В результаті процесні інновації дозволяють змінити ті обмеження проекту з розробки ПЗ, які визначаються внутрішніми можливостями компанії-розробника.

С. Макконелл, порівнюючи сферу розробки ПЗ з іншими сферами підприємницької діяльності зазначає, що більшість непрограмних проектів сфокусовані на оптимізації продуктових цілей, оскільки витрати власне на розробку є порівняно не важливими, однак велика вартість праці для проектів з розробки ПЗ переміщує фокус на проектні цілі, тобто на необхідність забезпечувати економічну доцільність розробки з урахуванням обмежень на якість і строки розробки продукту [136]. Це твердження передбачає посилення уваги до вдосконалення внутрішніх процесів розробки ПЗ, відповідно зростає значення процесних інновацій по відношенню до продуктових.

Таким чином, діяльність з розробки, впровадження і супроводу ПЗ – це інтелектуальна діяльність, яка, як правило, здійснюється колективно, включає в себе окремі складові (етапи моделі SLC), що передбачають поділ праці та інтенсивний обмін інформацією між учасниками, характеризується інноваційною природою, складністю, високою здатністю до змін, відносно великою долею витрат на оплату праці у загальній структурі витрат. Дана діяльність передбачає не лише розробку і поставку програмного продукту, а й супровідної документації до нього, надання послуг з впровадження, що передбачають встановлення програмного продукту у замовника, його інтеграцію з існуючою апаратно-програмною інфраструктурою, проведення навчання персоналу, а також послуги супроводу, які передбачають модифікацію продукту після його поставки, усунення помилок у самому продукті та супрові-

дній документації до нього, здійсненні будь-яких інших операцій по підтримці продукту, передбачених домовленостями між розробником і клієнтом.

Слід зазначити, що в цілому галузь розробки програмного забезпечення характеризується надзвичайно високою інтенсивністю науково-дослідницьких розробок (НДР). Зокрема, згідно з проведеним аналізом показників 123-х глобальних компаній світу різних галузей промисловості та сфери послуг станом на 1998 р., галузь розробки програмного забезпечення разом з фармацевтикою займають лідируючі позиції за показником витрат на НДР по відношенню до обсягу продаж (12,1% і 12,6% відповідно), майже вдвічі перевищуючи за цим показником галузь, що йде наступною у рейтингу [19].

Висока інтенсивність НДР свідчить про високу інтенсивність інноваційної діяльності. На відміну від фармацевтики, де від початку наукових досліджень до виходу продукту на ринок можуть проходити десятки років, галузь розробки ПЗ відрізняється дуже коротким життєвим циклом продукту: рекомендований цикл зміни програмно-апаратної платформи становить близько 3-х років, відповідно цикл оновлення ПЗ має бути, як мінімум, не тривалішим за даний строк [130]. З іншого боку, на практиці цей строк виявляється значно тривалішим, зокрема, окремі дослідження наводять середній строк сприйняття інновацій галуззю ПЗ в межах від 10 до 15 років [154], що, на нашу думку, свідчить про недостатній рівень інноваційного менеджменту у галузі ПЗ і вимагає суттєвого його підвищення.

Таким чином, висока інтенсивність впровадження інновацій у галузі розробки ПЗ передбачає високі вимоги до управління даним процесом. Інновації виступають безпосереднім інструментом ведення конкурентної боротьби, конкурентне положення компанії, що працює в галузі високих технологій прямо залежить від ефективності її інноваційного менеджменту [68, 47].

В результаті, на основі результатів проведеного аналізу, визначимо основні ознаки економічного змісту поняття «інновації на етапах життєвого циклу програмного забезпечення», яке, відповідно, буде предметом даного дослідження.

По-перше, на нашу думку, для галузі розробки ПЗ у якості економічної категорії «інновації» слід вважати, насамперед, ті інновації, які стосуються удоскона-

лення внутрішніх процесів розробки ПЗ для визначеного ринку і задоволення визначеної потреби (процесні інновації). Оскільки процес розробки ПЗ є інноваційним за своєю природою, то продуктові інновації не будуть розглядатися у якості предмета даного дослідження і, на нашу думку, мають обмежену сферу використання у даній галузі.

По-друге, це має бути закінчений продукт, що підпадає під визначення економічної категорії товару чи послуги, незалежно від того, чи є він результатом цілеспрямованих науково-технічних розробок, чи випадковим винаходом, створеним як побічний продукт іншої діяльності, однак, як будь-який товар чи послуга він повинен мати свою ціну і повністю або частково задовольняти конкретну потребу споживача, у ролі якого в даному разі виступає компанія-розробник ПЗ.

По-третє, серед усіх можливих нововведень під час розробки ПЗ у якості економічної категорії «інновації» слід вважати лише ті, які можна розглядати як об’єкт управлінського впливу, тобто їх впровадження не є неминучим, можна розглядати як інші альтернативи, так і можливість взагалі відмовитися від подібного нововведення. Управлінське рішення має прийматися на основі кількісних показників, а за неможливості їх розрахунку – на основі методів, що передбачають подібну ситуацію.

У рамках даної роботи основна увага буде приділятися інноваціям, що витікають зі специфіки такої сфери діяльності, як розробка ПЗ. Тобто до уваги дослідження не будуть братися такі сфери застосування інновацій як пошук нових ринків, нових способів задоволення існуючих потреб споживачів чи розробки нових товарів (продуктові інновації). На наш погляд, подібне обмеження дозволить сфокусуватися на більш глибокому розкритті вузького кола задач у порівнянні з тим можливим застосуванням категорії «інновації» до такої інноваційної за своєю природою сфери діяльності, якою є розробка ПЗ.

1.2. Аналіз інноваційної складової моделей і стандартів життєвого циклу програмного забезпечення

Згідно з результатами аналізу Standish Group, проведеного за період з 1994 р. по 2000 р. на основі вивчення близько 30 тис. проектів з розробки ПЗ у різних країнах світу, значна кількість проектів з розробки ПЗ зазнала повної невдачі чи призвела до суттєвих перевитрат фінансових ресурсів і часу (рис. 1.5) [108].

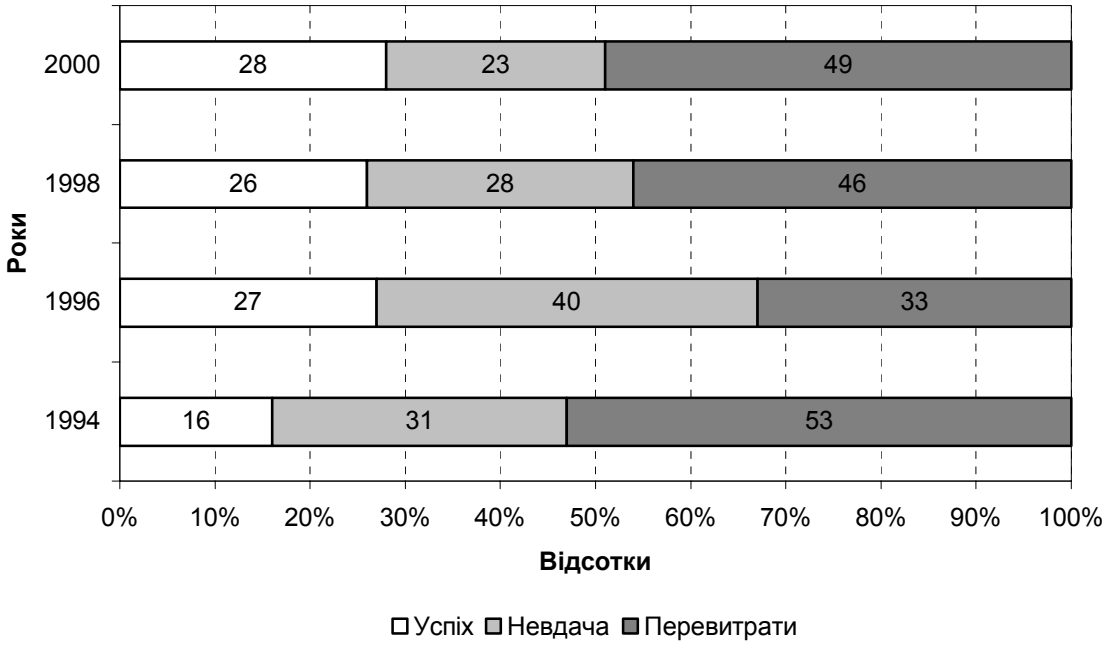


Рис. 1.5. Розподіл між проектами, що закінчилися успішно, невдачею чи призвели до суттєвих перевитрат фінансових ресурсів і часу за період 1994-2000 рр. [108]

Незважаючи на тенденцію зростання кількості успішних проектів і скорочення кількості проектів, що закінчилися невдачею у 2000 р. в порівнянні з 1994 р., в цілому ситуація є вкрай негативною, оскільки повністю успішними залишається значно менше третини проектів. Серед основних причин, згідно з результатами дослідження Standish Group, називається невміле чи неефективне управління проектами з розробки ПЗ.

Ефективне управління процесом розробки ПЗ можливе за умов повного розуміння процесів, що відбуваються в процесі роботи над проектом. Специфіка діяльності з розробки ПЗ викликала значну активність як окремих науковців, так і цілих

інституціональних закладів під час вирішення питання представлення адекватної моделі даної діяльності.

На початкових етапах розробки ПЗ процес не був структурованим, його модель можна було представити як постановку завдання та його виконання до того часу, доки воно не буде зроблено, чи відмінено [74].

Як було зазначено раніше (п. 1.1), першою загальновизнаною моделлю характеристики процесу управління розробки ПЗ вважається модель SLC. Однак незворотній характер етапів моделі не відображав прийняту у практиці більшості компаній-розробників ПЗ можливість повернення на попередні етапи.

Модель SLC була модифікована в 1970-х рр. У. Ройсом з урахуванням вказаного зауваження і отримала вигляд, представлений на рис. 1.6 [74].

Петлі зворотного зв'язку між етапами моделі передбачають тісну взаємодію між двома сусідніми етапами, наприклад, проектування може зворотнім чином впливати на визначення вимог, а розробка, в свою чергу, може вимагати уточнення і модифікацію проекту.

Однак навіть з урахуванням можливості повернення на попередній етап, модель SLC виявилася такою, що слабко відповідає вимогам до ефективного управління процесом розробки ПЗ.

Перехід з однієї фази SLC на іншу супроводжувався передачею проекту іншій команді учасників, виникали проблеми невідповідності вимог і специфікацій розробленому продукту, команди працювали розрізнено і з низькою ефективністю комунікацій.

Значна проблема полягала у суттєвій невідповідності отриманого в результаті продукту і вимог, які до нього висувалися.

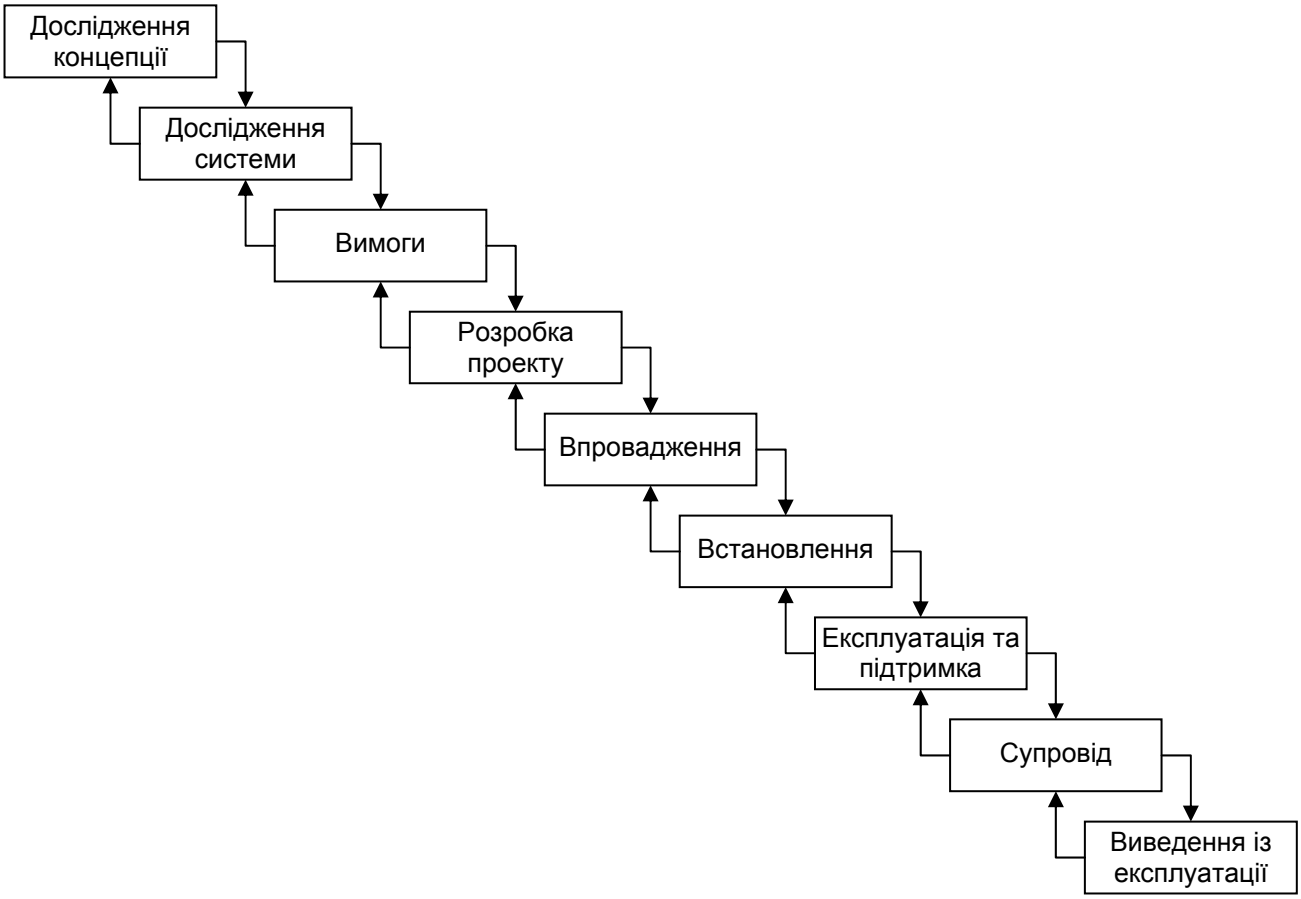


Рис. 1.6. Модифікована модель життєвого циклу розробки ПЗ SLC [74]

На заміну каскадній моделі була запропонована V-подібна (шарнірна) модель, яка покликана усунути зазначені недоліки, зокрема поєднати етапи постановки завдання та перевірки його відповідності специфікаціям (рис. 1.7) [27].

Однак «інженерний» підхід до розробки ПЗ, реалізований у моделі SLC та V-подібній моделі, виявився неконкурентноздатним, особливо для крупних проектів – тривала поетапна розробка проекту призводила до того, що процес розробки часто виходив за рамки встановленого бюджету та часових обмежень, а кінцевий продукт часто виявлявся таким, що запізнювався з виходом на ринок.

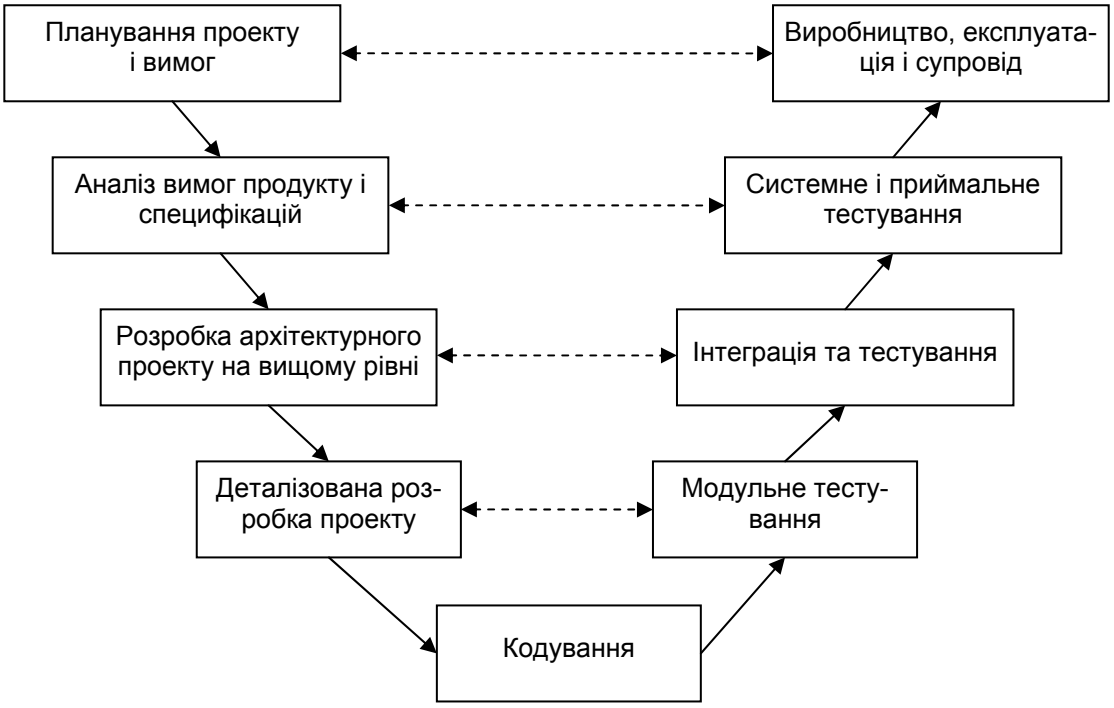


Рис. 1.7. V-подібна модель управління процесом розробки ПЗ [27]

З погляду управління інноваціями, слід зазначити, що моделі SLC та V-подібна модель не пристосовані до можливості впровадження ефективного інноваційного менеджменту. Реалізація проекту може тривати декілька років, однак, відповідно до даних моделей, розроблені на його початку вимоги та план реалізації не враховують науково-технічний прогрес, вони орієнтовані на використання тих технологій, які були актуальними на початок робіт над проектом, що в результаті призводить до створення продукту який є морально застарілим та неконкурентноздатним ще до свого виходу на ринок.

Вирішити основні недоліки моделі SLC покликана спіральна модель, яка відповідно до Б. Боема являє собою рухомий ризиком генератор процесної моделі, тобто являє собою модель процесу, в якій ризик виступає у якості фактору, що примушує її до розвитку [92].

Дана модель використовується для управління проектами з розробки програмного забезпечення, в яких бере участь значна кількість розробників. Вона має дві головні особливості: перша полягає в тому, що використовується циклічний підхід до поступового нарощування рівня визначеності та реалізації системи в умовах зни-

ження її рівня ризику; друга являє собою набір певних віх (чи контрольних точок), котрі необхідні для того, щоб була можливість відслідковувати реальність впровадження та двостороннє узгодження рішень.

Ризик у даному випадку являє собою ситуації чи можливі випадки, що можуть стати на заваді проекту при досягненні ним певної мети (рис. 1.8).

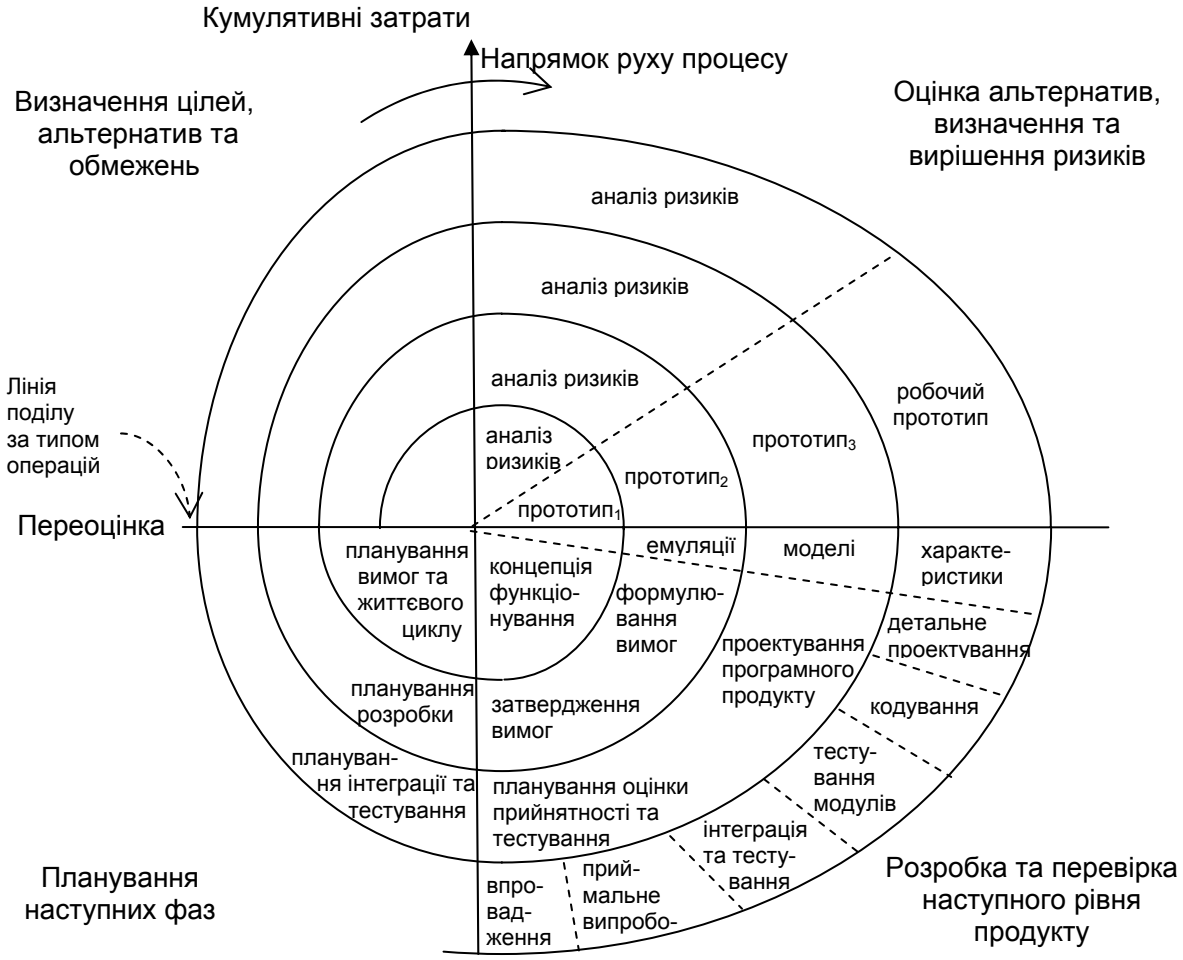


Рис. 1.8. Діаграма спірального розвитку [91]

Відповідно до рис. 1.8 можна відзначити, що рух до кінцевої точки, який відбувається у напрямку руху годинникової стрілки на даній діаграмі, проходить значну кількість стадій, на кожній з яких, по-перше, відбувається уточнення функціональності кінцевого продукту, а, по-друге, спостерігається постійне зростання кумулятивних затрат на проект, які можна визначити у кожен конкретний момент часу роботи над проектом як абсолютну величину відстані між кожною точкою на спіралі та точкою перетину осей на діаграмі.

На відміну від попередніх розглянутих моделей, спіральна модель значно краще підходить до ефективного управління інноваціями. Окремі витки спіралі розпочинаються з етапів оцінки ризику та планування, які можуть включати також і підбір та оцінку можливих інновацій у діяльності підприємства. Таким чином, кожен виток спіралі може передбачати інноваційну діяльність і розробка може бути спланована з урахуванням переваг, які будуть отримані в результаті впровадження інновацій у роботі підприємства-розробника.

Щоб підтвердити переваги спіральної моделі саме з погляду можливостей впровадження інновацій, Б. Боем наводить наступний приклад [91], згідно з яким на початку 1980-х рр. одна урядова установа стала замовником потужної інформаційної системи, що повинна була забезпечувати роботу понад однієї тисячі користувачів, розміщувалася у великому будівельному комплексі, мала потужні аналітичні та запитові можливості до великої динамічної бази даних. У контракті на розробку даної інформаційної системи була зафіксована вимога, згідно з якою час реакції системи на дії користувача не повинен перевищувати однієї секунди.

Використовуючи класичний підхід (каскадну модель) до проектування ПЗ і розробивши близько двох тисяч сторінок документів з вимогами, архітектори з програмного забезпечення визначили, що поставлене завдання можна реалізувати, побудувавши систему за Архітектурою А, приблизна вартість якої склала близько 100 млн. дол. США. Однак зазначена сума перевищувала рамки встановленого бюджету і реалізація проекту за таких умов була неможливою.

Лише завдяки створенню прототипу інтерфейсу користувача дослідним шляхом було встановлено, що вимога щодо часу реакції системи на дії користувача, який не повинен перевищувати однієї секунди, є завищеною, і достатньо затримки до чотирьох секунд. В результаті переробки вимог та вибору принципово іншої архітектури системи оціночна вартість системи була знижена до 30 млн. дол. США (рис. 1.9).

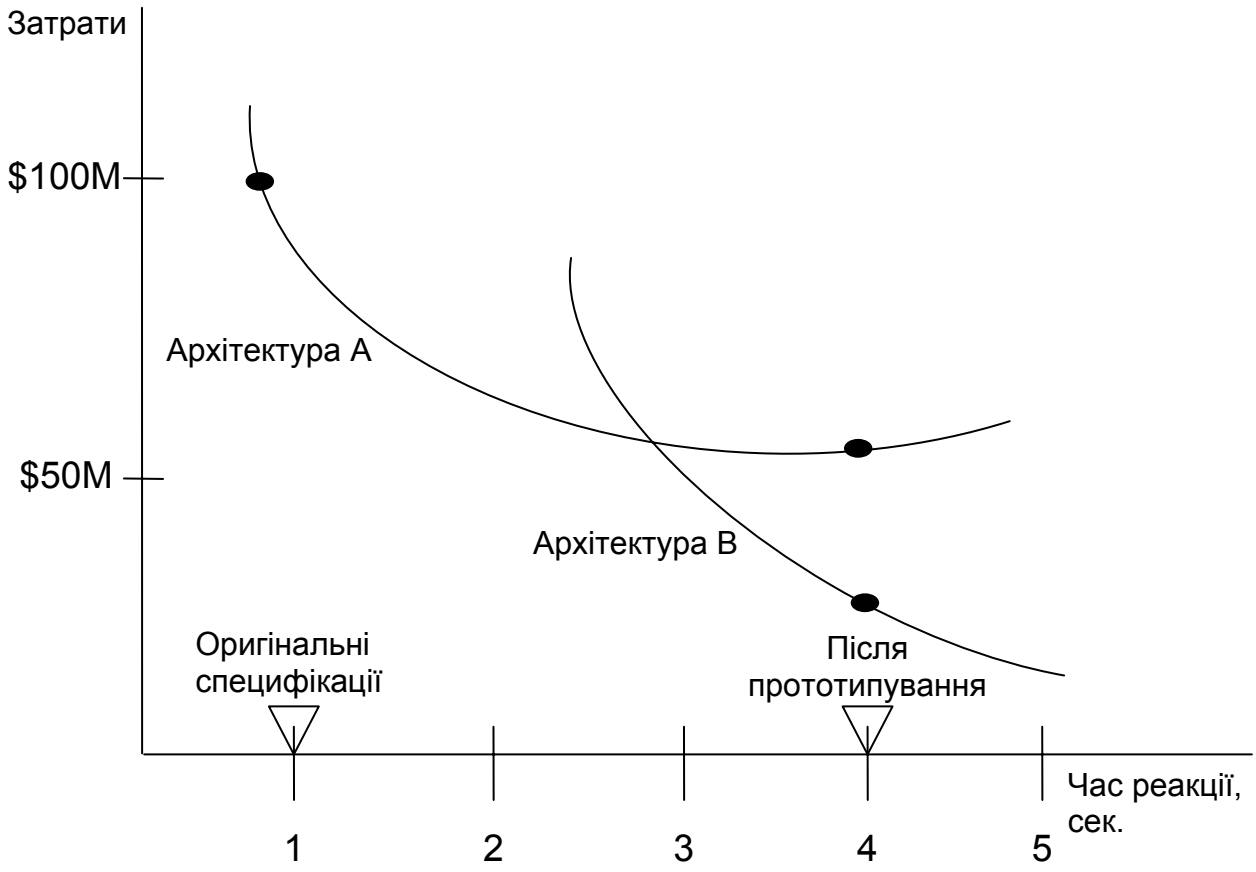


Рис. 1.9. Дві архітектури системи: затрати у порівнянні з часом реакції [91]

На рис. 1.9 зображено дві можливі архітектури побудови системи. Кожна з них має свою сферу застосування: архітектура А здатна забезпечити більш високу швидкість реакції системи, проте архітектура В за умов невисоких вимог до швидкості реакції має значно нижчу вартість реалізації.

У тому випадку, коли при проектуванні використовується каскадна модель, можливі варіанти, коли після проектування за умов зміни вимог до системи рівень економії виявиться не достатньо високим, оскільки не розглядаються інші, можливо інноваційні, підходи (у даному випадку, коли використовується архітектура А і рух йде по верхній кривій на рис. 1.9 зліва направо).

Якщо ж за основу використовується спіральна модель, то прототип системи співвідноситься з ризиками. У даному випадку на початку кожного нового витку проектування системи визначається проблема й відшукуються її найбільш оптимальні шляхи вирішення: архітектура В буде обрана як найбільш оптимальний варіант на одній з ітерацій проектування за умов ідентифікації ризику завищених вимог (часу реакції системи менше однієї секунди) та їх зміщення до 4 сек (рис. 1.9).

Таким чином, початок кожного нового витка спіралі може бути здійснений з переоцінки ризиків та визначення рішень і технологій, які доцільно застосувати у проекті. Саме з цієї точки зору спіральна модель значним чином відповідає основним положенням ефективного інноваційного менеджменту: початок кожного нового витка може бути здійснено з визначення нововведень, впровадження яких у процес розробки принесе найбільший економіко-технічний ефект.

Подальший розвиток ідей спіральної моделі був здійснений у так званому «ітеративному підході», який вперше був запропонований Ф. Крачтеном [44].

Оригінальна спіральна модель Б. Боема використовує ітератівний підхід лише по відношенню до процесу проектування, інші складові процесу розробки ПЗ є послідовними, як у каскадній моделі. Ф. Крачтен поширив ітеративний підхід на усі інші складові процесу розробки ПЗ, виділивши при цьому чотири фази життєвого циклу:

1. Початок. На даній стадії команда приділяє основну увагу розумінню бізнес-варіанта проекту, визначенню його масштабу, можливості досягнення реалізації. Проводиться аналіз проблеми, складається документ-концепція, попередньо оцінюються графік, бюджет, а також фактори ризику проекту.
2. Дослідження. Уточнюються вимоги до системи, задається вихідна архітектура, демонструється попередній прототип системи.
3. Побудова. Головна увага приділяється реалізації. На цій стадії створюється основна частина програмного коду, закінчується планування та доробка архітектури.
4. Впровадження. Проводиться бета-тестування; користувачі та команда супроводження системи отримують досвід роботи з нею. Протестована базова версія ПЗ передається користувачам та розгортається для використання [44].

Ітеративна модель передбачає послідовний випуск життєздатних версій по закінченню кожної ітерації на усіх фазах життєвого циклу процесу розробки (рис. 1.10). В своїй основі ітеративна модель у значній мірі подібна спіральній, за ті-

єю відмінністю, що окремі ітерації (витки спіралі) охоплюють не лише процес проектування, а й усі інші процеси, для яких це можливо (процеси поділяються на робочі та допоміжні, кожен процес може мати свій цикл ітерацій, що не завжди має відповідати циклу інших процесів).

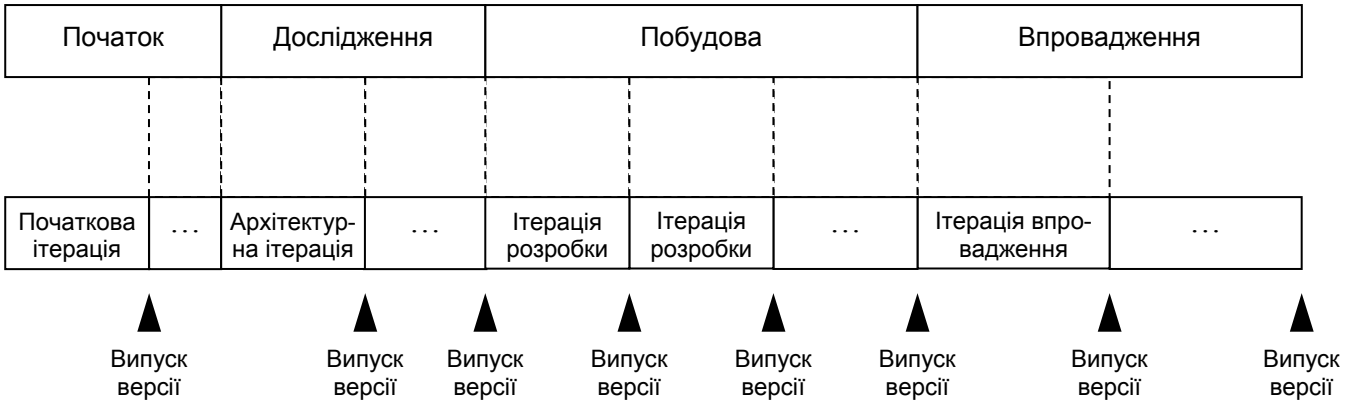


Рис. 1.10. Фази та ітерації, що призводять до появи життєздатних версій [44]

Оскільки ітеративна модель розвинула основі положення спіральної моделі і, фактично, замінила її, то звичайно у сучасних умовах під спіральною моделлю процесу розробки ПЗ розуміють не оригінальну спіральну модель, розроблену Б. Боемом, а ітеративну модель Ф. Крачтена. Сам підхід набув широкої популярності і став відноситися до управління будь-якими проектами, а не лише проектами з розробки ПЗ [107].

У 1998 р. Б. Боем у співавторстві з іншими дослідниками запропонував модифікацію спіральної моделі, яка отримала назву спіральної моделі «Win-Win» [94]. Спіральна модель «Win-Win» фактично є представленням гри двох учасників, причому умовою виграшу для них є успішне закінчення проекту.

На наш погляд, спіральна та ітеративна моделі, на відміну від попередніх, мають значні можливості для ефективного впровадження інновацій у діяльності з розробки ПЗ. Початок кожної нової ітерації доцільно розпочинати з процесу вибору та оцінки нововведень з максимальним економіко-технічним ефектом як для конкретного проекту, так і для фірми-розробника взагалі з урахуванням можливих ризиків, зокрема, затримки у реалізації проекту з причин впровадження інновацій.

Слід зазначити, що існують також інші моделі процесу розробки ПЗ, серед яких доцільно виділити наступні:

- модель прототипування – подібна спіральній моделі, основна увага приділена питанням тісної взаємодії з замовником на основі розробки робочих моделей системи – прототипів;
- модель швидкої розробки (Rapid Application Development, RAD) – сфокусована на використанні спеціалізованих інструментів, що передбачають візуальні методи розробки ПЗ;
- інкрементна модель – поступове нарощування функціональності за рахунок ділення задачі на окремі функціональні складові, які, в свою чергу, управляються каскадною моделлю;
- гібридні та адаптовані моделі – створені на основі поєднання та модифікації інших моделей [27, 44, 74].

В основному вказані моделі подібні або є модифікаціями розглянутих, тому мають їх особливості.

Популярним в сучасних умовах є підхід до розробки ПЗ, що має назву «швидка розробка» чи «швидке управління проектами з розробки ПЗ», один з найпопулярніших варіантів якого має назву «екстремальне програмування» [2, 6, 89, 118, 150]. В основному модель подібного підходу включає в себе тісну взаємодію з замовником та ітераційне виконання проекту.

У подібних методологіях управління інноваціями визначається як один з невід'ємних процесів успішної розробки. Зокрема, у [118] відзначається, що для успішної розробки мають впроваджуватися «надійні інновації» як постійний процес отримання прийнятного результату, тобто такого, що не обов'язково має бути прогнозованим, однак за умов ефективного управління здатен становити постійну цінність для організації. При визначенні критеріїв доцільності впровадження інновацій як головні називаються цінність для кінцевого споживача, витрати і час на впровадження.

Крім розглянутих моделей процесу розробки, значний вплив на діяльність з розробки ПЗ становлять створені спеціалізованими організаціями стандарти, які по-

єднують певне бачення (модель) процесу розробки і конкретні вимоги та рекомендації стосовно його організації. На рис. 1.11 представлено перелік найважливіших стандартів та моделей управління процесом розробки.

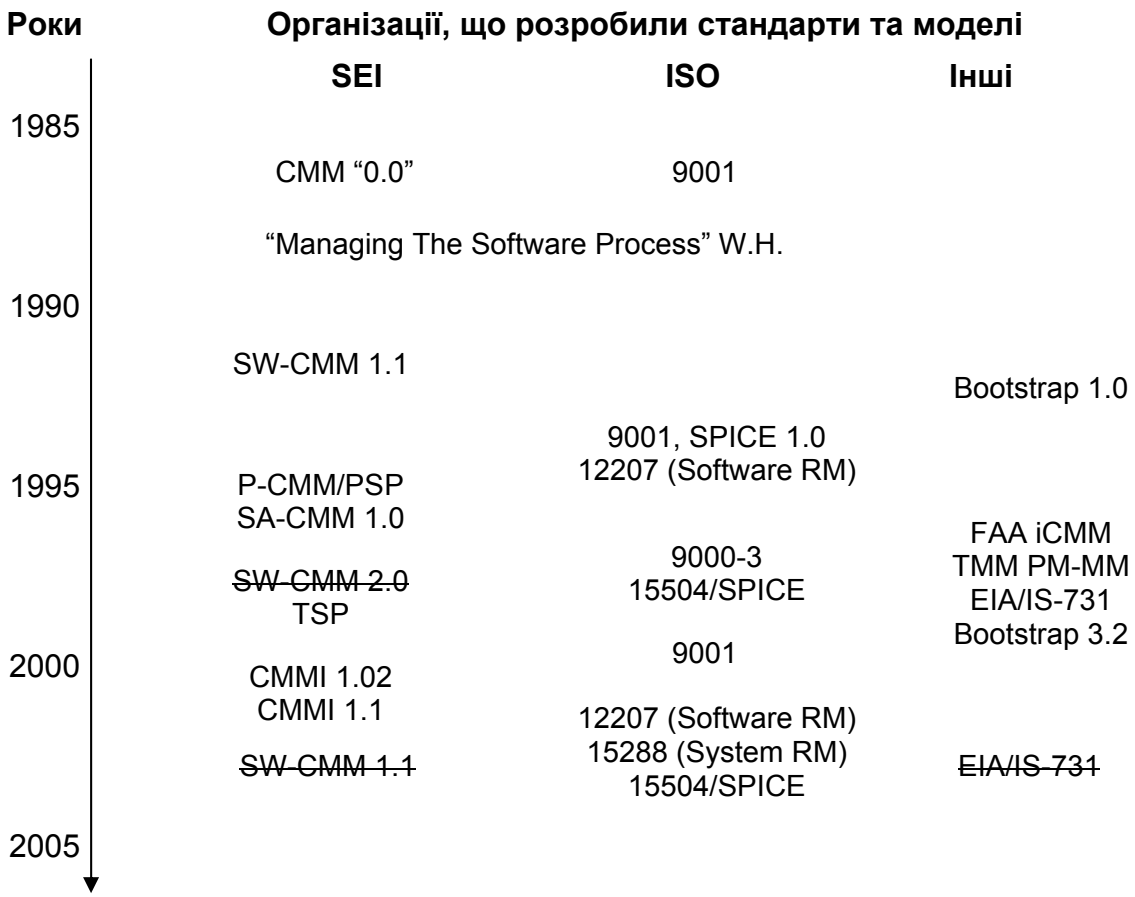


Рис. 1.11. Важливі стандарти і моделі управління розробкою програмного забезпечення, починаючи з 1985 р. [131]

Примітка: закресленими позначено стандарти, які втратили своє значення у сучасних умовах.

