УДК 004.4

МЕТОДОЛОГІЧНІ СТРАТЕГІЇ МЕНЕДЖМЕНТУ ПРОЕКТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Р.І. Храбатин, М.М. Яцишин, О.Ф. Козак, О.В. Храбатин, Г.В. Григорчук

ІФНТУНГ, 76019, м. Івано-Франківськ, вул. Карпатська, 15, тел. (0342) 754252,
e-mail: romankhr@yahoo.com

Дане дослідження спрямоване на вирішення питань менеджменту проектів програмного забезпечення. Менеджмент проекту ґрунтується на використанні визначених знань, умінь, засобів і методів для досягнення поставленої мети. Досвід реалізації навіть відносно нескладних проектів свідчить, що труднощі, з якими стикаються розробники, обумовлені браком знань про сучасні методи та засоби ведення проекту, відсутності методології створення інформаційних систем, а також невмінням організувати ефективну проектну команду. Досліджено принципи теорії управління проектами з розробки програмного забезпечення, планування, контролю та оптимізації процесів розробки програмного забезпечення. Обґрунтовано основні поняття управління проектами, класифікація проектів та задач управління проектами, проаналізовано системні підходи та методи управління проектами.

Ключові слова: програмне забезпечення, інформаційні системи, інформаційно-обчислювальна система, менеджмент.

Данное исследование направлено на решение вопросов менеджмента проектов программного обеспечения. Менеджмент проекта основывается на использовании определенных знаний, умений, средств и методов для достижения поставленной цели. Опыт реализации даже относительно несложных проектов свидетельствует, что трудности, с которыми сталкиваются разработчики, обусловленные недостатком знаний о современных методах и средства ведения проекта, отсутствия методологии создания информационных систем, а также неумением организовать эффективную проектную команду. Исследованы принципы теории управления проектами по разработке программного обеспечения, планирования, контроля и оптимизации процессов разработки программного обеспечения. Обоснованы основные понятия управления проектами, классификация проектов и задач управления проектами, проанализированы системные подходы и методы управления проектами.

Ключевые слова: программное обеспечение, информационные системы, информационно-вычислительная система, менеджмент.

The research work is intended to solve the issues in management of software projects. The project management is based on application of the certain knowledge, skills, means and methods for the goal to be achieved. The implementation experience of even comparatively simple projects proves that the difficulties which are faced developers with are caused by lack of knowledge about modern methods and means of running the project, absence of methodology for creating information systems as well incapability to arrange an efficient project team. The theory principles of projects management in software development, planning checking and optimization of software development processes have been researched. The main notions for projects management, projects classification and tasks for projects management have been proved and the system concepts and methods of projects management have been investigated.

Key words: software, information systems, data processing system, management.

Можливі варіанти розвитку проекту розробки програмного забезпечення представляються як безліч операційних маршрутів, серед яких виділена область допустимих траєкторій. Управління розглядається як діяльність, що перешкоджає виходу траєкторії з області допустимості. У рамках цих угод описуються стратегії управління, прийняті в існуючих методологіях. З позицій стратегічних концепцій обговорюються жорсткі та гнучкі методології.

Актуальність теми. Якість планування розвитку проекту залежить, від точності встановлення для кожного етапу операційних маршрутів, які є оптимальні для досягнення цілей проекту (або виробничої функції). При такому розгляді розробки програмного забезпечення, *методика* (точніше, метод) - це здатність підтримати проходження операційного маршруту за допомогою певних приписів, угод, рекомендацій та регламентів. Не слід плутати цей термін з поняттям методики використання інструменту, яке пов'язується не з конкретною функцією, а з потенційною реалізацією різних функцій. *Методологія* - це, по-перше, стратегія, яка відображає напрями побудови та виконання системи діяльностей, тобто угоди про те, яким чином має розвиватися система; по-друге, набір методів, засобів та інструментів, які узгоджені зі стратегією.

Мета і завдання дослідження. Мета – це дослідження методологічних стратегій менеджменту проектів програмного забезпечення та ефективного управління проектами інформаційно-обчислювальної системи, що використовується в нафтогазовій галузі.

