

УДК 658:330.101.52

## АНАЛІЗ МОДЕЛЕЙ ЖИТТЄВОГО ЦИКЛУ ПРОЕКТІВ ГАЛУЗІ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

М. Дацко, Г. Семенів

*Львівський національний університет імені Івана Франка,  
економічний факультет,  
кафедра економічної кібернетики*

*Стаття присвячена проблемам управління проектами на підприємствах галузі інформаційних технологій (ІТ). Проаналізовано поняття життєвого циклу проекту, процесу управління проектом. Здійснено огляд існуючих моделей життєвого циклу ІТ проектів.*

*Ключові слова: управління проектами, моделі життєвого циклу, підприємства галузі інформаційних технологій.*

Успіх ІТ проекту залежить від ефективного управління. В межах теорії управління проектами базовим є поняття процесу. Процес управління – це сукупність взаємопов'язаних дій та операцій, що виконуються для отримання продукту, послуги або іншого визначеного результату проекту. Кожний з процесів отримує вхідні дані і генерує відповідні вихідні, які знову можуть бути вхідними даними для інших процесів.

Відповідно до стандартів американського Інституту управління проектами (Project Management Institute), викладених у PMBOK® Guide, виділяють 44 процеси управління проектом. Усі ці процеси об'єднані в п'ять груп, які називаються групами процесів управління проектом (або групами проектних процесів): ініціації, планування, виконання, моніторингу і контролю, а також закриття [1].

Життєвий цикл ІТ проекту визначається як термін часу, який починається від моменту прийняття рішення про створення програмного продукту й закінчується у момент його повного вилучення з експлуатації, скоріш за все, з метою встановлення та інсталяції нового продукту. Під моделлю життєвого циклу ІТ проекту сприймається структура, що визначає послідовність виконання і взаємозв'язок процесів, дій і задач упродовж життєвого циклу [2].

Головним нормативним документом, що регламентує склад процесів життєвого циклу, є міжнародний стандарт ISO/IEC 12207: 1995 «Information Technology – Software Life Cycle Processing» та відповідний йому Державний стандарт України ДСТУ 3918-99 «Інформаційні технології. Процеси життєвого циклу програмного забезпечення». Серед інших важливих стандартів, які визначають процеси управління ІТ проектами, можна виділити:

- IEEE 1540: Standard for Software Risk Management – управління ризиками програмного забезпечення;
- IEEE 1517: Standard for Software Reuse Processes – процеси повторного використання програмного забезпечення;
- ISO/IEC 15939: Standard for Software Measurement Process – процеси обчислення в області програмного забезпечення.

Програмне забезпечення в процесі своєї розробки й експлуатації проходить ряд певних етапів: виникнення та дослідження ідеї, аналіз вимог і проектування, безпосередньо кодування, тестування та налагодження, введення програми в дію, експлуатація та супровід, виведення з експлуатації [3]. Залежно від обраної моделі життєвого циклу ІТ проекту, ці фази можуть бути розбиті на декілька складових частин або навпаки об'єднані. Модель життєвого циклу ІТ проекту схематично пояснює, яким чином будуть виконуватися дії з

розробки програмного продукту за допомогою опису “послідовності” цих дій. Така послідовність може бути як лінійною, так і нелінійною, оскільки фази можуть слідувати одна за одною, повторюватися або виконуватися одночасно. Моделі життєвого циклу ІТ проектів визначають загальні стадії розробки програмного забезпечення. Вони концентруються на основних роботах і їх взаємовідношеннях. Моделі повинні враховувати специфіку галузі інформаційних технологій, яка характеризується високим рівнем невизначеності та надзвичайною динамічністю.

Проведений аналіз показав, що найбільш відомими та широко використовуваними моделями життєвого циклу проектів галузі інформаційних технологій є: каскадна модель (waterfall), інкрементна модель (incremental), рап модель (RUP – rational unified process model), спіральна модель (spiral). Окрему підгрупу утворюють еджайл моделі (agile).