Позначення: A – Acquisition (Злиття); CMM – Capability Maturity Model (Модель характеристики зрілості); CMMI – CMM Integrated (Інтегрована модель CMM); EIA/IS – Electronic Industries Alliance Interim Standard; PM-MM – Project Management Maturity Model (Модель зрілості управління проектом); PSP – Personal Software Process (Персональний програмний процес); RM – Reference Model (Модель для посилання); SE – Systems Engineering (Системний інжиніринг); Software – Програмне забезпечення; SPICE – Software Process Improvement Capability dEtermination (Визначення можливостей для вдосконалення програмного процесу); SW – Software Engineering (Програмний інжиніринг); TSP – Team Software Process (Командний програмний процес).

Слід зазначити, що визначаючий вплив на процеси розробки ПЗ і життєвий цикл в цілому становлять не лише стандарти і моделі, а також і технології, які використовуються при реалізації проектів [128].

Відповідно до рис. 1.11, найбільш значний вплив становили стандарти розроблені міжнародною організацією ISO та Інституту розробки програмного забезпечення (Software Engineering Institute, SEI), що є структурною одиницею Університету Карнегі-Меллона, США. Інститут SEI працює у сфері дослідження питань менеджменту процесу розробки ПЗ, вивчає техніко-економічні аспекти процесу розробки ПЗ та його готовність до подальшого розвитку. Інститут розробив Модель характеристики зрілості процесу розробки ПЗ (Software Capability Maturity Model, SW-CMM), котра набула широкого визнання [98].

В даний час модель SW-CMM поступово замінюється більш сучасною Інтегрованою моделлю зрілості процесу розробки (Capability Maturity Model, CMMI), яка поєднує основні положення SW-CMM, стандарту EIA/IS 731 та окремих стандартів ISO, зокрема ISO 15504/SPICE [96, 112, 149]. Проте основу CMMI становить модель SW-CMM, яку, на наш погляд, доцільно розглянути детальніше.

Фактично модель SW-CMM має вигляд стандарту, що характеризує п’ять рівнів зрілості процесу розробки ПЗ, які визначають рівень організації діяльності компанії-розробника ПЗ (табл. 1.2).

Нинішній фокус та ключові процеси компанії залежать від того, якого рівня зрілості вона досягла. На найбільш високому рівні зрілості фокус компанії зосереджений на завданні постійного поліпшення процесу, ключовими процесами є менеджмент впровадження інновацій та постійне поліпшення процесу.

Таким чином, модель SW-CMM прямо вказує на місце інноваційного менеджменту у діяльності компанії. Однак, відповідно до моделі, управління інноваціями як ключовий процес можливе лише за умови досягнення компанією найвищого рівня зрілості.

Таблиця 1.2

Описова характеристика рівнів зрілості процесу розробки ПЗ відповідно до моделі SW-CMM [98]

Номер та назва рівня	**Характеристика рівня**
1. Початковий	Бізнес компанії має неструктурований та інноваційний характер. Успіх, якщо його вдається досягнути, залежить лише від ділових якостей лідера та членів команди. Головні інформаційні зв'язки сконцентровані на керівництві та носять спонтанний характер. Планування не є ефективним. Процес розробки непрозорий та орієнтований на невдачу. Звільнення окремого співробітника ставить рід загрозу виконання всього проекту.
2. Повторюваний	Характерними рисами виступають жорстке управління, оперативне планування та контроль, що стають доступними через досвід, який був набутий при реалізації попередніх проектів. Проте інформаційні зв'язки не є формалізованими, а прозорості процесу розробки також не вдається досягнути. Компанія здатна передбачити та протистояти певним невдачам. Організаційна структура може бути будь-якою, проте найбільшого поширення набули функціональна та дивізіонна. Розпочинає формуватися організаційна культура.
3. Визначений	Процес розробки ПЗ на всіх стадіях є документованим. Формалізованими є розробка ПЗ та управління даним процесом. Процеси виконання окремих робіт стають прозорими, звільнення окремих співробітників не загрожує успіху проекту. Розпочинається процес управління внутрішньофірмовими знаннями, що дає підстави для набуття компанією певних конкурентних переваг.
4. Керований	Формуються принципи внутрішнього контролю та стандартизації. Налагоджується розвинена взаємодія з зовнішнім середовищем, особливо з постачальниками, від яких вимагається сумлінне та бездоганне виконання обов'язків. Обов'язковою умовою виступає наявність постійних надійних клієнтів. Планування носить довгостроковий характер. Характерна особливість – впровадження кількісних методів в управлінні процесом, що дає підстави для введення наукових методів прийняття рішень. Стратегічні та оперативні плани тісно взаємопов'язані. Компанія здатна заздалегідь передбачити можливі проблеми та уникнути їх.
5. Оптимізуючий	На цьому рівні компанія сфокусована на постійному поліпшенні процесу. За допомогою кількісних методів проводиться управління всіма процесами, що пов'язані з основною діяльністю. Процес розробки ПЗ здатен уникати можливих дефектів у кінцевому продукті. Характерна особливість – постійне вдосконалення, як за рахунок внутрішніх ресурсів, так і зовнішніх, наприклад, нових технологій.

Залежність цілей компанії та її пріоритетних завдань від рівня зрілості процесу розробки ПЗ подано в табл. 1.3.

Таблиця 1.3

Залежність цілей компанії та її пріоритетних завдань
від рівня зрілості процесу розробки ПЗ [98]

Номер та назва рівня	**Фокус компанії**	**Площини ключових процесів**
1. Початковий	Компетентні особи та співробітники, що докладають надзвичайні зусилля	
2. Повторюваний	Базовий менеджмент проекту	Перехід до підтримки Визначення цінності Спостереження та нагляд за виконанням контракту Проектний менеджмент Розробка та управління вимогами Ведення проекту Планування розробки ПЗ
3. Визначений	Стандартизація процесу	Тренувальна програма Управління ризиками Управління виконанням контракту Управління виконанням проекту Визначення та підтримка процесу
4. Кількісний	Кількісний менеджмент	Кількісні методи управління процесом Кількісний менеджмент досягнень
5. Оптимізуючий	Постійне поліпшення процесу	Менеджмент впровадження інновацій Постійне поліпшення процесу

Слід зазначити, що в цілому в світі кількість компаній, які досягли п’ятого, найвищого рівня відповідно до SW-CMM є порівняно невелика, зокрема у табл. 1.4 представлено інформацію щодо кількості компаній, які пройшли сертифікацію за моделями SW-CMM/CMMI, причому доля тих, які досягли п’ятого рівня SW-CMM становить близько 10% від загальної кількості сертифікованих компаній.

Однак порівняно незначна кількість компаній, що відповідають п’ятому рівню зрілості згідно з SW-CMM у дійсності не означає, що управління інноваціями не розглядається як ключовий процес іншими компаніями. Наприклад, лідер ринку ПЗ компанія Microsoft не проходила сертифікацію на відповідність вимог SW-CMM, проте управління інноваціями у цій компанії розглядається як один з ключових процесів [17, 57].

Таблиця 1.4

Кількість організацій, що пройшли сертифікацію за моделями SW-CMM/CMMI станом на червень 2004 р. в розрізі країн світу [122]

№ п/п	Країна	Кількість організацій, що пройшли сертифікацію SW-CMM/CMMI
1	США	1896
2	Індія	359
3	Китай	182
4	Франція	135
5	Великобританія	135
6	Японія	131
7	Канада	73
8	Південна Корея	64
9	Німеччина	56
10	Австралія	35
11	Ізраїль	30
12	Сінгапур	22
13	Ірландія	11
14	Усі інші країни разом	231
	Всього:	3360

Таким чином, модель SW-CMM не достатньо точно здатна описати реальні процеси підприємства-розробника, вона в основному визначає вимоги, яким мають відповідати підприємства, щоб досягти певного рівня зрілості.

На відміну від SW-CMM, модель, яка призвана її замінити, CMMI здатна оцінювати окремі процеси і передбачати різний їх рівень зрілості в залежності від потреб організації. Поняття «ключових площин процесів» було замінено поняттям «площин процесів», кількість яких була збільшена, головна їх відмінність полягає у тому, що організація може приділяти різну увагу площинам процесів, залежно від власних потреб [99, 96]. Менеджменту інноваціями у CMMI приділено значне місце, зокрема, CMMI передбачає спеціальний метод оцінки SCAMPI (Standard CMMI Appraisal Method for Process Improvement), призначений для визначення ефективності процесів покращення розробки ПЗ, до яких відносяться інноваційні процеси [102].

Підхід SPI (Software Process Improvement, Покращення процесу розробки ПЗ), сфокусований на питаннях покращення процесу розробки ПЗ, у тому числі й на ос-

нові інноваційних процесів, дуже часто базується на SW-CMM/CMMI. За оцінками фахівців вдале використання SPI може привести до отримання вигоди у співвідношенні з витратами від 2:1 до 10:1 [126].

Інститут SEI розробив модель для практичної реалізації SPI, котра отримала назву IDEAL (Initiating, Diagnosing, Establishing, Acting & Learning). Модель IDEAL являє собою еталонну модель впровадження інновацій, яка дозволяє управляти процесами ініціації, планування та впровадження інновацій (рис. 1.12) [96, 113].

Рис. 1.12. Графічне представлення моделі IDEAL [113]

Відповідно до рис. 1.12 модель IDEAL є циклічною і складається із п’яти фаз, які поділяються на окремі етапи.

Фаза «Ініціація» є початковою і містить наступні етапи:

– стимул до змін – внутрішня або зовнішня подія, що ініціює процес;

– задати контекст – визначити, яким чином інновація відповідає стратегії бізнесу організації;
– організувати підтримку – отримати підтримку зі сторони керівництва і відповідальних осіб як у вигляді ресурсів, так і управлінського впливу на виконавців процесу;
– зафрахтувати інфраструктуру – визначити і зафіксувати у вигляді письмової угоди чітко визначені очікування від впровадження інновації, вигоди та відповідальність [113].

Фаза «Діагностика» містить наступні етапи:

– охарактеризувати нинішній і цільовий стан – необхідно визначити нинішній стан проблеми і окреслити, що саме буде досягнуто при її вирішенні;
– розробити рекомендації – визначити ключові рекомендації щодо впровадження інновації. Рекомендації мають бути розроблені фахівцями предметної області і можуть бути використані керівництвом при прийнятті рішень.

Фаза «Створення» містить наступні етапи:

– встановити пріоритети – необхідно визначити пріоритети дій з урахуванням багатьох факторів, зокрема обмеженості ресурсів;
– розробити підхід – скласти стратегію виконання роботи з урахуванням доступних ресурсів. Мають бути розглянути технічні фактори (специфіка впровадження нової технології та необхідні для її використання знання і навики), а також нетехнічні (організаційна культура, ймовірні джерела опору, рівні підтримки та ринкові сили);
– спланувати дії – необхідно розробити детальний план, що має включати графік, завдання, віхи, точки прийняття рішень, ресурси, відповідальність, систему вимірювання досягнень, механізми відслідковування, ризики та стратегії міграції та будь-які інші елементи, які можуть вимагатися організацією.

Фаза «Дії» містить наступні етапи:

– створити рішення – на основі поєднання усіх ключових елементів необхідно розробити рішення, що має відповідати потребам організації, визначе-

ним на попередніх етапах. Рішення має бути комплексним і якомога більш детальним, враховувати усі необхідні дії і ресурси, у тому числі й передбачати зовнішню допомогу;

– перевірити рішення – рішення має бути перевірене, наприклад, з використанням пілотного проекту;
– уточнити рішення – на основі досвіду, отриманого на попередньому етапі необхідно внести корективи у рішення, якщо це необхідно;
– реалізувати рішення – на основі розробленого плану необхідно реалізувати рішення на практиці.

Фаза «Навчання» містить наступні етапи:

– аналіз і перевірка – зібрати і проаналізувати інформацію стосовно реалізованого рішення. Інформація має бути зібрана відповідально, отримані результати мають бути зіставлені з цілями, які ставилися на фазі ініціації і задокументовані;
– скласти пропозиції щодо подальших дій – зробити висновки і оформити їх у вигляді звіту, який може бути використаний керівництвом для прийняття рішень стосовно подальших дій.

Аналізуючи структуру моделі IDEAL, слід зазначити, що модель прямо не передбачає питання оцінки економічної доцільності інновацій і вибір альтернатив, однак, на нашу думку, її важливим досягненням є орієнтація на безперервність процесу і необхідність цілеспрямовано збирати інформацію про досвід реалізації інноваційного проекту, яка в подальшому може бути використана у наступних ітераціях чи окремих проектах.

Корпорація Microsoft є розробником двох взаємодоповнюючих стандартів – MSF (Microsoft Solutions Framework) та MOF (Microsoft Operations Framework). Стандарт MSF відповідає за розробку ПЗ, а MOF – за розгортання і використання ПЗ [141, 142].

MSF являє собою «виважений і дисциплінований підхід до технологічних проектів, оснований на наборі принципів, моделей, дисциплін, концепцій, рекомендацій і доведених практик від Microsoft» [141, С. 4]. MOF розроблено на основі ста-

ндарту ITIL (IT Infrastructure Library), який створений за участі уряду Великобританії і є найбільш поширеним у даний час підходом до управління службами інформаційних технологій у світі [17].

В основі MSF лежать наступні ідеї:

- управління ризиками і їхнє планування;
- випуск проміжних версій;
- планування активності;
- чітко позначені контрольні віхи;
- проектні групи невеликої чисельності.

Моделі, що входять до складу MSF (модель проектної групи, модель процесу, модель програмного забезпечення) допомагають знайти відповіді на критично важливі питання на кожному з етапів проектування.

Модель проектної групи розглядає питання, яким чином повинна бути побудована і структурована проектна група, для того щоб можна було оптимізувати процес проектування за строками, якістю і витратами.

Характеристики ефективної проектної групи відповідно до MSF:

- кожний спілкується з кожним, і кожен робить реальну роботу;
- загальні для всіх членів групи цілі і плани;
- кожний розуміє як проблеми кінцевого користувача, так і проблеми розробника;
- кожний несе відповідальність за свою роботу, у тому числі і перед групою.

Суттєвою особливістю підходу MSF є заперечення ієрархічного способу структурування проектної групи. Модель не встановлює підпорядкування учасників, вона визначає основні ролі, закріплені за членами проектної групи (рис. 1.13).

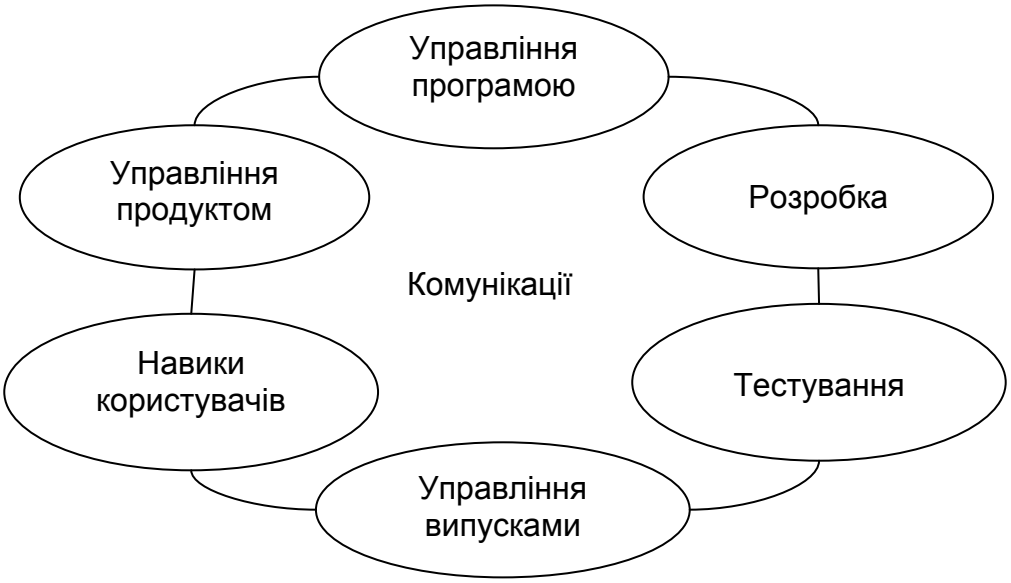


Рис. 1.13. Основні ролі проектної групи MSF [141]

Модель проектної групи MSF ніяк не співвідноситься з організаційною структурою. За кожним членом проектної групи закріплюється конкретна роль, для якої будується специфічний план робіт, що потім входить у загальний план проекту як складова частина.

Параметри різних ролей у складі проектної групи формалізовані в табл 1.5.

Модель проектної групи MSF має на увазі постійну роботу всіх ролей над проектом. Ролі мають різне навантаження протягом циклу проекту, відповідають за досягнення відповідних віх і т.п., але працюють над проектом від моменту його початку і до завершення.

Дотримання формалізованих правил побудови взаємовідносин між членами у команді розробників, жорсткий контроль над можливістю об’єднання ролей у проектній групі дозволяють у значній мірі позитивно вплинути на очікувані результати.

Модель процесу MSF є окремим випадком ітераційної моделі. Вона використовує поняття „віх” (моментів синхронізації проектної групи і замовника), скорочуючи цикл розробки за допомогою механізму випуску версій і разом з моделлю проектної групи визначаючи ясну і чітку відповідальність ролей.

В основі цієї моделі лежить кілька основних ідей:

- конструювання рішень з доступних для огляду і керованих частин;
- фази проекту завершуються віхами;

- випускаються версії продукту;
- планування виконується з урахуванням ризиків.

Модель процесу проектування MSF (рис. 1.14) складається з п’яти фаз і п’яти віх, якими завершуються ці фази (слід зазначити, що модель постійно еволюціонує, у попередній редакції вона мала чотири фази і чотири віхи) .

Таблиця 1.5

Ролі у складі проектної групи MSF [143]

Роль	Відповідальність	Навички	Пріоритети
Управління продуктом	Визначення проблеми. Просування продукту	Висока комунікабельність	Задоволеність замовника
Управління програмою	Керування специфікаціями. Координація робіт. Відстеження стану проекту	Досвід керування проектами. Висока комунікабельність. Уміння писати тексти	Розробка якісного продукту в термін і в рамках виділеного бюджету Виявлення і вирішення проблем
Розробка	Проектування функціональності. Створення продукту. Тестування продукту	Рішення проблем. Досвід розробки	Надійний і повний продукт
Тестування	Визначення стратегії тестування. Проведення тестування. Відстеження результатів тестування	Уміння розрізняти причини і наслідки, навички діагностування ситуації. Уміння моделювати критичні ситуації. Розуміння принципів функціонування продукту	Погоджений і надійний продукт
Навики користувачів	Розробка документації. Ведення глосарію . Тестування. Навчання користувачів	Технічний письменник	Продукт, який можна використовувати і супроводжувати
Управління випусками	Прогнозування ситуації. Підготовка впровадження. Супровід продукту на етапі впровадження. Забезпечення адекватної інфраструктури, коли в ній є необхідність	Розуміння оперативної ситуації. Комунікабельність. Розуміння і відстеження динаміки взаємин між організаціями.	Забезпечення впровадження і розвитку продукту

Орієнтація на віхи визначає, що невелика, заздалегідь визначена частина загального рішення буде отримана й перевірена вчасно, ризики планування і якості бу-

дуть відомі заздалегідь, що надасть можливості їх усунення. Цим вона дуже схожа до основних положень спіральної та ітераційної моделей.

Управління проектом згідно MSF – це управління зовнішніми і внутрішніми віхами, а також процесом, що призводить до їхнього досягнення [141].

Необхідно відмітити, що на діаграмі процесу (рис. 1.14) зображені саме фази проекту і немає прив'язки до часу. Крім того, дана діаграма не затверджує, що тривалість усіх чотирьох фаз збігається.

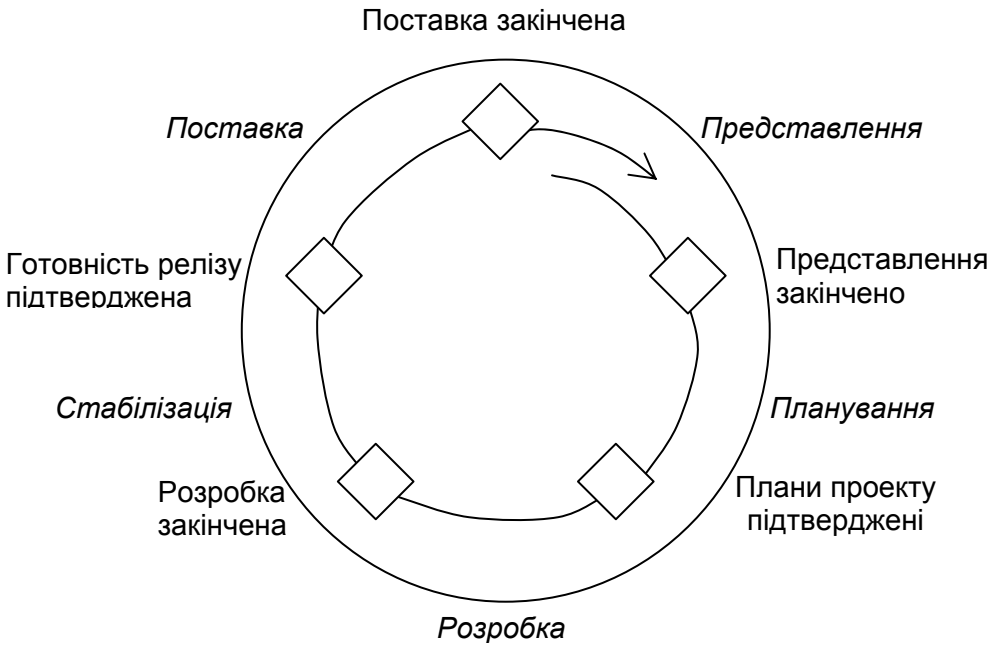


Рис. 1.14. Діаграма моделі процесу MSF [141]

Віхи більше орієнтовані на замовника, ніж на розробника. Кожна віха – це точка синхронізації, коли команда заново переглядає очікування замовника і ризики. Це точки обговорення і ревізії, а не точки фіксації прийнятих рішень, вони дозволяють проектній групі і замовникам порівняти границі проекту і очікування користувачів.

Стандарт MSF має цілий комплекс заходів, що спрямовані на ефективне управління інноваціями.

По-перше, це планування з урахуванням ризиків:

- MSF заохочує розроблювачів створювати макети і прототипи якомога раніше. Це знижує ризик одержання неповноцінного продукту, нереального графіку або краху всього проекту;
- визначається, які можливості і коли повинні бути реалізовані; пріоритети задачам розставляються на підставі:
  - управлінських і бізнесів-ризиків – гарантія того, що в черговій версії будуть реалізовані особливо важливі для бізнесу функції;
  - технічних ризиків – гарантія того, що потенційно ризиковані функції будуть реалізовані в першу чергу, і це надасть достатній ресурс часу для ретельного тестування і, якщо буде потрібно, пошуку компромісів;
- розробляється план по запобіганню найбільш ймовірних і небезпечних ситуацій і, про усякий випадок, план „ліквідації катастрофи”;
- обов'язково переглядається проектний план для включення в нього цього „пом’якшуючого” ризики плану. Як правило, це вимагає збільшення необхідних ресурсів і часу, а також виконання додаткових задач.

Якщо компоненти, пов'язані з високим ступенем ризику, потребують більше часу, ніж було розраховано, то планування з урахуванням ризиків надасть додатковий час для реагування.

Оцінка ризиків і наявність плану «ліквідації катастрофи» допомагає визначати доцільність впровадження інновацій, а також надає можливість відмінити їх впровадження, якщо очікувані цілі не були досягнуті, чи в процесі впровадження виникли проблеми.

По-друге, це орієнтація на випуск версій продукту. Випуск декількох версій продукту означає, що не уся функціональність реалізується відразу, процес розробки є ітерованим і проектна група приймає рішення про включення тієї або іншої функціональності в міру необхідності, орієнтуючись на версії і дати випуску.

При послідовному випуску версій проектна група може почати з реалізації функцій, найбільш критичних для замовника, і планомірно одержувати від користувачів зворотний зв'язок, що буде потрібно при розробці наступних версій. Модель

процесу проектування MSF стимулює проектні групи до розгляду розроблювального продукту нарівні з існуючим, що підтримується як версії.

Перший випуск продукту містить базовий набір функцій, а наступні – включають все більше і більше додаткових функцій, аж до фінального продукту. Наступні версії дозволяють проектній групі в черговий раз перевірити або змінити образ продукту (наприклад, якщо змінилися вимоги бізнесу).

Крім того, випуск версій дозволяє здійснювати впровадження процесних інновацій в перервах між випусками окремих версій, підвищуючи технічний потенціал компанії-розробника і надаючи їй можливості починати розробку нової версії з використанням нових технологій.

По-третє, MSF окремо виділяє такі ролі проектної групи як управління продуктом та управління програмою. Подібний поділ дозволяє менеджеру продукту вирішувати питання продуктових інновацій, а менеджеру програми – процесних, усуваючи таким чином можливі

В цілому MSF є орієнтованою на продукт методологією, на відміну від CMM/CMMI, які орієнтовані на процес. Орієнтація на продукт означає, що методологія значно більше уваги приділяє кінцевому результату, ніж орієнтовані на процес підходи.

Таким чином, в результаті проведеного аналізу інноваційної складової моделей і стандартів життєвого циклу програмного забезпечення можна зробити наступні висновки.

Існує достатньо велика кількість моделей, призначених описати процес розробки ПЗ. Розбіжності в поглядах на процес розробки з точки зору кожної моделі в основному викликані достатньо високою складністю процесу розробки ПЗ, а також зміною домінуючих в процесі розробки підходів.

Застарілі моделі, такі як каскадна та V-подібна не передбачали інноваційні процеси, що являється одним із факторів, які сприяли їх неефективності.

Серед сучасних моделей спіральна та ітераційна найбільш пристосовані до інноваційних процесів, оскільки здатні переоцінювати ризики і технологічний потенціал розробника, що змінюється в процесі впровадження інновацій.

Поширені стандарти управління розробкою ПЗ включають питання управління інноваціями як один із ключових процесів. Модель SW-CMM передбачала комплексне управління інноваціями лише на останньому рівні зрілості, що не відповідало дійсності для багатьох успішних компаній-розробників. Її наступниця, модель CMMI передбачає більш гнучкий підхід і передбачає можливість впровадження інноваційного менеджменту без прямого зв'язку з рівнем зрілості.

Модель IDEAL, що є однією з моделей CMMI, призначена для постійного управління процесом впровадження інновацій. Модель не фокусується на питаннях дослідження економічної доцільності, що може бути виправдано з урахуванням великої складності цього процесу, однак орієнтується на безперервність і постійне поліпшення процесу.

Методологія MSF є досить виваженим підходом, основаним на ітеративному підході, який містить цілий комплекс заходів, спрямованих на ефективне управління інноваціями.

В цілому слід зробити висновок, що підходи, які базуються на постійному аналізі процесу розробки ПЗ, виявленню його слабких та сильних сторін й узагальненню отриманих результатів у вигляді практичних рекомендацій з жорстким режимом їх виконання, є одним з найбільш поширених серед сучасних методів ефективного управління інноваціями.

1.3. Аналіз підходів до визначення економічних параметрів процесу розробки програмного забезпечення

Однією з основних причин комерційного неуспіху інновацій називається відсутність ефективної системи управління інноваційними процесами [54].

Побудова ефективної системи управління можлива лише за умови використання кількісних методів прийняття рішень, основаних на здійсненні економічної оцінки параметрів процесу розробки ПЗ.

Питання економічної оцінки параметрів процесу розробки ПЗ має високу актуальність та активно обговорюється у науковій літературі [79, 88, 120, 117, 135, 95,

123, 109, 132, 156, 162, 167, 159, 166]. В цілому, загальна риса досліджень, присвячених даним питанням, полягає в тому, що вони визнають надзвичайно високу складність отримання достовірних оцінок у порівнянні з аналогічними показниками сфери матеріального виробництва.

Згідно з даними SEI, близько 80% усіх систем кількісної оцінки процесу розробки ПЗ виявляються недієздатними протягом перших двох років використання [161].

У найпростішому випадку оцінити вартість розробки ПЗ можна виходячи з оцінки трудовитрат на розробку та вартості одиниці часу на розробку:

$$B = T \times Ц \quad (1.1),$$

де B – вартість розробки ПЗ;

T – трудовитрати на розробку ПЗ (людино-міс);

$Ц$ – ціна одиниці трудовитрат.

Ціна одиниці трудовитрат звичайно формується з заробітної плати та нарахувань на заробітну плату, оскільки в сучасних умовах інші витрати є порівняно незначними, їх не приймають до уваги.

Простий розрахунок трудовитрат оснований на хронологічних даних [74]:

$$T = P \times П \quad (1.2),$$

де T – трудовитрати на розробку ПЗ;

P – розмір ПЗ;

$П$ – хронологічна продуктивність.

Для оцінки величини розміру ПЗ (Р) використовуються різні одиниці величин, які залежать від методики оцінки проекту та не можуть довільним чином бути перетворені між собою, оскільки між ними складно встановити однозначну тотожність.

Крім того, одиниці оцінки розміру ПЗ також передбачають використання відповідних методик для проведення оцінки, результатом виконання яких є певне значення, що вимірюється у відповідних одиницях.

Найбільш поширеними є наступні одиниці оцінки розміру ПЗ:

– кількість рядків коду (*Lines Of Code, LOC*);

- функціональні точки;
- точки властивостей;
- кількість «пухирців» на діаграмі потоку даних (Data Flow Diagram*, DFD*);
- кількість сутностей на діаграмі сутностей (Entity Relationship Diagram*, ERD*);
- кількість «квадратиків», що відповідають процесу/контролю на структурному графіку;
- кількість різних елементів у складі управлінської специфікації;
- обсяг документації;
- кількість об'єктів, атрибутів та служб на об'єктній діаграмі [74].

Кількість рядків коду (Lines Of Code*, LOC*; Source Lines of Code*, SLOC*) є найпростішою і найпоширенішою серед зазначених одиниць виміру. У загальному випадку *LOC* означає кількість рядків коду на відповідній мові програмування, які мають бути написані для того, щоб проект був виконаний [83]. На практиці звичайно використовується похідна від *LOC – KLOC (KSLOC)*, що означає кількість тисяч рядків коду.

Оскільки різні мови програмування мають різні можливості і різну продуктивність по вирішенню задач програмування, то одиниці *LOC* мають бути приведені до певної співрозмірної величини. Приведення здійснюється з використанням таблиць перетворень, які дозволяють перевести кількість рядків коду більшості найбільш поширених мов програмування у еквівалентну кількість рядків коду мови асемблера, кожна строчка коду якої відповідає одній машинній команді. Такі таблиці періодично поновлюються, щоб враховувати еволюцію мов програмування [125].

Результати дослідження окремих вчених, зокрема К. Джонса встановили, що метрика *LOC* не може бути використана для ефективної оцінки проектів, виконаних з використанням сучасних об'єктно-орієнтованих мов програмування, кількість яких останнім часом суттєво зростає [124].

Вирішити проблему обмеженості *LOC* по відношенню до сучасних мов програмування покликана інша одиниця оцінки розміру ПЗ – функціональні точки.

Головна ідея використання функціональних точок полягає в тому, що розмір ПЗ краще оцінювати в термінах кількості та складності функцій, а не кількості рядків коду.

Перша наукова робота, присвячена функціональним точкам, була опублікована в кінці 1970-х рр., її автором був А. Альбрехт, що представляв компанію IBM [82]. Подальші дослідження були здійснені К. Джонсом, який суттєво розвинув дану ідею [80].

Створена в 1986 р. некомерційна група International Function Point User Group (IFPUG) займається розробкою і підтримкою стандартів, зокрема IFPUG Standard Function Point Counting Practices Manual та IFPUG Standard Guidelines to Software Measurement [74].

При використанні методу функціональних точок здійснюється вимірювання категорій бізнес-функцій. На відміну від використання *LOC*, даний метод зосереджений на визначенні не абстрактного розміру ПЗ, а лише таких його показників, які визначають функціональність. Дана особливість дозволяє ефективно використовувати метод функціональних точок на етапі планування розробки.

Характерно, що метод функціональних точок має можливості для переведення кількості функціональних точок в одиниці *LOC*. Подібне переведення дозволяє порівнювати розмір проектів, оцінених за різними методиками, однак слід зазначити, що результатом переведення є величина, яка може зіставлятися для різних методик лише з урахуванням певної погрішності.

Зокрема, використання різних методик для оцінки розміру одного й того ж самого проекту, а потім приведення результату до одиниць *LOC* може дати значення, що суттєво відрізняються для різних методик.

Певним недоліком методу функціональних точок, що обмежує його використання, є порівняно висока складність даного методу. На відміну від оцінки кількості рядків коду, даний метод вимагає кваліфікованого підходу і порівняно значних зусиль при проведенні оцінки.

Також певним недоліком даного методу є його орієнтація на оцінювання бізнес-функцій. Існує певна категорія програмних продуктів, які мають порівняно про-

стий набір входів/виходів, однак характеризуються значною внутрішньою складністю.

Подолати обмеженість методу функціональних точок в оцінюванні розміру складного ПЗ покликаний метод точок властивостей. Даний метод в цілому дуже подібний до методу функціональних точок, однак він більш адаптований до високої складності алгоритмів, оскільки крім врахування параметрів функціональних властивостей ПЗ, він здатен оцінити складність його внутрішньої організації [147].

Результатом адаптації методів функціональних точок і точок властивостей до особливостей об'єктно-орієнтованої технології розробки ПЗ є метод об'єктних точок, який дуже схожий до базових методів, однак при оцінці оперує об'єктними параметрами проекту, такими як класи і об'єкти [74].

Якщо для розробника ПЗ доступні хронологічні дані по виконаним проектам, то для здійснення оцінки розміру наступного проекту може бути використаний метод побудови бліц-моделі, розроблений Т. ДеМарко [153].

В основі процесу побудови бліц-моделі знаходиться припущення, що оцінити розмір проекту можна на основі хронологічних даних по проектам, які вже були закінчені. Бліц-модель може бути побудована на будь-якому етапі життєвого циклу проекту і оперувати показниками, що відповідають етапу циклу (наприклад, кількість «пухирців» на діаграмі потоку даних, кількість сутностей на діаграмі сутностей, кількість «квадратиків», що відповідають процесу/контролю на структурному графіку та ін.).

Спрощено побудова бліц-моделі відбувається як процес розрахунку показників розміру проекту, виходячи з відповідних показників раніше закінчених проектів, які коригуються відповідно до вже відомих характеристик проекту, що аналізується. Процес побудови можна здійснювати постійно протягом роботи над проектом і його точність буде зростати одночасно із прогресом у виконанні проекту.

Результативність методики побудови бліц-моделей збільшується при її системному використанні, оскільки вона передбачає можливість коригування параметрів на основі післяпроектного аналізу.

Суттєвим недоліком даної методики є її залежність від хронологічних даних. Вважається, що зростання кількості хронологічних даних за рахунок виконаних проектів підвищує точність методики. Однак, на нашу думку, використання хронологічних даних має свої обмеження, зокрема розвиток технологій і впровадження інновацій компанією-розробником ПЗ змінює процес розробки і самі поняття розміру і складності ПЗ для нього. Оцінка сучасного проекту на основі параметрів проектів, які були виконані з використанням застарілих технологій, втрачає смисл.

Іншим поширеним методом оцінювання розміру ПЗ є методика Wideband Delphi, розроблена як адаптація методики експертної оцінки Delphi до особливостей процесу розробки ПЗ [168].

В цілому Wideband Delphi дозволяє скоординувати дії експертів для проведення оцінки параметрів проекту точно також, як і при звичайній методиці Delphi. Єдина її суттєва відмінність полягає у тому, що завершальним етапом є групові дискусії експертів, що можуть покращити достовірність оцінки шляхом досягнення консенсусу між експертами.

Перевагою методики є її порівняна простота і можливість проведення оцінки без наявності хронологічних даних по виконаним проектам.

Суттєвим недоліком методики є можливість надання невірної оцінки, коли експерти поступаються своєю думкою і готові прийняти іншу думку без належних на те підстав. Крім того, існує ризик взагалі не досягти єдиної думки і, відповідно, не виробити єдиної оцінки. Важливим питанням також є відповідальний підбір експертів, оскільки, за умов неправильного підбору, група експертів може однобічно оцінювати проект і не враховувати усієї повноти діючих факторів.

Загальна характеристика більшості моделей оцінки вартості процесу розробки ПЗ – створення та адаптація параметричних математичних моделей на основі набору зібраних даних [112].

Відповідно до формули (1.2) трудовитрати лінійно залежать від розміру ПЗ, оскільки можна припустити, що продуктивність роботи над проектом є сталою величиною, однак дане припущення було спростовано Б. Боемом, який розробив і опублікував у 1981 р. одну з найбільш популярних моделей оцінки витрат на розро-

бку ПЗ, що отримала назву COCOMO (Constructive COst Model, Конструктивна модель витрат) [93].

Модель COCOMO була розроблена на основі аналізу статистичних параметрів 63 проектів з розробки ПЗ різних типів. Параметрами, що підлягали оцінці були розмір ПЗ (*LOC*), понесені в процесі реалізації трудовитрати, а також фактична тривалість процесу розробки.

Фактично під загальною назвою COCOMO було розроблено три рівня моделі, які передбачають різний ступінь деталізації: базовий, проміжний та деталізований рівні.

Згідно з моделлю COCOMO виділяються також три режими її використання:

- органічний режим – невеликий проект розробляється невеликою командою, для якої не характерні нововведення, середовище розробки залишається стабільним;
- зблокований режим – порівняно невелика команда працює над проектом середнього розміру, в процесі розробки потрібні певні інновації, середовище розробки характеризується незначною нестабільністю;
- впроваджений режим – велика команда розробників працює над великим проектом, потрібен великий обсяг інновацій, середовище розробки складається із значної кількості елементів, які не характеризуються стабільністю [93].

Для оцінки трудовитрат на базовому рівні моделі COCOMO використовується наступна формула [93]:

$$T = a \times P^{\,b} \qquad (1.3),$$

де T – трудовитрати на розробку ПЗ;

a, b – константи, які залежать від режиму моделі, що визначається типом проекту;

P – розмір ПЗ, KLOC.

Константи a і b встановлюються відповідно до режиму використання моделі COCOMO, який визначається типом проекту в залежності від його розміру і середовища розробки.