Проект загалом можна представити у вигляді конуса операційних маршрутів, центр якого відповідає задуму, а підстава - безлічі всіх варіантів завершення проекту. На підставі конуса виділяється цільова область - варіанти

завершення проекту, що відповідає меті розробки. Траєкторії, які ведуть в цільову область, є допустимими (рис. 1).

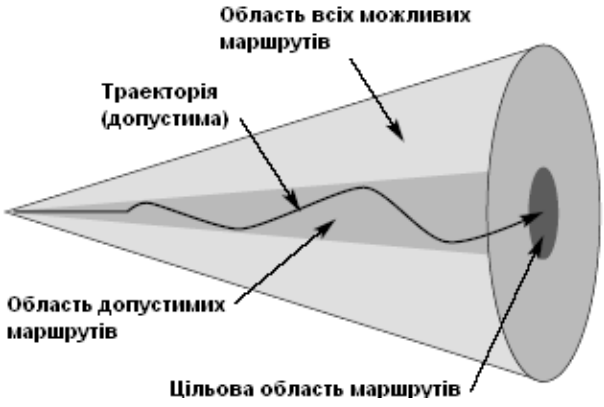

Рисунок 1 – Конус операційних маршрутів проекту

Якщо розробка проекту не керована менеджером, то ймовірність того, що його траєкторія виявиться допустимою, вельми невисока - занадто розрізняються цілі діяльностей, які спонтанно формуються у виконавців. З цієї причини впроваджується діяльність менеджера, одна з цілей якої - не допустити відхилення траєкторій діяльності виконавців від цільової області проекту. Для цього у керівника повинні бути засоби, що дають змогу виявляти відхилення, та інструменти впливу, призначені для коректування відхилень. Багато різних методик, що використовуються на практиці, можна враховувати до даних засобів і інструментів. Так, саме за принципом виявлення відхилень і швидкого коректування будується робота менеджера в рамках підходу екстремального програмування. Науковець Бек у викладі цього підходу наводить метафору водіння автомобіля, якому уподібнюється розвиток проекту. Водій просто коректує рух таким чином, щоб машина не відхилялася від полотна шосе. У вигляді схеми це може бути зображено так, як зображено на рис. 2.

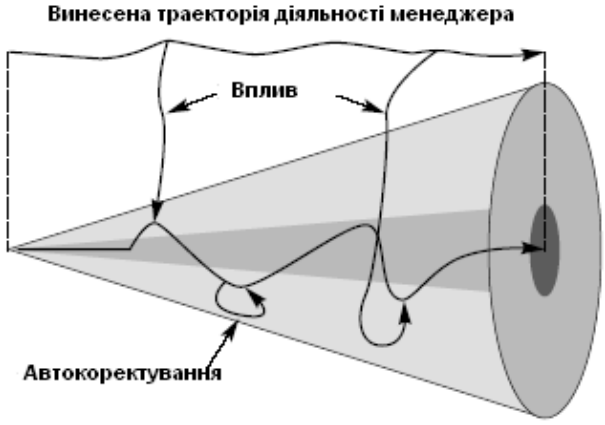

Рисунок 2 – З'ясування відхилень і коректування траєкторії

Водночас контроль діяльності проекту загалом є надскладним завданням, а тому потрібні як кошти (багато з методик екстремального програмування та інших методологій призначено саме для цього), так і спеціальні заходи. Останнє призводить до двох стратегій розвитку проекту, які розглядаються нижче.

Для реалізації загальної глобальної задачі проекту вона розбивається на під задачі, які вирішуються послідовно. Кожне рішення призводить до результатів, що використовуються наступним завданням. Зокрема, воно дає можливість уточнити постановку задачі і коло діяльностей, які доцільно активізувати. Таким чином, операційні маршрути проекту загалом розбиваються на послідовність етапів зі своїми локальними цілями.

Постановка завдання кожного етапу характеризується:

- суб'єктом-виконавцем;
- термінами вирішення завдання;
- виділеними ресурсами;
- засобами, інструментами та методами вирішення завдань;
- контрольними заходами, що дають можливість упевнитися, що завдання етапу вирішені.