Розглянемо детальніше класичні моделі життєвого циклу проекту в ІТ галузі.

Однією з класичних моделей є **каскадна модель (waterfall)**. Вона побудована логічно і послідовно: усі виробничі процеси у такій моделі ідуть один за одним. Виробничий процес проходить через фази аналізу вимог, дизайну, реалізації, тестування, інтеграції та підтримки.

В оригінальній версії каскадної моделі (розробником якої є Winston W. Royce) визначено наступні стадії життєвого циклу проекту:

- розробка документів з вимогами;
- дизайн продукту;
- розробка;
- інтеграція;
- тестування і виправлення поточних помилок;
- впровадження продукту;
- підтримка існуючого продукту.

Якщо слідувати цій моделі (див. Рис.1), то початок кожної наступної стадії відбувається лише після повного завершення попередньої навіть якщо для виконання даних стадій потрібні абсолютно різні ресурси. Фази даної моделі є дискретними і неможливо стрибнути назад чи перестрибнути на кілька стадій вперед.

Існують багато різновидів каскадної моделі, у яких є незначні видозміни базових процесів і фаз.

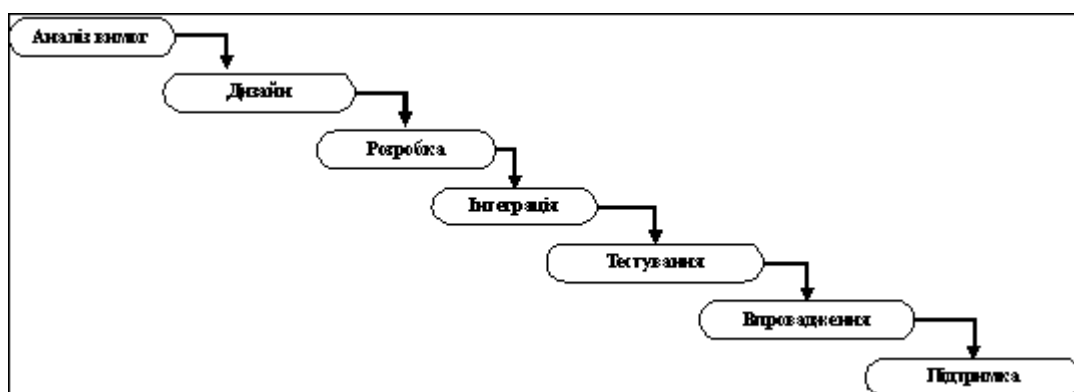
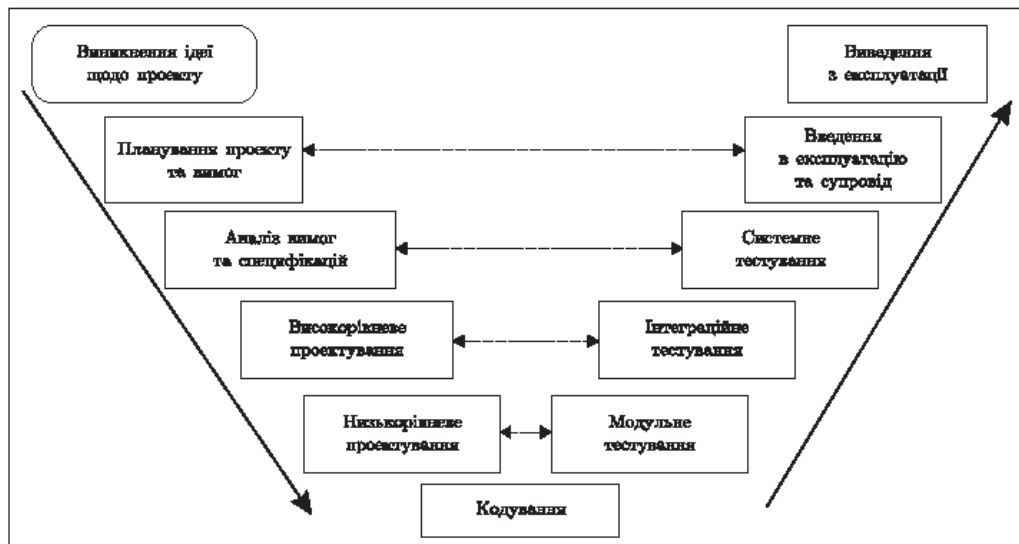