Значення коефіцієнтів *a* і *b*, що відповідають залежності режиму моделі COCOMO представлені у табл. 1.6.

Таблиця 1.6

Значення коефіцієнтів *a* і *b* в залежності від режиму моделі COCOMO [93]

Назва режиму моделі COCOMO	**Значення коефіцієнту *a***	**Значення коефіцієнту *b***
Органічний	2,4	1,05
Зблокований	3,0	1,12
Впроваджений	3,6	1,20

Відповідно до даних табл. 1.6 значення коефіцієнтів *a* і *b* зростають при переході від органічного до впровадженого режимів моделі COCOMO, що, відповідно, означає експоненціальне зростання розміру трудовитрат при рівних значеннях розміру ПЗ.

Таким чином, модель COCOMO враховує нелінійність зростання трудовитрат на розробку відповідно до зростання розміру проекту.

Також модель враховує необхідність у проведенні інновацій, яка зростає із зростанням розміру проекту. Відповідно до зростання інтенсивності інновацій відбувається зростання і трудовитрат проекту.

Важливий висновок, який можна отримати з аналізу моделі COCOMO, полягає в тому, що зростання трудовитрат на реалізацію проекту при переході до іншого режиму моделі (в тому числі і викликане необхідністю впровадження інновацій) не означає відповідне зростання тривалості виконання проекту. Зокрема, розрахунок тривалості виконання проекту згідно моделі COCOMO виконується відповідно до формули:

$$F = 2{,}5 \times T^k \qquad (1.4),$$

де F – час, необхідний на виконання проекту;

T – трудовитрати на виконання проекту, розраховані згідно з формулою (1.3);

k – величина, що залежить від режиму моделі COCOMO і дорівнює для органічного режиму – 0,38, для зблокованого – 0,35, для впровадженого – 0,32.

Відповідно до формули (1.4) величина k зменшується при переході від органічного до впровадженого режиму і, відповідно, зменшується час, необхідний для виконання проекту за умов рівних трудовитрат.

Для проміжної моделі COCOMO формула (1.3) змінюється, до неї вводиться коефіцієнт, що враховує фактор коригування трудовитрат, формула отримує такий вигляд:

$$T = a \times P^{b} \times C \qquad (1.5),$$

де T – трудовитрати на розробку ПЗ;

a, b – константи, які залежать від режиму моделі, що визначається типом проекту;

P – розмір ПЗ, *KLOC*;

C – фактор коригування трудовитрат.

Величина фактору коригування трудовитрат (C) визначається, виходячи з 15-ти показників, які враховують параметри програмного продукту, комп'ютерного забезпечення, персоналу та проекту.

Деталізований рівень моделі COCOMO відрізняється від проміжного тим, що весь проект розбивається на складові на основі ієрархії із трьох рівнів: система, підсистема модуль. Для кожного рівня ієрархії оцінка трудовитрат здійснюється окремо.

Також окремо оцінюються фази життєвого циклу по розробці проекту. Виділяється чотири основних фази: вимоги, розробка проекту продукту, деталізований дизайн продукту, кодування та тестування модулів. Такі складові життєвого циклу як інтеграція та тестування, а також підтримка описуються протягом всього життєвого циклу. Поділ на фази і оцінка кожної з них окремо може бути здійснена для всіх трьох рівнів ієрархії.

Враховуючи той факт, що модель COCOMO була створена на основі хронологічних даних, а значення її факторів і коефіцієнтів були встановлені емпіричним

шляхом, то модель передбачає калібрування цих показників, для того, щоб більш точно відповідати особливостям компанії-розробника програмного забезпечення.

Однак для калібрування необхідні хронологічні дані мінімум по п’яти закінченим проектам, а тому цей процес можливий лише у тих умовах, коли компанія-розробник використовує кількісні методи управління процесом розробки і збирає статистичні дані протягом певного часу.

В цілому, модель COCOMO значно краще підходить для визначення трудовитрат на розробку проекту, ніж лінійна модель, однак вона має певні досить суттєві недоліки:

- по-перше, модель побудована емпіричним шляхом і орієнтована на певну категорію «типових» проектів, її не можна в оригінальному вигляді використовувати для проектів, які не підпадають під цю категорію, для таких проектів модель слід калібрувати, для чого потрібна відповідна статистика;
- по-друге, модель COCOMO орієнтована на каскадну модель життєвого циклу і передбачає певне співвідношення трудовитрат між основними фазами: розробка проекту (30%), кодування (30%), інтеграція та тестування (40%);
- по-третє, модель не враховує певні фактори, що суттєвим чином впливають на процес розробки, зокрема, взаємодію з замовником і його рівень участі у виконанні проекту;
- по-четверте, модель не враховує науково-технічний прогрес, зміну середовища розробки і ріст продуктивності за рахунок впровадження інновацій, вона основана на певних хронологічних даних, які можуть бути застарілими по відношенню до сучасних проектів;
- по-п’яте, модель не охоплює весь життєвий цикл програмного забезпечення, її використання починається з проектування системи і закінчується інтеграцією та тестуванням, фази системного аналізу, розробки вимог, впровадження та супроводу залишаються поза увагою моделі.

Незважаючи на зазначені недоліки, модель COCOMO набула значної популярності і є однією з класичних методик, що використовуються при оцінці економічних параметрів проекту.

У другій половині 1990-х рр. була створена модель COCOMO II, що являє собою покращений варіант оригінальної моделі COCOMO, адаптований до сучасних методологій розробки ПЗ, а також придатний для використання зі спіральною та ітераційною моделями життєвого циклу [155].

Також модель COCOMO II, на відміну від багатофакторного регресійного аналізу у моделі COCOMO, використовує Бейесовський аналіз для обробки статистичних даних, який у більшому ступені відповідає особливостям статистичних даних програмних проектів, що характеризуються неповнотою і неоднозначністю [101].

У якості одиниці для визначення розміру ПЗ модель COCOMO II поряд із одиницями LOC використовує також вимірювання розміру за допомогою об'єктних точок.

Модель COCOMO II включає три різні моделі:

- композиційна прикладна модель – придатна для використання у проектах, що розробляються з використанням сучасних інструментальних засобів, основаних на об'єктно-орієнтованих технологіях;
- модель ранньої розробки проекту – придатна для приблизного оцінювання витрат на розробку проекту до того, як була визначена його архітектура;
- пост-архітектурна модель – найбільш деталізована модель, що використовується після розробки загальної архітектури проекту і включає нові правила оцінки розміру проекту, а також нові рівняння визначення трудовитрат порівняно до моделі COCOMO [160].

Модель COCOMO II, на відміну від оригінальної моделі COCOMO, при оцінці розміру ПЗ враховує використання і адаптацію сторонніх компонентів, що відповідає сучасному підходу до створення прикладного ПЗ, який в значній мірі орієнтований на використання сторонніх компонент.

Для оцінки розміру ПЗ (P) у моделі COCOMO II використовується наступна формула [163]:

$$P = KNSLOC + \left[KASLOC \cdot \left(\frac{100 - AT}{100} \right) \cdot \left(\frac{AA + SU + 0{,}4 \cdot DM + 0{,}3 \cdot CM + 0{,}3 \cdot IM}{100} \right) \right] \quad (1.6),$$

де $KNSLOC$ – розмір модуля, тис. строчок коду;

$KASLOC$ – розмір адаптованих компонент, тис. строчок коду;

AT – доля компонент, що були автоматично трансльовані, %;

AA – доля зусиль на оцінку і асиміляцію модуля, %;

SU – розуміння внутрішньої будови модуля (нуль, якщо CM = 0 та IM = 0), %;

DM – доля дизайну, яка була модифікована, %;

CM – доля коду, яка була модифікована, %;

IM – доля інтеграції та тестів, що була модифікована, %.

При оцінці розміру трудовитрат модель COCOMO II враховує фактори масштабу, сумарна дія яких розраховується згідно з наступною формулою [104]:

$$B = 0{,}91 + 0{,}01 \times \sum_{i=1}^{5} W_i \quad (1.7),$$

де W_i – i-й фактор масштабу (модель враховує 5 факторів).

Фактори масштабу визначають експоненціальну залежність між розміром ПЗ та трудовитратами на його розробку. Фактори масштабу замінюють і розширюють поняття режимів для попередньої моделі COCOMO. Вони охоплюють такі питання, що впливають на трудовитрати процесу розробки, як спадкоємність по відношенню до попередніх проектів, гнучкість середовища розробки, систему виявлення і оцінки ризиків, згуртованість команди, а також зрілість процесу розробки у відповідності з моделями SEI CMM/CMMI [104].

Трудовитрати на розробку розраховуються згідно з формулою:

$$T = \prod_{i=1}^{17} (EM_i) \times A \times \left[\left(1 + \frac{BRAK}{100} \right) \times P \right]^B + \left(\frac{ASLOC \times \left(\frac{AT}{100} \right)}{ATPROD} \right) \quad (1.8),$$

де EM – фактор, що модифікує трудовитрати (модель враховує 17 факторів);

A – константа, що визначається в процесі калібрування, її початкове значення становить 2,45;

$BRAK$ – доля коду, що був забракований і не ввійшов у кінцевий продукт, %;

P – розмір ПЗ, розраховується згідно з формулою (1.6), тис. строчок коду;

B – сумарна дія факторів масштабу, розраховується згідно з формулою (1.7);

SF – фактори масштабу (модель враховує 5 факторів);

$ASLOC$ – розмір адаптованого коду, тис. рядків коду;

$ATPROD$ – продуктивність автоматичної трансляції коду;

AT – доля коду, що був автоматично трансльований, %.

Слід зазначити, що модель COCOMO II постійно оновлюється, зміні підлягають значення коефіцієнтів, які коригуються з урахуванням даних, отриманих з нових проектів, по яким була зібрана статистика. Зокрема, у варіанті моделі від 1997 р. стале значення формули (1.7) становило не 1,01, а 0,91 [105].

Для розрахунку часу (F), необхідного для виконання проекту, у моделі COCOMO II використовується наступна формула [104]:

$$F = \left[3{,}67 \times T^{(0{,}28+0{,}2\times(B-1{,}01))}\right] \times \frac{SCED}{100} \qquad (1.9),$$

де T – трудовитрати, розраховується у відповідністю з формулою (1.8);

B – сумарна дія факторів масштабу, розраховується згідно з формулою (1.7);

$SCED$ – фактор, що визначає запізнення чи випередження графіку виконання проекту, %.

Як видно з формули (1.9), її загальний вигляд у порівнянні з формулою (1.4) принципово не змінився, формула була доповнена новими коефіцієнтами і до неї був введений показник, що визначає вплив графіку виконання проекту. Однак в цілому формула (1.9) означає експоненціальну залежність між трудовитратами і часом, необхідним на виконання проекту.

Таким чином, модель COCOMO II вирішила основні недоліки моделі COCOMO, вона інтегрується з сучасними моделями розробки ПЗ, такими як спіральна та ітераційна, а також стандартами та методологіями, такими як SEI SW-CMM/CMMI та RUP. Модель враховує науково-технічний прогрес і ріст продуктивності компанії-розробника за рахунок факторів масштабу, що впливають на оцінку трудовитрат.

Основним недоліком нової моделі залишилася необхідність використання статистичних даних, зібраних на основі завершених проектів, які можуть не в повній мірі відповідати особливостям організації процесу компанії-розробника ПЗ. Хоча модель і передбачає калібрування параметрів, для здійснення даної процедури необхідно зібрати відповідні статистичні дані.

Слід відзначити, що модель COCOMO II містить певні рекомендації стосовно провадження інноваційного менеджменту компанією-розробником, рекомендації відповідають основним положенням моделей SEI SW-CMM/CMMI, зокрема інноваційний менеджмент розглядається з двох позицій: управління технологічними змінами і управління змінами процесу розробки [105].

Іншою моделлю, яка набула популярності для оцінки економічних параметрів процесу розробки ПЗ, побудованою на основі емпіричних даних, є модель SLIM (Software Life-cycle Model, Модель життєвого циклу ПЗ).

Модель SLIM була розроблена в 1970-х рр. Л. Патнамом, який використав формулу Рейлайха по відношенню до життєвого циклу ПЗ [151].

Для встановлення зв’язку між розміром ПЗ та трудовитратами і обмеженням на строк поставки була використана наступна формула [151]:

$$S = C \times K^{1/3} \times t_d^{4/3} \qquad (1.10),$$

де S – розмір ПЗ, строчок коду;

C – фактор середовища, що залежить від стану технології;

K – загальні трудовитрати для всього проекту;

t_d – обмеження на строк поставки (графік), років.

Фактор середовища (C) може бути розрахований за наступною формулою [151]:

$$C = \frac{S}{K^{1/3} \times t_d^{4/3}} \quad (1.11),$$

де S, K, t_d – значення розміру, трудовитрат та обмеження на строк поставки для попередніх проектів.

Для визначення розміру ПЗ (S) у моделі SLIM можуть бути використані різні підходи, а сам розмір може бути визначений не лише як кількість строчок коду, а як і інші одиниці виміру, такі як, наприклад, функціональні точки.

Один із рекомендованих підходів для визначення розміру ПЗ у моделі SLIM полягає у використанні бета-розподілу [151]:

$$S_n = (S_{min} + 4S_i + S_{max}) / 6 \quad (1.12),$$

де S_n – прогнозне значення для номінального розміру ПЗ;

S_{min} – мінімально можливий розмір;

S_i – найбільш ймовірний розмір;

S_{max} – максимально можливий розмір.

Якщо відомі значення розміру, обмеження на строк поставки продукту та визначений фактор середовища, розрахунок трудовитрат можна здійснити наступним чином (виходячи з формули 1.10) [151]:

$$K = \left(\frac{S}{C}\right)^3 \times \frac{1}{t_d^4} \quad (1.13).$$

Фактор середовища (C) моделі SLIM враховує також і продуктивність персоналу, яка може суттєво відрізнятися як для різних проектів, так і для різних компаній, що працюють над проектами подібної складності. Врахування продуктивності персоналу було одним з основних відмінностей моделі SLIM від моделі COCOMO, до тих пір, поки не була розроблена модель COCOMO II.

Незважаючи на значні досягнення у побудові математичних параметричних моделей останнього часу, проведене на початку 2000-х рр. канадськими вченими дослідження доцільності використання оцінки складності проекту як величини, що

може бути самостійно використана для оцінки його тривалості, показало, що подібну оцінку з прийнятним рівнем вірогідності здійснити дуже складно [146].

Тому для оцінки економічних параметрів ПЗ активно використовуються також підходи, орієнтовані на експертну оцінку, метод аналогій, побудову нейронних мереж та ін. [159, 109]

В результаті проведеного аналізу підходів до визначення економічних показників процесу розробки програмного забезпечення можна зробити наступні висновки.

По-перше, головним економічним параметром процесу розробки є вартість, яка може бути розрахована на основі трудовитрат проекту. Трудовитрати, в свою чергу, прийнято розраховувати, виходячи з такого параметру як розмір ПЗ.

По-друге, розмір ПЗ є неоднозначною величиною і для його визначення існують різні підходи, основані на використанні різних одиниць виміру. Одиниці виміру для оцінювання розміру ПЗ не можуть довільним чином зіставлятися і перетворюватися між собою, оскільки в їх основі знаходяться різні підходи до того, що саме брати за критерій оцінки. Найбільш поширеною і універсальною є одиниця вимірювання розміру ПЗ на основі кількості строчок коду, яка в той же час є і найбільш грубішою, оскільки не враховує значну кількість важливих аспектів ПЗ, таких, як наприклад, нерівномірність складності коду і відсутність прямої відповідності між задачею і кількістю необхідного програмного коду, який її здатен вирішити. Останнім часом популярності набувають такі схожі за принципом вимірювання одиниці вимірювання розміру ПЗ як функціональні точки, точки властивостей та об’єктні точки.

По-третє, в підходах до оцінки трудовитрат зберігається основна ідея, яка полягає в тому, що трудовитрати нелінійно залежать від розміру ПЗ. Розмір ПЗ є найважливішим фактором, який формує трудовитрати, однак він не являється єдиним з них. Крім факторів, що визначають складність програмного продукту і вимоги до нього, на трудовитрати впливає також рівень розвитку технологій, які використовуються при розробці і ступінь зрілості процесу створення ПЗ компанії-розробника. Важливим висновком, який можна отримати з математичних параметричних моделей, побудованих емпіричним шляхом, зокрема, COCOMO, є той факт, що зростан-

ня трудовитрат, викликаних серед багатьох факторів і необхідністю впровадження інновацій, не означає зростання строків виконання проекту, а, навпаки, може призвести до їх скорочення за рахунок підвищення продуктивності.

По-четверте, більшість математичних моделей оцінки економічних параметрів розробки ПЗ основана на емпіричному визначенні параметрів, ґрунтуючись на статистичних даних, зібраних із завершених проектів. Сучасні моделі, такі як, наприклад, COCOMO II враховують особливості сучасного процесу розробки ПЗ та інтегруються з поширеними методологіями розробки. Проте використання навіть найскладніших і найсучасніших математичних моделей оцінки не гарантує правильності оцінки і, найголовніше, відповідності можливостям компанії-розробника, оскільки поняття розміру, складності, трудовитрат і, відповідно, вартості, визначаються не лише зовнішніми, а й внутрішніми факторами для компанії-розробника, такими як кваліфікація персоналу, інструментарій для розробки та рівень зрілості процесу розробки.

По-п’яте, інноваційні процеси порівняно слабо враховуються сучасними підходами оцінки економічних параметрів процесу розробки ПЗ. Модель COCOMO використовує досить негнучкий підхід, оскільки передбачає три фіксованих режими функціонування, які відрізняються основними параметрами процесу розробки, в тому числі і необхідністю у інноваційному менеджменті, що передбачає невиправдану залежність між необхідністю управління інноваціями і іншими параметрами процесу розробки. Модель COCOMO II використовує більш гнучкий підхід, інновації розглядаються як один з факторів, що визначають характер залежності між розміром ПЗ та трудовитратами. Схожий підхід використовує також модель SLIM. Однак параметри інноваційного процесу в цілому не розглядаються як одні з основних економічних показників життєвого циклу ПЗ, скоріше їх сприймають як коригуючий фактор, який слід враховувати з метою підтримки в актуальному стані параметрів емпіричних моделей за умов науково-технічного прогресу і старіння хронологічних даних, які лежать в основі моделей.

Висновки до першого розділу

1. Проведене у підрозділі 1.1 дослідження сутності і місця інновацій на етапах життєвого циклу програмного забезпечення, показало, що інновації становлять один з найважливіших факторів, який є джерелом розвитку галузі розробки ПЗ.

2. ПЗ як продукт інтелектуальної праці людини характеризується надзвичайно високим рівнем складності, що, в свою чергу, є основною причиною проблем, які виникають при побудові ефективної системи управління створенням ПЗ.

3. Наявність в науковій літературі порівняно значної кількості принципово різних підходів до характеристики життєвого циклу ПЗ, а також визначення економічних параметрів процесу його розробки, пояснюється як швидкими змінами, що відбуваються в процесі розвитку галузі ПЗ, так і визнанням того факту, що на даний момент відсутні рішення, які б могли повністю розрішити проблему ефективного управління розробкою ПЗ, в тому числі, і в сфері інноваційного менеджменту.

4. На нашу думку, управлінням інноваціями на етапах життєвого циклу ПЗ приділяється недостатня увага, в основному управління здійснюється або з суто технічних позицій, або з позицій, що не враховують специфіку галузі ПЗ. Економічний механізм управління інноваціями має бути одним з центральних елементів загальної системи управління процесом створення ПЗ і враховувати специфіку даної діяльності, що пояснює необхідність проведення подальших досліджень у цьому напрямку.

5. Враховуючи критичний стан із проведенням НДР і впровадженням нових технологій в Україні [12, 14, 16, 23, 24, 52, 53, 55, 64, 65, 71, 72], для забезпечення конкурентноздатності вітчизняної економіки слід зосередити увагу на інтенсивному розвитку окремих високотехнологічних галузей народного господарства, до яких відноситься розробка ПЗ.

Основні положення даного розділу знайшли відображення у наступних наукових працях здобувача [7, 30, 32, 34, 36, 78]:

1. Богдановський В. Г., Колдовський В. В. Софтверний фактор у питаннях створення ефективної системи автоматизованого управління бізнес-процесами // Вісник Сумського державного аграрного університету. Серія: „Економіка та менеджмент". Вип. 2. – Суми, 2001. – С. 177-180.
2. Кислий В. М., Колдовський В. В. Проблеми використання новітніх технологій у сфері розробки програмного забезпечення // Тезисы докладов научно-технической конференции преподавателей, сотрудников, аспирантов и студентов экономического факультета Сумского государственного университета. – Сумы: Изд-во СумГу, 2002. – С. 64-65.
3. Козьменко С. М., Колдовський В. В. Сучасні моделі управління процесом розробки програмного забезпечення // Проблеми і перспективи розвитку банківської системи України: Збірник наукових праць. Т. 12. – Суми: ВВП «Мрія-1» ЛТД, УАБС НБУ, 2005. – С. 43-49.
4. Колдовский В. В. Разработка ПО: модели жизненного цикла // Компьютерное обозрение. – 2005. – № 24. – С. 56-59.
5. Колдовский В. В., Козьменко С.Н. Перспективы и возможности отечественного рынка бесплатного программного обеспечения // Збірник наукових праць Сумського державного університету/ Механізм регулювання економіки, економіка природокористування, економіка підприємства та організація виробництва. 1999 – Випуск 3 (99). – С.139-141.
6. Ярошенко С. П., Колдовський В. В. Умови ефективної реалізації проектів автоматизації в промисловості // Механізм регулювання економіки, економіка природокористування, економіка підприємства та організація виробництва. – Суми: Вид-во СумДУ. – 2002. – № 1-2. – С. 152-155.

РОЗДІЛ 2
ТЕОРЕТИКО-МЕТОДИЧНІ ОСНОВИ УПРАВЛІННЯ ІННОВАЦІЯМИ НА ЕТАПАХ ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Науково-методичні підходи до розробки економічного механізму управління інноваціями на етапах життєвого циклу програмного забезпечення

Проведений у першому розділі аналіз науково-методичних підходів до управління інноваціями на етапах життєвого циклу програмного забезпечення, свідчить про необхідність розробки комплексного механізму управління інноваціями, побудованого на принципах економічного підходу до розробки і прийняття рішень.

Система управління інноваціями має відповідати місії і стратегічним цілям компанії-розробника ПЗ, вона має бути дієвим і практичним інструментом, який забезпечує конкурентні переваги фірми.

На наш погляд, економічний механізм управління інноваціями на етапах життєвого циклу ПЗ має вирішувати такі задачі:

- забезпечувати пошук та розробку інновацій за рахунок як зовнішніх, так і внутрішніх джерел;
- здійснювати оцінку доцільності впровадження інновацій з урахуванням як економічних, так і технічних факторів, що складно піддаються економічній оцінці;
- розроблювати план впровадження інновацій, враховуючи можливість відміни впровадження, якщо будуть виявлені раніше не передбачені фактори, дія яких призводить до недоцільності подальшого впровадження інновацій;
- забезпечувати реалізацію інновацій на практиці, здійснюючи контроль за ходом процесу і вносячи відповідні корективи, якщо процес відхиляється від запланованого курсу;
- якщо це необхідно, то забезпечити відміну інновацій відповідно до розробленого плану;

– надати можливості для аналізу процесу, збору інформації і формування висновків для подальшого використання;
– забезпечити безперервність процесу і його органічне поєднання із загальною системою управління компанією на усіх рівнях.

На рис. 2.1 представлено схематичне зображення етапів інноваційного процесу, які він має проходити циклічно і безперервно.

Рис. 2.1. Етапи інноваційного процесу

На наш погляд, основною ідеєю економічного механізму управління інноваціями має бути контрольованість процесу і його відповідність як короткостроковим, так і довгостроковим цілям фірми. Інноваційні процеси не мають бути спонтанними

чи безконтрольними, оскільки в такому разі вони становитимуть ризик для стабільної діяльності фірми.

Відповідно до рис. 2.1 нами пропонується виділити 8 етапів, які мають вирішувати вище наведений перелік задач, що стоять перед економічним механізмом управління інноваціями на етапах життєвого циклу ПЗ.

Склад задач, які необхідно вирішувати на кожному з етапів, пропонується такий:

- на етапі визначення потреб, цілей, задач, пріоритетів інноваційного менеджменту слід проаналізувати потреби компанії, визначити цілі інноваційного менеджменту на різних рівнях – від стратегічного до оперативного, визначити пріоритети інноваційної діяльності і сформулювати конкретні задачі, які необхідно вирішити;
- на етапі пошуку та замовлення розробки нововведень у зовнішніх джерелах, чи розробки на основі внутрішніх джерел слід створити дієвий механізм, який би слугував джерелом нововведень для компанії. Необхідно одночасно сфокусуватися на двох напрямках, що можуть слугувати джерелом нововведень: зовнішньому та внутрішньому;
- на етапі оцінки доцільності впровадження необхідно здійснити комплексну оцінку і співставлення витрат і вигод, які будуть отримані в результаті впровадження інновацій, можливо, доцільно розглянути і зіставити декілька альтернативних варіантів;
- на етапі розробки плану дій необхідно розробити детальний план подальших дій з реалізації інновацій, зокрема, необхідно передбачити виділення необхідних ресурсів, а також розробити план відміни впровадження, який може бути використаний у тому разі, якщо в процесі реалізації нововведення були виявлені складнощі, що роблять подальше продовження процесу недоцільним;
- на етапі створення рішення необхідно відповідно до розробленого на попередньому етапі плану здійснити розробку конкретного рішення з впровадження інновації в діяльність компанії, здійснити, якщо це необхідно, ада-

птацію інноваційної технології до особливостей діяльності компанії-розробника ПЗ;

- на етапі управління реалізацією рішення необхідно забезпечити впровадження інноваційної технології у діяльність компанії, особлива увага має приділятися контрольованості і прозорості процесу, проблеми, що стають на перешкоді процесу мають виявлятися і оперативно усуватися, велике значення має робота з персоналом, який повинен сприймати інноваційні процеси і бути їх активним прихильником;
- на етапі відміни рішення здійснюється відміна реалізації впровадження інноваційного рішення, якщо процес впровадження зіткнувся з перешкодами, які неможливо, чи було недоцільно усувати. Також відміна може бути здійснена у тому разі, якщо в процесі впровадження була отримана нова інформація, що свідчить про недоцільність реалізації інновації;
- на етапі аналізу результатів ітерації, формування висновків, забезпечення безперервності процесу здійснюється комплексний аналіз інформації, зібраної в процесі проходження результатів ітерації незалежно від того, був досвід реалізації успішним, чи закінчився невдачею. Усі проблеми, що виникли на шляху впровадження, мають бути задокументовані і проаналізовані, а отримана у вигляді висновків і рекомендацій інформація має бути основою для подальших ітерацій, які мають здійснюватися безперервно, становлячи тим самим основу для постійного підвищення конкурентоспроможності компанії.

Слід зазначити, що інноваційні процеси є джерелом змін до процесу розробки ПЗ, а тому мають розглядатися через призму механізмів, які відповідають за управління змінами.

Управління змінами залежить від специфіки програмного коду і особливостей реалізації програмного продукту, зокрема його життєвого циклу.

Наприклад, що стосується внесення змін у програмні продукти, які працюють з базами даних, то для них виділяються такі рекомендації [139].

По-перше, більшість подібних програмних продуктів базуються на використанні поняття об’єкту (сутності), який у більшості випадків виражений у вигляді глобального для проекту переліку однотипних елементів, що пов’язує навколо себе компоненти програми.

По-друге, подібні об’єкти мають набір певних властивостей, однак у більшості випадків даний набір не є статичним. Програмні продукти мають передбачати зміну набору властивостей, насамперед, у напрямку до його розширення, проте використання традиційного реляційного підходу до побудови баз даних суттєво ускладнює даний процес. Зокрема, вирішення відносно простих завдань зі зміни поведінки об’єкта, пов’язаних зі зміною набору його властивостей, може передбачати внесення надзвичайно комплексних змін до всього проекту.

З метою вирішення даної проблеми з урахуванням специфіки життєвого циклу подібного ПЗ пропонується відмовитися від зберігання властивостей об’єкта у вигляді полів у одній таблиці з власне самим об’єктом. Пропонується виділити три окремі таблиці: першу – з переліком об’єктів, другу – з переліком властивостей та третю таблицю, що пов’язує об’єкт з його властивостями і зберігає значення властивостей. У даному разі зміна переліку властивостей об’єкта не передбачатиме реструктуризацію таблиць даних, а вимагатиме лише від програмного коду врахування появи нових властивостей у об’єкта чи зникнення старих [139].

Однак суттєвим недоліком даного підходу можна вважати значне зростання складності доступу до подібних даних, зокрема при побудові звітів, що негативно вплине на трудомісткість операцій з таблицями даних та неминуче призведе до зростання собівартості проекту. Для вирішення проблем, що можуть виникнути у даному разі, слід запропонувати повністю відмовитися від прямого доступу до таблиць даних і перенаправляти усі запити виключно до відображень та процедур, що зберігаються у базі даних. Подібний крок, направлений на відокремлення служб даних від інших служб програмного проекту, дозволяє ще більш значним чином спростити внесення змін до структур даних, не впливаючи при цьому на складність та трудомісткість інших служб проекту.

По-третє, у якості однієї з найважливіших проблем розглядається проблема дублікатів у системі. Для будь-якого проекту, що має складну базу даних, слід передбачити механізми виявлення і усунення дублікатів. Дублікати можуть виникнути зі значної кількості причин: від помилок користувачів, до збоїв у системі, не зважаючи на наявність відповідних механізмів підтримки унікальності. Основна ідея, полягає в тому, щоб відмовитися від ручного усунення дублікатів і сфокусуватися на розробці відповідних інструментів [139].

Інший спосіб уникнути дублікатів полягає в тому, щоб розмістити їх на одному рівні ієрархії, однак це можливо лише за умови, якщо база даних побудована за ієрархічним принципом.

По-четверте, при розробці баз даних автор рекомендує орієнтуватися на ієрархічні структури даних, що, на відміну від загально прийнятого підходу при розробці баз даних, не вимагає зміни структури таблиць даних під час зміни кількості рівнів ієрархії. Ієрархічний принцип суттєво ускладнює маніпулювання даними, проте дану проблему можна вирішити за рахунок використання процедур, що зберігаються у базах даних.

Однак головна перевага ієрархічного принципу – надзвичайно висока гнучкість при побудові структур даних. За допомогою даного підходу можна зручним чином усунути дублікати, не видаливши їх з бази даних повністю, адже вони можуть мати хоч і не суттєві, проте деякі відмінності; за допомогою даного підходу можна навпаки, визначити відмінності для екземплярів одного і того ж об’єкту, розмістивши різні гілки на ієрархії, рівень якої нижчий за рівень об’єкту.

По-п’яте, увага має бути зосереджена на тому, що система має бути спроектована з урахуванням можливості для тривалого функціонування, а тому має містити механізми зменшення кількості даних, якими необхідно постійно оперувати, розмістивши їх у архіві. При цьому слід з самого початку орієнтуватися на оптимальне використання дискового простору, зокрема, якщо це можливо, то зберігати в історії даних лише відмінності від попередніх значень, а не повні записи на певний проміжок часу.

По-шосте, рекомендується передбачати певні типові проблеми для функціонуючої системи, що знизить трудовитрати на підтримку системи на відповідних етапах її життєвого циклу.

Зокрема, усі об’єкти у таблицях даних є абстракцією реальних об’єктів, а тому завжди відображають лише окремі властивості реальних об’єктів і не можуть точно відповідати ним, в результаті завдання впізнати кожен конкретний об’єкт за певних обставин не може бути вирішене. Лише частково у даному разі може допомогти зміна властивостей об’єкта.

Далі, при побудові проекту рекомендується максимальним чином сфокусуватися на можливостях до його розширення і настройці обслуговуючим персоналом. Перенесення певних задач з програмування на плечі обслуговуючого персоналу дозволить суттєво зменшити навантаження на команду розробників проекту і у окремих випадках значним чином знизити вартість розробки.

Крім того, при побудові проекту слід жодним чином не обмежувати його можливості для розвитку, зокрема важливим завданням має бути передбачення росту масштабу проекту у перспективі, що дозволить скоротити витрати на підтримку проекту на завершальних етапах життєвого циклу.

На наш погляд, при розробці механізму управління інноваціями на етапах життєвого циклу ПЗ особливу увагу слід приділити обмеженості суто економічного підходу до управління інноваціями у галузі розробки програмного забезпечення.

Зокрема, проекти по розробці ПЗ з відкритим вихідним кодом не забороняють використання власних розробок у інших проектах, що, фактично, має стримувати інноваційні процеси, якби вони керувалися лише економічними методами прийняття рішень щодо обґрунтування доцільності інвестицій, оскільки інвестиції у розробку чи придбання інновацій стають недоцільними, якщо є можливості для безоплатного використання доступних розробок як компанією-розробником, так і її конкурентами.

Однак на практиці економічна недоцільність не стримує проведення активних інноваційних заходів, компанії продовжують їх здійснювати, що можна пояснити

дією не лише економічних, а і інших факторів, які складно піддаються кількісній оцінці [133].

Також обмеження використання суто економічних методів при прийнятті рішень стосовно доцільності впровадження інновацій є дія таких факторів, які складно спрогнозувати, наприклад, перспективність технології у майбутньому.

Слід зазначити, що обрання тієї чи іншої технології, яка буде використовуватися у процесі розробки, обов'язково має здійснюватися з урахуванням перспективності даної технології, чи буде вона конкурентноздатною і яким чином вона буде розвиватися у майбутньому.

Наприклад, компанія Apple, що носить неофіційний статус «найбільш інноваційної компанії світу», неодноразово змінювала постачальника, і, відповідно, архітектуру мікропроцесорів для своїх систем. Подібні зміни дуже болісно відзначаються на бізнесі компанії та її партнерів, оскільки вимагають значних витрат. Однак відмовитися від подібних змін неможливо, оскільки існуючі технології компанії виявлялися неконкурентноздатними і становили загрозу для неї взагалі втратити ринок.

Таким чином, необхідність здійснення інновацій у вигляді переходу на нові технології для компанії Apple пояснювалася втратою конкурентноздатності існуючих технологій, що, в свою чергою, було помилкою при здійсненні стратегічних інноваційних рішень у минулому, коли були обрані технології, які не змогли забезпечити конкурентноздатний розвиток продуктів компанії.

На нашу думку, в сфері високих технологій взагалі і при розробці ПЗ зокрема, проблема вибору інновацій на основі не лише економічних факторів, тобто таких, які можна виразити у вигляді кількісних показників, що характеризують співвідношення доходів і витрат, а й інших, що не піддаються такому вираженні, стоїть досить гостро. Подібна проблема активно обговорюється у наукових колах і стосовно інших інвестиційних проектів [64].

Існують межі, в рамках яких не можна визначити з достатнім рівнем вірогідності, чи буде технологія конкурентноздатною у майбутньому, однак можливість технології розвиватися і мати конкурентноздатні характеристики не лише у корот-

костроковому, а і у довгостроковому періоді може розглядатися як найважливіший фактор, що впливає на прийняття рішення про її вибір до впровадження.

Таким чином, при прийнятті інноваційного рішення недоцільно приймати до уваги лише економічні показники альтернативних варіантів, на нашу думку, доцільно ввести також «неекономічну складову», що має включати такі фактори, як перспективність технології, її потенціал до розвитку, прийняття персоналом фірми та ін.

Для визначення показників і вагомості даної складової доцільно використовувати методи експертної оцінки, залучаючи експертів як з числа персоналу фірми, так і зі сторони.

Також у даному ракурсі суттєве значення має розуміння загальної широти наслідків, які приймає інноваційний процес у інших площинах, крім економічної.

Оскільки, як уже зазначалося, розробка ПЗ являє собою суто інтелектуальний процес, то інновації, які змінюють технології, що використовуються в процесі розробки, становлять значний вплив на інтелектуальну діяльність учасників даного процесу, на їх спосіб мислення, прийняття рішень, сприйняття світу та ін.

Таким чином, технічні інновації виходять за рамки суто технічних процесів, вони набувають нових властивостей, оскільки ефект поширюється на інші сфери людської діяльності, причому соціальна сфера найбільше відчуває вплив інноваційних процесів.

В результаті, у сфері інформаційних технологій взагалі, і при розробці ПЗ зокрема, слід використовувати поняття соціально-технічних інновацій, що віддзеркалює поєднання змін технологічного процесу і характеру людської діяльності [84].

Враховуючи комплексний характер наслідків, що викликають за собою соціально-технічні інновації, зростає важливість здійснення пілотних проектів, які дозволяють визначити ефект від впровадження інновацій як результат дії багатьох факторів і зміни характеру інновацій по відношенню до використаного контексту і отриманого ефекту (рис. 2.2).

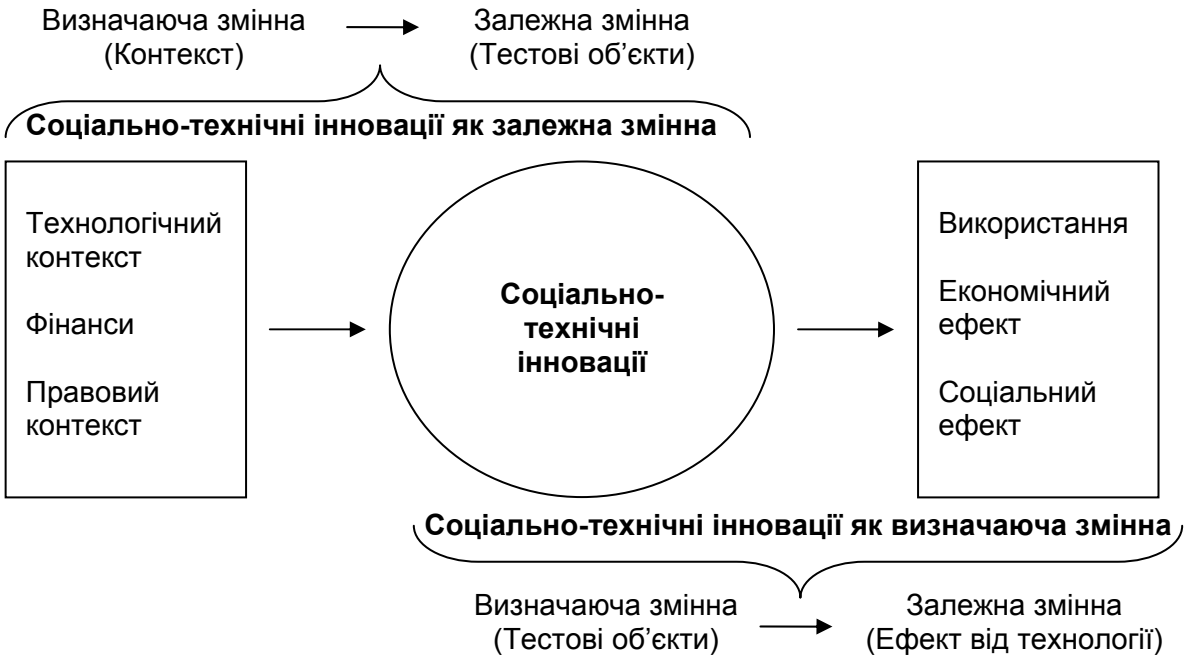


Рис. 2.2. Соціально-технічні інновації в процесі реалізації пілотного інноваційного проекту [84]

Пілотні проекти є значно більш ефективним способом оцінки доцільності здійснення соціально-технічних інновацій, ніж, наприклад, прототипування, оскільки дозволяють розглянути такі складові інноваційного процесу, які можуть залишитися поза увагою при використанні інших підходів [84].