До кінця етапу ставляться так звані контрольні точки, які призначені для того, щоб підвести локальні підсумки і перехідити до робіт наступного етапу. Інструментами діяльності в контрольних точках є контрольні заходи. Якщо з'ясовується, що завдання етапу може бути не виконано, тобто траєкторія, що реалізує діяльність етапу, вийшла з допустимої області, то виконуються дії з метою повернення траєкторії в цю область. Зокрема, використовуються схеми з'ясування відхилень і коригування траєкторії. Контрольна точка - це своєрідний бар'єр, який треба подолати, щоб перейти до робіт наступного етапу.

Продовження робіт після заходів в контрольних точках залежить від того, в якому стані опинився проект. Так визначається зміст робіт чергового етапу і поділ подальших робіт надалі. Так, якщо задача пройденого етапу включала з'ясування ринкової ситуації, то результатом її рішення може виявитися, що слід віддати перевагу придбанню продукту, а не його розробці. Як наслідок, подальша робота щодо досягнення кінцевих цілей проекту повинна змінити характер: місце діяльності з виготовлення повинно бути зайнято закупівлею та постачанням готового виробу. Інші варіанти результату вирішення цього завдання – форсування робіт, додаткові інвестиції (наприклад, в рекламу) і т.д.

Якщо поділ роботи на етапи залишається незадовільним, то він, у свою чергу, розбивається на локальні етапи, роботи або завдання, що допускають відносно автономне керування тим чи іншим способом, у тому числі шляхом з'ясування відхилень і коригування, а також можуть бути враховані додаткові контрольні точки. Якщо ресурсне забезпечення проекту, можливості виконавців і залежності між роботами за результатами допускають суміщення робіт, то вони можуть виконуватися паралельно, а для синхронізації встановлюються додат-

Рисунок 3 – Послідовний розвиток проекту

кові контрольні точки. Розпаралелювання може підтримуватися діяльністю менеджерів за напрямками.

Стратегія визначення етапів отримала назву послідовного розвитку проектів. Це історично перша стратегія, якої дотримувалися підчас розробки програмних проектів, а тому для неї існує багато методик та їх варіантів, але всі вони укладаються в схему, зображену на рис. 3.

Зі схеми видно, що послідовний розвиток скорочення обсягу конуса операційних маршрутів досягається за рахунок розбиття його на ряд більш дрібних конусів.

Скорочення обсягу конуса операційних маршрутів можна добитися шляхом виокремлення в загальному завданні проекту таких її частин, які стають прийнятними (доступними) для управління. На відміну від попередньої стратегії, тут як перша, так і наступні виділяють завдання, які частково вирішують проблеми користувача. Кожного разу будується програмний виріб, який в застосовуватися на практиці, навіть, якщо неповністю задовольняють потреби користувача. Таким чином, в даній стратегії замість підзадач виділяються вимоги, які планується реалізовувати. У сукупності вимог виділяється перша порція, потім наступна і т.д., які послідовно, від релізу (випуск програмного забезпечення) до релізу наближають програмний виріб до стану, що задовольняє проект, і потреби користувачів у вирішенні їхніх завдань.

Кожне з часткових рішень називається ітерацією проекту, а стратегія такої розробки - ітеративним розвитком проекту або ітеративним нарощуванням можливостей системи. В результаті виконання однієї ітерації будується самостійний програмний виріб, залежний від результатів попередніх ітерацій. Принципово: реліз ітерації ніколи не скасовує користувальницьких можливостей, досягнутих на попередніх ітераціях.

Принципи та методи вибору вимог для реалізації на поточній ітерації в різних методологіях можуть розрізнятися істотно, але всі сходяться в одному: завдання ітерації повинні бути досяжні для менеджменту. В іншому, спектр підходів простягається від обліку актуальності для користувача і до глибокого аналізу всіх вимог, відомих і передбачуваних на даний момент. Це проводиться з метою визначення такої архітектури системи, яка допускає послідовне ітеративне нарощування можливостей без переробки результату попередніх ітерацій. Взагалі, стосовно архітектури можна класифікувати всі ітеративні методології, що проявляються в тому, яке місце в проекті займають підготовчі роботи до ітерації.