Рис.1 Життєвий цикл каскадної моделі

Перевагами розглянутої моделі є стабільний і закінчений набір проектної документації, який відповідає критеріям повноти та відповідності. В процесі використання даного підходу виявилось багато недоліків. До прикладу, під час використання моделі рідко вдається вкластись у тісні рамки схеми, тому постійно виникає необхідність повернення до попередніх етапів і уточнення чи перегляду раніше прийнятих рішень. Ще одним недоліком є суттєве запізнення кінцевого результату, оскільки замовник і кінцеві користувачі можуть внести свої нові пропозиції лише після завершення проекту.

На сьогодні велика увага приділяється тестуванню програмного забезпечення. Сучасні технології проектування вимагають, щоб процес тестування починався на ранніх фазах

життєвого циклу ПЗ. Найбільшу увагу процесам тестування приділено у **V-подібній моделі** життєвого циклу ІТ проекту. Розглянемо її докладніше. V-подібна модель була створена для допомоги працюючій над проектом “команді” в плануванні та забезпеченні подальшої можливості тестування системи. В цій моделі особливе значення надається діям, спрямованим на верифікацію й атестацію продукту. План приймання кінцевого програмного продукту замовником розробляється на етапі планування, а планування решти системи – на фазах аналізу, розробки проекту тощо. Процес розробки планів тестування позначений пунктирною лінією між прямокутниками V-подібної моделі (рис. 2).



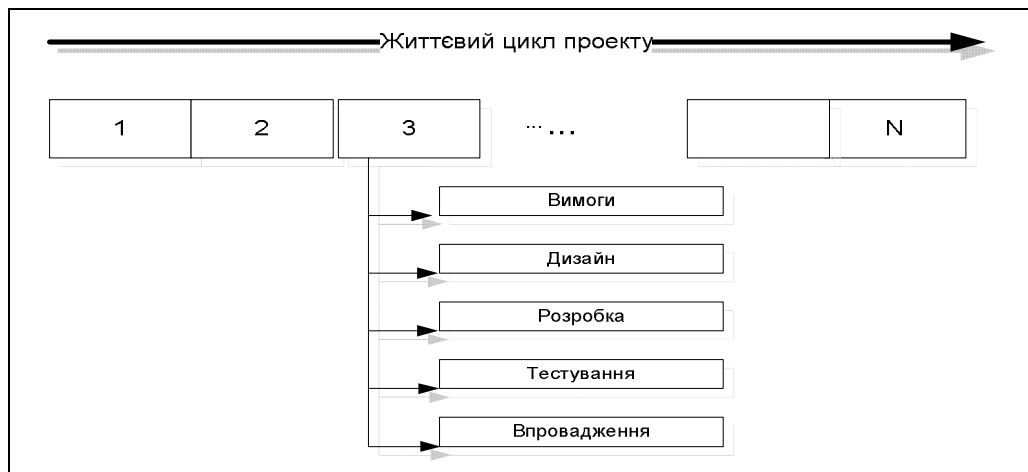
**Рис. 2 Життєвий цикл V-подібної моделі**

Проте, незважаючи на всі свої переваги та спрямованість на тестування, V-подібна модель має послідовну структуру: кожна наступна фаза починається лише по завершенню попередньої. Незважаючи на те, що передбачається план тестування розробляти на ранніх етапах життєвого циклу програмного забезпечення, саме тестування здійснюється вже після того, як продукт створено. Отже, помилки, виявлені на цьому етапі (пов'язані з неправильним проектуванням) спричинять повернення на попередні етапи життєвого циклу, а отже призведуть до значних фінансових втрат.