На наш погляд, під час характеристики основних показників процесу розробки ПЗ, що визначають основні його параметри, такі як, наприклад, надійність, доцільно використовувати поняття «кортеж програми», наведене у [31].

На рис. 2.3 зображено кортеж програми, що складається з наступних елементів:

- самої програми (у вихідних текстах або у машинних кодах);
- режиму експлуатації (включає вихідні дані та кваліфікацію користувача);
- середовища експлуатації (включає апаратну конфігурацію обладнання, системне програмне забезпечення, зокрема, операційну систему та драйвери обладнання, а також прикладне програмне забезпечення, яке може справляти вплив на роботу інших програм);

- документації, яка визначає допустиму множину вихідних даних, середовища, а також необхідну кваліфікацію користувачів програми.

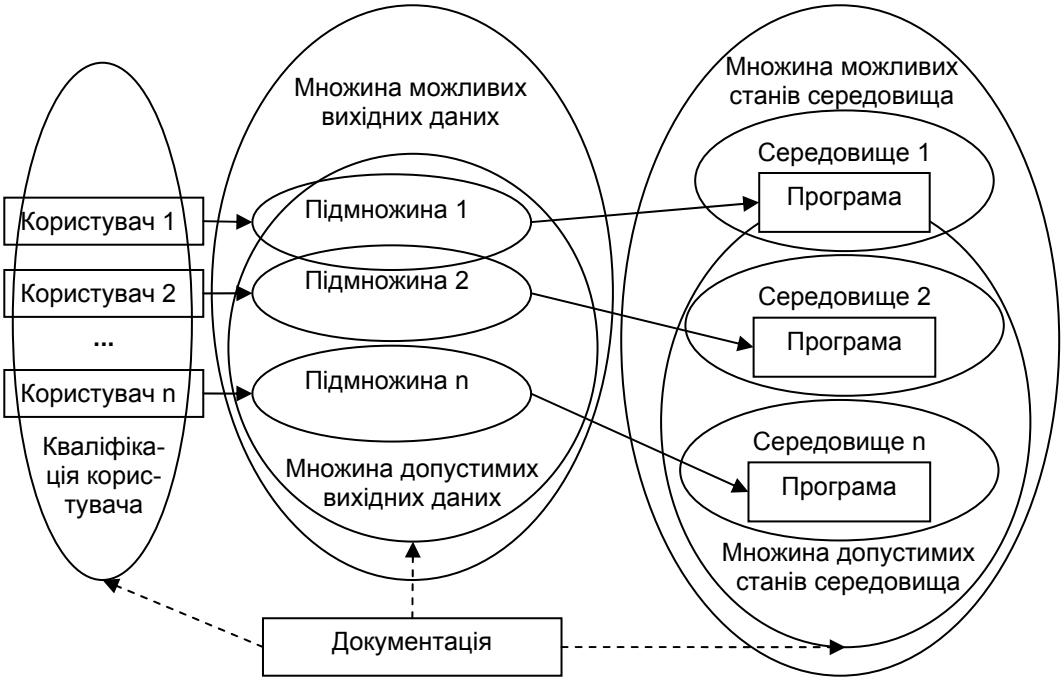


Рис. 2.3. Кортеж програми

Відповідно з рис. 2.3 інші складові кортежу, крім програмного коду та документації, такі як середовище виконання, вихідні дані та можлива кваліфікація користувачів можуть суттєво відрізнятися від одного випадку використання програмного забезпечення до іншого. Іноді ці складові мають непередбачувані та взагалі недопустимі значення, що, в кінцевому разі, негативно позначається на здатності програмного забезпечення функціонувати без збоїв, тобто відповідати критеріям якісної характеристики ПЗ, зокрема надійності.

Таким чином, надійність програмного забезпечення визначається не лише його програмним кодом, а усіма складовими кортежу програми, який являє собою конкретну реалізацію програмного забезпечення для конкретного режиму експлуатації та у конкретному середовищі. Документація обмежує допустиму множину вихідних даних, середовище експлуатації, а також кваліфікацію користувачів.

Аналізуючи кортеж програми, можна зробити висновок, що програмне забезпечення – це значно більше, ніж просто набір машинних кодів. Наприклад, документація є невід'ємною складовою програмного забезпечення, адже саме вона визначає умови, для яких використання програмного забезпечення є допустимим, а для яких – ні. Некоректно написана документація рівноцінна помилці у програмному коді, адже вона не дозволяє використати систему для вирішення тієї чи іншої задачі. Крім того, за рахунок внесення змін до документації можна надзвичайно дешево (у порівнянні із іншими методами) позбутися помилок у програмному коді – наприклад, обмежуючи середовище чи режим експлуатації програмного забезпечення.

В результаті можна стверджувати, що надійність, як складова якості програмного забезпечення, забезпечується шляхом підтримки кортежу програми в такому стані, в якому функціонування програмного забезпечення відповідає критеріям надійності. У процесі розробки програмного забезпечення для даного виду діяльності використовується спеціальний термін – «управління конфігурацією», а можливий прийнятний варіант кортежу програмного забезпечення має назву «конфігурація».

Виходячи з необхідності проведення тестування ПЗ, слід зауважити, що обсяги тестування мають бути обмежені тим рівнем затрат, що виробник ПЗ несе у зв'язку з тестуванням: затримуючи випуск ПЗ у зв'язку з тестуванням, виробник втрачає певну долю ринку, однак випустивши неякісний виріб, що не був достатнім чином протестований, виробник також втратить певну долю ринку через відмову потенційних покупців від придбання ненадійного ПЗ.

Для того, щоб визначити той обсяг тестування, що є достатнім для випуску ПЗ, у роботі [91] вводиться поняття „Впливу ризику" (англ. „Risk exposure", *RE*), який можна розрахувати за формулою:

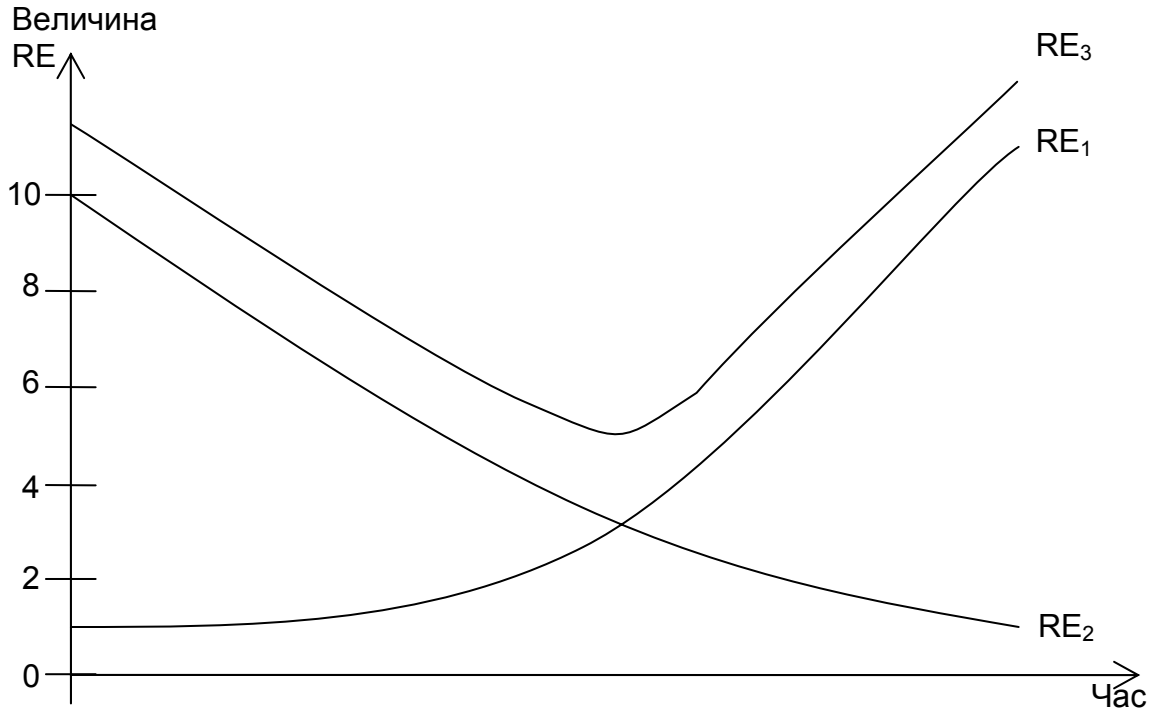
$$RE = I_B \times O_B, \tag{2.1}$$

де I_B – імовірність втрат;

O_B – обсяг втрат.

На рис. 2.4 зображено графічне представлення залежності трьох різних величин впливу ризику: Вплив ризику від втрати ринкової долі у зв'язку з затримкою

випуску (RE_1), Вплив ризику від втрати ринкової долі у зв'язку з випуском неякісного виробу (RE_2),загальна величина Впливу ризику (RE_3).



Де:
RE_1 – вплив ризику від втрати ринкової долі у зв'язку з затримкою випуску;
RE_2 – вплив ризику від втрати ринкової долі у зв'язку з випуском неякісного виробу;
RE_3 – загальна величина впливу ризику.

Рис. 2.4. Вплив ризику під час тестування ПЗ [91]

Як видно з рис. 2.4, загальна величина впливу ризику (*RE_3*) набуває мінімального значення в точці часу, що відповідає перетину *RE_2* та *RE_1*. Це означає, що компанія-виробник ПЗ має оцінити величини *RE_2* та *RE_1*, що дасть їй можливість розрахувати оптимальну величину часу, котрий доцільно використати на тестування.

Таким чином, якість – це значно більше, ніж відсутність помилок. Границі продукту і дата, коли він передається в експлуатацію, також є важливими елементами якості. Під час розробки доводиться постійно знаходити компроміси між набором можливостей, що повинні бути реалізовані, термінами виконання і надійністю продукту. Іноді дата випуску продукту більш важлива, і в цьому випадку необхідно скорочувати перелік можливостей, реалізованих у даній версії. В окремих випадках можливості продукту більш важливі, ніж дата випуску, і в такому разі доцільно переглядати плани-графіки проекту.

Великого інтересу питання управління інноваціями при реалізації програмних проектів викликає стосовно роботи лідера ринку програмного забезпечення – корпорації Microsoft. Витік частини вихідного коду операційної системи MS Windows 2000, що стався на початку 2004 р., дозволив досить детально проаналізувати окремі деталі управління змінами в програмних продуктах.

Зокрема, за опублікованими деталями вихідного коду було встановлено, що корпорація приділяє значну увагу забезпеченню сумісності між різними версіями програмних продуктів, навіть у такому питанні, як виправлення помилок [158].

Причому особливої уваги заслуговує той факт, що в однаковій мірі значна увага приділяється наступним напрямкам забезпечення сумісності:

– по-перше, забезпечення сумісності з програмним забезпеченням, що веде себе некоректно і використовує особливості роботи операційної системи чи недокументовані функції, замість використання стандартних і документованих шляхів вирішення тих же завдань (зокрема, забезпечується сумісність з такими програмними продуктами як Microsoft Exchange 2, Microsoft Office 95, Microsoft Access, IBM Antivirus та ін.);
– по-друге, забезпечення сумісності з програмним забезпеченням, що враховує помилки та некоректність роботи попередніх версій операційної системи, і використовує різноманітні «хитрощі» для їх обходу (зокрема, таким чином забезпечується сумісність з Borland Java Builder).

Забезпечення сумісності як за першим, так і за другим напрямком роботи з економічної точки зору суттєво збільшує вартість розробки програмного забезпечення, саме тому доцільно розглянути, які саме цілі переслідує корпорація Microsoft, витрачаючи додаткові ресурси на інвестиції у даному напрямку.

Більш детальний аналіз коду показує, що найбільшу увагу приділено забезпеченню сумісності з продуктами самої корпорації Microsoft, навіть якщо продукт був замінений більш новими версіями. Подібні дії можна однозначно охарактеризувати як один з інструментів нечесної конкурентної боротьби та прикладів використання монопольного положення на ринку: в той час як якість самих програмних продуктів корпорації не є достатньо високою, Microsoft штучно «завищує» якість продукту

шляхом «тонкої наладки» під нього операційної системи. Звісно, що конкуренти подібних переваг не отримують, а кінцевий користувач, роблячи вибір між одним чи іншим програмним продуктом, в дійсності не може знати, наскільки якісним він є «зсередини», він розглядає роботу продукту комплексно.

Таким чином, подібний підхід є одним з найбільш дієвих інструментів конкурентної боротьби, що застосовуються корпорацією Microsoft, і досить часто ставить конкурентів у невигідне положення, коли з виходом нової версії операційної системи від Microsoft, єдиними програмними продуктами серед багатьох конкурентних, що нормально у ній функціонують, є продукти самої корпорації. Хоча насправді це не означає, що продукти конкурентів є менш якісними, просто Microsoft зробила певні інвестиції в те, щоб забезпечити сумісність з власними програмними продуктами на рівні операційної системи і, звісно, не займалася тим, щоб забезпечити подібним чином сумісність для продуктів конкурентів.

Якщо ж корпорація Microsoft займається налагодженням операційної системи по відношенню до програмних продуктів сторонніх постачальників, то в даному разі також простежується помітна тенденція: додаткові витрати ресурсів здійснюються на забезпечення сумісності лише з програмними продуктами відомих і крупних компаній, таких як IBM та Borland.

Дії Microsoft у даному напрямку також достатньо просто пояснити з економічної точки зору: якщо кінцевий користувач не матиме змоги використовувати потрібний йому програмний продукт з нової версією операційної системи, то він швидше відмовиться від придбання та встановлення подібної операційної системи, ніж програмного продукту. Звідси можна зробити висновок, що забезпечення сумісності по відношенню до програмних продуктів відомих компаній – один з маркетингових кроків для забезпечення продаж нової версії операційної системи.

Таким чином, успішна ринкова стратегія компанії-розробника ПЗ можлива за умови здійснення виваженої стратегії управління процесом створення програмних продуктів на усіх етапах життєвого циклу.

2.2. Управління впливом сторонніх компонентів на процес розробки програмного забезпечення

Сучасні проекти з розробки ПЗ відрізняються надзвичайною складністю, яка має стійку тенденцію до зростання.

На рис. 2.5 представлено графік, що відображає зростання розміру відомих програмних проектів з відкритим вихідним кодом RedHat та Debian за період з 1998 по 2002 р.

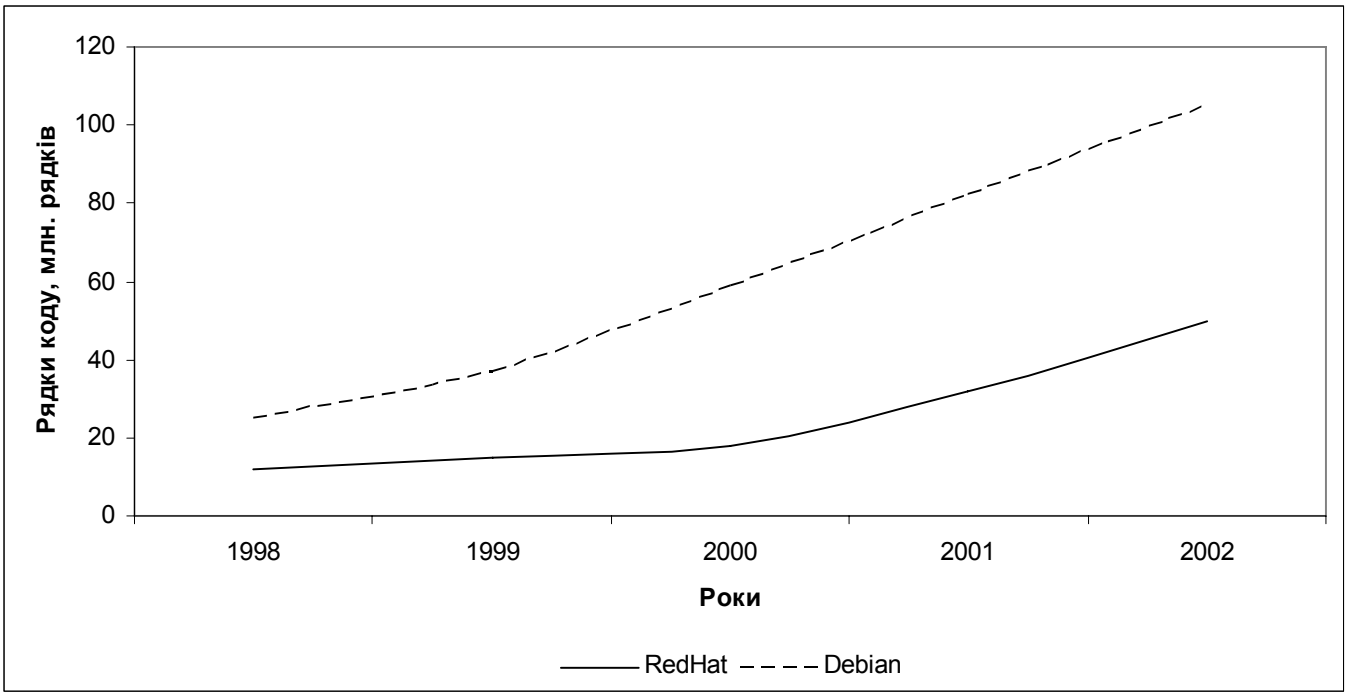


Рис. 2.5. Зростання розміру програмних проектів з відкритим вихідним кодом RedHat та Debian за період з 1998 по 2002 р. [133]

Як видно з рис. 2.5 за чотири роки розвитку проекти суттєво збільшилися у обсязі, зростання кількості рядків коду за цей період становило 3-5 разів по відношенню до 1998 р.

Значне зростання розміру програмних проектів також означає і відповідне підвищення складності, що, в свою чергу, призводить до росту витрат на розробку.

Як зазначалося у підрозділі 1.3, зростання витрат на розробку нелінійно залежить від розміру проекту.

Оскільки проекти RedHat та Debian є проектами з відкритим вихідним кодом, що розробляються переважно на безоплатній основі, то оцінка вартості їх розробки може бути здійснена лише умовно, припускаючи, що праця учасників проекту має оплачуватися таким чином, якщо проекти були б повністю комерційними.

На рис. 2.6 представлено графік, що відображає зростання оціночної вартості розробки програмних проектів з відкритим вихідним кодом RedHat та Debian за період з 1998 по 2002 р. Оцінка вартості була здійснена з використанням моделі COCOMO [133].

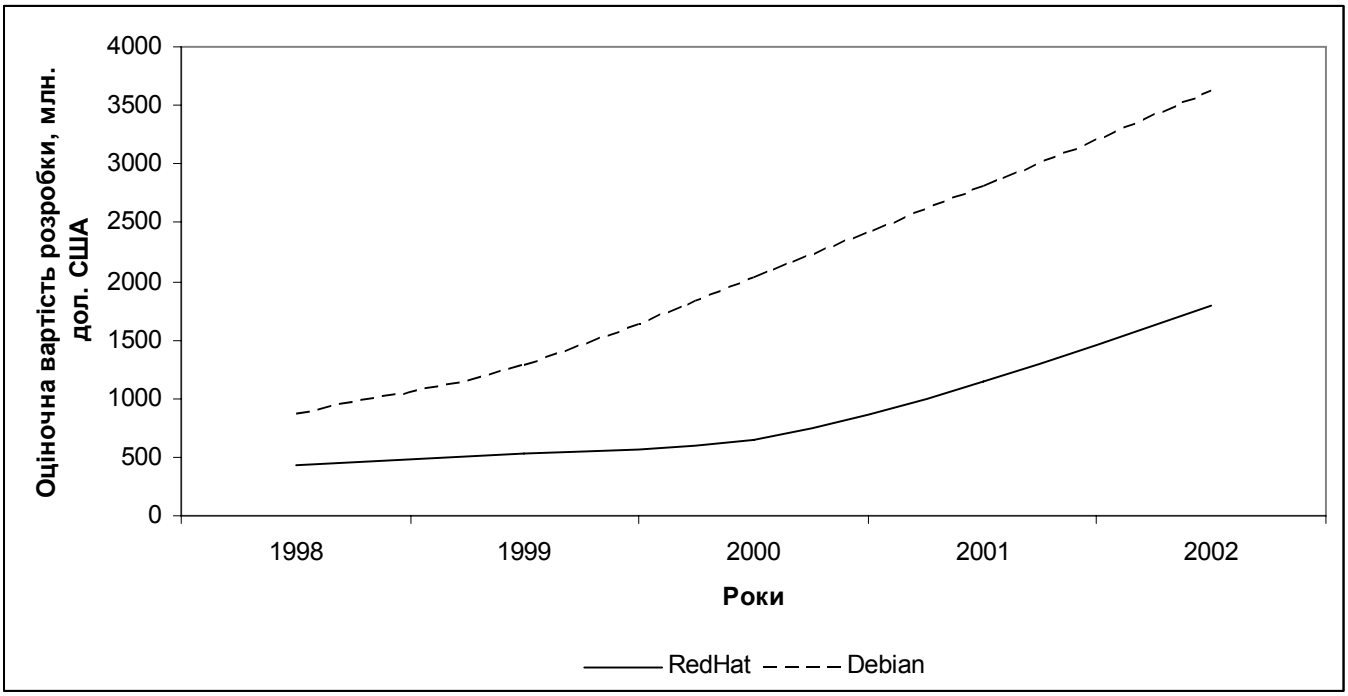


Рис. 2.6. Ріст оціночної вартості розробки програмних проектів з відкритим вихідним кодом RedHat та Debian за період з 1998 по 2002 р. [133]

Як видно з рис. 2.6 за період у чотири роки оціночна вартість розробки перевищує для обох проектів 1 млрд. дол. США, що свідчить про надзвичайно високу капіталоємність великих проектів з розробки ПЗ.

Суттєво зменшити вартість розробки можна за допомогою використання сторонніх компонентів як готових програмних модулів, розроблених сторонніми поста-

чальниками, які можна використовувати при реалізації власних програмних проектів.

Роль сторонніх компонентів при розробці сучасного ПЗ постійно зростає, однак, на нашу думку, питання управління використанням сторонніх компонентів при реалізації програмних проектів у сучасній науці розкрито не достатньо повно, звичайно більше уваги приділяється питанням управління змінами, що не відповідає в повній мірі основним завданням управління інноваціями, до яких слід віднести управління сторонніми компонентами.

З метою управління змінами в процесі розробки ПЗ використовуються підходи, що отримали назву «управління конфігураціями» [115, 90, 86].

Управління конфігураціями являє собою комплекс методів і інструментів, призначених для підтримки управління і розробки ПЗ, які дозволяють забезпечувати ідентифікацію коректного вихідного коду, контроль за вихідним кодом, аудит змін до вихідного коду, можливість здійснювати перегляд вихідного коду, управління компіляцією коду, управління процесом розробки і забезпечувати можливість колективної роботи над вихідним кодом [90].

Однак для традиційного підходу управління конфігураціями програмного забезпечення не застосовуються окремі заходи стосовно управління інноваційними процесами під час розробки програмного забезпечення. У якості найпростішого прикладу подібного інноваційного процесу можна розглянути перехід з однієї версії допоміжної бібліотеки, що використовується піж час розробки програмного забезпечення, на іншу, більш досконалу. З точки зору управління конфігураціями даний процес не підпадає під процес контролю за змінами, а тому в традиційному підході не розглядається як випадок набуття програмним продуктом нових функцій чи властивостей, що мають бути піддані детальному аудиту управління конфігурацією.

Дійсно, програмний продукт у випадку застосування нової версії допоміжної бібліотеки звичайно не набуває якихось додаткових функціональних властивостей, однак це не означає що перехід з однієї версії допоміжної бібліотеки на іншу є настільки ординарним процесом, що роботою над ним під час управління конфігурацією можна знехтувати.

Річ у тім, що з підвищенням складності програмного забезпечення, яке відбувається еволюційно у зв’язку з об’єктивним зростанням обсягів програмних кодів сучасних програмних проектів, роль допоміжних бібліотек суттєво зростає. Зокрема, код допоміжних бібліотек, що використовуються при розробці сучасних програмних продуктів, займає надзвичайно значну долю у загальному коді проекту.

У табл. 2.1 наведено порівняльний аналіз кількості рядків основного коду по відношенню до загальної кількості рядків (з урахуванням сторонніх бібліотек) для десяти довільно нами обраних дрібних та середніх проектів, виконаних з використанням сучасного інструментарію та високорівневих мов програмування.

Таблиця 2.1

Порівняльний аналіз кількості рядків основного коду по відношенню до загальної кількості рядків деяких проектів

Номер проекту	Загальна кількість рядків програмного коду проекту, з урахуванням коду допоміжних бібліотек	Кількість рядків основного коду	Частка рядків основного коду у загальній кількості рядків проекту, %
1	283209	3015	1,06
2	192376	2725	1,42
3	165810	2617	1,58
4	126204	2352	1,86
5	315743	10931	3,46
6	126746	5289	4,17
7	144600	6434	4,45
8	128625	6524	5,07
9	295129	20955	7,10
10	500298	50367	10,07

Відповідно до даних табл. 2.1 можна зробити висновок, що для переважної більшості дрібних та середніх проектів частка кількості рядків основного коду в загальній кількості рядків проекту з урахуванням сторонніх бібліотек становить приблизно від 1% до 10%, а це означає, що частка кількості рядків допоміжних бібліотек становить відповідно від 99% до 90% у загальній кількості рядків проекту.

Подібні показники є характерними для сучасних проектів, спрямованих на вирішення конкретних задач кінцевих користувачів, тобто найтиповіших для більшості компаній-розробників програмного забезпечення.

На нашу думку, подібна тенденція характеризує сам підхід до сучасного процесу розробки програмного забезпечення, що корінним чином відрізняється від того, який використовувався у минулому: в сучасних умовах економічно недоцільно розробляти самостійно всі складові програмного продукту власними силами, на противагу цьому підходу більш раціональним з погляду витрат часу, матеріальних та інших ресурсів є підхід, коли в ході роботи над проектом підбираються спеціалізовані бібліотеки компонентів стороннього виробництва, а основний код програми являє фактично оболонку, що використовує функціональність даних бібліотек для вирішення поставлених задач.

Підвищення значущості сторонніх бібліотек компонентів при розробці програмного забезпечення вимагає і підвищити увагу до тих процесів, що супроводжують використання сторонніх бібліотек, зокрема, процес обновлення їх версій.

Зібрана нами статистика щодо випуску нових версій для 12 довільно обраних популярних бібліотек коду, які використовуються розробниками на високорівневій мові програмування Borland Delphi, за період в один календарний рік (з 01.06.2003 р. по 01.06.2004 р.) представлена в табл. 2.2.

Відповідно до даних табл. 2.2, можна зробити висновок, що випуск нових версій для сторонніх бібліотек коду відбувається досить часто. Серед розглянутих бібліотек коду мінімальна кількість випусків нових версій становила три за рік, в той час як максимальна – двадцять дев’ять. Середнє значення в місяць на рівні 1,06 свідчить, що нові версії бібліотек коду в середньому виходять не рідше, ніж один раз на місяць.

В тому, що бібліотеки коду постійно обновляються, немає нічого негативного, скоріше навпаки: це свідчить про постійну роботу їх розробників над вдосконаленням бібліотек, підвищенням їх стабільності та функціональності. Саме для найпопулярніших бібліотек практика частих випусків нових версій є досить поширеною.

Таблиця 2.2

Вихід нових версій бібліотек коду, що використовуються розробниками на Borland Delphi за період з 01.06.2003 р. по 01.06.2004 р.

Назва бібліотеки коду	**Кількість нових версій**	
	всього за період	**в середньому в місяць**
Absolute Database	17	1,42
BusinessSkinForm	29	2,42
EurekaLog	12	1
FastReport VCL	3	0,25
FIBPlus	5	0,42
IBPhoenix OpenSource ODBC Driver	20	1,67
IntraWeb	19	1,58
KOL & MCK	17	1,42
PGP Components	3	0,25
SQLDirect	8	0,67
TPlanner	4	0,33
Unified Interbase	16	1,33
Середнє значення	**12,75**	**1,06**

Однак з погляду розробників програмного забезпечення постійний випуск нових версій використовуваних компонентів та інструментарію несе за собою наступні актуальні проблеми:

- по-перше, процес стеження за новими версіями потребує значних витрат часу, він вимагає періодичного ознайомлення з анонсами компаній-розробників бібліотек компонент та інструментарію, стрічками новин спеціалізованих сайтів та списків розсилки;
- по-друге, після того, як було помічено вихід нової версії сторонньої бібліотеки чи інструментарію, її необхідно отримати, що може бути пов’язане не лише зі значними витратами часу (посилання запиту на отримання нової версії, очікування ключів, доставка та ін.), а й з суттєвими матеріальними витратами (придбання нової версії, завантаження дорогими каналами зв’язку, пересилка поштою чи кур’єрською службою зі сплатою мита та митних зборів залежно від вимог законодавства);

- по-третє, чи не найскладнішою проблемою є процес переходу на нову версію, що пов’язаний з надзвичайно трудомісткими процесами забезпечення сумісності розробленого коду проекту з новими версіями бібліотек чи інструментарію, а також здійснення регресивного тестування проекту на предмет коректної поведінки з урахуванням поновлення версій бібліотек та інструментарію;
- по-четверте, необхідно забезпечувати сумісність для «унаслідуваних» проектів, тобто тих, для яких не відбувається розширення функціональності, а забезпечується лише супровід з виправленням виявлених помилок; для цих проектів досить часто взагалі здійснювати перехід на нові версії бібліотек та інструментарію робити економічно недоцільно, а тому розробники вимушені витрачати додатковий час на перехід з нових версій бібліотек та інструментарію на попередні для обслуговування даних проектів, і повернення на актуальні версії для роботи з основними проектами;
- по-п’яте, досить часто виявляється, що нова версія містить критичні помилки які роблять її використання неможливим до того часу, доки вони не будуть виправлені її розробником, у даному разі слід здійснити процес повернення на попередню версію, який нерідко виявляється нетривіальним;
- по-шосте, якщо компанія-розробник програмного забезпечення однозначно прийняла рішення про поновлення версій використовуваних бібліотек та інструментарію, то слід здійснити даний процес дуже відповідально, виключивши випадки, коли деякі з розробників на робочих місцях мають різні версії сторонніх бібліотек чи інструментарію, в противному разі це може привести до помилок у проекті, джерело яких надзвичайно складно виявити.

Наведений перелік не є вичерпним, однак лише вказаних проблем достатньо, для того, щоб зрозуміти, що до такого, інноваційного за своєю природою процесу, як перехід на нові версії використовуваного інструментарію та бібліотек коду, слід відноситися з такою ж відповідальністю, як і до будь-якого іншого виду інноваційної діяльності.

Однак, зважаючи на свою відносну новизну, проблема управління впровадженням нових версій бібліотек та інструментарію поки що не має добре відомих прикладів успішного і ефективного вирішення, а тому компанії-розробники програмного забезпечення вимушені самостійно, на свій страх і ризик вживати заходи, які насправді дуже часто негативним чином впливають на успішність проектів з розробки програмного забезпечення.

Серед таких, на нашу думку, надзвичайно пагубних рішень слід виділити повну або часткову відмову від поновлення версій сторонніх бібліотек та інструментарію, що використовується. Якщо компанія обрала подібний шлях, то вона свідомо йде на сумнівний компроміс: незручності з відмовою від проблем з поновленням версій компенсується економією часу та ресурсів на проведення подібних поновлень.

Проте на практиці виявляється, що подібне рішення найчастіше приймається адміністративно і дуже часто суперечить точці зору багатьох розробників, що працюють над проектом і вимушені безпосередньо стикатися з різноманітними проблемами попередніх версій, а тому природно зацікавлених у їх поновленні. Адміністративний тиск, спрямований лише на забезпечення виконання проекту в строк, без врахування актуальних для розробників проблем, однією з головних серед яких є забезпечення потреби працювати лише з сучасним інструментарієм та досконалими бібліотеками компонентів, деморалізує роботу колективу, призводить до виникнення конфліктів і в кінцевому випадку негативно позначається на успіху проекту.

Іноді ініціатива про відмову від поновлень версій йде безпосередньо від розробників, що працюють над проектом. У більшості випадків це свідчить про низький рівень організації праці у такому колективі, відсутність зацікавленості учасників проекту у кінцевому результаті, їх відмову від сприйняття інновацій, що взагалі неприпустимо для галузі розробки програмного забезпечення. Проект, що розробляється з такої позиції не може бути успішним.

У будь-якому разі відмова від використання сучасного інструментарію та бібліотек коду призводить до зниження якості кінцевого продукту, його низької конкурентноздатності. А тому подібний шлях, на нашу думку, є однозначно неприйнятним.

На нашу думку, управління впливом сторонніх компонентів на процес розробки ПЗ має здійснюватися у рамках єдиної системи управління інноваціями, що має бути органічно інтегрованою у систему управління конфігураціями та змінами.

На рис. 2.7 представлено схематичне зображення основних складових системи управління впливом сторонніх компонентів на процес розробки ПЗ.

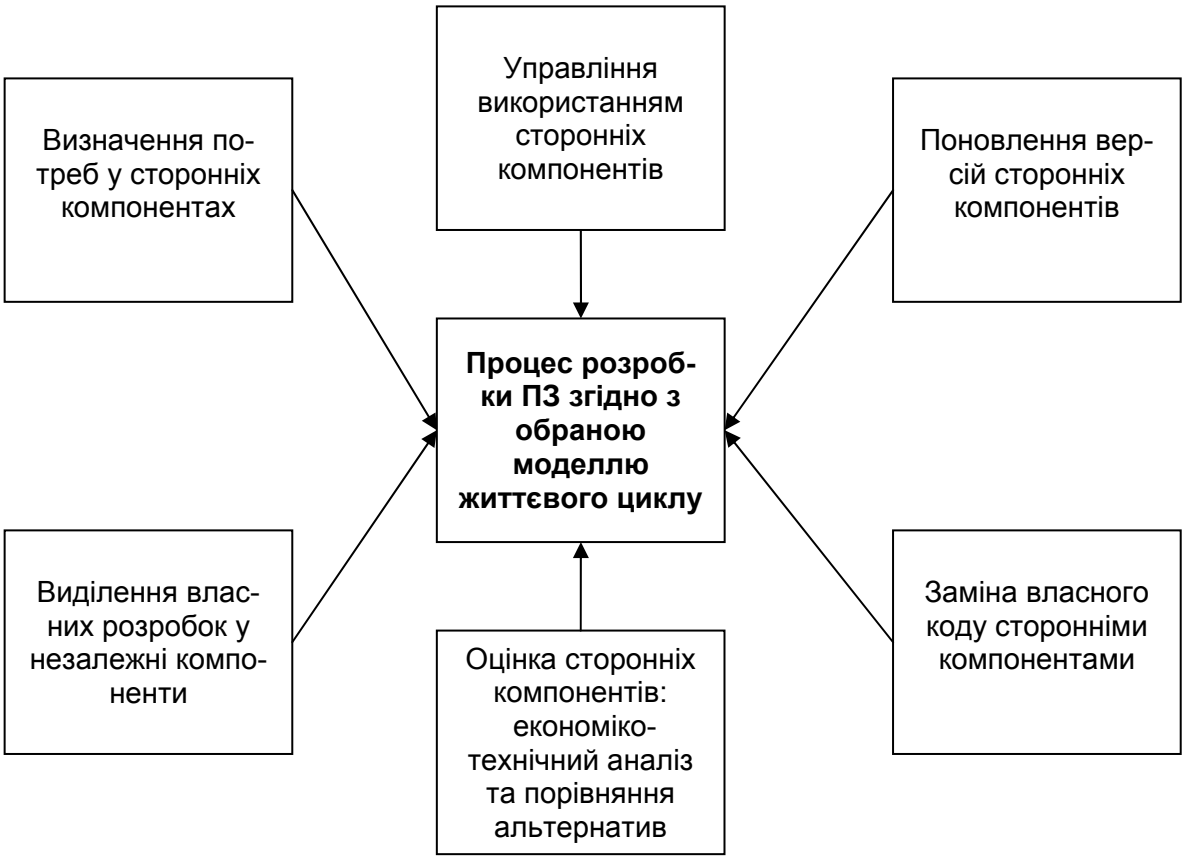


Рис. 2.7. Основні складові системи управління впливом сторонніх компонентів на процес розробки ПЗ

Відповідно до рис. 2.7, на нашу думку, доцільно виділити такі складові системи управління впливом сторонніх компонентів на процес розробки ПЗ:

– управління використанням сторонніх компонентів – має передбачати комплекс заходів, спрямованих на популяризацію використання і забезпечення ефективності використання сторонніх компонентів, що може бути здійснено як за рахунок навчання персоналу, ознайомлення його зі структурою та особливостями використання бібліотек коду, так і за рахунок певних орга-

нізаційних заходів, що обмежують створення розробниками власних рішень, якщо для цього можна використати готові бібліотеки коду. Подібним заходам має бути приділена особлива увага, оскільки розробники як по причині недостатньої інформованості стосовно особливостей використання готових бібліотек коду, так і по причині недостатньої дисциплінованості, намагаються уникати готових рішень, розроблюючи власні. Звичайно подібна практика призводить до збільшення витрат на розробку і супровід продукту, оскільки необґрунтовано зростає його розмір. Крім того, самостійна розробка рішення замість використання готового коду звичайно призводить до необхідності більше ресурсів витрачати на тестування подібного рішення у порівнянні з уже розробленим і, відповідно, перевіреним;

– визначення потреб у сторонніх компонентах – у процесі розробки програмного продукту, особливо на ранніх етапах життєвого циклу, необхідно періодично здійснювати ідентифікацію потреб у сторонніх компонентах та бібліотеках готового коду. Своєчасне виявлення можливості для відмови від власних розробок на користь уже готових компонентів може суттєво скоротити як вартість, так і тривалість розробки, а також у більшості випадків може привести до підвищення якості готового продукту;
– виділення власних розробок у незалежні компоненти – задача виділення власних розробок у незалежні компоненти має розглядатися як з позицій підвищення долі повторного використання програмного коду, так і з погляду виявлення нових джерел доходів за рахунок розповсюдження створених бібліотек коду як окремих продуктів. Обидва напрямки дозволяють підвищити дохідність капіталовкладень у розробку проекту, у першому випадку це здійснюється за рахунок скорочення витрат на розробку за рахунок повторного використання створеного коду, у другому випадку – за рахунок зростання обсягу грошових потоків в результаті реалізації проекту;
– оцінка сторонніх компонентів: економіко-технічний аналіз та порівняння альтернатив – оцінку сторонніх компонентів та здійснення порівняння з альтернативними варіантами необхідно провадити надзвичайно відповіда-

льно з урахуванням значної кількості важливих факторів як технічного, так і економічного характеру. Крім того, в процесі здійснення оцінки необхідно здійснювати юридичний аналіз ліцензій, на умовах яких пропонуються до використання сторонні бібліотеки коду, а також враховувати такі фактори, які складно піддаються кількісній оцінці, наприклад, перспективність технологій, рівень сприйняття бібліотек персоналом та ін.;

– заміна власного коду сторонніми компонентами – в процесі розробки необхідно періодично перевіряти створених програмний код на предмет доцільності його заміни сторонніми компонентами. Інспекція програмного коду з цією метою має здійснюватися постійно, оскільки готові бібліотеки стороннього коду могли з'явитися на ринку пізніше проведення початкового проектування програмного продукту і, відповідно, не були враховані;
– поновлення версій сторонніх компонентів – необхідно забезпечити прозорий і безперервний процес здійснення оцінки доцільності і переходу на нові версії сторонніх бібліотек компонентів. Слід відзначити, що сторонні компоненти і бібліотеки коду розвиваються незалежно від ходу проекту компанії-розробника, і у процесі розвитку можуть набувати нових якісних властивостей, які позивним чином здатні вплинути на характеристики продукту, що розробляється з їх використанням. Подібний розвиток звичайно здійснюється без суттєвих витрат зі сторони розробника – звичайно нові версії можна отримати безкоштовно, чи зі значними знижками, що, в свою чергу, є вагомим джерелом для підвищення дохідності капіталовкладень у процес розробки ПЗ.

Слід зазначити, що збільшуючи кількість сторонніх компонентів і бібліотек коду, які використовуються у проекті з розробки програмного забезпечення, компанія-розробник в певній мірі ставить себе в залежність від постачальника сторонніх компонентів, що, в свою чергу, може привести до зростання ризиків проекту.

Однак, на наш погляд, за умови виваженого підходу до підбору сторонніх компонентів, можна не лише уникнути невиправданого зростання сумарних ризиків проекту, а і досягти їх зменшення.

Серед найбільш поширених ризиків, які впливають на програмні проекті, прийнято виділяти наступні:

– технічні ризики;
– операційні ризики;
– політичні ризики;
– юридичні ризики;
– управлінські ризики;
– ринкові ризики;
– соціальні ризики;
– внутрішні ризики;
– зовнішні ризики [74].

Сторонні компоненти впливають на такі ризики: технічні, юридичні, внутрішні та зовнішні.

Рівень технічного ризику залежить від використання сторонніх компонентів, оскільки їх можливості можуть бути недостатні для вирішення поставленої технічної задачі. З іншого боку, за умов адекватного підбору сторонніх компонентів, можна досягти зменшення сумарної величини технічного ризику за рахунок відмови від самостійного вирішення задачі і використання вже готового стороннього ризику.

Юридичні ризики при використанні сторонніх компонент звичайно проявляються у можливостях порушення ліцензійних угод на них, що може тягти за собою певні негативні юридичні наслідки. На нашу думку, компанія-розробник ПЗ у змозі добитися мінімізації юридичних ризиків за рахунок уважного і кваліфікованого вивчення ліцензій, на умовах яких розповсюджуються сторонні компоненти, і прийняття лише тих із з них, що не суперечать основним принципам організації роботи компанії.

Також можуть виникнути юридичні ризики, пов’язані з порушенням прав на інтелектуальну власність розробників сторонніх компонентів, уникнути яких можна за рахунок проведення роз’яснювальної роботи з персоналом щодо правильності і допустимих способів використання сторонніх компонентів, а також відповідного управлінського контролю.

Однак, з іншого боку, за рахунок використання інтелектуальної власності, що вже захищена певними угодами, компанія-розробник ПЗ може зменшити юридичні ризики, які можуть виникнути внаслідок вирішення питань інтелектуальної власності з власним персоналом, який може претендувати у приватному порядку на окремі технічні рішення.

Внутрішні та зовнішні ризики в процесі використання сторонніх компонент, як правило, взаємопов'язані. Звичайно, при відмові від використання власних розробок і переході на зовнішні компоненти зменшуються внутрішні ризики і навпаки.

Важлива задача, яка має бути вирішена в процесі підбору сторонніх компонент, полягає в тому, щоб знайти компроміс між здешевленням вартості розробки програмного продукту і залежністю від використання сторонніх компонентів.

Залежність від використання сторонніх компонентів може проявлятися наступним чином:

- сторонні компоненти можуть втрачати конкурентоспроможність у зв'язку з появою більш прогресивних аналогів, що може вплинути на конкурентоспроможність програмного продукту, який їх використовує;
- технічний розвиток сторонніх компонентів може відбуватися в такому напрямку, що не відповідає, або вступає в протиріччя із потребами компанії-розробника ПЗ, яка використовує дані компоненти у власних продуктах;
- компанія-постачальник сторонніх компонентів може відмовитися взагалі від розвитку компонент, чи змінити умови ліцензії у наступних версіях таким чином, що це робить недоцільним їх використання у попередній ролі (наприклад, може суттєво зрости вартість компонентів, чи можуть з'явитися вимоги сплачувати ліцензійні відрахування з кожної реалізованої копії програмного продукту, що може стати зробити подальше їх використання економічно недоцільним);
- якість сторонніх компонентів може не відповідати вимогам до якості, які ставляться до створюваного програмного продукту;
- можливість супроводу сторонніх компонентів може не відповідати тим потребам, які існують у компанії-розробника ПЗ.

Щодо конкретних рекомендацій, які можна сформулювати для зменшення впливу вищенаведених факторів, можна запропонувати наступне:

- необхідно віддавати перевагу тим стороннім компонентам, які мають ліберальні ліцензії, а також доступні у вихідних кодах, що дасть змогу вносити зміні до їх коду і займатися супроводом самостійно, не порушуючи при цьому ліцензійних угод;
- можна розглянути можливість придбати всі права на сторонні компоненти, чи отримати ліцензію, що має мінімальні обмеження;
- можна розглянути можливість повного чи часткового придбання фірми-розробника сторонніх компонентів, для того, щоб становити безпосередній вплив на її рішення, якщо буде доведена економічна доцільність подібного заходу;
- якщо виникне необхідність, то можна розглянути варіант пошуку додаткових угод між компанією-постачальником сторонніх компонент, за якими можна отримувати інші умови використання компонент, що відрізняються від, які використовуються на загальних підставах;
- при розробці технічних рішень важливо додержуватися підходу, що надасть можливість здійснити перехід на альтернативні рішення з мінімальними витратами ресурсів і часу.

Відповідно, за умов виваженого підходу до підбору і використання сторонніх компонент, можна не лише уникнути невиправданого зростання ризиків проекту, а також і домогтися їх зменшення.

Таким чином, можна отримати наступні висновки:

- по-перше, у зв’язку зі зростанням складності розробки сучасних програмних продуктів використання сторонніх компонент може слугувати одним із найважливіших напрямків, що дають змогу скоротити вартість і тривалість розробки програмних продуктів;
- по-друге, в результаті проведеного дослідження встановлено, що тенденція до зростання долі сторонніх компонент у загальному обсязі коду програм-

них проектів є сталою, і сягає для проектів, призначених для використання кінцевими користувачами, до 99% у загальній кількості строчок проекту;

– по-третє, використання сторонніх компонент може бути одним із ефективних джерел інновацій з порівняно невисоким рівнем ризику і собівартості для програмних проектів компанії-розробника ПЗ;
– по-четверте, при прийнятті рішення про доцільність використання сторонніх компонентів необхідно керуватися значною кількістю факторів різного характеру, значна кількість яких складно піддається кількісній оцінці.

2.3. Розробка системи економіко-технічних показників для оцінки управління інноваціями на етапах життєвого циклу програмного забезпечення

Слід відзначити, що необхідність постійного поліпшення процесу за рахунок впровадження інновацій є стратегічною метою компанії, досягнення якої вимагає певних зусиль і витрат ресурсів.

За умови неефективного менеджменту в компанії, короткострокові і довгострокові цілі можуть вступати в протиріччя, оскільки потребують для свого досягнення залучення ресурсів, які у більшості випадків є обмеженими.

Протиріччя між довгостроковими і короткостроковими цілями також може призводити до виникнення конфліктних ситуацій між різними рівнями управління, а також між різними функціональними підрозділами компанії.

Якщо система управління інноваціями не буде побудована таким чином, щоб усунути протиріччя між короткостроковими і довгостроковими цілями фірми, чи суттєво зменшити його, то така система не зможе бути ефективною, чи взагалі дієздатною.

Усунути протиріччя довгострокових цілей впровадження інновацій і оперативних задач з ефективного створення програмних продуктів, на наш погляд, можна за рахунок використання підходів, що встановлять чіткий і однозначний взаємозв’язок між стратегічними цілями компанії і її оперативними задачами.

Одним з підходів, що допомагає встановити такий зв'язок, є система збалансованих показників ефективності, яка була розроблена американськими вченими Р. Капланом і Д. Нортоном [29].

Збалансована система показників ефективності надає схему для перекладу загальної стратегії компанії в терміни операційного процесу за допомогою чотирьох складових:

- фінансова складова – фінансові показники оцінюють економічні наслідки прийнятих рішень і є індикаторами їх відповідності стратегії компанії. Фінансові показники віддзеркалюють фінансові цілі компанії і вимірюються, наприклад, операційним прибутком, доходністю залученого капіталу, доданою вартістю;
- клієнтська складова – розглядається як споживацька база та сегмент ринку, в яких конкурує підприємство, а також як показники результатів його діяльності в цільовому сегменті ринку. До цієї групи показників відносяться задоволеність потреб клієнтів, збереження споживацької бази, залучення нових клієнтів, доходність, обсяг та доля цільового сегменту ринку, а також такі фактори, які становлять цінність для клієнта з погляду збереження його лояльності: своєчасна доставка, постійний потік нових товарів та послуг, здатність постачальника повною мірою задовольняти потреби клієнтів;
- складова внутрішніх бізнес-процесів – визначає найголовніші внутрішні бізнес-процеси компанії, які необхідно вдосконалювати з метою розробки таких бізнес-пропозицій клієнтам, які зможуть зберегти і розширити клієнтську базу, а також задовольнити очікування акціонерів відносно високої фінансової доходності компанії. Показники цього напрямку зосереджені на оцінці внутрішніх процесів, від яких у значній мірі залежить задоволення потреб клієнтів та досягнення фінансових задач компанії в цілому;
- складова навчання і розвитку персоналу – визначає інфраструктуру, яку необхідно створити, для того, щоб забезпечити довготривалий ріст і вдосконалення. Розглядаються три головних джерела: люди, системи та органі-

заційні процедури, для їх оцінки використовуються показники, що являють собою комплекс загальних критеріїв (задоволеність роботою, плинність кадрів, навчання, підвищення кваліфікації), а також специфічних факторів, таких як детальний, характерний для предметної діяльності набір знань, навичок і вмінь, що потрібні для підтримки конкурентноздатності компанії [29].

Виходячи з основних тверджень системи збалансованих показників ефективності, у якості орієнтирів для прийняття рішень мають використовуватися не лише фінансові показники, а й інші складові сучасного бізнесу, в тому числі й такі, що орієнтовані на його специфіку.

Стратегічною ціллю в сфері інноваційного управління на етапах життєвого циклу програмного забезпечення має бути постійне вдосконалення якісних характеристик продуктів, що розробляються, а також технологій, які використовуються.

Забезпечити ефективне управління інноваціями можна за допомогою системи економіко-технічних показників, які відображають стан інноваційних процесів при виконанні програмних проектів компанією-розробником ПЗ.

В високотехнологічній компанії Hewlett-Packard, яка займається розробкою одночасно програмного і апаратного забезпечення, для оцінки ефективності циклу розробки продукту використовується показник періоду окупності – BET (Break-Even Time, період окупності). Цей показник визначається як час від процесу розробки нового продукту до того моменту, коли був отриманий прибуток, достатній для повернення інвестицій [119].

Показник BET включає три елементи, що характеризують і орієнтують на ефективність процесу розробки нових продуктів:

– по-перше, показник орієнтує на здійснення ефективних капіталовкладень у дослідження і розробки, оскільки витрати безпосередньо зіставляються з доходами;
– по-друге, підкреслюється значення фактора прибутковості, що заохочує до співробітництва усіх учасників, що працюють над одним продуктом – від дослідників, до менеджерів, що займаються питаннями продаж;

– по-третє, одиницею виміру для показника є час, покращення значення показника сприяє прискоренню процесу розробки нових продуктів у порівнянні з конкурентами, забезпечуючи цим самим високий рівень продаж та швидку окупність інвестицій.

Показник BET призначений, насамперед, для оцінки ефективності процесу розробки нових продуктів і технологій, а не для визначення кінцевих результатів діяльності. У такому ракурсі він є ефективним інструментом для визначення тактики діяльності компанії, вдосконалення вже існуючих технологій компанії [29].

На наш погляд, використання показника BET доцільно при реалізації програмних проектів, особливо на етапах планування діяльності, оскільки дозволяє визначити строк повернення капіталовкладень на здійснення науково-дослідницьких робіт.

Рекомендується використовувати показник тривалості життєвого циклу продуктів компанії. Слід зазначити, що продукти з тривалим життєвим циклом можна розглядати як більш інноваційні продукти у порівнянні з тими, для яких життєвий цикл є короткостроковим.

На нашу думку, при розробці тактичних рішень, що основані на фінансових показниках і мають на меті вирішення фінансових задач, компанія-розробник ПЗ має враховувати джерела доходів.

Для компанії-розробника ПЗ існує два основних джерела доходів: від продажу нових копій продуктів і супроводу встановлених копій продуктів. Незалежно від тенденцій і характеру загальної динаміки доходів, важливо враховувати динаміку співвідношення між вказаними потоками.

Для визначення характеру структури грошових надходжень компанії рекомендується використовувати формулу:

$$K_{ПН} = \frac{H_H}{H_H + H_C}, \tag{2.2}$$

де $K_{ПН}$ – коефіцієнт перспективності грошових надходжень, $0 \le K_{пн} \le 1$;

H_H – надходження від реалізації нових копій продуктів, грош. од.;

H_C – надходження від супроводу існуючих копій продуктів, грош. од.

Показник $К_{ПН}$, розрахований станом на певну дату, показує, наскільки перспективними є доходи компанії станом на цю дату, тобто, чи вказують вони на тенденцію до зростання у майбутньому.

Особливого значення набуває динаміка показника $К_{ПН}$ – якщо компанія успішно працює на ринку, то це означає, що для неї одночасно зростають надходження як від реалізації нових копій продуктів ($Н_Н$), так і від супроводу існуючих копій продуктів ($Н_С$). В такій ситуації значення показника $К_{ПН}$ є сталим, або змінюється несуттєво.

Зростання значення показника $К_{ПН}$ одночасно із збільшенням доходів компанії свідчить про її успіх на ринку і має сприйматися позитивно.

Однак зменшення показника $К_{ПН}$ навіть в той час, як загальні доходи зростають, свідчить про зменшення перспективності надходжень і до появи підстав до їх зменшення у майбутньому.

Дане твердження пояснюється зміною структури доходів компанії – доходи від супроводу ПЗ слід сприймати як неосновні, оскільки клієнти не схильні оплачувати супровід ПЗ на невизначено тривалий термін. Звичайно клієнти намагаються оплачувати супровід на термін, що не перевищує 1-3 роки, а потім займатися супроводом власними силами, підготувавши для цього власних фахівців.

Крім того, зменшення надходжень від реалізації нових копій продуктів свідчить про втрату продуктами компанії конкурентноздатності на ринку. Якщо показник $К_{ПН}$ має сталу тенденцію до зменшення, то це означає, що незалежно від того, чи відбувається ріст доходів у даний час, вони обов’язково почнуть зменшуватися у недалекому майбутньому, оскільки частина клієнтів, що приносить доход від супроводу, відмовиться від стороннього супроводу і почне здійснювати його власними силами, а інша частина – перейде на альтернативні рішення, оскільки вони виявляються більш конкурентноздатними на ринку.

Оптимальне значення показника $К_{ПН}$ може залежати як від специфіки ринку, на якому працює компанія (для різних ринків існує різне співвідношення між ціною продукту і вартістю його супроводу, що звичайно визначається конкурентною ситуацією), а також стратегічних цілей і пріоритетів компанії (компанія може встанов-

лювати порівняно низьку початкову ціну, для того, щоб розширити коло клієнтів і збільшити доходи від супроводу, чи, навпаки, бути мінімально зацікавленою у грошових надходженнях від супроводу, орієнтуючись лише на надходження від реалізації нових копій).

На наш погляд, для більшості компаній з розробки ПЗ, що зацікавлені як у грошових надходженнях від реалізації нових копій, та і від супроводу, оптимальне значення для $К_{ПН}$ буде знаходитися в межах 0,7-0,9. Значне наближення $К_{ПН}$ до 1 слід також розглядати як негативне явище, оскільки в даному разі компанія втрачає важливі джерела доходів, які становить супровід ПЗ і є вразливою до зменшення доходів від продажу нових копій, що може бути викликане різними причинами.

Іншим важливим показником структури доходів, що відображає ефективність інноваційного менеджменту компанії-розробника ПЗ є показник, який відображає співвідношення між різними версіями ПЗ у структурі доходів компанії.

Ефективна ринкова стратегія компанії-розробника ПЗ полягає у постійному оновленні версій програмних продуктів, які вона пропонує ринку, а також забезпечення стабільного попиту на нові версії продуктів.

Нова версія, як суттєве оновлення продуктового ряду компанії, є результатом її науково-технічних розробок і відображає, наскільки ефективною є політика компанії по розробці і виведенні нового продукту на ринок.

При випуску нової версії компанія зацікавлена у масовому переході користувачів на неї з декількох причин:

- перехід на нову версію користувачів попередніх версій є джерелом доходів, для розвитку якого звичайно не потрібні значні інвестиції у просування продукту на ринок;
- відмова від використання користувачами попередніх версій продуктів за рахунок переходу на нові дозволяє компанії-розробнику ПЗ скоротити витрати на супровід різних версій продуктів;
- у разі роботи на конкурентному і насиченому ринку перехід на нові версії програмних продуктів є найменш ризиковим і найбільш привабливим джерелом доходів.

Відмова клієнтів від використання старих версій продуктів на користь нових свідчить про успіх інноваційної політики компанії-розробника ПЗ. В той же час, якщо користувачі не бажають оновлювати встановлені копії програмних продуктів, а також існує попит на постачання нових копій попередніх версій, то це означає, що компанія-розробник ПЗ не змогла запропонувати ринку продукт, який відповідає його очікуванням, а її інноваційна політика виявилася неефективною.

Для визначення успішності нової версії програмного продукту рекомендується використовувати наступний коефіцієнт:

$$K_{УНВ} = \frac{H_{ВК} + H_{ВО}}{H_{Н}}, \tag{2.3}$$

де $K_{УНВ}$ – коефіцієнт успішності нової версії програмного продукту, $0 \le К_{унв} \le 1$;

$H_{ВК}$ – надходження від реалізації нових копій нової версії програмного продукту, грош. од.;

$H_{ВО}$ – надходження від оновлення попередніх версій новими версіями програмних продуктів, грош. од.;

$H_{Н}$ – надходження від реалізації нових копій продуктів, грош. од.

Компанія-розробник ПЗ має забезпечити зростання значення коефіцієнта $K_{УНВ}$ після виходу нової версії і його наближення до 1. Якщо нова версія програмного продукту виявилася вдалою, то це означає, що вона має повністю замінити попередні версії і значення $K_{УНВ}$ має дорівнювати 1.

В той же час, якщо значення $K_{УНВ}$ не має тенденцій до швидкого наближення до 1 чи стабілізувалося на певній величині і не зростає, це означає, що нова версія програмного продукту виявилася невдалою, вона не сприймається позитивно ринком, а ситуація, що склалася, вимагає негайної реакції зі сторони менеджменту фірми, оскільки компанія може втратити ринкові позиції.

Іншим важливим показником, який, на наш погляд, доцільно використовувати при здійсненні управління інноваціями для компаній-розробників ПЗ, є показник актуальності середовища розробки.

Актуальність середовища розробки – відповідність версій інструментальних засобів, бібліотек коду, компонентів та ін. найсучаснішим розробкам, які доступні на ринку.

В результаті розглядання у підрозділі 2.2 питання управління сторонніми компонентами, був отриманий висновок, що компанія-розробник ПЗ має організувати свою діяльність таким чином, щоб постійно оновлювати середовище розробки, підтримуючи його в актуальному стані.

На нашу думку, для визначення актуальності середовища розробки доцільно використати такий показник:

$$B_C = \frac{\sum_{i=1}^{n} \Delta T_i}{n}, \tag{2.4}$$

де B_C – показник «відставання» середовища розробки, років;

n – кількість компонентів, які формують середовище розробки;

ΔT_i – «відставання» у часі і-го компонента, який використовується при розробці ПЗ від його актуальної версії, років. ΔT слід розраховувати як різницю між датою виходу актуальної версії компонента і датою виходу тієї версії, яка використовуються при розробці ПЗ, виражену в роках.

Значення для показника B_C, розраховане на певний момент часу, буде відображати, на скільки років в середньому середовище розробки «відстає» від актуальних версій інструментарію і компонентів, які представлені на ринку в даний момент часу.

Компанія-розробник ПЗ має намагатися мінімізувати значення B_C, оскільки відставання від актуальних версій складових середовища розробки є фактором, що негативно впливає на конкурентноздатність компанії.

На наш погляд, значення показника B_C для компанії-розробника ПЗ має не перевищувати одного року, в іншому разі можна вважати, що середовище розробки не є конкурентноздатним.

Також слід зазначити, що вдосконалення значення показника B_C не слід розглядати як самоціль, оскільки постійне оновлення версій інструментальних засобів і компонентів може відволікати значні ресурси компанії і перешкоджати здійсненню її основної діяльності.

З урахуванням відносно високого рівня використання сторонніх компонент при розробці програмних продуктів пропонується використовувати такий показник, як коефіцієнт залежності програмного продукту від сторонніх компонентів. Розраховувати його пропонується таким чином:

$$K_{ЗК} = \frac{K_{СК}}{З_{КП}}, \tag{2.5}$$

де $K_{ЗК}$ – коефіцієнт залежності продукту від сторонніх компонентів, $0 \le K_{ЗК} \le 1$;

$K_{СК}$ – розмір коду сторонніх компонентів, одиниць розміру коду;

$З_{КП}$ – загальний розмір коду програмного продукту, одиниць розміру коду.

Чим вищим є значення показника $K_{ЗК}$, розраховане відповідно до формули (2.5), тим більший вплив на кінцевий результат програмного проекту становить сторонній код, і тим вищою є цінність ефективного інноваційного менеджменту для досягнення поставлених результатів.

Економіко-технічні показники і основні економічні задачі мають визначатися з урахуванням моделі і етапу життєвого циклу, на якому знаходиться проект. В табл. 2.3 представлено основні економічні задачі відповідно до стану програмного проекту.

Відповідно до методик розрахунку вартості реалізації проекту, які були нами розглянуті у підрозділі 1.3, доцільно здійснювати розрахунок бюджету проекту як відповідно до стану (табл. 2.3), так і відповідно до етапу життєвого циклу, який залежить від обраної моделі життєвого циклу.

На наш погляд, показники, які використовуються в сфері інноваційного менеджменту компанією-розробником ПЗ, необхідно обирати, виходячи з етапу життєвого циклу, на якому знаходиться проект з розробки ПЗ.

Таблиця 2.3

Основні економічні задачі відповідно до стану програмного проекту [88]

Стан програмного проекту	Основні економічні задачі
Проект ще не розпочався	Розрахунок пропозицій клієнтів, зацікавлених у придбанні ПЗ; оцінка витрат на розробку програмного проекту; розрахунок бюджету на розробку програмного проекту і визначення основних його віх.
Проект виконується	Визначення витрат на реалізацію проекту і коригування бюджету проекту; Контроль за дотриманням бюджету проекту.
Проект закінчено	Аналіз витрат на реалізацію проекту і відхилень від запланованих показників для: – покращення реалізації майбутніх проектів; – визначення відповідності бюджету проекту.

Відповідно до визначених у підрозділі 1.1 узагальнених етапів розробки ПЗ (проектування, розробка, впровадження та супровід) розглянемо економіко-технічні показники, які доцільно використовувати на кожному з них.

На етапі проектування ПЗ важливо визначити ті технології і нововведення, використання яких надасть нові властивості програмному продукту, чи дасть змогу здійснити його створення більш ефективним чином.

З метою оцінки доцільності впровадження інноваційних можна використати традиційні показники для оцінки інвестиційних проектів, такі як чиста дисконтована вартість (*NPV*), індекс прибутковості (*PI*), період окупності (*PP*), внутрішня норма доходності (*IRR*).

Однак під час використання даних показників для оцінки альтернативних проектів з впровадження інновацій у діяльність компанії-розробника ПЗ, на наш погляд, необхідно використовувати коригуючі коефіцієнти, які враховують так звану «неекономічну складову», визначену нами у підрозділі 2.1. Для визначення даних коригуючих коефіцієнтів раціонально використати метод експертної оцінки.

Крім того, важливого значення набуває коректний вибір горизонту планування [5]. Горизонт планування при здійсненні оцінки доцільності впровадження інновацій може перевищувати тривалість реалізації одного проекту з розробки ПЗ, оскільки впровадження нових технологій надасть змогу для їх використання і у наступних проектах.

В цілому, на етапі проектування важливо зосередитися на визначенні вартості реалізації програмного проекту з урахуванням прийнятих вимог і обмежень [58].

На етапі розробки програмного продукту необхідно приділити увагу забезпеченню відповідності запланованих показників ефективності від впровадження інновацій реальним показникам діяльності компанії.

У даному разі доцільно здійснити пряме порівняння витрат і доходів від впровадження інновацій у діяльність компанії з плановими показниками, які використовувалися при прийнятті рішення про доцільність здійснення впровадження інновацій.

Крім того, варто забезпечити контроль за дотриманням намічених планів по впровадженню інновацій, зокрема здійснювати постійний моніторинг середовища розробки згідно з формулою (2.4).

Також доцільно сфокусуватися на своєчасному виявленню і усуненні дії ризиків, які можуть впливати на успіх проекту, таких, як:

- порушення запланованих строків досягнення віх проекту;
- порушення, зміна і зростання вимог;
- високий рівень звільнень серед персоналу, що призводить до низького рівня знань особливостей реалізації проекту серед нових робітників;
- відсутність ентузіазму у колективі при виконанні проекту;
- непередбачувана складність задач під час виконання проекту.

Важливим питанням під час забезпечення виконання проекту має бути бажання і можливості для того, щоб іти на компроміс і давати можливості поступатися певними показниками проекту для того, щоб забезпечити можливість його виконання взагалі [26].

На етапі впровадження ПЗ варто зосередитися на визначенні характеристик отриманого продукту з точки зору його конкурентноздатності і відповідності ринковим очікуванням.

Для вирішення цієї задачі необхідно здійснювати постійний моніторинг доходів компанії відповідно до формул (2.2) та (2.3).

Своєчасне виявлення невідповідності сприйняття ринком нової версії та зміну структури доходів з реалізації нових копій програмних продуктів до їх супроводу надасть можливості для того, щоб здійснити відповідні застережні дії і не втратити ринок у конкурентній боротьбі.

На етапі супроводу ПЗ слід зосередитися на визначенні місця у структурі доходів компанії, яке займає дохід від супроводу ПЗ відповідно до формули (2.2).

З погляду трудовитрат на супровід ПЗ, на наш погляд, важливо враховувати який саме супровід здійснює компанія-розробник ПЗ.

Існують наступні різновиди супроводу ПЗ:

– коригуючий супровід: коригування функціонування, продуктивності, виправлення помилок у функціонуванні ПЗ;
– адаптивний супровід: зміна ПЗ для того, щоб врахувати зміни у середовищі, такі як нове апаратне забезпечення, чи випуск наступної версії операційної системи. Адаптивний супровід не передбачає зміну функціональності ПЗ;
– вдосконалюючий супровід: вдосконалення ПЗ з метою покращення його споживчих характеристик, у тому числі й враховуючи нові вимоги клієнтів, чи зміну раніше визначених вимог [127].

Якщо у структурі трудовитрат компанії велику долю займає коригуючий супровід, то це свідчить про низьку якість ПЗ, оскільки коригуючий супровід передбачає здійснення доопрацювання ПЗ після закінчення основних стадій його розробки.

Таким чином, доцільно використовувати такий коефіцієнт, що відображає долю коригуючого супроводу у загальних трудовитратах на супровід ПЗ:

$$K_{KC} = \frac{K_C}{K_C + A_C + B_C}, \tag{2.6}$$

де K_{KC} – коефіцієнт коригуючого супроводу – відображає долю коригуючого супроводу у загальних трудовитратах на супровід ПЗ, $0 \le K_{KC} \le 1$;

K_C – трудовитрати на коригуючий супровід ПЗ, людино-год.;

A_C – трудовитрати на адаптивний супровід ПЗ, людино-год.;

B_C – трудовитрати на вдосконалюючий супровід ПЗ, людино-год.

Компанія-розробник ПЗ має зосередитися на зменшенні показника K_{KC} за рахунок вдосконалення процесу на етапах проектування та розробки ПЗ. Зниження показника K_{KC} сприятиме вивільненню ресурсів для реалізації нових проектів компанії, а також позитивно вплине на сприйняття розробленого програмного продукту ринком, який матиме більш високі показники якості.

Також важливим показником для оцінки показників інноваційного менеджменту програмних проектів має бути відповідність процесу організації розробки ПЗ рівню зрілості відповідно до моделей CMM/CMMI, розглянутих у підрозділі 1.2.

Відповідно до основних положень CMM/CMMI компанія, яка знаходиться на більш високому рівні зрілості, має більш ефективний процес впровадження інновацій у свою діяльність.

Як свідчать проведені дослідження, дане твердження підтверджується реальними фактами: компанії, які знаходяться на більш високому рівні зрілості, є більш успішними при досягненні поставлених цілей, у тому числі й у питаннях управління інноваціями [127].

Таким чином, в систему економіко-технічних показників, яку доцільно використовувати при здійсненні управління інноваціями для компанії-розробника ПЗ, доцільно включити показники, які використовуються при визначенні відповідності рівня організації процесу розробки рівням зрілості згідно з моделями CMM/CMMI.

Таким чином, в результаті розробки системи економіко-технічних показників для оцінки управління інноваціями на етапах життєвого циклу програмного забезпечення можна отримати наступні висновки.

По-перше, система управління інноваціями має бути побудована таким чином, щоб усунути протиріччя між короткостроковими і довгостроковими цілями фірми, чи суттєво зменшити його за рахунок встановлення чіткої відповідності між цілями стратегічного рівня і задачами оперативного рівня.

По-друге, система економіко-технічних показників має бути побудована з урахуванням стану і етапів життєвого циклу програмного проекту. Кожен етап вимагає фокусування на окремих ланках інноваційного менеджменту компанії.

По-третє, використання загальноприйнятих методик для оцінки інвестиційних проектів, таких як чиста дисконтована вартість, індекс прибутковості, період окупності, внутрішня норма доходності має здійснюватися з урахуванням особливостей інноваційних проектів у галузі ПЗ, зокрема, слід значну увагу приділити вивченню факторів, які можна віднести до «неекономічної складової», а також визначенню горизонту планування.

Висновки до другого розділу

1. Економічний механізм управління інноваціями на етапах життєвого циклу програмного забезпечення має вирішувати такі задачі як пошук та розробку інновацій за рахунок зовнішніх і внутрішніх джерел, здійснювати оцінку доцільності впровадження інновацій з урахуванням економічних і технічних факторів, забезпечувати розробку плану впровадження інновацій, враховуючи можливість відміни впровадження, забезпечувати управління реалізацією інновацій на практиці, здійснювати аналіз процесу і формування висновків для подальшого використання, забезпечувати безперервність процесу і його органічне поєднання із загальною системою управління компанією на усіх рівнях.

2. Основною ідеєю економічного механізму управління інноваціями має бути контрольованість процесу і його відповідність як короткостроковим, так і довгостроковим цілям фірми. Інноваційний процес має бути безперервним і циклічно повторюватися.

3. Інноваційні процеси є джерелом змін до процесу розробки ПЗ, а тому мають розглядатися через призму механізмів, які відповідають за управління змінами.

4. Визначено, що існують обмеження використання суто економічних методів при прийнятті рішень стосовно доцільності впровадження інновацій, причиною яких є дія таких факторів, які складно спрогнозувати, наприклад, перспективність технології у майбутньому.

5. Було запропоновано у підрозділі 2.1 при прийнятті інноваційного рішення приймати до уваги також «неекономічну складову», що має включати такі фактори, як перспективність технології, її потенціал до розвитку, прийняття персоналом фірми та ін.

6. В результаті проведеного у підрозділі 2.2 аналізу було встановлено, що для сучасних програмних проектів характерним є високий рівень складності, який має тенденцію до зростання. Одним з найбільш ефективних способів подолання складності є використання сторонніх компонентів і бібліотек коду, тенденція до зростання долі яких у загальному обсязі коду програмних проектів є сталою, і сягає 99% по відношенню до загального обсягу проекту.

7. Використання сторонніх компонент може бути одним із ефективних джерел інновацій з порівняно невисоким рівнем ризику і собівартості для програмних проектів компанії-розробника ПЗ. Однак при прийнятті рішення про доцільність використання сторонніх компонентів необхідно керуватися значною кількістю факторів різного характеру, значна кількість яких складно піддається кількісній оцінці.

8. Система управління інноваціями має бути побудована таким чином, щоб усунути протиріччя між короткостроковими і довгостроковими цілями фірми, чи суттєво зменшити його за рахунок встановлення чіткої відповідності між цілями стратегічного рівня і задачами оперативного рівня. Досягти виконання подібної задачі можна за допомогою використання таких підходів, як, наприклад, системи збалансованих показників ефективності, яка надає схему для перекладу загальної стратегії компанії в терміни операційного процесу.

9. Система економіко-технічних показників має бути побудована з урахуванням стану і етапів життєвого циклу програмного проекту. Кожен етап вимагає фокусування на окремих ланках інноваційного менеджменту компанії. У підрозділі 2.3. нами було розроблено і запропоновано до використання такі показники як коефіцієнт перспективності грошових надходжень (2.2), коефіцієнт успішності нової версії програмного продукту (2.3), показник «відставання» середовища розробки (2.4), коефіцієнт залежності продукту від сторонніх компонентів (2.5), коефіцієнт коригуючого супроводу (2.6).

10. Було встановлено, що використання загальноприйнятих методик для оцінки інвестиційних проектів, таких як чиста дисконтована вартість, індекс прибутковості, період окупності, внутрішня норма доходності має здійснюватися з урахуванням особливостей інноваційних проектів у галузі ПЗ, зокрема, слід значну увагу приділити вивченню факторів, які можна віднести до «неекономічної складової», а також визначенню горизонту планування.

Основні положення даного розділу знайшли відображення у наступних наукових працях здобувача [7, 30, 38, 39, 40, 78]:

1. Богдановський В. Г., Колдовський В. В. Софтверний фактор у питаннях створення ефективної системи автоматизованого управління бізнес-процесами // Вісник Сумського державного аграрного університету. Серія: „Економіка та менеджмент". Вип. 2. – Суми, 2001. – С. 177-180.
2. Кислий В. М., Колдовський В. В. Проблеми використання новітніх технологій у сфері розробки програмного забезпечення // Тезисы докладов научно-технической конференции преподавателей, сотрудников, аспирантов и студентов экономического факультета Сумского государственного университета. – Сумы: Изд-во СумГу, 2002. – С. 64-65.
3. Колдовський В. В. Окремі аспекти управління інноваційними процесами при розробці ПЗ // Проблеми і перспективи розвитку банківської системи України: Збірник наукових праць. Т. 13. – Суми: ВВП «Мрія-1» ЛТД, УАБС НБУ, 2005. – 270 с., С. 185-198.
4. Колдовський В. В. Створення ефективної системи автоматизованого управління бізнес-процесами підприємства // Збірник матеріалів Першої міжнародної науково-практичної конференції „Сучасні проблеми управління". – К.: „Політехніка", 2001. – С. 291-293.

5. Колдовський В. В., Шамота Г. М. Інтенсифікація інновацій: безперервність процесу і орієнтація на персонал // Вісник Хмельницького національного університету. – 2005. – № 3. – С. 165-168.
6. Ярошенко С. П., Колдовський В. В. Умови ефективної реалізації проектів автоматизації в промисловості // Механізм регулювання економіки, економіка природокористування, економіка підприємства та організація виробництва. – Суми: Вид-во СумДУ. – 2002. – № 1-2. – С. 152-155.

РОЗДІЛ 3
ПРАКТИЧНІ АСПЕКТИ РЕАЛІЗАЦІЇ СИСТЕМИ УПРАВЛІННЯ ІННОВАЦІЯМИ У СФЕРІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Інструменти та методи для управління інноваціями: кортеж та паспорт проекту, стандартизація внутрішніх процесів

В результаті проведеного нами в першому розділі аналізу підходів до управління інноваціями на етапах життєвого циклу ПЗ, а також сформульованих у другому розділі рекомендацій стосовно вдосконалення науково-методологічних підходів до управління інноваціями були визначені висновки, згідно з якими встановлено як важливість побудови ефективної системи управління інноваціями на етапах життєвого циклу ПЗ, так і окреслено основні характеристики складових цієї системи.

З погляду практичної реалізації основних ідей побудови системи управління інноваціями на етапах життєвого циклу ПЗ, на наш погляд, доцільно розглянути набір рекомендацій для вирішення конкретних задач інноваційного менеджменту, які були апробовані у практичній діяльності компаній-розробників ПЗ і зарекомендували себе з позитивної сторони.

Такі рекомендації, які мають вигляд цілісної концепції для вирішення конкретних задач управління інноваціями на етапах життєвого циклу ПЗ, на нашу думку, доцільно називати інструментами, оскільки вони мають визначені характеристики у сфері вирішення поставлених задач, рекомендації по використанню, а також визначену сферу використання.

В питанні управління інноваціями зокрема, і в тому, що стосується управління змінами при розробці ПЗ взагалі, на нашу думку, доцільно запровадити термін «кортеж проекту» по аналогії з розглянутим у підрозділі 2.1 терміном «кортеж програми».

Використання кортежу проекту може бути корисним для вирішення проблеми поновлення версій використовуваного інструментарію та бібліотек коду.

Як було визначено у підрозділі 2.2, крім використання великої кількості сторонніх компонент, робота над сучасним проектом ведеться з використанням значної кількості інструментальних засобів, за допомогою яких розробляється проектна документація, створюється вихідний та машинний код, проводиться тестування і виправлення помилок, готується дистрибутив продукту та ін. Всі ці компоненти становлять безпосередній вплив на процес розробки програмного продукту і на відповідність його вимогам якості та, зокрема, надійності.

Особливого значення кортеж, який становлять ці компоненти, набуває в тому випадку, якщо над проектом працює значна кількість розробників. В даному разі невідповідність версій бібліотек стороннього коду, чи інструментарію, що використовуються розробниками, може призвести до значних складнощів з узгодженням проекту. При збільшенні кількості учасників проекту, актуальність даної проблеми зростає.