Модель життєвого циклу, де усі фази ІТ проекту виконуються поступово наз. **інкрементною моделлю (incremental)**. Модель відображає процеси двох видів ІТ проектів: проект розробки нового і проект підтримки існуючого програмного продукту. Проект вважається завершеним, коли усі початкові вимоги були виконані. Ця модель базується на каскадній моделі. Продукт ділиться на певну кількість версій, кожна з яких розробляється окремо. Кожна з версій продукту постачається клієнту поступово, коли вона вважається закінченою. Це забезпечує часткове реальне використання продукту і дає можливість уникнути довгого періоду розробки програмного забезпечення. Модель забезпечує досить швидку оплату проекту. Така модель також уникає негативного психологічного ефекту постачання клієнтам усього продукту відразу і значного терміну необхідного на розробку кінцевого продукту в цілому.

Перевагами використання моделі є стабільний графік виконання проекту та суттєві показники прогресу розробки проекту.

Одним із недоліків моделі є те, що кожна нова версія продукту при такій моделі має бути інтегрована з попередньою і будь-якими існуючими системами. Завдання декомпозиції продукту на версії теж не з простих, адже невелика кількість версій спричиняє велику відмінність кожної наступної версії від попередньої, у той же час завелика їх кількість призведе до втрати ефективності.



**Рис.3 Життєвий цикл інкрементної моделі**

Творці рап моделі (RUP – rational unified process model) сфокусувалися на діагностиці невдач ІТ проектів. Вони намагалися визначити і попередити причини невдач. У даній моделі життєвий цикл проекту ділиться на індивідуальні невеликі цикли – фази. Відповідно фази розбиваються на ітерації. Кожна ітерація має визначений термін завершення, а кожна фаза має визначені цілі.

Виділяють наступні фази:

- «Inception» - ініціація;

Початкова стадія. Період встановлення необхідного програмного середовища та аналізу усіх бізнес вимог. Після детального аналізу керівник проекту або інша відповідальна особа створює усю необхідну початкову документацію проекту (бачення проекту, визначення необхідних ресурсів і часу виконання робіт, початковий графік виконання проекту).

- «Elaboration» - фаза планування і детального аналізу;

Група програмістів детально аналізує вимоги до продукту і визначає технології, що будуть використані під час розробки нового продукту. Аналітики визначають план тестування. При цьому також створюється відповідна документація.

- «Construction» - розробка і моніторинг за виконанням;

Основна фаза виконання проекту, у якій здійснюється розробка і тестування нового продукту. Керівник проекту та інші відповідальні особи слідкують за виконанням проекту.

- «Transition» - закриття.

Завершальна стадія проекту. Фаза передачі продукту клієнтам. Створюється уся необхідна кінцева документація (наприклад інструкція по встановленню програмного продукту, інструкція по використанню, звіт про кількість знайдених дефектів програмного забезпечення під час виконання проекту тощо).

Перевагами рап моделі є зниження проектних ризиків та збільшення якості програмного продукту. Серед недоліків є те, що деколи проектна команда фокусується в основному на документації, забуваючи при цьому приділяти належну увагу основним роботам – кодуванню, тестуванню. У результаті затримуються терміни завершення проекту.