Зафіксувати складові кортежу проекту, задокументувати його допустимі значення можна за допомогою відповідного документу, який має бути доступним для всіх учасників проекту і повинен мати силу наказу. У найпростішому випадку даний документ може мати вигляд таблиці, поділеної на розділи, кожен з яких відповідає елементам кортежу, для складних проектів даний документ може бути більш структурованим і розділеним на окремі, відносно незалежні частини.

На нашу думку доцільно назвати подібний документ «паспорт проекту», адже цей термін найбільше відповідає його призначенню – приводити у відповідність власне проект і середовище, в якому він проходить всі стадії розвитку.

На рис. 3.1 представлено графічне зображення запропонованого поняття «кортеж проекту» та зазначено місце документу «паспорт проекту», що визначає його допустимі значення.

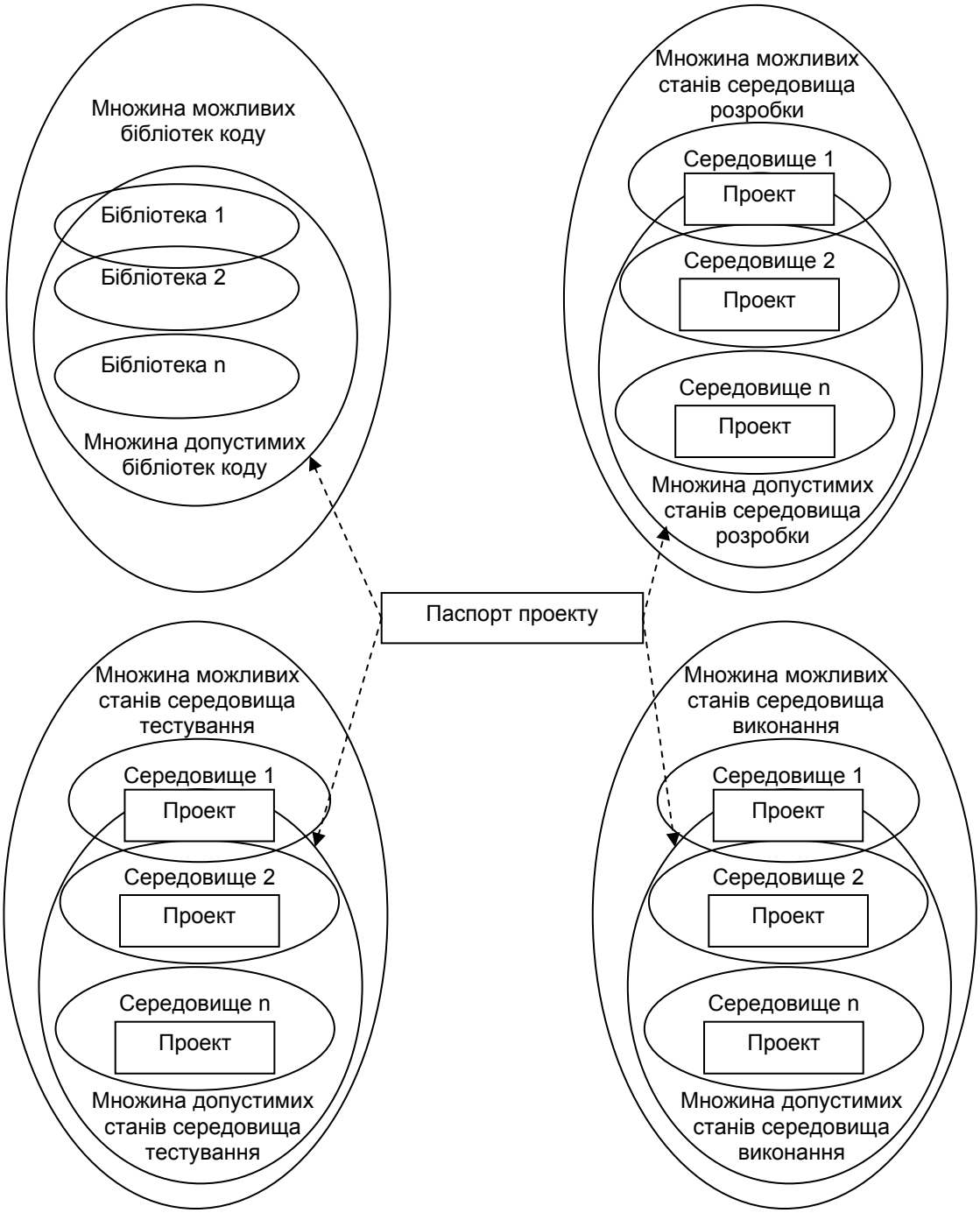


Рис. 3.1. Кортеж проекту

Розглянемо складові кортежу проекту, які визначаються паспортом проекту:

– допустимі бібліотеки коду – наводиться перелік бібліотек коду, які використовуються в проекті з обов'язковим зазначенням їх номерів версій та модифікацій, якщо такі є. Це можуть бути як сторонні бібліотеки коду, так і бібліотеки внутрішньої розробки;
– множина допустимих станів середовища розробки – дається перелік складових апаратно-програмного середовища, в якому ведеться робота над про-

ектом, зокрема обов'язково вказуються номери версій інструментальних засобів, за допомогою яких здійснюється робота над окремими компонентами проекту, зазначається, чи використовуються обновлення та виправлення помилок;

- множина допустимих станів середовища тестування проекту – зазначається, в яких умовах та за допомогою якого інструментарію здійснюється перевірка функціональності проекту;
- множина допустимих станів середовища виконання проекту – зазначається, в яких умовах буде функціонувати кінцевий продукт. Ця складова кортежу проекту цілком відповідає складовій «середовище» кортежу програми, описаному у підрозділі 2.1.

Підтримувати паспорт проекту в актуальному стані та слідкувати за його дотриманням у команді, що працює над проектом, слід одному з керівників проекту, делегувати дані функції на більш низькі рівні управління, на нашу думку, недоцільно. Хоча делегування може бути здійснено, якщо робота виконується у великому колективі, в такому разі питанням підтримки паспорту проекту має займатися учасник проекту, за яким закріплено дані повноваження.

Паспорт проекту визначає такі ключові для проекту питання як:

- середовище виконання проекту – апаратні вимоги, вимоги до ОС, вимоги до мережного середовища, вимоги до СУБД й іншим зовнішнім по відношенню доданої системам, з якими повинні взаємодіяти отримані в результаті реалізації проекту прикладні програми;
- середовище розробки проекту – вказуються засоби автоматизованого проектування, використані мови програмування, компілятори й транслятори мов, інтегровані середовища програмування, засоби для проектування БД, у тому числі візуальні конструктори запитів і дизайнери БД, використані компоненти, що не входять у стандартну поставку середовища програмування;

- засоби створення документації – використані інструменти для генерації документації, що входить у кінцеву поставку прикладного програмного забезпечення;
- засоби створення дистрибутивів – використане середовище для генерації дистрибутиву, що поставляється кінцевому користувачеві створеного у результаті реалізації проекту програмного продукту;
- середовище для тестування проекту – описуються середовище, а також інструментальні засоби, що застосовувалися для тестування проекту [25].

Під час реалізації великих проектів, у яких задіяне значне число розробників (у тому числі й залучених зі сторони), паспорт проекту дозволяє гарантувати, що всі вони працюють у єдиному середовищі, що виключає проблеми несумісності окремих інструментальних засобів, конфлікт версій, неможливість відтворення програмних помилок і т.д.

Паспорт проекту дозволяє відстежити проблеми, породжені інструментальними засобами й іншими засобами, створеними поза фірмою-розробником програмного забезпечення. Будь-які проблеми, що виникають із тим або іншим інструментом, або середовищем експлуатування готового програмного продукту документуються й приймаються заходи для їхнього усунення.

Окремі проблеми усуваються шляхом документації особливостей роботи тих або інших засобів, для інших – необхідно прийняти заходи для їхньої заміни.

Паспорт проекту є вихідною точкою для управління процесом впровадження інновацій у процесі роботи над проектом – будь-які зміни, що стосуються ключових питань розробки відбиваються в даному документі й повинні бути прийняті всіма учасниками проекту до уваги.

У випадку, якщо робота над проектом припиняється або відкладається на тривалий строк, використання документа паспорт проекту дозволяє при необхідності відтворити актуальне середовище розробки при поверненні до робіт над проектом.

З огляду на можливість роботи тих самих розробників над різними проектами, у процесі якої вони можуть використовувати різне середовище розробки, у тому числі й таке, що виключає одне одного, документ паспорт проекту дозволяє їм відтво-

рити актуальне середовище розробки для роботи над кожним конкретним проектом. Також паспорт проекту може бути використаний розробником при початковій підготовці свого робочого місця до роботи над конкретним проектом. Приклад паспорту проекту представлений в додатку А.

Сам процес переходу на нові версії бібліотек коду та інструментарію має бути також впорядкованим і задокументованим. Його характер дуже сильно залежить від масштабу проекту та розміру компанії-розробника. Можливо, його слід проводити поетапно, чи з певною «тестовою» фазою, такою, як, наприклад, реалізація пілотного проекту.

Проте, на нашу думку, в процесі переходу на нову версію інструментарію, бібліотек коду чи будь-якої іншої складової кортежу проекту слід заборонити написання нового коду та розширення функціональності продукту за виключенням виправлення критично важливих помилок.

Також слід порекомендувати використовувати систему контролю версій компанією-розробником програмного забезпечення для збереження не лише вихідного коду проекту та пов’язаних з ним файлів, а й сторонніх бібліотек коду, що дозволить полегшити контроль за впровадженням нових версій використовуваних бібліотек коду розробниками, а також спростити підтримку «унаслідуваних» проектів, для яких перехід на нові версії бібліотек коду та інструментарію є недоцільним.

Як використання паспорту проекту, так і систем контролю версій може допомагати коректному впровадженню інновацій, зокрема, забезпечувати певне обмеження необґрунтованих нововведень, визначене у [28]. Необхідно орієнтуватися на вивчення можливості детальної формалізації процедури поновлення версій бібліотек компонентів та використовуваного інструментарію, її органічного включення до сучасних методик управління програмними проектами.

Іншим важливим питанням при реалізації програмних проектів є питання стандартизації внутрішніх процесів компанії. Згідно проведеного Ф. Бруксом дослідження, якщо реальний графік виконання програмного проекту відстає від планових показників, то збільшення кількості людських ресурсів на пізніх етапах реалізації

проекту лише погіршить ситуацію, і строки виконання проекту будуть відкладені на більш пізній термін [8].

Основна причина погіршення часових показників виконання проекту при збільшенні кількості доступних людських ресурсів для його виконання полягає у необхідності ознайомлення нових співробітників із особливостями реалізації проекту, що вимагає додаткових витрат часу як для нових учасників проекту, так і призводить до того, що відволікаються ресурси, які вже задіяні у реалізації проекту.

На нашу думку, важливим механізмом, який може сприяти значному скороченню неефективного використання робочого часу під час ознайомлення нових співробітників із особливостями реалізації проекту, а також зниженню неоднозначності у сприйнятті інформації як новими, так і існуючими учасниками проекту, має бути запровадження комплексу процедур і заходів, спрямованих на стандартизацію внутрішніх процесів.

Необхідність стандартизації процесів розробки ПЗ, зокрема стандартизації програмного коду, який створюється різними розробниками, відзначається у наступних джерелах [66, 140, 137, 45, 121, 70, 73, 111, 97, 67, 35, 81].

Стандартизація підходів до створення програмного забезпечення в рамках організації, зокрема стандартизація написання програмного коду, дозволяє отримати наступні переваги:

- спрощується процес заміни персоналу, зокрема підготовки нового персоналу для участі у проекті, вивчення особливостей реалізації проекту;
- програмний код стає більш прозорим і зрозумілим, а також містить меншу кількість помилок за рахунок використання випробуваних на практиці і визнаних фахівцями підходів до його створення;
- зникає проблема «приватного володіння» програмним кодом, коли пояснювати деталі реалізації і вносити зміни до вихідного коду здатен лише розробник, який його створював;
- ефективність розробки ПЗ може суттєво зрости за рахунок зміни моделі володіння кодом – будь-який розробник може змінювати код будь-якого іншого учасника проекту, якщо визнає його недосконалим;

- забезпечується підтримка так званого «захищеного програмування», спрямованого на створення ПЗ таким чином, щоб на початку написання коду закласти механізми простого його розширення, пошуку різного виду помилок, налагодження і тестування, а також програмування з мінімальною кількістю помилок;
- суттєво спрощується процес управління розробкою ПЗ, в тому числі й інноваційною діяльністю за рахунок зростання прозорості процесу розробки.

Під стандартизацією у даному разі слід розуміти формування внутрішньої документації (стандартів), яка вимагає від учасників проекту дотримуватися визначених нею правил при виконанні робіт, які регламентовані стандартами.

На наш погляд, стандарти мають регламентувати наступні елементи процесу розробки:

- створення проектної документації;
- створення документації по використанню програмних модулів;
- розміщення файлів проекту;
- надання назв файлам, програмним модулям, перемінним, структурам даних і т.д.;
- написання програмного коду;
- структура файлів вихідного коду;
- створення елементів інтерфейсу користувача;
- коментування вихідного коду;
- створення типових програмних конструкцій;
- створення критично важливого програмного коду;
- взаємодію учасників проекту при здійсненні типових операцій;
- використання інструментарію управління конфігураціями;
- середовище розробки ПЗ;
- процес оновлення компонентів і бібліотек коду;
- процес тестування ПЗ розробниками (юніт-тести);
- процес фіксування і документування помилок.

Також стандартами мають бути передбачені питання пріоритетності у дотримані їх вимог, визначатися можливості і умови для надання виключень із встановлених стандартів.

В результаті можна зробити наступні висновки.

Для вирішення проблеми поновлення версій сторонніх бібліотек коду та інструментарію, що використовується при розробці програмного забезпечення, доцільно ввести термін «кортеж проекту» по аналогією з терміном «кортеж програми».

Для контрою над актуальним станом кортежу проекту слід використати спеціальний документ, паспорт проекту. Цей документ має підтримуватися одним із керівників проекту і мати силу наказу в колективі розробників.

Також має бути забезпечена стандартизація внутрішніх процесів організації-розробника ПЗ.

3.2. Підходи до створення ефективної системи управління персоналом для забезпечення інтенсифікації інноваційного процесу

Для компанії, що займається розробкою ПЗ, найбільш важливим ресурсом є її персонал. Впровадження інновацій і забезпечення постійного поліпшення процесу можливе лише за рахунок створення ефективної системи управління персоналом.

Що стосується системи управління персоналом в рамках інноваційного менеджменту, то, на наш погляд, слід виділити дві головні задачі:

- по-перше, орієнтувати персонал таким чином, щоб співробітники підприємства були прихильниками, а не противниками інновацій;
- по-друге, мотивувати персонал на активну участь в інноваційних процесах, щоб співробітники самі були джерелами інноваційних ідей і розробок.

Що стосується першої задачі, то, на наш погляд, проблема відношення персоналу до інновацій стоїть дуже гостро і заслуговує особливої уваги.

Інновації є джерелом змін в організації і у більшості випадків впливають на інтереси її робітників. Фахівці з менеджменту відзначають, що «інновації є по свої суті підривними, бунтівними по відношенню до статусу-кво і кидають виклик тим, хто емоціонально, інтелектуально чи політично відданий існуючому стану речей» [169, С. 155]

У більшості випадків впровадження інновацій викликає опір зі сторони персоналу компанії, який може бути дуже активним і становити загрозу інноваційному процесу [145].

Наприклад, В. Москаленко зазначає, що інновації часто зустрічають опір у найбільш досвідчених і авторитетних фахівців, оскільки від впровадження інновацій вони можуть втратити свої передові позиції, які були набуті, можливо, за застарілих технологій [51].

На нашу думку, вирішити дану проблему можна декількома способами, які доцільно використати одночасно.

По-перше, слід проводити активну роз'яснювальну роботу серед ключових співробітників, намагатися добитися не лише зникнення у них недовіри до нововведень, а й зробити їх активними прибічниками [51].

По-друге, на нашу думку, для вирішення цієї проблеми дуже важливо мати і підтримувати сильну позитивну корпоративну культуру на підприємстві. Саме позитивна корпоративна культура буде заохочувати персонал до оволодіння новими технологіями, в той час як негативна може цьому перешкоджати.

По-третє, слід приділяти значну увагу організованим процесам навчання новим технологіям і методам роботи. Персонал має постійно підвищувати свій рівень кваліфікації і заохочуватися та контролюватися цей процес має зі сторони керівництва. Можна створити групу найбільш кваліфікованих працівників, які б займалися просвітницькою діяльністю [165].

По-четверте, система мотивації праці має бути побудована таким чином, щоб вона заохочувала персонал опановувати і використовувати нові технології.

Що стосується другої задачі (мотивації персоналу на створення інноваційних ідей і розробок), то Р. Каплан і Д. Нортон відзначають, що за сучасних умова навіть

простий робітник здатен становити особливу цінність для організації, якщо він здатен запропонувати ідеї яким чином покращити якість, зменшити витрати та скоротити виробничий цикл.[29]

Тому задача менеджменту фірми – зорієнтувати роботу подібних фахівців, щоб вони були зацікавлені в тому, щоб пропонувати свої ідеї і розробки для покращення діяльності підприємства на будь-якому етапі його бізнес-процесів.

Цікаві, на наш погляд, ідеї стосовно вирішення даної проблеми викладені у [59]. Автори даної роботи акцентують увагу не лише на адекватній фінансовій винагороді розробникам нововведень, а на побудові комплексного механізму, що дозволяє усунути розбіжності в інтересах таких учасників інноваційного процесу як розробника і менеджера. На наш погляд, подібний підхід, з урахуванням раніше наведених міркувань, слід доповнити також механізмами зацікавленням безпосередніх виконавців інновацій, які впроваджуються.

В тому, що стосується розробки інновацій робітниками підприємства, нерідко виникає проблема їх небажання з тих чи інших причин повністю передавати результати своєї інтелектуальної праці компанії, в якій вони працюють, відмовляючись таким чином від вигід, які принесе інновація її власнику.

Для вирішення цієї проблеми, на нашу думку, слід застосувати позитивний досвід служб комерціалізації результатів досліджень в наукових організаціях, наприклад, викладений у [1]. Зокрема, слід на договірній основі (наприклад, при укладанні контракту зі співробітником) чітко визначити процедуру оформлення і передачі авторського і майнового права зацікавленим сторонам. На наш погляд, слід договірним чином чітко визначити ті вигоди, які отримає співробітник, якщо він стане автором інновацій на підприємстві. У даному разі може бути корисним досвід деяких західних компаній, які встановлюють матеріальне заохочення для подібних співробітників у вигляді відсотку від загальної суми у грошовому вираженні тих вигід чи економії, які отримує компанія в результаті впровадження інновації, розробленої її співробітником.

Крім того, не слід акцентувати увагу лише на матеріальному інтересі подібних робітників. Як свідчить практика, для людей, схильних до наукової творчості, важ-

лива також можливість самовираження. Підприємство має з розумінням підходити до потреб подібних людей, оснащати їх робочі місця чи давати можливість користуватися необхідним обладнанням і інформацією, а також давати можливість навчатися, розвивати свої здібності і приймати участь у конференціях, на яких вони зможуть обмінятися досвідом зі своїми колегами.

Дослідження, покликане встановити взаємозв'язок стимулів і факторів, які визначають успіх проекту з розробки і впровадження ПЗ, представлено у [116]. Автори визначили фактори, які впливають на успіх проекту, виділивши з них декілька узагальнюючих груп, і встановили на основі опитування рівень кореляції між стимулами (які були поділені на монетарні і немонетарні) і значеннями факторів (табл. 3.1).

Таблиця 3.1

Кореляція між стимулами і факторами, які визначають успіх проекту з розробки і впровадження ПЗ [116]

Група факторів	**Назва фактору**	**Кореляція між стимулом і фактором**
Успіх проекту	Відповідність проекту запланованому бюджету витрат	0,935
	Відповідність тривалості проекту запланованому значенню	0,835
	Відповідність функціональності проекту очікуванням користувачів	0,750
Мотивація	Відчуття ентузіазму під час роботи над проектом	0,915
	Бажання думати над проектом у вільний від роботи час	0,778
Бажання діяти	Більш успішне управління командою	0,837
	Підвищення ефективності комунікацій у команді	0,722
	Підвищення контролю за продуктивністю у команді	0,704
	Бажання докладати більше зусиль, ніж звичайно очікується	0,806
Відданість	Відчуття свободи і комфорту під час роботи над проектом	0,710
	Проект справляє більш персональне значення для виконавця	0,866

Відповідно до даних табл. 3.1 можна зробити висновок, що рівень кореляції між стимулами виконавців і факторами, які визначають успіх проекту, є безсумнівно високим, що, в свою чергу, надає підстави стверджувати про наявність безпосереднього впливу системи мотивації та заохочування до праці виконавців проекту на його кінцевий результат.

Важливо розуміння з позицій управління фірмою витрат ресурсів на побудову ефективної системи мотивації як необхідних інвестицій у забезпечення вирівнювання інтересів найманих робітників і власників фірми [138].

Згідно з результатами дослідження [152], стимули до виконання проектів з розробки і впровадження ІС розподілені наступним чином (табл. 3.2).

Таблиця 3.2

Стимули до виконання проекту згідно опитування учасників проектів з розробки і впровадження ІС [152]

Стимули	**Доля респондентів, яка визнала стимул, %**
Грошові бонуси	80,58
Кар'єрний ріст	48,54
Гордість за добре виконану роботу	43,69
Умови роботи	38,83
Щорічна винагорода відповідно до досягнень	36,89
Технічне навчання	33,98
Відчуття внеску в роботу організації	33,98
Гнучкий робочий графік	32,04
Публічне визнання	30,10
Використання нових технологій	27,18
Робота на дому	22,33
Інше	11,65

Слід звернути увагу, що відповідно до табл. 3.2 понад 27% опитуваних вважають використання нових технологій одним з важливих факторів мотивації, що дає підстави вважати, що інновації у компанії мають не лише значну кількість противників, а і прибічників, прихильність яких необхідно розумно використати при впровадженні нових технологій.

Таким чином, можна порекомендувати наступні нематеріальні засоби мотивації, які доцільно використовувати при розробці ПЗ в умовах впровадження інновацій:

– просування кар'єрними сходами;
– проведення змагань між командами розробників;
– надання більш творчої і відповідальної роботи;

- надання можливості приймати участь у клубах з обмеженим доступом;
- надання кращих умов роботи, у тому числі й окремого робочого простору;
- надання вільного робочого графіку і можливості виконувати роботу на дому;
- надання додаткового відпочинку;
- сумісне відвідування командами розробників культурних заходів;
- надання можливості навчання і проведення сертифікації;
- надання символічних подарунків;
- надання можливості для участі у конференціях, виставках;
- визначення виконавців із найкращими та найгіршими показниками;
- використання перехідних символів, які видаються у якості догани за допущені помилки, а також у якості заохочення за особливі досягнення;
- здійснення позачергового чергування по виконанню рутинних задач для тих робітників, які заслуговують на покарання з того чи іншого поводу.

Крім того, в питаннях мотивації робітників інтелектуальної праці, на нашу думку, важливе значення має індивідуальний підхід. Для того, щоб забезпечити успішність системи управління персоналом, важливо докласти певних зусиль для виявлення індивідуальних стимулів кожного окремого робітника і розробити персоніфіковані підходи до стимулювання діяльності. Необхідність персоніфікованого підходу до стимулювання праці робітників високотехнологічної компанії відзначається також у [60, 63, 170].

Навчання персоналу також може бути важливим фактором мотивації [164]. Рекомендується розглядати навики персоналу в наступних областях:

- знання предмету – знання специфічних технологій розробки системи;
- знання розробки – досвід в проектуванні програмних систем та знання особливостей мов програмування;
- знання середовища – знання засобів і процесу, що використовуються при розробці проекту;

– знання функцій – володіння конкретними навиками, необхідними для виконання власної функції у колективі, наприклад, знання архітектури чи питань, пов’язаних з тестуванням [28].

Необхідно ідентифікувати, які саме інтереси можуть бути порушені в процесі впровадження інновацій, зокрема:

– необхідність проведення навчання і додаткової підготовки для співробітників, які мають використовувати нові технології;
– можливість втрати авторитет як фахівця, що добре знається на технологіях;
– можливість зміни характеру роботи;
– можливість зникнення певних посад і переведення на іншу роботу, чи поява загрози звільнення.

В середині 1990-х рр. Б. Куртіс розробив модель, що базується на CMM, проте сфокусована на кадрових питаннях – People Capability Maturity Model (P-CMM) [106]. Дана модель розглядає людський фактор у розробці програмного забезпечення та його взаємозалежність і взаємопов’язаність з загальною характеристикою рівня зрілості процесу розробки програмного забезпечення.

Відповідність задач менеджменту персоналу рівню зрілості процесу розробки ПЗ згідно з P-CMM представлена на рис 3.2.

<table>
<tr><td colspan="4">5. Оптимізуючий</td></tr>
<tr><td colspan="3">4. Керований</td><td rowspan="5">Постійне підвищення рівня компетенції та розвиток персоналу на рівні конкретної особистості і організації в цілому, навчання персоналу та впровадження інноваційних технологій.</td></tr>
<tr><td colspan="2">3. Визначений</td><td rowspan="4">Впровадження та розвиток кількісних методів управління персоналом, концентрація уваги на організацію ефективної взаємодії у групах учасників проекту.</td></tr>
<tr><td>2. Повторюваний</td><td rowspan="3">Відповідність цілям організації, оцінка знань і вмінь, планування потреб персоналу, навчання, планування кар’єри, формування стійкої організаційної культури.</td></tr>
<tr><td>1. Початковий
Систематичні методи відсутні.</td><td>Запровадження базового кадрового менеджменту. Добір кадрів, комунікації, навчання персоналу, компенсаційний пакет.</td></tr>
</table>

Рис. 3.2. Відповідність задач менеджменту персоналу рівню зрілості процесу розробки програмного забезпечення

В цілому, існує значна кількість досліджень, що сфокусовані на аспектах мотивації персоналу розробників ПЗ та ефективного управління ним. Відомі спроби представити програмування як мистецтво, чи, навпаки, як просте ремесло. Однак, практично всі фахівці у даній галузі погоджуються, що розробка інформаційної системи вимагає певних якісних характеристик робочої сили виконавця проекту.

Найбільш важливими виділяють наступні вимоги:

- наявність чіткого алгоритмічного мислення, знання сучасного інструментарію з розробки програмного забезпечення, здатність чіткого уявлення проблеми та можливість уявити послідовність взаємопов’язаних дій, що приводять до її розв’язання;
- здатність до роботи в команді, комунікабельність, ініціативність, здатність захистити свою точку зору, а також вміння визнати свою неправоту та погодитися з думкою інших;
- впевненість у власних силах і здатність не розгублюватися перед неочікуваними труднощами, що можуть виникнути при реалізації складних проектів, проте й вміння тверезо оцінити власні можливості, не переоцінити, чи недооцінити їх;
- здатність до самовдосконалення та бажання розвиватися інтелектуально, схильність до вивчення нових методів організації роботи та вміння не зупинятися на досягнутому;
- творчій підхід до вирішення завдань, вміння ефективно використовувати набутий досвід, будувати такі елементи системи, що здатні до вдосконалення у майбутньому, здатність до абстрактного мислення та досконале знання об’єктно-орієнтованих методів проектування програмного забезпечення;
- бажання виконувати свою роботу, наявність внутрішніх мотиваторів [25].

В тому, що стосується практичних рекомендацій з побудови організаційної структури компанії, яка займається розробкою ПЗ, то, на наш погляд, доцільно запропонувати наступні конкретні рекомендації.

Організаційна структура компанії має враховувати методологію і модель життєвого циклу, яка використовується компанією при роботі над програмними проектами. У тому випадку, якщо використовуються жорсткі підходи до постановки задач і контролю за їх виконанням, а також в основі моделі життєвого циклу лежить водоспадна модель, то у даному разі доцільно буде використовувати таку організаційну структуру, яка буде забезпечувати жорсткий контроль і забезпечення виконання поставлених завдань, наприклад, лінійну.

В той же час, якщо використовуються підходи, що передбачають більш вільну і твору роботу співробітників, то організаційна структура має надавати їм такі можливості.

Можна порекомендувати наступні вимоги до організації проектної групи:

- взаємозалежні ролі в малій групі;
- визначення ролі, особливої місії і зони відповідальності для кожного члена проектної групи;
- розподілені управління проектом і відповідальність;
- кожен сфокусований на успіху проекту і орієнтований на безперервну роботу протягом усього циклу проекту;
- комунікації між членами проектної групи є ключовим чинником успіху;
- користувачі і навчальний персонал мають бути включені в проектну групу;
- рівнобіжна робота всіх учасників групи над проектом.

Бажано, щоб модель проектної групи ніяк не співвідносилася з організаційною структурою. На практиці часті випадки, коли в одній проектній групі працюють люди з різних організацій, що є підлеглими різних керівників. За кожним членом проектної групи закріплюється конкретна роль, для якої будується специфічний план робіт, що потім входить у загальний план проекту як складова частина.

Приймаючи рішення про склад групи, важливо в першу чергу подбати про те, щоб у неї ввійшли люди, що володіють необхідними навичками і знаннями для виконання задачі.

Кожна роль проектної групи має свою особливу компетенцію і, взаємодіючи з іншими ролями, забезпечує створення якісного продукту.

Рекомендується використовувати наступні ролі, по аналогії з розглянутою у підрозділі 1.2 моделлю проектної групи MSF.

Менеджер продукту. Ця роль забезпечує комунікаційний канал між замовником і проектною групою. Менеджер продукту керує очікуваннями замовника, розробляє і підтримує бізнес-контексту проекту. Його робота не пов'язана прямо з продажем продукту, він сфокусований на продукті, його задача – визначити і забезпечити задоволення замовника. Краща кандидатура на цю роль – існуючий користувач, співробітник комерційного відділу або інший представник замовника, якщо він розуміє задачі і механіку бізнесу.

Менеджер програми. Ця роль керує комунікаціями і взаємовідносинами в проектній групі, є в деякому роді координатором, розробляє функціональні специфікації і керує ними, веде графік проекту і звітує про стан проекту, ініціює прийняття критичних для ходу проекту рішень.

Розробник. Розробник приймає технічні рішення, що можуть бути реалізовані і використані, створює продукт, що задовольняє специфікаціям і очікуванням замовника, консультує інші ролі в ході проекту. Він бере участь в оглядах, реалізує можливості продукту, бере участь у створенні функціональних специфікацій, відслідковує і виправляє помилки за прийнятний час. У контексті конкретного проекту роль розроблювача може включати, наприклад, інсталяцію програмного забезпечення, настроювання продукту або послуги.

Розробка складних інформаційних систем вимагає детального знання високорівневих мов програмування, візуального програмування, мережних технологій і проектування баз даних. Звичайно, одна людина не може бути експертом у всіх областях цих технологій. І важливо, щоб експертиза у всіх областях була представлена відповідними технічними фахівцями, що входять у групу розроблювачів, а керівник цієї групи знав і розумів ключові моменти кожної з цих технічних областей.

Тестувальник. Тестування повинне містити в собі не тільки перевірку коду. Тестувати потрібно функціональні специфікації, систему забезпечення продуктивності, інтерфейси користувача, плани впровадження і використовувану термінологію. Тестер забезпечує те, що всі особливості і задачі будуть відомі до випуску вер-

сії продукту, розробляє стратегію тестування і плани тестування для кожної з фаз проекту. Плани і процедури тестування для клієнт-серверних систем повинні бути комплексними. Ще більш комплексними вони повинні бути у випадку програмування, орієнтованого на події, декількох мережних транспортів і цільових серверів, задач адміністрування даних і баз даних і т.д.

Дуже важливо розрізняти тестування і контроль якості. Тестування зосереджене на проекті й оперує деталями і технікою роботи.

Інструктор. Ця роль відповідає за зниження витрат на подальший супровід продукту, забезпечення максимальної ефективності роботи користувача. Важливо, що мова йде про продуктивність користувача, а не системи. Для забезпечення оптимальної продуктивності інструктор збирає статистику по продуктивності користувачів і створює рішення для підвищення продуктивності. Інструктор бере участь у обговореннях інтерфейсу користувача й архітектури продукту.

Логістик. Задача цієї ролі – забезпечити гладке впровадження і розвиток продукту. Звичайною є ситуація, коли впровадження продукту коштує дорожче його розробки. Логістик повинен забезпечити, щоб замовник був готовий до впровадження, щоб вчасно були виконані всі підготовчі роботи й існувала необхідна інфраструктура.

Крім того, для ефективного забезпечення інноваційних процесів на етапах життєвого циклу ПЗ можна виділити додаткову роль – *інноватор*, яка може бути суміщена з іншими ролями, однак передбачає цілеспрямовану інноваційну діяльність при реалізації проектів командою [110].

Таким чином, можна сформувати наступні висновки.

По-перше, що стосується персоналу компанії, то слід не лише перебороти недовіру, а й зорієнтувати співробітників на підтримку інноваційних процесів. Дуже часто інновації зустрічають опір зі сторони найбільш досвідчених і авторитетних фахівців, оскільки вони здобули свої позиції за минулих, можливо суттєво застарілих технологій, і намагаються їх не втратити. Для вирішення цієї проблеми слід використовувати комплекс заходів, спрямованих на те, щоб кардинальним чином змінити відношення робітників до інновацій.

По-друге, слід активно заохочувати тих робітників, які є носіями інноваційних ідей і розробок. Необхідно не лише створювати для них всі необхідні умови, а й давати їм можливість приймати участь у вигодах підприємства, отриманих за рахунок їх розробок. Важливим моментом є орієнтація не лише матеріальні методи стимулювання праці подібних робітників, оскільки для них особливо важливою є потреба в самовираженні.

3.3. Інтеграція системи управління інноваціями в загальну систему управління життєвим циклом програмного забезпечення

Система управління інноваціями буде дієвим і функціональним механізмом в роботі організації лише за умови, якщо вона ефективно інтегрована у систему управління життєвим циклом ПЗ.

Слід зазначити, що важливим визначальним фактором діяльності сучасних компаній з розробки ПЗ є процеси глобальної інформатизації, які істотно впливають на всі форми економічної діяльності і вимагають певних зусиль для підготовки організацій до діяльності в новому бізнес-середовищі [76].

В тому, що стосується загальних підходів до управління, відзначається, що необхідне переосмислення, радикальна перебудова організацій з метою виключення непотрібної роботи, усунення затримок, підвищення гнучкості організаційних структур і зниження операційних витрат.

Рух, спрямований на досягнення даних цілей, що виник на початку 1990-х рр., одержав назву «реінжиніринг бізнес-процесів». Засновники даного напрямку визначають реінжиніринг як фундаментальне переосмислення і радикальну перебудову бізнес-процесів для досягнення корінних покращень критичних показників продуктивності, таких як собівартість, якість, обсяг і номенклатура наданих послуг, швидкість обслуговування [114]. Під словами “корінних покращень” мається на увазі багаторазове (у десятки або навіть сотні разів) підвищення основних показників ефективності роботи організації.

Кінцева мета реінжинірингу бізнес-процесів складається в корінній зміні підприємства, у прийнятті нової філософії управління, орієнтованої на процеси, у внесенні радикальних і необоротних змін у діяльність підприємства, що дозволяють зробити його більш ефективним і більш пристосованим до виживання в умовах швидко мінливого зовнішнього середовища.

У визначенні реінжинірингу бізнес-процесів покладено три ключових положення: фундаментальність, радикальність і орієнтація на процеси [114].

Під фундаментальністю розуміється, що реінжиніринг бізнес-процесів вимагає від підприємства переосмислення головних основ свого існування. Ціль реінжинірингу – відкинути все непотрібне, зайве, неефективне і сконцентруватися на тому, як можна зробити те ж саме якомога найкращим чином.

Під радикальністю розуміється головне завдання реінжинірингу – воно перебуває не в тому, щоб поліпшити старі способи роботи, а в тому, щоб відкинути все старе і створити радикально нове з метою досягнення багаторазового поліпшення основних показників ефективності.

На нашу думку, третє положення реінжинірингу – орієнтація на бізнес-процеси, вимагає більш детального розгляду.

Традиційне прийняття управлінських рішень відбувається в рамках організаційної структури, що заснована на ієрархії, котра базується на основі поділу підприємства на функціональні підрозділи.

Реінжиніринг бізнес-процесів дозволяє відмовитися від багаторівневої ієрархічної системи управління, зробивши її більш плоскою. Замість функцій підприємство починає орієнтуватися на процеси, що являють собою фундаментальні види діяльності підприємства з погляду кінцевого покупця продукції чи послуг. Процес інтерпретується як набір операцій, що створюють на основі кінцевого числа входів кінцевий набір виходів, котрі представляють цінність для покупця.

Реінжиніринг бізнес-процесів дозволив розглядати конструювання бізнесу як особливий вид інженерної діяльності. Аналіз характеристик бізнес-процесів дозволяє по-новому підійти до задачі побудови ефективно працюючих підприємств.

Відбулася зміна порядку прийняття рішень в організації – за умови спроектованого належним чином бізнес-процесу – менеджери середньої та низової ланки отримали більше повноважень для прийняття рішень: відбувся «вертикальний стиск процесів». Замість звертання «наверх» у нових умовах виконавець уповноважений прийняти значну групу рішень самостійно. Прийняття рішень стає частиною функціональних обов'язків виконавця.

У результаті вертикального стиску процесів зменшуються тимчасові затримки, знижується вартість, прискорюється реакція на запити замовника.

Етапи процесу виповнюються в природному порядку. На практиці це означає, що порядок виконання робіт не визначається директивним розпорядженням і не повинен бути лінійно упорядкованим. Виконавці самостійно визначають природний для них порядок робіт з ходу виконання і відповідно до реальної обстановки.

Усунення лінійності бізнес-процесів, по-перше, дозволяє виконувати ряд робіт паралельно, а, по-друге, зменшує час, що витрачається на усунення невідповідностей між попередніми і наступними етапами процесу.

Процеси мають різні варіанти виконання. Традиційні процеси були спроектовані з орієнтацією на масове виробництво. Такий процес повинен виконуватися ідентично для всіх типів входів, враховувати різні виключення й окремі випадки, і тому був дуже складним. У сучасних умовах нестабільного, мінливого зовнішнього оточення необхідно, щоб процес мав різні версії в залежності від конкретного набору входів і станів ринку.

Нові процеси, у відповідністю з концепцією реінжинірингу, починаються з деякого перевірочного етапу, на якому визначається яка з існуючих версій процесу найбільше відповідає його вхідним параметрам (сформованої ситуації). У результаті кожна версія орієнтована тільки на один конкретний варіант, а тому є простою і ясною, проте й дозволяє значну гнучкість в залежності від компетенції виконавця.

Таким чином, бізнес-процес став безпосереднім об'єктом управління менеджерів середньої та нижньої ланки

У традиційних процесах робота організується навколо спеціалістів, організованих у функціональні підрозділи, що можуть розміщатися у всіляких місцях, іноді

досить територіально вилучених. У результаті економічна вигода від спеціалізації праці може бути непорівнянною з грошовими і часовими витратами на між функціональне узгодження і переміщення робіт і/або спеціалістів.

Реінжиніринг бізнес-процесів усуває надмірну функціональну інтеграцію там, де вона не потрібна або не приносить очікуваної вигоди, що приводить до підвищення ефективності бізнес-процесу.

Зменшується кількість перевірок і управлінських впливів. Тому що управління не створює прямої доданої споживчої вартості, тому воно зберігається тільки на тих ділянках робіт, де це має економічний зміст. Замість перевірки усіх виконуваних робіт бізнес-процес, отриманий у результаті проведення реінжинірингу, характеризується агрегованими відкладеними в часі перевірками і управлінськими впливами. Подібний підхід дозволяє скоротити час і вартість перевірок.

Відбувається децентралізація підрозділів на основі централізованої інформації. Питання про співвідношення централізації і децентралізації в підходах до управління завжди було одним з найбільш спірних [48]. Кожний із двох підходів має свої переваги та недоліки. Реінжиніринг бізнес-процесів, заснований на застосуванні інформаційних технологій, дозволяє використовувати переваги обох підходів. Зберігаючи децентралізовані підрозділи, він дозволяє забезпечити їм доступ до централізованих даних [114].

На нашу думку, велике значення реінжинірингу бізнес-процесів по відношенню до управління інноваціями на етапах життєвого циклу ПЗ полягає в тому, що реінжиніринг дозволяє «сконструювати» бізнес-процеси компанії таким чином, щоб вони відповідали визначеним вимогам до інноваційного менеджменту компанії.

Зокрема, Р. Каплан і Д. Нортон, розробники збалансованої системи показників (BSC), розглядають інноваційний процес як невід'ємну складову внутрішніх бізнес-процесів [29]. Відповідно до рис. 3.3 інноваційний процес передує операційному процесу і процесу післяпродажного обслуговування.

Позитивна риса подібного підходу – розуміння інновацій як невід'ємної складової бізнес-процесів організації.

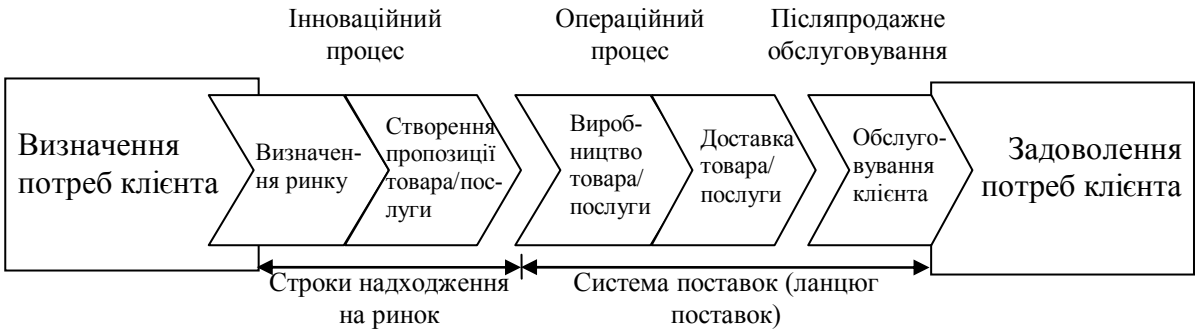


Рис. 3.3. Складові внутрішніх бізнес-процесів –
загальна модель ланцюга вартості [29]

Однак, на наш погляд, подібний підхід є досить вузьким і штучно відокремлює у часі науково-дослідницькі розробки від інших складових бізнес-процесів організації. Якщо виробничій цикл є досить тривалим, чи компанія працює на ринку, на якому технології швидко розвиваються, подібний підхід призведе до того, що вона у своїх бізнес-процесах завжди буде впроваджувати «інновації вчорашнього дня».

На наш погляд, вирішити проблему відокремленості в часі розробки інновацій від їх впровадження можливо за умови, якщо інновації будуть розглядатися не лише як невід'ємна складова бізнес-процесів, а як їх постійна складова.

Слід відзначити, що як операційний процес, так і процес післяпродажного обслуговування вимагають постійного вдосконалення і впровадження нових технологій. З випуском продукту на ринок компанія має сфокусуватися на визначенні і реалізації підходів, які б дали змогу набути і утримувати конкурентні переваги за рахунок постійного покращення основних показників операційного процесу і процесу післяпродажного обслуговування, таких як якість, собівартість і їх тривалість. Зокрема, це стосується системи показників, розробленої нами у підрозділі 2.3.

Саме тому ми вважаємо, що доцільно модифікувати схему складових бізнес-процесів, наведену на рис. 3.3 таким чином (рис. 3.4).

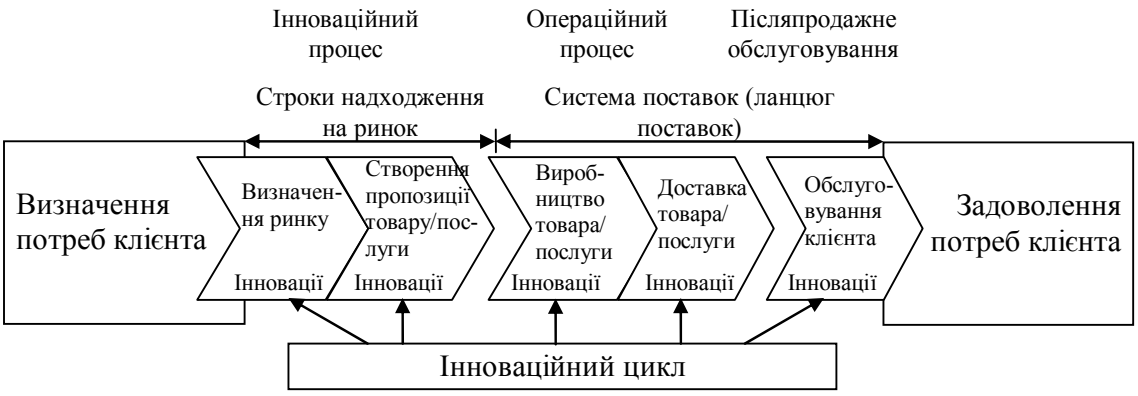


Рис. 3.4. Інноваційний цикл як постійна складова бізнес-процесів

Зазначимо, що запропонований нами інноваційний цикл як складова кожного етапу бізнес-процесів має безперервно повторюватися і є доцільним навіть на етапі розробки товару чи послуги. Будь-які нововведення, реалізація яких буде мати позитивний економічний ефект з урахуванням витрат на їх впровадження та, можливо, затримок, викликаних даних процесом, мають бути впровадженні незалежно від того, на якому етапі знаходиться бізнес-процес.

Саме такий підхід до інновацій, на нашу думку, дозволить надати їм нового значення у визначенні конкурентноздатності підприємства. Зокрема, безперервність інноваційного процесу дозволить подовжити життя товару чи послуги на ринку за рахунок зниження собівартості чи покращення якості, що, відповідно, призведе до підвищення віддачі від інвестицій у розробку і виведення на продуктів компанії.

Особливої уваги заслуговує питання інновацій на етапі розробки товару чи послуги. Процес розробки за своєю природою є інноваційним і в сучасних умовах його вплив на кінцевий результат суттєво зростає, оскільки вузьким місцем стає так званий «людський фактор», який на даному етапі є найбільш вираженим. Саме тому зростає роль нових технічних засобів, організаційних підходів та методів, які застосовуються на даному етапі. Інновації дозволять знизити вплив людського фактору і отримати керований та передбачуваний кінцевий результат процесу науково-технічних досліджень і розробок.

Використання формалізованих підходів до аналізу і опису бізнес-процесів компанії дозволить спростити процес інтеграції системи управління інноваціями у загальну систему управління бізнес-процесами компанії [85, 37].

Зокрема, згідно з BPML бізнес-процес розглядається як взаємодія між учасниками у ході виконання діяльності у відповідності зі встановленим набором правил з метою досягнення спільних цілей. Характерна особливість мови BPML полягає в тому, що учасник бізнес-процесу визначається не як певний суб'єкт, а як певна сутність, під якою можна розуміти й інший процес, що дає змогу моделювати так звані "динамічні" бізнес-процеси, для яких учасники на момент моделювання можуть бути невідомими, а відомо лише певні вимоги до них. Саме тому учасники у термінології BPML можуть бути статичними і динамічними [85].

Під "статичними" учасниками розуміють учасників, що відомі на момент моделювання процесу, процес завжди взаємодіє з одним і тим же набором статичних учасників. Абстракція досягається за рахунок використання організаційних ролей, ділових каналів і т. ін. Прикладом таких учасників можуть бути "менеджер з продаж", "місце продажу", "розрахункові послуги".

Оскільки динамічні учасники невідомі на момент моделювання процесу, то це дозволяє моделювати такі процеси, що цілком залежать від зовнішнього середовища та ніколи не повторюються (наприклад, ринок). Коли покупець звертається з запитом до місця проведення торгів, відбувається створення нового ринкового процесу з метою пошуку відповідної пропозиції. Різні ринкові процеси можуть бути використані, щоб знайти найкращу відповідність (у відповідності з певними критеріями).

Місце проведення торгів визначає прийнятний перелік постачальників та інформує їх про існування нового ринку. Процес комунікації між постачальником і місцем проведення торгів є важливим, так як не існує двох однакових ринків.

Коли процес пошуку завершується, найкраща з пропозицій надсилається покупцеві (виграє процес), інформуючи покупця про те, який з постачальників запропонував найкращі умови. Покупець далі може безпосередньо взаємодіяти з постачальником.

У такій прояві гнучкості BPML може виступати практичним інструментом, який здатен забезпечити інтеграцію складових системи управління інноваціями у практичну діяльність компанії зі створення ПЗ.

Важливо забезпечити поєднання системи управління інноваціями із іншими складовими, що забезпечують реалізацію проекту. Управління фінансами проекту передбачає контроль і управління бюджетом проекту, іншими фінансами, а також має передбачати фінансову звітність для проекту. Успішне управління фінансами передбачає, що всі витрати, які мають відношення до прогресу проекту, плануються і відстежуються, зокрема експлуатаційні витрати, капіталовкладення, і витрати на оплату праці.

Враховуючи, що звичайно основні фінансові питання пов’язані з діяльністю основної проектної команди, однак не підконтрольні їй, (наприклад, розвиток інфраструктури), необхідно забезпечити такі умови для функціонування проекту, щоб управління фінансами дозволяло команді визначити необхідну кількість ресурсів і підтримувати всі фінансові аспекти бізнес-середовища проекту. Важливо забезпечити учасникам проекту, зокрема, керівникам проекту, можливість приймати участь у вирішенні питань, пов’язаних із наданням ресурсів для його реалізації, що має забезпечити більш успішне виконання проекту.

На нашу думку, слід відокремити основні питання, такі як забезпечення діяльності по виконанню задач проекту, від допоміжних, таких як ведення звітності, і перекласти допоміжні задачі на супроводжуючий персонал, що не приймає участі у реалізації основних задач проекту.

Слід зазначити, що важливо задати загальні директиви стосовно бюджетних обмежень на проект, надавши можливість учасникам проекту самостійно спланувати конкретні витрати з урахуванням власних поглядів на те, яким чином має виглядати структура витрат і потреба у ресурсах на виконання проекту. Дотримання даної вимоги надасть можливості здійснити оцінку реалістичних показників бюджету проекту і сприятиме підвищенню мотивації за рахунок відчуття власної участі у прийнятті важливих рішень.

Основою кошторису витрат на реалізацію проекту має бути розроблений керівником проекту бюджет, що має передбачати конкретні джерела витрат на реалізацію програмного проекту. Це мають бути як прямі, так і непрямі витрати, які мають бути предметом обговорення із спонсором проекту.

Крім власне запропонованого бюджету, важливо також визначити ті його елементи, які можуть уточнюватися і змінюватися в процесі реалізації проекту. Наприклад, оцінка витрат на впровадження нової технології може бути грубою, її реальне значення можна визначити лише в певному діапазоні. Якщо цей діапазон буде запропонований і узгоджений із особами, що забезпечують фінансування проекту, то у даному разі невизначеність витрат не буде являтися джерелом додаткового ризику для успіху проекту.

Крім того, має бути запропонований оціночний підхід, на основі якого буде здійснюватися виключення і включення витрат до бюджету проекту. Якщо основні принципи виключення і включення витрат будуть узгоджені із керівництвом, то це дозволить при реалізації проекту у майбутньому уникати додаткових витрат часу на узгодження в питаннях, що не є критичними з погляду виконання проекту.

Саме менеджер проекту, на нашу думку, мають бути делеговані повноваження контролю і управління проектними витратами по ходу реалізації проекту. Менеджер проекту має здійснювати процес схвалення накладних, забезпечувати їх збереження, відповідати за своєчасність оплат по проекту і звітувати про фінансовий статус проекту.

Наступним важливим питанням є питання управління постачальниками. Управління постачальниками передбачає участь менеджерів проекту у процесі з вибору і управління ресурсами, які організація отримує із зовнішніх джерел. Сюди відносяться питання добору і взаємодії із консультантами, постачальниками та підрядчиками. Головна мета управління постачальниками полягає в тому, щоб досягти виконання завдань проекту, допомагаючи у виборі постачальників і встановлюючи ділові відносини з ними для того, щоб отримати відповідні технології, вироби, і послуги. Управління постачальниками має передбачати укладання контракту і управ-

ляти будь-якими змінами до контракту, враховуючи динаміку зовнішнього середовища, зокрема, зміну конкурентного становища постачальника.

Вигоди від ефективного управління постачальниками полягають у зниженні вартості створюваного ПЗ з урахуванням безпосередньої взаємодії постачальників технологій і їх клієнтів, налагодження партнерства між учасниками одного продуктового ланцюга, що надає можливості до їх ефективної взаємодії, розвиток стандартів якості, побудованих на кількісних показниках.

Ці стандарти гарантують, що всі учасники погоджуються із певними обмеженнями на показники продуктів, які є предметом контракту, і можуть брати їх за основу в процесі прийняття важливих рішень стосовно розробки ПЗ.

Укладання контрактів має здійснюватися у тісній взаємодії учасників проекту із юридичним відділом компанії. Юридичний відділ має транслювати вимоги учасників проекту у форму угоди, яка забезпечує їх дотримання.

Слід встановити прямі відносини між учасниками проекту і постачальниками, щоб забезпечити максимум ефективності комунікацій між ними, а також прискорення вирішення будь-яких проблем, які мають, як правило, технічний характер.

Слід розглядати наступні ключові моменти в питаннях управління постачальниками:

- планування потреб:
- здійснення вибору;
- встановлення договірних відносин;
- контроль за дотриманням договірних відносин;
- зміна ухвалених угод;
- припинення угод.

У процесі здійснення планування потреб у сторонніх постачальниках необхідно ідентифікувати всі продукту і послуги, потрібні для проекту. Необхідно враховувати як основні продукти і послуги, від яких безпосередньо залежить успіх проекту, наприклад, інструментарій і технології, які будуть використовуватися в процесі його реалізації, так і допоміжні, які мають сприяти успішному виконанню проекту та пришвидшенню його реалізації.

У процесі планування потреб важливо не лише ідентифікувати потреби у конкретних продуктах і технологіях, необхідних для успішної реалізації проекту, також важливо запропонувати конкретний перелік постачальників, продукція яких в тій чи іншій мірі задовольняє встановлені вимоги.

Варіантом для складання подібного переліку також може бути оголошення тендеру із формуванням конкретних вимог і розсилання їх потенційним постачальникам для отримання конкретних пропозицій. Будь-які зміни у вимогах мають одночасно бути повідомлені усім постачальникам, яким запропоновано прийняти участь у побідному тендері.

У процесі здійснення вибору необхідно провести комплексний аналіз можливих варіантів використання тих чи інших технологій з урахуванням критеріїв, запропонованих нами у підрозділі 2.2. Звичайно основою для прийняття рішення має бути розробка технічного завдання на проект з використання технології, яке має бути створене на основі технічного завдання основного проекту.

Як зазначалося раніше, важливим кроком може бути здійснення пілотного проекту, який може надати інформацію стосовно практичного використання тієї чи іншої технології, що складно зробити, базуючись лише на теоретичних оцінках. Реалізація пілотного проекту може провадитися у рамках частини бюджету, виділеного на придбання нової технології.

Також важливим моментом у процесі вибору постачальника є надання повної інформації потенційним постачальникам стосовно критеріїв, згідно з якими була встановлена невідповідність продукції вимогам. Якщо постачальник зацікавлений у отриманні контракту, то він може запропонувати інші умови, що дасть можливість переглянути рішення.

Як зазначалося раніше, контракт має бути сформований одночасно проектною командною і юридичним відділом компанії. Також у процесі встановлення договірних відносин можливе здійснення декількох ітерацій по модифікації форми контракту, оскільки певні умови можуть не влаштовувати одночасно обидві сторони.

В процесі контролю за дотриманням договірних відносин менеджер проекту має вимагати від постачальника виконання усіх пунктів, обумовлених контрактом.

Якщо постачальник відмовляється від дотримання певних пунктів, то до нього можуть бути застосовані санкції, обумовлені умовами контракту. Будь-які відхилення від умов контракту слід відповідним чином документувати.

У певних випадках умови виконання проекту, чи вимоги до нього можуть змінитися, що, в свою чергу, може викликати необхідність перегляду контрактів із постачальниками. Якщо постачальник погоджується на зміну положень контракту, то його може бути змінено і зміни мають бути зафіксовані у вигляді нового контракту, чи доповнення до існуючого. Якщо постачальник відмовляється, то в такому разі необхідно припинити контракт.

Припинення контракту може бути здійснено у зв'язку його виконанням, чи достроково. Менеджер проекту має забезпечити захист інтересів власної компанії і гарантувати, що припинення контракту у зв'язку з його виконанням відбудеться лише за умови, що виконані всі його вимоги зі сторони постачальника. Він відповідає за здіснення оплат по контракту.

В тому, що стосується відношення моделі життєвого циклу ПЗ до можливостей органічної інтеграції з системою управління інноваціями, то, на наш погляд, доцільно запропонувати можливість такої інтеграції (табл. 3.3)

Таблиця 3.3

Здатність моделей життєвого циклу бути інтегрованими
із системою управління інноваціями

№ п/п	Назва моделі життєвого циклу ПЗ	Здатність інтеграції із системою управління інноваціями
1	Каскадна модель SLC	Низька
2	Модифікована каскадна модель SLC	Низька
3	V-подібна модель	Низька
4	Інкрементна модель	Середня
5	Модель швидкої розробки	Середня
6	Модель прототипування	Середня
7	Спіральна модель	Висока
8	Ітеративна модель	Висока

Як видно з табл. 3.3, високий рівень інтеграції із системою управління інноваціями мають моделі, побудовані з використанням ітераційних підходів до управління життєвим циклом ПЗ.

Таким чином, система управління інноваціями має бути побудована з урахуванням стану і етапів життєвого циклу програмного проекту.

Крім того, на нашу думку, що підтверджується іншими незалежними дослідженнями, зокрема [157], не слід фокусуватися на абсолютизації кількісних методів управління інноваціями при розробці ПЗ.

3.4. Практична перевірка запропонованої системи управління інноваціями на етапах життєвого циклу програмного забезпечення

В підрозділі 2.3 даної дисертаційної роботи нами була запропонована система економіко-технічних показників для управління інноваціями на етапах життєвого циклу ПЗ, створена на основі визначених у підрозділі 2.1 науково-методичних підходів до розробки ПЗ і запропонованих у підрозділі 2.2 підходів до управління впливом сторонніх компонентів на процес розробки ПЗ. У підрозділі 3.1 запропоновано конкретні інструменти та методи для управління інноваціями у сфері ПЗ, у підрозділі 3.2 запропоновано конкретні підходи до створення ефективної системи управління персоналом для забезпечення інтенсифікації інноваційного процесу, а у підрозділі 3.3 розглянуті питання інтеграції системи управління інноваціями на етапах життєвого циклу ПЗ у загальну систему управління життєвим циклом ПЗ.

Нижче буде здійснена практична перевірка запропонованої системи управління інноваціями на етапах життєвого циклу ПЗ на основі даних програмних проектів, які реалізуються фахівцями відділу розробки ПЗ АТ «Датекс Україна».

Вибір саме цієї компанії для апробації теоретичних висновків дисертаційного дослідження пояснюється наявністю ряду об’єктивних факторів:

– по-перше, АТ «Датекс Україна» є провідною компанією, яка працює одночасно на декількох суміжних ринках як апаратного, так і програмного забезпечення (насамперед, це обладнання і програмне забезпечення для ав-

томатизації торгівлі і обліку – від невеликих торгівельних підприємств до крупних мереж супермаркетів міжнародного масштабу) і є одним із лідерів у кожному із секторів ринку, на яких вона працює;

– по-друге, АТ «Датекс Україна» працює на ринках з надзвичайно жорстким конкурентним середовищем, яке вимагає від учасників проводити агресивну маркетингову і технологічну політику, приділяти значну увагу інноваційному менеджменту. Ефективне управління інноваціями на подібних ринках є однією із ключових складових успішної конкурентної боротьби, від якої у значній мірі залежать ринкові позиції компанії;
– по-третє, АТ «Датекс Україна» одночасно працює над реалізацією великої кількості програмних проектів різного масштабу, які використовують різні моделі і знаходяться на різних етапах життєвого циклу, і призначені як виключно для внутрішнього використання, так і для поставки кінцевим користувачам у вигляді повністю закінчених і самодостатніх рішень чи окремих складових комплексних систем автоматизації.

У якості вихідних даних для перевірки теоретичних положень даного дослідження виступають матеріали внутрішньої документації і проведеного анкетування фахівців АТ «Датекс Україна», які приймають участь у виконанні одинадцяти програмних проектів.

Вихідні дані для розрахунків представлені у додатку Б. Табл. Б.1 містить інформацію про проекти, які повністю чи частково використовували запропоновані у даній дисертаційній роботі науково-методичні підходи і практичні рекомендації управління інноваціями на етапах життєвого циклу ПЗ, а табл. Б.2 містить інформацію про проекти, які не використовували результати даного дослідження для управління інноваціями. Такі показники як коефіцієнт перспективності грошових надходжень ($К_{ПН}$) і коефіцієнт успішності нової версії програмного продукту ($К_{УНВ}$) розраховувалися згідно з формулами 2.2 і 2.3 працівниками АТ «Датекс Україна», оскільки для їх розрахунку необхідна інформація, яка вважається закритою згідно з внутрішніми вимогами компанії до безпеки інформації.

Показники $К_{ПН}$ і $К_{УНВ}$ для проекту 5 табл. Б.1 додатку Б і проектів 5 та 6 табл. Б.2 додатку Б.2 не розраховувалися, оскільки проекти призначені виключно для внутрішнього використання і не передбачають оцінку грошових надходжень від їх реалізації.

Для проектів, представлених у табл. Б.1 додатку Б повністю або частково використовувалися наступні складові системи управління інноваціями на етапах життєвого циклу програмного забезпечення, представлені у даній дисертаційній роботі:

– поетапне проходження інноваційний циклу, описане у підрозділі 2.1;
– система управління впливом сторонніх компонентів під час розробки ПЗ, описана у підрозділі 2.2;
– система економіко-технічних показників для оцінки управління інноваціями на етапах життєвого циклу ПЗ, описана у підрозділі 2.3;
– інструменти та методи для управління інноваціями, викладені у підрозділі 3.1 (паспорт проекту для фіксування кортежу проекту, стандарти на здійснення внутрішніх процесів при реалізації програмних проектів, зокрема, стандарти на написання вихідного коду, розроблені на основі авторських стандартів, опублікованих у [35]);
– підходи до створення ефективної системи управління персоналом для інтенсифікації інноваційних процесів, описані у підрозділі 3.2;
– підходи до інтеграції системи управління інноваціями у загальну систему управління життєвим циклом ПЗ, описані у підрозділі 3.3.

У табл. 3.4 представлено результати розрахунку економіко-технічних показників для оцінки управління інноваціями згідно методики, описаної у підрозділі 2.3 даного дослідження, по відношенню до проектів, які використовували вищенаведені складові системи управління інноваціями на етапах життєвого циклу ПЗ (на основі даних табл. Б.1 додатку Б).

Таблиця 3.4

Результати розрахунку економіко-технічних показників для оцінки управління інноваціями для проектів, при реалізації яких використовувалися складові запропонованої системи управління інноваціями на етапах життєвого циклу ПЗ

№ п/п	Показник	Проект					Середнє значення
		1	2	3	4	5	
1	Коефіцієнт перспективності грошових надходжень ($K_{ПН}$)	0,73	0,56	0,44	0,74	н/д	0,62
2	Коефіцієнт успішності нової версії програмного продукту ($K_{УНВ}$)	0,95	0,84	0,87	0,86	н/д	0,88
3	Показник «відставання» середовища розробки (B_C), років	0,9	1,5	1,0	0,7	0,8	0,99
4	Коефіцієнт залежності продукту від сторонніх компонентів ($K_{ЗК}$)	0,69	0,92	0,89	0,91	0,96	0,87
5	Навантаження вихідного коду на одного учасника проекту, *KSLOC*	17,41	6,48	5,51	12,35	6,51	9,65
6	Середня тривалість випуску однієї версії, років	0,3	0,3	0,5	1	0,5	0,52

У табл. 3.5 представлено результати розрахунку відповідних економіко-технічних показників для управління інноваціями по відношенню до проектів, які не використовували результати даного дослідження для управління інноваціями (на основі даних табл. Б.1 додатку Б).

Значення $K_{КС}$ (коефіцієнту коригуючого супроводу) при побудові табл. 3.4 і табл. 3.5 не розраховувалося, оскільки для його розрахунку необхідні дані, збирання яких при реалізації програмних проектів у АТ «Датекс Україна» не передбачається.

При розрахунку показників $K_{ПН}$, $K_{УНВ}$, B_C та $K_{ЗК}$ табл. 3.4 і табл. 3.5 використовувалися наведені нами у підрозділі 2.3 відповідні формули 2.2, 2.3, 2.4 та 2.5.

Таблиця 3.5

Результати розрахунку економіко-технічних показників для оцінки управління інноваціями для проектів, при реалізації яких не використовувалися складові запропонованої системи управління інноваціями на етапах життєвого циклу ПЗ

№ п/п	Показник	Проект						Середнє значення
		1	2	3	4	5	6	
1	Коефіцієнт перспективності грошових надходжень ($K_{ПН}$)	0,65	0,30	0,25	0,52	н/д	н/д	0,43
2	Коефіцієнт успішності нової версії програмного продукту ($K_{УНВ}$)	0,82	0,74	0,76	0,63	н/д	н/д	0,74
3	Показник «відставання» середовища розробки (B_C), років	1,1	1,3	2,4	1,4	1,3	1,6	1,5
4	Коефіцієнт залежності продукту від сторонніх компонентів ($K_{ЗК}$)	0,91	0,93	0,85	0,89	0,94	0,93	0,9
5	Навантаження вихідного коду на одного учасника проекту, *KSLOC*	5,60	2,99	5,29	3,82	6,36	6,31	5,1
6	Середня тривалість випуску однієї версії, років	0,5	0,7	0,7	1,2	1,5	1,7	1,1

Під час розрахунку показника «Навантаження вихідного коду на одного учасника проекту» використовувалася наступна формула:

$$N = \frac{S}{n} \tag{3.1}$$

де N – обсяг навантаження вихідного коду на одного учасника проекту, одиниць розміру коду;

S – розмір вихідного коду проекту, одиниць розміру коду;

n – кількість учасників проекту.

Показник, розрахований у відповідністю із формулою 3.1, дозволяє визначити, який обсяг вихідного коду програмного проекту приходиться на одного його учасника. За умов незмінності інших показників, більш високе значення показника N для певного проекту у порівнянні з іншими проектами буде свідчити про відповідно більш високий рівень ефективності управління даним проектом у порівнянні з іншими проектами. Це твердження випливає із встановленою у підрозділі 1.3 даної

роботи залежністю між обсягом вихідного коду і зростанням трудовитрат (згідно з формулами 1.2, 1.3, 1.5, 1.8, 1.13).

Аналізуючи дані табл. 3.4 і табл. 3.5, можна зробити висновок, що значення ключових показників для програмних проектів, при реалізації яких використовувалася запропонована у даному дисертаційному дослідженні система управління інноваціями на етапах життєвого циклу програмних проектів (табл. 3.4), відрізняються у кращу сторону у порівнянні з показниками програмних проектів, при реалізації яких дана система не використовувалася, зокрема:

- більш високе середнє значення коефіцієнту перспективності грошових надходжень $K_{ПН}$ (0,62 у порівнянні з 0,43) свідчить про більш високу здатність проектів приносити грошові надходження у майбутньому, оскільки для них надходження від реалізації нових копій продуктів є вищими у порівнянні з надходженнями від супроводу існуючих копій продуктів;
- більш високе середнє значення коефіцієнту успішності нової версії програмного продукту $K_{УНВ}$ (0,88 у порівнянні з 0,74) свідчить про краще сприйняття ринком нових версій продуктів і є непрямим доказом їх вищої якості;
- значно нижче середнє значення показника «відставання» середовища розробки B_C (0,99 року у порівнянні з 1,5 року) свідчить про використання більш сучасного середовища розробки і, відповідно, наявності кращих конкурентних позицій з технологічної точки зору;
- більш високе середнє значення показника навантаження вихідного коду на одного учасника проекту (9,65 *KSLOC* у порівнянні з 5,1 *KSLOC*) свідчить вищу ефективність управління програмними проектами, здатність долати складність ПЗ, відволікаючи менший обсяг трудових ресурсів, що в результаті сприяє зниженню собівартості реалізації програмних проектів;
- скорочення середньої тривалості випуску однієї версії програмного продукту (0,52 року у порівнянні з 1,1 року) свідчить про здатність команди проекту виводити на ринок нові версії програмних продуктів у більш короткий термін, що означає можливість швидше реагувати на тенденції ринку і зміну споживацьких уподобань, зміцнюючи при цьому конкурентні позиції.

Крім отриманих кількісних показників, які свідчать про практичне підтвердження отриманих теоретичних висновків даного дисертаційного дослідження, в результаті анкетування співробітників, які приймали участь при виконанні проектів, що досліджувалися, були також отримані і узгоджені з ними якісні показники.

Зокрема, учасниками проектів, при реалізації яких випробовувалися теоретичні положення даної роботи, було відзначено наступне:

- значно спростився і став прозорим процес управління інноваціями, чого вдалося досягти за рахунок формалізації найбільш критичних процедур даного процесу;
- спостерігається підвищення якості програмних продуктів, а також ріст їх конкурентноспроможності, чого вдалося досягти за рахунок використання сучасних інструментальних засобів і технологій розробки ПЗ;
- відбулося зростання мотивації персоналу при здійсненні інноваційних процесів, учасники проектів почали позитивно сприймати нові інструментальні засоби і технології розробки, а також стали активними прибічниками інноваційних процесів;
- з'явилася можливість використовувати елементи кількісного управління при реалізації програмних проектів, формалізуючи процеси прийняття рішень і зменшуючи вплив суб'єктивних факторів на прийняття важливих проектних рішень;
- за рахунок використання документа «Паспорт проекту» спростилося узгодження середовища розробки учасників проекту, скоротилися витрати часу на виконання непродуктивних операцій, зменшилися витрати часу на підготовку середовища розробки, виконання і тестування проекту, зменшилася кількість помилок, викликаних неуважністю учасників проекту при підготовці відповідного середовища проекту;
- спростилися комунікації між учасниками проекту, зменшилася необхідність у здійсненні неефективних комунікацій, чого вдалося досягти за рахунок стандартизації важливих внутрішніх процесів;

– за рахунок використання запропонованої системи управління використанням сторонніх компонентів значно спростилися процеси оновлення версій бібліотек сторонніх компонент, виділення власних розробок у незалежні бібліотеки коду та ін.

Слід зазначити, що запропонована система управління інноваціями дозволяє впровадити елементи кількісного управління процесом розробки ПЗ, що свідчить про наближення системи управління програмними проектами в цілому до високих вимог 4-го і 5-го рівнів зрілості згідно з міжнародними стандартами якості процесу розробки ПЗ SEI CMM/CMMI та 15504/SPICE.

В результаті, можна стверджувати, що запропонована нами система управління інноваціями на етапах життєвого циклу ПЗ дозволяє спростити управління інноваційними процесами при реалізації програмних проектів, що, насамперед, сприяє зростанню якості програмних продуктів і росту їх конкурентноздатності на ринку.

Висновки до третього розділу

1. Для вирішення проблеми поновлення версій сторонніх бібліотек коду та інструментарію, що використовується при розробці програмного забезпечення, доцільно ввести термін «кортеж проекту» по аналогією з терміном «кортеж програми».

2. Для контрою над актуальним станом кортежу проекту слід використати спеціальний документ, паспорт проекту. Цей документ має підтримуватися одним із керівників проекту і мати силу наказу в колективі розробників.

3. Має бути забезпечена стандартизація внутрішніх процесів організації, яка спростить контроль як за внутрішніми, так і за зовнішніми процесами, а також полегшить взаємодію учасників проектів.

4. Що стосується персоналу компанії, то слід не лише перебороти недовіру, а й зорієнтувати співробітників на підтримку інноваційних процесів. Дуже часто інновації зустрічають опір зі сторони найбільш досвідчених і авторитетних фахівців, оскільки вони здобули свої позиції за минулих, можливо суттєво застарілих техно-

логій, і намагаються їх не втратити. Для вирішення цієї проблеми слід використовувати комплекс заходів, спрямованих на те, щоб кардинальним чином змінити відношення співробітників до інновацій.

5. Слід активно заохочувати тих робітників, які є носіями інноваційних ідей і розробок. Необхідно не лише створювати для них всі необхідні умови, а й давати їм можливість приймати участь у вигодах підприємства, отриманих за рахунок їх розробок. Важливим моментом є орієнтація не лише матеріальні методи стимулювання праці подібних робітників, оскільки для них особливо важливою є потреба в самовираженні.

6. Система управління інноваціями має бути побудована таким чином, щоб усунути протиріччя між короткостроковими і довгостроковими цілями фірми, чи суттєво зменшити його за рахунок встановлення чіткої відповідності між цілями стратегічного рівня і задачами оперативного рівня;

7. Система економіко-технічних показників має бути побудована з урахуванням стану і етапів життєвого циклу програмного проекту.

8. В результаті практичної перевірки теоретичних положень дисертації, виконаної у підрозділі 3.4, було встановлено, що розроблена нами система управління інноваціями на етапах життєвого циклу ПЗ дозволяє отримати зростання важливих показників програмних проектів, що свідчить про вирішення нами поставленої наукової задачі.

Основні положення даного розділу знайшли відображення у наступних наукових працях здобувача [41, 37, 38, 39, 40]:

1. Колдовський В.В. Визначення економічних параметрів інноваційних процесів на етапах життєвого циклу програмного забезпечення // Вісник Української академії банківської справи. – 2005. – № 1. – С. 105-113.
2. Колдовський В. В. Використання BPML як універсальної мови опису бізнес-процесів // Тезисы докладов научно-технической конференции преподавателей, сотрудников, аспирантов и студентов экономического факультета Сумс-

кого государственного университета. – Сумы: Изд-во СумГу, 2002. – С. 121-122.

3. Колдовський В. В. Окремі аспекти управління інноваційними процесами при розробці ПЗ // Проблеми і перспективи розвитку банківської системи України: Збірник наукових праць. Т. 13. – Суми: ВВП «Мрія-1» ЛТД, УАБС НБУ, 2005. – С. 185-198.
4. Колдовський В. В. Створення ефективної системи автоматизованого управління бізнес-процесами підприємства // Збірник матеріалів Першої міжнародної науково-практичної конференції „Сучасні проблеми управління”. – К.: „Політехніка”, 2001. – С. 291-293.
5. Колдовський В. В., Шамота Г. М. Інтенсифікація інновацій: безперервність процесу і орієнтація на персонал // Вісник Хмельницького національного університету. – 2005. – № 3. – С. 165-168.

ВИСНОВКИ

У дисертації наведені теоретичне узагальнення і нове вирішення наукової задачі, що виявляється в необхідності вдосконалення підходів до управління програмними проектами на основі розробки системи управління інноваціями на етапах життєвого циклу ПЗ. Вирішення задачі полягає у розробці і обґрунтуванні науково-методичних підходів і конкретних практичних рекомендацій, які можуть бути використані з метою покращення показників програмних проектів, за рахунок орієнтації на постійне поліпшення процесу на основі управління інноваціями на етапах життєвого циклу ПЗ.