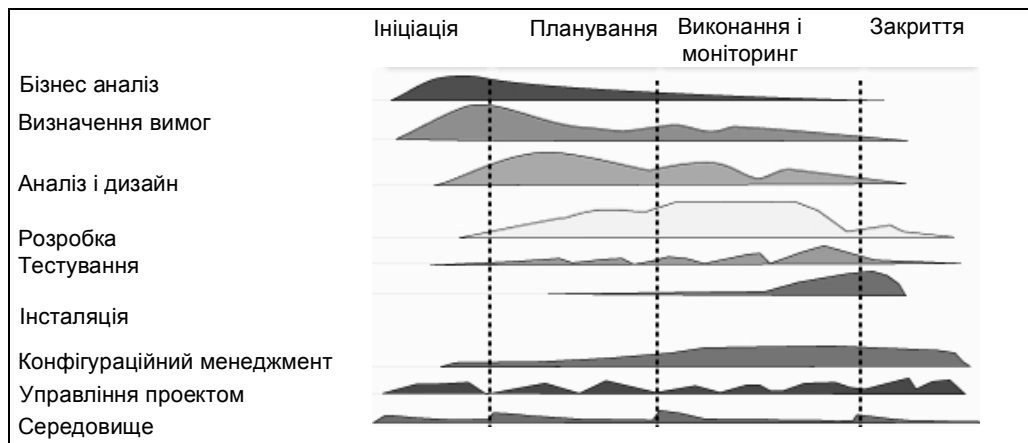


Рис.4 Життєвий цикл рап моделі

**Спіральна модель** життєвого циклу розробки програмної системи є подальшим розвитком водоспадної або каскадної моделі, в якій кожне коло спіралі відповідає одній з версій розробленої системи [2]. Тобто кожне коло спіралі відповідає одному з етапів життєвого циклу, під час якого розробляється версія програмної системи. Спіральна модель використовується для управління великими, дорогими і складними проектами. Ця модель позбавляє замовника і розробника від необхідності повного і точного формулювання вимог до системи на початковій стадії, оскільки вони уточнюються на кожному колі спіралі (ітерації або етапі розробки версії системи). Таким чином, постійно поглиблюються та конкретизуються деталі проекту і, в результаті, вибирається обґрунтований варіант, який доводиться до реалізації [3]. Спіральна модель (Рис.5), тобто кожен з її етапів розробки, складається з чотирьох стадій: аналізу вимог, проектування, реалізації та тестування. Під стадією розробки розуміють частину етапу процесу розробки програмної системи, що обмежена у часі і завершується створенням конкретного продукту, визначеного у вимогах. Проходження всіх стадій повторюється на кожному етапі розробки, кожному колі спіралі, окрім останнього, який завершується введенням системи в експлуатацію. Відмінною рисою такої моделі від попередньої (каскадної) є можливість багаторазового повернення до стадії формування вимог для розробки з будь-якої стадії робіт, якщо виявиться необхідність внесення змін. Це позитивна властивість моделі, бо на кожній стадії життєвого циклу може виникнути потреба змін, а внесення змін на будь-якій стадії обов'язково починається з внесення змін до попередньо зафіксованих вимог. Це може бути пов'язане з виявленням у процесі розробки недостатньої обґрунтованості сформульованих вимог або неузгодженості окремих позицій такого формулювання, відсутності потрібних інформаційних або технічних ресурсів, зниженні обсягів фінансового забезпечення тощо. Такі зміни носять локальний поточний характер і не приводять до принципних змін вимог, зафіксованих у документах. Але і в цьому випадку рішення про внесення змін повинно бути виваженим, ретельно обґрунтованим і не виходити за рамки розроблюваної версії, оскільки велика кількість змін і, таким чином, часте повернення до попередніх стадій можуть внести додаткові труднощі в організацію робіт, пов'язаних з розробкою. Більш суттєві пропозиції щодо змін, які пов'язані з розширенням функціональних можливостей програмної системи, накопичуються у замовника, узгоджуються з розробником і вносяться у список для подальшого включення у нові вимоги до розробки системи з метою створення її нової, більш досконалої, версії.



**Рис.5 Життєвий цикл спіральної моделі**

Деякі процеси життєвого циклу надають особливого значення швидкому введенню в експлуатацію програмних систем і глибокому залученню користувачів до процесу розробки. Такі процеси називаються **agile-методами**. Виділяють багато різновидів даного типу моделей. До них можна віднести: Scrum, Extreme Programming, Crystal Clear, Adaptive Software Development моделі. Перевагою даних методів є гнучкість у плануванні: план проекту активно корегується по мірі просування до мети проекту.