Одержані результати проведеного дослідження дозволили зробити наступні висновки:

- інновації становлять один з найважливіших факторів, який є джерелом розвитку галузі розробки ПЗ. ПЗ як продукт інтелектуальної праці людини характеризується надзвичайно високим рівнем складності, що, в свою чергу, є основною причиною проблем, які виникають при побудові ефективної системи управління на етапах життєвого циклу ПЗ;
- більшість існуючих підходів до управління програмними проектами визначають у якості однієї із найважливіших ціль постійного поліпшення процесу розробки програмного забезпечення, однак, як правило, не сфокусовані на питаннях, яким саме чином можна досягти даної цілі, вважаючи, що фірма-розробник має досягти її еволюційно. На наш погляд, подібний підхід не відповідає умовам сучасного конкурентного середовища, які ставлять в нерівні умови невеликі і починаючі компанії у порівнянні з тими, які тривалий час працюють на ринку, мають стабільні фінансові показники і володіють пакетом патентів, що захищають інтелектуальну власність. У даному разі слід відмовитися від поступового еволюційного росту показників компанії-розробника ПЗ і зосередитися на впровадженні системи управління інноваціями, яка надасть змогу здійснити якісний перехід до нового рівня її конкурентоспроможності;

– економічний механізм управління інноваціями на етапах життєвого циклу програмного забезпечення має вирішувати такі задачі як пошук та розробку інновацій за рахунок зовнішніх і внутрішніх джерел, здійснювати оцінку доцільності впровадження інновацій з урахуванням економічних і технічних факторів, забезпечувати розробку плану впровадження інновацій, враховуючи можливість відміни впровадження, забезпечувати управління реалізацією інновацій на практиці, здійснювати аналіз процесу і формування висновків для подальшого використання, забезпечувати безперервність процесу і його органічне поєднання із загальною системою управління компанією на усіх рівнях;
– основною ідеєю економічного механізму управління інноваціями має бути контрольованість процесу і його відповідність як короткостроковим, так і довгостроковим цілям фірми. Інноваційний процес має бути безперервним і циклічно повторюватися на усіх етапах життєвого циклу програмного забезпечення;
– визначено, що існують обмеження використання суто економічних методів при прийнятті рішень стосовно доцільності впровадження інновацій, причиною яких є дія таких факторів, які складно спрогнозувати. Було запропоновано при прийнятті інноваційного рішення приймати до уваги також «неекономічну складову», що має включати такі фактори, як перспективність технології, її потенціал до розвитку, прийняття персоналом фірми та ін.;
– було встановлено, що для сучасних програмних проектів характерним є високий рівень складності, який має тенденцію до зростання. Одним з найбільш ефективних способів подолання складності є використання сторонніх компонентів і бібліотек коду, тенденція до зростання долі яких у загальному обсязі коду програмних проектів є сталою, і сягає 99% по відношенню до загального обсягу проекту. У підрозділі 2.2 запропоновано комплексну систему управління впливом сторонніх компонентів на процес розробки ПЗ, яка має включати такі елементи, як: визначення потреб у сто-

ронніх компонентах, управління використанням сторонніх компонентів, передбачати поновлення версій сторонніх компонентів, розглядати механізм виділення власних розробок у незалежні компоненти, передбачати процедуру заміни власного коду сторонніми компонентами і здійснювати оцінку сторонніх компонентів на основі техніко-економічного аналізу і порівняння альтернатив;

– система управління інноваціями має бути побудована таким чином, щоб усунути протиріччя між короткостроковими і довгостроковими цілями фірми, чи суттєво зменшити його за рахунок встановлення чіткої відповідності між цілями стратегічного рівня і задачами оперативного рівня. У підрозділі 2.3 нами запропонована система економіко-технічних показників для оцінки управління інноваціями у сфері ПЗ, яка враховує дане завдання і дозволяє побудувати систему управління інноваціями на етапах життєвого циклу ПЗ таким чином, щоб забезпечити досягнення стратегічних цілей узгоджено із виконанням оперативних завдань;
– визначено, що система управління інноваціями має бути побудована з урахуванням стану і етапів життєвого циклу програмних проектів, передбачати нестабільність зовнішніх факторів, від яких залежить успіх проекту, що відображено у запропонованому нами у підрозділі 3.1 терміну «кортеж проекту» та документі «паспорт проекту»;
– у підрозділі 3.1 визначено необхідність стандартизації внутрішніх процесів під час реалізації програмних проектів, зокрема, запропоновано визначати внутрішньофірмовими стандартами ключові складові етапів життєвого циклу ПЗ;
– визначено, що значний вплив на кінцевий результат інноваційного процесу становить відношення до нього персоналу фірми, його бажання сприяти впровадженню нових технологій, чи, навпаки, відстоювати застарілі методи роботи і технології, перешкоджаючи розповсюдженню інновацій. У зв’язку з цим у підрозділі 3.2 запропоновано конкретні підходи до створення ефективної системи управління персоналом для забезпечення інтенсифі-

кації інноваційного процесу при реалізації програмних проектів, зокрема, головна увага сфокусована на активізації участі персоналу у інноваційних процесах;

- враховуючи, що система управління інноваціями має бути невід'ємною складовою загальної системи управління життєвим циклом ПЗ, у підрозділі 3.3 нами розглянуто, яким чином мають поєднуватися внутрішні бізнес-процеси компанії-розробника ПЗ і запропонована нами комплексна система управління інноваціями на етапах життєвого циклу програмного забезпечення. Здійснена оцінка здатності інтеграції системи управління інноваціями у загальну систему управління програмними проектами в залежності від моделі життєвого циклу, яка використовується при роботі над проектом;
- у підрозділі 3.4 здійснено перевірку запропонованої системи управління інноваціями на етапах життєвого циклу програмного забезпечення, що дало змогу перевірити правильність отриманих нами теоретичних висновків;
- отримані в результаті проведеного дослідження науково-методичні підходи і практичні рекомендації можуть бути використані для покращення показників управління життєвим циклом програмного забезпечення компаній-розробників ПЗ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алдошин С. М., Зинов В. Г. Службы коммерциализации результатов исследований в научных организациях // Инновации. – 2003. – № 4. С. 15-23.
2. Астелс Д., Миллер Г., Новак М. Практическое руководство по экстремальному программированию.: Пер. с англ. – М.: «Вильямс», 2002. – 320 с.
3. Бабенко Л. П., Лавріщева К. М. Основи програмної інженерії: Навч. посіб. – К.: Т-во «Знання», КОО, 2001. – 269 с.
4. Бабий А. О термине «программное изделие» // Тезисы 2-й Всесоюзной конференции «Технология программирования». – К.: ИК АН УССР. – 1986. – С. 21-23.
5. Багриновский К.А., Бусыгин В.П. Математика плановых решений. – М.: Наука, 1980. – 224 с.
6. Бек К. Экстремальное программирование: разработка через тестирование. Библиотека программиста – СПб.: Питер, 2003. – 224 с.
7. Богдановський В. Г., Колдовський В. В. Софтверний фактор у питаннях створення ефективної системи автоматизованого управління бізнес-процесами // Вісник Сумського державного аграрного університету. Серія: „Економіка та менеджмент”. Вип. 2. – Суми, 2001. – С. 177-180.
8. Брукс Ф. Мифический человеко-месяц или как создаются программные системы. – СПб.: Символ-Плюс, 2000. – 306 с.
9. Буч Г. Объектно - ориентированный анализ и проектирование. – СПб.: Невский диалект, 1998. – 560 с.
10. Буч Г., Рамбо Д., Джекобсон А. UML. Руководство пользователя. – СПб.: ДМК Пресс, Питер, 2004. – 432 с.
11. Василенко В. О., Шматько В. Г. Інноваційний менеджмент: Навчальний посібник. За ред. В. О. Василенко. – К.: ЦУЛ, Фенікс, 2003 – 440 с.

12. Гальчинський А., Геєць В., Семиноженко В. Україна: наука та інноваційний розвиток. – К.: Вища школа, 1997. – 67 с.

13. Гейтс Б. Бизнес со скоростью мысли. Изд. 2-е, испр. – М.: ЭКСМО-Пресс, 2001. – 480 с.

14. Герасимчук М. Проблеми інвестиційної та інноваційної політики України // Економіка України. – 1997 – № 8. – С. 94-95.

15. Грибачев К. Г. Delphi и Model Driven Architecture. Разработка приложений баз данных. – СПб.: Питер, 2004. – 348 с.

16. Економіка України: підсумки перетворень та перспективи зростання. За редакцією академіка НАН України В.М.Гейця. – Х.: Форт, 2000. – 432 с.

17. Еталонні архітектури MSA. – К.: Майкрософт Україна; К.: Видавнича група BHV, 2005. – 352 с.

18. Жалдак М. І., Рамський Ю. С. Елементи програмування. – К.: «Радянська школа», 1976. – 208 с.

19. Житенко Е. Д. Эффективность стимулирования инноваций // Инновации. – 2004. – № 3. – С. 20-25.

20. Инновационный менеджмент: Справ. пособие/ под ред. П.Н. Завлина, А.К. Казанцева, Л.Э. Миндели. Изд. 2-е, пере-раб. и доп. – М., ЦИСН, 1998. – 568 с.

21. Инновационный менеджмент: Учебник для вузов/ С. Д. Ильенкова, Л. М. Гохберг, С. Ю. Ягудин и др.; Под ред. С. Д. Ильенковой. – М.: ЮНИТИ-ДАНА, 2002. – 327 с.

22. Инновационный менеджмент: Учебное пособие / Под ред. д.э.н. проф. Л. Н. Оголевой. – М.: ИНФРА-М, 2003. – 238 с.

23. Ілляшенко С.М. Управління інноваційним розвитком: проблеми, концепції, методи: Навчальний посібник. – Суми: ВТД «Університетська книга», 2003. – 278 с.

24. Ілляшенко С.М., Божкова В.В. Управління ризиками інновацій: Монографія / За ред. д.е.н., проф. С. М. Ілляшенка. – Суми: ВТД «Університетська книга», 2004. – 214 с.

25. Інформаційні системи в менеджменті: Навчальний посібник / Глівенко С.В., Лапін Є.В., Павленко О.О. та ін. – Суми: ВТД «Університетська книга», 2005. – 407 с.

26. Йордон Э. Путь камикадзе. Как разработчику программного обеспечения выжить в безнадежном проекте. – М.: Лори, 2002. – 272 с.

27. Калбертсон Р., Браун К., Кобб Г. Быстрое тестирование.: Пер. с англ. – М.: Вильямс, 2002. – 384 с.

28. Кантор М. Управление программными проектами. Практическое руководство по разработке успешного программного обеспечения.: Пер. с англ. – М.: Вильямс, 2002. – 176 с.

29. Каплан Р., Нортон Д. Сбалансированная система показателей. От стратегии к действию. – М.: ЗАО «Олимп-бизнес», 2003. – 210 с.

30. Кислий В. М., Колдовський В. В. Проблеми використання новітніх технологій у сфері розробки програмного забезпечення // Тезисы докладов научно-технической конференции преподавателей, сотрудников, аспирантов и студентов экономического факультета Сумского государственного университета. – Сумы: Изд-во СумГу, 2002. – С. 64-65.

31. Коганов А., Романюк С. Экономический подход к понятию надежности программы // Открытые системы. – 1995. – № 3. – С. 25-37.

32. Козьменко С. М., Колдовський В. В. Сучасні моделі управління процесом розробки програмного забезпечення // Проблеми і перспективи розвитку банківської системи України: Збірник наукових праць. Т. 12. – Суми: ВВП «Мрія-1» ЛТД, УАБС НБУ, 2005. – С. 43-49.

33. Кокурин Д. И. Инновационная деятельность – М.: Экзамен, 2001. – 576 с.

34. Колдовский В. В. Разработка ПО: модели жизненного цикла // Компьютерное обозрение. – 2005. – № 24. – С. 56-59.

35. Колдовский В. В. Стандарты написания исходного кода в Delphi // Портал разработчиков в среде Delphi DelphiPlus: http://www.delphiplus.org/articles/delphi/source_code_standards/index.html

36. Колдовский В., Козьменко С.Н. Перспективы и возможности отечественного рынка бесплатного программного обеспечения // Збірник наукових праць Сумського державного університету/ Механізм регулювання економіки, економіка природокористування, економіка підприємства та організація виробництва. 1999 – Випуск 3 (99). – С.139-141.

37. Колдовський В. В. Використання BPML як універсальної мови опису бізнес-процесів // Тезисы докладов научно-технической конференции преподавателей, сотрудников, аспирантов и студентов экономического факультета Сумского государственного университета. – Сумы: Изд-во СумГу, 2002. – С. 121-122.

38. Колдовський В. В. Окремі аспекти управління інноваційними процесами при розробці ПЗ // Проблеми і перспективи розвитку банківської системи України: Збірник наукових праць. Т. 13. – Суми: ВВП «Мрія-1» ЛТД, УАБС НБУ, 2005. – 270 с., С. 185-198.

39. Колдовський В. В. Створення ефективної системи автоматизованого управління бізнес-процесами підприємства // Збірник матеріалів Першої міжнародної науково-практичної конференції „Сучасні проблеми управління”. – К.: „Політехніка”, 2001. – С. 291-293.

40. Колдовський В. В., Шамота Г. М. Інтенсифікація інновацій: безперерв-ність процесу і орієнтація на персонал // Вісник Хмельницького націо-нального університету. – 2005. – № 3. – С. 165-168.

41. Колдовський В.В. Визначення економічних параметрів інноваційних процесів на етапах життєвого циклу програмного забезпечення // Вісник Української академії банківської справи. – 2005. – № 1. – С. 105-113.

42. Коренной А. А., Карпов В. И. Курс инновационного менеджмента. – К.: НИИ Статистики, 1997. – 336 с.

43. Ларман К. Применение UML и шаблонов проектирования.: Пер. с англ. – М.: Вильямс, 2001. – 496 с.

44. Леффингуэлл Д., Уидриг Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход.: Пер. с англ. – М.: «Вильямс», 2002. – 448 с.

45. Литвинов Ю. С. Частное мнение о стиле оформления программного кода // Записки профессионала. Авторский сайт: http://lus.hostingcenter.ru/rus/01/0119/0119.htm

46. Медынский В. Г., Шаршукова Л. Г. Инновационное предпринимательство: Учебное пособие. – М.: ИНФРА-М, 1997. – 240 с.

47. Мельников О. А. Механизмы планирования и управления крупными инновационными проектами, обеспечивающие сокращение сроков их реализации // Инновации. – 2002. – № 5. – С. 44-45.

48. Мескон М. Х., Альберт М., Хедоури Ф. Основы менеджмента: Пер. с англ. – М.: Дело, 1997. – 704 с.

49. Морозов Ю. П. Инновационный менеджмент. – М.: ЮНИТИ-ДАНА, 2000. – 446 с.

50. Морозов Ю. П., Гаврилов А. И., Городнов А. Г. Инновационный менеджмент: Учеб. пособие для вузов. – 2-е изд. перераб. и доп. – М.: ЮНИТИ-ДАНА, 2003. – 471 с.

51. Москаленко В. П. Экономические новации: поиск и внедрение. – Сумы: «Довкілля», 2004. – 366 с.

52. Несторенко О. Про перехідний період економічного розвитку України // Економіка України. – 1994. – №11. – С. 39-43.

53. Онишко С. Структура інвестиційного ресурсу України та перспективи економічного зростання //Економіст. – 2001.– № 11.– С. 58-61.

54. Павленко Л. А. Корпоративні інформаційні системи: Навчальний посібник. – Х.: ВД «ІНЖЕК», 2003. – 260 с.

55. Пересада А.А. Інвестиційні процеси в Україні. – К.: "Лібра", 1998. – 389 с.

56. Портер М. Международная конкуренция: Пер. с англ. – М.: Прогресс, 1993. – 895 с.

57. Програмні документи MSA. – К.: Майкрософт Україна; К.: Видавнича група BHV, 2005. – 56 с.

58. Проектування інформаційних систем: Посібник / За ред. В. С. Пономаренка. – К.: «Академія», 2002. – 488 с.

59. Рогалев Д., Лебедев И., Хабалова Н. Концептуальный механизм стимулирования инновационных процессов // Инновации. – 2004. – № 3. – С. 26-31.

60. Роджерс Б. Путь успеха: Как работает корпорация IBM./ Пер. с англ. – СПБ.: Азбука-Терра, 1997. – 256 с.

61. Савельєв Є.В., Куриляк В.Є. Галузева структура української економіки: нові підходи та прогнози. // Фінанси України. – 1997. –№ 1.– С. 33.

62. Сакс Дж., Пивоварський О. Економіка перехідного періоду. Уроки для України: Пер. з англ. – К.: Основи, 1996. – 345 с.

63. Салливан Э. Время – деньги. Создание команды разработчиков программного обеспечения / Пер. с англ. – М.: ИТД Русская редакция, 2002. – 368 с.

64. Смоляк С. А. Три проблемы теории эффективности инвестиций // Экономика и математические методы. – 1999. – т 35, № 4. – С. 87-104.

65. Статистика науки и инноваций. Краткий терминологический словарь/Под ред. Л.М.Гохберга – М.: Центр исследований и статистика науки, 1996. – С. 30-31.

66. Тексейра С., Пачеко К. Delphi 5. Руководство разработчика: В 2 т. – М.: Вильямс, 2000. – Т.1: Основные методы и технологии программирования. – 832 с.

67. Ткаченко А. В. Стандарт стилевого оформления исходного кода DELPHI // Королевство Дельфи: http://www.delphikingdom.com/article/coderules.htm

68. Трифилова А., Коршунов И. Современный инновационный менеджмент // Инновации. – 2003. – № 2-3. – С. 85-90.

69. Уткин Э. А., Морозова Н. И., Морозова Г. И. Инновационный менеджмент – М.: АКАЛИС, 1996. – 208 с.

70. Фирма «Олис». Стандарт оформления кода Delphi // Официальный сайт фирмы «Олис»: http://download.olis.ru/download/standard-delphi.zip

71. Філіпченко А.С., Бандера В.З. та ін. Перехідна українська економіка: стан і перспективи // За ред. А.Філіпченка, В.Бандери. – К.: Академія, 1996. – 224 с.

72. Хаустов В., Панфілова Т. Інноваційні процеси в Україні: реалії і перспективи розвитку // Економіст. – 2002. – № 3. – С. 54-59.

73. Чернявская Н., Чернявский Ю. Требования к проекту // Компьютеры+Программы. – 2005. – № 2. – С. 66-74.

74. Шафер Д., Фатрелл Р., Шафер Л. Управление программными проектами: достижение оптимального качества при минимуме затрат.: Пер. с англ. – М.: «Вильямс», 2003. – 1136 с.

75. Шумпетер Й. Теория экономического развития (Исследование предпринимательской прибыли, капитала, кредита, процента и цикла конъюнктуры). – М.: Прогресс, 1982. – 455 с.

76. Юрчук Н.И. Современные технологии и интеграция в международную банковскую систему // Банковские технологии. – 1998. – № 1. – С.13-20.

77. Янсен Ф. Эпоха инноваций: Пер. с англ. – М.: ИНФРА-М, 2002. – 308 с.

78. Ярошенко С. П., Колдовський В. В. Умови ефективної реалізації проектів автоматизації в промисловості // Механізм регулювання економіки, економіка природокористування, економіка підприємства та організація виробництва. – Суми: Вид-во СумДУ. – 2002. – № 1-2. – С. 152-155.

79. Aaen I., Arent J., Mathiassen L., Ngwenyama O. (2001). A Conceptual MAP of Software Process Improvement // Scandinavian Journal of Information Systems, Special Issue on Trends in the Research on Software Process Improvement in Scandinavia. – 2001. – №.13. – P. 123-146.

80. Abran A., Robillard P. Function Points: A Study of their Measurement Processes and Scale Transformations // Journal of Systems and Software. – 1994. – № 2. – P. 171-184.

81. Aen I., Pries-Heje J. Standardizing Software Processes – An Obstacle for Innovation // IT Innovation for Adaptability and Competitiveness / Edited by B. Fitzgerald, E. Wynn. – Boston: Kluwer Academic Publishers, 2004. – P. 117-133.

82. Albrecht A. Measuring Application Development Productivity // Proc. Joint SHARE/GUIDE/IBM Application Development Simposium. – Monterey (USA). – 1979. – P. 83-92.

83. Albrecht A., Gaffney J. Software Function, Source Lines of Code, and Development Effort Prediction // IEEE Transactions on Software Engineering, 1983. – № 6. – P. 639-648.

84. Andersen K., Vendelo M. The Past and Future of Information Systems. – Burlington (USA): Elsevier Butterworth-Heinemann, 2004. – 274 p.

85. Arkin A. Business Process Modeling Language (BPML) Specification (Version 1.0) – Business Process Management Initiative. – Alameda de las Pulgas, 2002. – 98 p.

86. Baca C. Project Manager's Spotlight on Change Management. – Alameda (USA): Harbor Light Press, 2005. – 146 p.

87. Baines, Robin. Across Disciplines: Risk, Design, Method, Process, and Tools // IEEE Software. – 1998. – July/August. – P 61-64.

88. Baumeister A., Ilg M. Cost Management of Software Developments – an Activity-Based Approach // IFSAM http://www.ifsam.org/2004/papers/227%20final.pdf. – 10 p.

89. Beck K., Fowler M. Planning Extreme Programming. – Addison Wesley, 2000. – 160 p.

90. Berchuk S., Appleton B. Software Configuration Management Patterns: Effective Teamwork, Practical Integration. – Addison Wesley, 2002. – 256 p.

91. Boehm B. Spiral Development: Experience, Principles, and Refinements / Spiral Development Workshop, Feb. 9, 2000 // Software Engineering Institute. – Pittsburgh (USA). – 2000. – 37 p.

92. Boehm B. A Spiral Model of Software Development and Enhancement // IEEE Computer. – 1988. – № 5. – pp. 61-72.

93. Boehm B. Software Engineering Economics. – Prentice Hall PTR, 1981. – 767 p.

94. Boehm B., Egyed A., Kwan J., Port D., Shah A., Madachy R. Using the WinWin Spiral Model: A Case Study // IEEE Computer. – 1998. – № 7. – pp. 33-44.

95. Briand L., Emam K., Wieczorek I. Explaining the Cost of European Space and Military Projects // Proc. International Conference on Software Engineering. – 1999. – P. 303-312.

96. Bush M., Dunaway D. CMMI Assessments: Motivating Positive Change. – Addison Wesley Professional, 2005. – 432 p.

97. Calvert C. Delphi Style Guide // Borland Developer Network: http://community.borland.com/soapbox/techvoyage/article/1,1795,10280,00.html

98. Capability Maturity Model for Software, Version 1.1 / Paulk M., Curtis B., Chrissis M., Weber C. – Pittsburgh (USA): Software Engineering Institute, 1993. – 91 p.

99. Capability Maturity Model Integration (CMMI^{SM}), Version 1.1. (CMMI-SE/SW/IPPD/SS, V1.1) / CMMI Product Team. – Pittsburg (USA): Software Engineering Institute, 2002. – 709 p.

100. Chabchoub N., Niosi J. Explaining the propensity to patent computer software // Technovation. – 2005. – № 25. – P. 971-978.

101. Chulani S. Bayesian Analysis of Software Cost and Quality Models // Proc. International Conference on Software Maintenance (ICSM 2001). – Florence (Italy). – 2001. – P. 565-569.

102. CMMI SCAMPI Distilled Appraisals for Process Improvement / Ahern D., Armstrong J., Clouse A. and others. – Addison Wesley Professional, 2005. – 240 p.

103. Cockburn A. Crystal Clear: A Human-Powered Methodology for Small Teams. – Addison-Wesley Professional, 2004. – 336 p.

104. COCOMO II Model Definition Manual / Boehm B., Abts C., Clark B., and others. – Los Angeles (USA): University of Southern California. – 1999. – 37 p.

105. COCOMO II Model Definition Manual. Version 1.4 / Boehm B., Abts C., Clark B., and others. – Los Angeles (USA): University of Southern California. – 1997. – 68 p.

106. Curtis B. A mature view of the CMM // American Programmer. – 1994. – № 7, P. 19-28.

107. Duncan W. A Guide to the Project Management Body of Knowledge. – Sylva (USA): PMI Publishing Division, 1996. – 176 p.

108. Extreme Chaos. The Standish Group International Inc. – The Report of the Standish Group International, 2005. – 12 p.

109. Finnie G., Wittig G. A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models // Journal of Systems and Software. – 1997. – № 39. – P. 281-289.

110. Fioravanti F. Skills for Managing Rapidly Changing IT Projects. – Hershey (USA): IRM Press, 2005. – 264 p.

111. Gabor P. When can we call a program properly commented? // Borland Developer Network: http://community.borland.com/article/0,1410,26678,00.html

112. Goodman P. Software Metrics: Best Practices for Successful IT Management. – Brookfield: Rothstein Associates, 2004. – 264 p.

113. Gremba J., Myers C. The IDEAL^{SM} Model: A Practical Guide for Improvement // Bridge. – 1997. – № 3. – pp. 19-23.

114. Hammer M., Champy J. Reengineering the Corporation. A Manifesto for Business Revolutions. HarperBusiness, 1993. – 256 p.

115. Hass A. Configuration Management Principles and Practice. – Addison Wesley, 2002. – 432 p.

116. Heales J., Raduescu C. The Role of Management Incentives in Successful Information Systems Development and Implementation // Proc. Americas Conference on Information Systems. – Atlanta (USA). – 2004, P. 806-810.

117. Henry W. Controlling Software Costs. White Paper. – Jamul (USA): Software Cost Control, Inc., 2002. – 34 p.

118. Highsmith J. Agile Project Management: Creating Innovative Products. – Addison Wesley, 2004. – 312 p.

119. House H. C., Price R. L. The Return Map: Tracking Product Teams // Harvard Business Review. – 1991. – № 1. – pp. 92-100.

120. Humphrey W. Why Big Software Projects Fail: The 12 Key Questions // CrossTalk. – 2005. – № 3. – pp. 25-29.

121. Indy Project. Indy Coding Conventions and Guidelines // Indy Project Official site: http://www.indyproject.org/Teams/Core/Standards/index.html

122. Jeffery R. Software Quality Accreditation in the Australian Context. – The Report of the Software Quality Accreditation Working Party, 2005. – 39 p.

123. Jensen R. A Comparison of the Jensen and COCOMO Schedule and Cost Estimation Models // Proc. International Society of Parametric Analysis. – 1984. – P. 96-106.

124. Jones C. The Economics of Object-Oriented Software. – Burlington (USA): Software Productivity Research Inc, 1997. – 19 p.

125. Jones C. Applied Software Measurement. – New York (USA): McGraw-Hill, 1991. – 494 p.

126. Jones L. Software Process Improvement and Product Line Practice: Building on Your Process Improvement Infrastructure. – Software Engineering Institute. – Pittsburgh (USA), 2004. – 18 p.

127. Jung H., Goldenson D. CMM-Based Process Improvement and Schedule Deviation in Software Maintenance. – Pittsburgh (USA): Software Engineering Institute, 2003. – 46 p.

128. Kaisler S. Software Paradigms. – New Jersey (USA): John Wiley and Sons, 2005. – 440 p.

129. Kaluzniacky E. Managing Psychological Factors in Information Systems Work: An Orientation to Emotional Intelligence. – Hershey (USA): Information Science Publishing, 2004. – 278 p.

130. Kay R., Del Prete C. The Corporate Refresh Cycle Grinds into Motion. – IDC Research Document, 2003. – 12 p.

131. Keefer G., Lubecka H. The CMMI in 45 minutes. Stuttgard: AVOCA GmbH, 2002. – 10 p.

132. Kitchenham B., Taylor N. Software Project Development Cost Estimation // Journal of Systems and Software. – 1985. – № 5. – P. 267-278.

133. Koch S. Free/Open Source Software Development. – Hershey (USA): Idea Group Publishing, 2005. – 309 p.

134. Marasco J. The Software Development Edge: Essays on Managing Successful Projects. – Addison Wesley Professional, 2005. – 335 p.

135. Mattson J., Barett B., Mellichamp J. Software Development Cost Estimation Using Function Points // IEEE Transactions on Software Engineering. – 1994. – № 4. – P. 275-287.

136. McConell S. Professional Software Development: Shorter Schedules, Higher Quality Products, More Successful Projects, Enhanced Careers. – Addison Wesley, 2003. – 272 p.

137. McConell S. Code Complete, Second Edition. – Microsoft Press, 2004. – 960 p.

138. McKenzie R., Lee D. Managing Through Incentives: How to Develop a More Collaborative, Productive, and Profitable Organization. – Oxford University Press, – 1998. – 337 p.

139. Meeting O. Life cycle of Database Application // Borland Developer Network, http://bdn.borland.com/article/0,1410,28994,00.html

140. Microsoft Corporation. Visual Studio Coding Techniques // MSDN Library: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vsent7/html/vxconCodingTechniques.asp

141. Microsoft Solutions Framework Version 3.0 Overview. White Paper / Lory G., Campbell D., Robin A. and Others. – Redmond (USA): Microsoft, 2002. – 30 p.

142. Microsoft Solutions Framework. MSF Process Model Version 3.1. White Paper / Getchell S., Hargrave L., Haynes P. and Others. – Redmond (USA): Microsoft, 2002. – 47 p.

143. Microsoft Solutions Framework. MSF Team Model Version 3.1. White Paper / Haynes P., Robin A., Paschino E. and Others. – Redmond (USA): Microsoft, 2002. – 44 p.

144. Narayanan V. K. Managing technology and innovation for competitive advantage – New Jersey (USA): Prentice-Hall, 2001. – 510 p.

145. Nelson B., Economy P. The Management Bible. – New Jersey: John Wiley & Sons, 2005. – 296 p.

146. Oligny S., Bourque P., Abran, A., Fournier, B. Exploring the Relation Between Effort and Duration in Software Engineering Projects // Proc. World Computer Congress. – Beijing (China). – 2000. – P. 175-178.

147. Oligny, S.; Abran, A., On the Compatibility Between Full Function Points and IFPUG Function Points // Proc. of the 10th European Software Control and Metric Conference (ESCOM SCOPE 99). – Herstmonceux Castle (England). – 1999. – P. 10-19.

148. Parnas D. L. Software Engineering Programs Are Not Computer Science Programs // IEEE Software. – 1999. – Nov/Dec – P. 19-30.

149. Paulk M. A Comparison of ISO 9001 and the Capability Maturity Model for Software. – Pittsburgh (USA): Software Engineering Institute, 1993. – 78 p.

150. Pugh K. Prefactoring.– O'Reilly, 2005. – 238 p.

151. Putnam L., Meyers W. Measures For Excellence : Reliable Software On Time, Within Budget. – Prentice Hall PTR, 1991. – 400 p.

152. Raduescu C., Heales J. Incentives and Their Effects on Information Systems Projects // Proc. Thirteenth European Conference on Information Systems. – Regensburg (Germany). – 2005: http://is.lse.ac.uk/asp/aspecis/20050040.pdf

153. Rask R., Laamanen P., Lyyttinen K. Simulation and Comparison of Albrecht's Function Point and DeMarco's Function Bang Metrics in a CASE Environment // IEEE Transactions on Software Engineering. – 1993. – № 7. – P. 661-671.

154. Redwine S., Riddle W. Software technology maturation // Proc. 8th International Conference on Software Engineering (ICSE-85). – Washington (USA). – 1985. – P. 189-200.

155. Reifer D., Boehm B., Chulani S. The Rosetta Stone: Making COCOMO 81 Estimates Work with COCOMO II // Crosstalk. – 1999. – № 2. – P. 11-15.

156. Rittinghouse J. Managing Software Deliverables: A Software Development Management Methodology. – New York: Digital Press, 2004. – 315 p.

157. Schalken J. Research Methods for the Empirical Assessment of Software Processes // Proc. 17th Conference on Advanced Information Systems Engineering (CAiSE'05). – Porto (Portugal). – 1995.: http://lbd.epfl.ch/caise05dc/Final_version/schalken-dc-caise-05.pdf

158. Selznak, S. We Are Morons: a quick look at the Win2k source // http://www.kuro5hin.org/story/2004/2/15/71552/7795

159. Shepperd M., Schofield C. Estimating Software Project Effort Using Analogies // IEEE Transactions on Software Engineering. – 1997. – № 23. – P. 736-743.

160. Software Cost Estimation with COCOMO II / Boehm B., Horowutz E., Madachy R. and others. – Prentice Hall PTR. – 2000. – 544 p.

161. Software Engineering Institute. Annual Report 2003 / Software Engineering Institute. – Pittsburg (USA): Software Engineering Institute, 2004. – 102 p.

162. Srinivasan K., Fisher D. Machine Learning Approaches to Estimating Software Development Effort // IEEE Transactions on Software Engineering. – 1995. – № 2. – P. 126-137.

163. USC COCOMO II Reference Manual / University of Southern California. – Los Angeles (USA): University of Southern California, 1999. – 86 p.

164. Using Knowledge Management to Drive Innovation / American Productivity & Quality Center. – APQC, 2003 – 194 p.

165. Viardot E. Successful Marketing Strategy for High-Tech Firms. – Boston: Artech House, 2004. – 306 p.

166. Walkerden F., Jeffery R. Software Cost Estimation: A Review of Models, Process, and Practice // Advances in Computers. – 1997. – № 44. – P. 59-125.

167. Walston C., Felix C. A Method of Programming Measurement and Estimation // IBM Systems Journal. – 1977. – № 1. – P. 54-73.

168. Wiegers K. Stop Promising Miracles // Software Development. – 2000. – № 2. – P. 49-57.

169. Wit B., Meyer R. Strategy: Process, Content, Context. 2nd edition. – London: Thomson Learning, 2002. – 1252 p.

170. Zultner R. TQM for technical teams // Communications of the ACM. – 1993. – № 10. – P. 79-91.

ДОДАТКИ

Додаток А
Приклад документу «Паспорт проекту» (рос. мов.)

Паспорт проекта «Рапорт+Статоценка»

версия документа: 1.0 от 03.08.2003
актуальная версия проекта: 1.5.1 от 03.08.2003

Ответственный

_____

Параметр	Значение	Примечание	Особенности работы, выявленные проблемы и пути их устранения
1. Среда исполнения проекта			
Окружение выполнения	Семейство операционных систем Microsoft Windows 2000/XP	Проект предназначен для семейства операционных систем архитектуры Windows NT, начиная с Windows 2000. Проект не тестировался в окружении Windows 2003, однако планируется его сопровождение под этой ОС. В качестве исключения, по требованию заказчика, допускается использование клиентской части приложения на операционных системах семейства Windows 9x/Me	
Сервер БД	FireBird 1.5 для платформы Win32	Проект тестировался под FireBird 1.5 RC4. Используется UDF-библиотека собственной разработки V_UDF_Date.dll При исполнении на локальной машине необходимо обязательно указывать имя компьютера или служебное слово «localhost» при соединении с БД (в настройках БД)	FireBird 1.5 RC4 не позволяет задать в параметрах подключения и создания БД относительные пути, в используемых скриптах и в модулях данных проекта используются абсолютные пути для подключения к БД
Сетевая среда	Интранет-среда, поддерживаемая FireBird	Любые сетевые соединения устанавливаются исключительно средствами клиента СУБД FireBird	
SCADA-система	Rockwell Software RSView 32 build 6.30.16	Для RSView 32 используется Add-on: Tools & Utilities 2.1 (Build 2.1.0)	
2. Среда разработки проекта			

Параметр	Значение	Примечание	Особенности работы, выявленные проблемы и пути их устранения
Среда разработки приложения и используемые языки программирования	Borland Delphi 7.0 build 4.453	Для увеличения номеров версий при каждой компиляции проекта используется компонент IDE эксперт Delphi собственной разработки IDE Compile Helper v 1.0	IDE Compile Helper v 1.0 не всегда заставляет Delphi перекомпилировать модули, выбранные для перекомпилирования, рекомендуется при перекомпиляции проекта осуществлять Build
Среда разработки БД	IBExpert v 2.5.0.55 (для скриптов – компонент IBExpert IBEscript v 1.65; для визуализации структуры БД используется дизайнер БД приложения IBExpert)	Вся структура БД обязательно представлена в соответствующих SQL-скриптах, позволяющих полностью собрать БД в актуальном состоянии. Скрипты позволяют также добавить в БД необходимые тестовые данные, позволяющие избежать дополнительной настройки и обработки данных при проведении тестирования определенных функций проекта. В скриптах использована особенность IBEscript, позволяющая загружать в BLOB-поля данные из файлов, для чего используется адрес :h00000000_00FFFFFF при вставке файла в BLOB-поле, позволяющий загрузить файл размером до 16 Мб.	IBEscript (как минимум, версии v 1.65 и ниже) не позволяет указывать максимально возможный адрес (:h00000000_FFFFFFFF) при импорте файла в BLOB-поле.
Компоненты и драйвера для доступа к БД	Доступ к FireBird – IBX (7.08)	Учитывая, что официально IBX не поддерживает работу с FireBird, в случае дальнейшего развития проекта планируется отказ от использования IBX	IBX версий более поздних, чем x.05 (в частности, 7.07 и 7.08) содержит ошибку при работе с NUMERIC-полями, которая приводит к переполнению используемого BCD-поля в IBX для хранения NUMERIC-поля БД. При использовании NUMERIC-полей в БД следует использовать IBX x.05, либо скорректировать модули IBX, заменив использование BCD на Float (как это было в более старых версиях)
	Доступ к .dbf – Microsoft VFP ODBC		
	Доступ к Watcom SQL-server – Watcom ODBC		
Компоненты для построения отчетов	FastReport 2.50		FastReport 2.50 некорректно выполняет функцию FieldIsNull при работе с IBX (результат функции – всегда true)

Параметр	Значение	Примечание	Особенности работы, выявленные проблемы и пути их устранения
Компонент для исполнения скриптов	Используется MS Script Control 2.6	Используются языки JScript и VBScript.	
Другие компоненты Delphi и нестандартные модули	Для криптографических целей используется TCryptLib 2.1	Используемая версия TCryptLib является opensource, более поздние являются закрытыми. Оригинальный TCryptLib 2.1 предназначен для Delphi 3 или 4. Для обеспечения совместимости с более поздними версиями Delphi используется самостоятельно модифицированная версия TCryptLib 2.1.	
	Отображение и печать DBGrid – используется EhLib 3.1		EhLib 3.1 содержит ошибки в модуле для сортировки данных в IBX. Для их устранения используется модуль собственной разработки, являющийся надстройкой над EhLib (EhLibIBXEx).
	Подсветка синтаксиса – SynEdit 1.0		
	Для общих целей используется RX 2.75	Используется port to Delphi 7, v1.0 (by Oleg Fyodorov)	
	Дополнительные модули собственной разработки	Используется Splashes.pas для создания Splash Forms. Используется uGLTools.pas – различные дополнительные процедуры и функции.	
3. Документация			
Создание документации	Для создания документации используется htm2chm v.1.6	Документация создается в chm-формате	
4. Создание дистрибутива			
Среда для создания дистрибутива	InstallShield Developer 8.0	Дистрибутив выглядит в виде единого исполняемого файла и включает все дополнительные компоненты, необходимые для	

Параметр	Значение	Примечание	Особенности работы, выявленные проблемы и пути их устранения
		работы приложения (в т.ч. FireBird, MS Script Control, Watcom SQL, компоненты доступа к данным и т.д.) При установке приложения возможно выбрать отдельные компоненты и варианты установки.	
5. Тестирование проекта			
Среда для тестирования проекта	OS: MS Windows 2000/XP	Для создания среды тестирования используется VMWare Workstation 4.0 build 4460. В качестве основной серверной платформы тестируется Windows 2000 Server SP 4. В качестве основной клиентской платформы тестируется Windows XP Professional SP 1	
	SCADA: Rockwell Software RSView 32 build 6.30.16	Для RSView 32 используется Add-on: Tools & Utilities 2.1 (Build 2.1.0)	
	Сетевая среда: Интранет-сеть MS Windows на протоколе TCP/IP	Тестируется работа как на локальном проекте, так и в сетевом окружении	

Додаток Б

Інформація про проекти АТ «Датекс Україна»

Таблиця Б.1

Окремі дані проектів, які використовували на практиці теоретичні положення даної дисертації

Показник	**Проект**				
	1	**2**	**3**	**4**	**5**
Розмір проекту, KSLOC	191,5	25,9	16,5	12,4	13,0
Розмір сторонніх компонентів, KSLOC	430,0	315,7	128,6	126,2	283,2
Коефіцієнт перспективності грошових надходжень ($К_{ПН}$)	0,73	0,56	0,44	0,74	н/д
Коефіцієнт успішності нової версії програмного продукту ($К_{УНВ}$)	0,95	0,84	0,87	0,86	н/д
Кількість компонентів, які формують середовище розробки	14	8	7	7	6
Кумулятивне відставання у часі компонент, які формують середовище розробки, років	12,5	12	7	5	5
Кількість співробітників, які приймають участь у проекті	11	4	3	1	2
Середня тривалість випуску однієї версії, років	0,3	0,3	0,5	1	0,5

Таблиця Б.2

Окремі дані проектів, які не використовували на практиці теоретичні положення даної дисертації

Показник	**Проект**					
	1	**2**	**3**	**4**	**5**	**6**
Розмір проекту, KSLOC	50,4	21,0	26,4	15,3	12,7	12,6
Розмір сторонніх компонентів, KSLOC	500,3	295,1	144,6	126,7	192,4	165,8
Коефіцієнт перспективності грошових надходжень ($К_{ПН}$)	0,65	0,30	0,25	0,52	н/д	н/д
Коефіцієнт успішності нової версії програмного продукту ($К_{УНВ}$)	0,82	0,74	0,76	0,63	н/д	н/д
Кількість компонентів, які формують середовище розробки	8	7	6	5	7	5
Кумулятивне відставання у часі компонент, які формують середовище розробки, років	8,5	9,2	14,5	7	9	8
Кількість співробітників, які приймають участь у проекті	9	7	5	4	2	2
Середня тривалість випуску однієї версії, років	0,5	0,7	0,7	1,2	1,5	1,7