Більшість моделей даного типу намагаються мінімізувати ризики через розробку програмного продукту частинами (що називаються ітераціями). Переважно ітерація триває не більше 4-х тижнів. Кожна ітерація – це мініатюрний ІТ проект, який включає усі фази звичайного проекту, тобто планування, аналіз вимог, дизайн, кодування, тестування і документацію. Хоча інколи у кінці ітерації здача продукту не відбувається такий agile проект повинен бути в змозі здати новий продукт в кінці кожної ітерації.

Agile-методи фокусуються на комунікації. Більшість “команди” розміщені, працюють разом, а також включають усіх необхідних людей для задачі проекту. Як мінімум ця команда включає програмістів і їхніх замовників (тобто людей, які визначають і аналізують продукт, це можуть бути бізнес аналітики, проектні менеджери чи власне замовники). Команда також може включати тестерів, архітекторів, менеджерів тощо.

Працездатність нового програмного забезпечення визначається як основний показник прогресу проекту. У поєднанні з високою комунікацією на проекті такі методи продукують незначну кількість документації. Це дуже часто є причиною критики даного підходу. Проте сучасні agile методи виникли як реакція і протест проти методів виконання проекту, що обтяжені значною кількістю документації.

Виділяють наступні принципи Agile-методів:

- максимальне задоволення вимог клієнта (у разі необхідності - продовження терміну задачі проекту);
- продукт часто показується клієнту (до прикладу кожен тиждень, а не кожен місяць);
- працююче програмне забезпечення є визначником прогресу проекту.
- навіть пізні зміни у вимогах до продукту сприймаються позитивно;
- тісна щоденна кооперація клієнтів, програмістів, аналітиків проекту.
- постійна комунікація групи учасників проекту;
- проект будується навколо вмотивованих індивідів, до яких повинна бути довіра;
- постійна увага до технічної сторони і хорошого дизайну продукту;
- простота;
- само організованість;
- постійна адаптація до зміни середовища.

Адаптивні методи фокусуються на швидкій адаптації до змін середовища, до змін реальності. Хоча на практиці вони відрізняються від інших класичних моделей життєвого циклу проте деякі важливі складові залишаються такими ж. А саме ітеративне кодування і розробка (при цьому команда фокусується на комунікації і зменшенні артефактів). Agile-методи варто застосовувати до проектів, де вимоги до продукту постійно змінюються в часі. Вони не дуже підходять до проектів, де початкові вимоги точніше визначені і надійні. Хоча ще досі не має консенсусу стосовно цього.

Розробка моделей життєвого циклу є актуальним для кожної ІТ компанії, в тому числі і для вітчизняних компаній галузі інформаційних технологій. Величезна різноманітність і складність розглянутих методів потребує додаткового дослідження. Управління проектами на ІТ підприємствах необхідно корегувати відповідно до типу проекту і його особливостей. У статті визначено актуальність проблеми вибору моделі життєвого циклу проекту, розглянуто приклади класичних моделей та їх основні характеристики, визначено основні переваги та недоліки. Подано узагальнене графічне представлення моделей. Розглянута тема потребує додаткового дослідження. У подальших дослідженнях планується більш детально розглянути agile-моделі та здійснити розробку економіко-математичної моделі на прикладі одного з видів agile-моделей.

1. A Guide to the Project Management Body of Knowledge, Third Edition (PMBOK Guides) by Project Management Institute, March 2004.
2. Алексеев В. А., Терещенко В. С. Развитие спиральной модели жизненного цикла программных систем // Проблемы программирования, 2003, №4. – с. 34-41.
3. Дідковська М.В. Y-подібна модель життєвого циклу програмного забезпечення // Reports of the National Academy of Sciences of Ukraine, 2007, №6. – с.32-35.
4. Дослідження компанії Standish Group.  
Доступно з: <http://www.computerra.ru/offline/2003/503/2874>

## SOFTWARE LIFE CYCLE MODELS ANALYZING

**M. Datsko, H. Semeniv**

*This article is dedicated to project management problems of software companies. Ideas of project life cycle, processes of project management are considered. Existing life cycle models of software projects are analyzed.*

*Key words: project management, life cycle models, software companies.*