На рисунку 4 у вигляді схеми зображено стратегію ітеративного нарощування можливостей системи.

Прямокутники, зображені поблизу основ конусів операційних маршрутів ітерацій, відповідають вимогам: виділено блок вимог, призначений для реалізації; пунктиром - відпрацьовані вимоги. Кількість прямокутників при переході від ітерації до ітерації зростає, що відображає надходження нових вимог в ході розвитку проекту.

При розгляді схеми ітеративного нарощування можливостей системи легко зауважити, що конуси ітерацій можна, принаймні візуально, представляти як конуси етапів. А чи не можна у зв'язку з цим вважати ітеративну стратегію окремим випадком послідовної схеми визначення етапів – в якому кожен етап є ітерація? Відповідь на це питання найчастіше негативна. Справа в тому, що поділ діяльності, яке ми назвали стратегією визначення етапів, припускає мету виконання, пов'язану з використанням отриманих результатів на наступному етапі. А ця умова для ітеративного нарощування не завжди виконується. Але, якщо крім того, що в ході ітерації готується реліз системи, вона передбачає побудову бази для реалізації наступної ітерації, то розгляд ітерацій як етапів проекту досить природній. Однак, залишається ще одна відмінність: при поетапному розподілі проекту зазвичай ціль і зміст усіх етапів визначаються на початкових стадіях, тоді як ітеративне нарощування не пов'язане з цим припущенням і, більш того, навіть кількість ітерацій, не кажучи вже про їх зміст, не фіксується заздалегідь.

Важливо зазначити, що стратегія ітеративного нарощування не фіксує методику роботи в рамках ітерації. Виконання вимоги видимості завдань ітерації дає підставу для того, щоб розробку ітерації підпорядковувати будь-якій стратегії. За рахунок звуження завдання про-

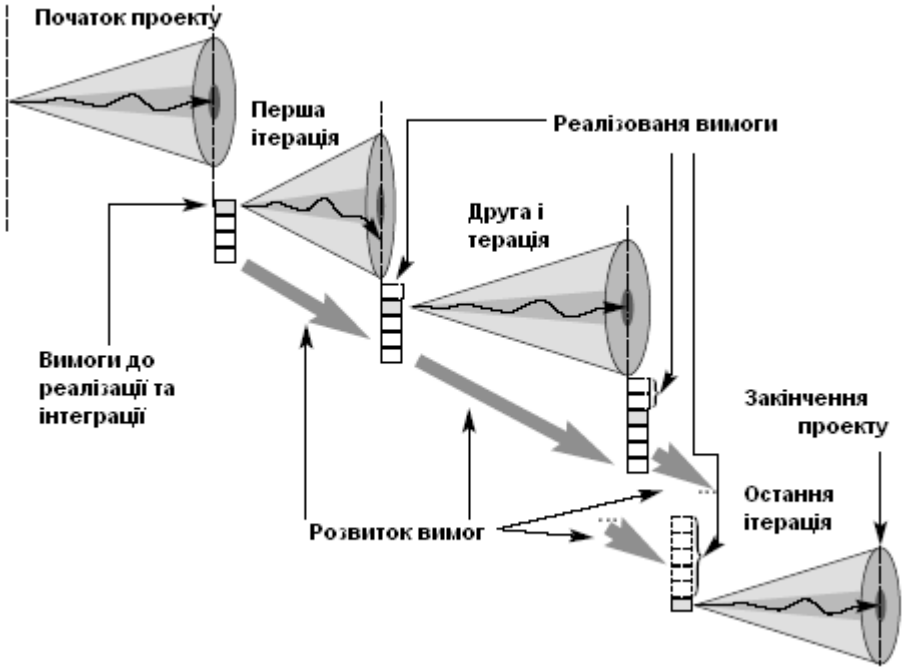

Рисунок 4 – Ітеративне нарощування можливостей системи.

екту розробники можуть послабити вимоги видимості і вирішити задачу ітерації як розробку самостійного проекту, наприклад підпорядкувати його послідовній стратегії, і це звичайний шлях реалізації. Найчастіше немає необхідності розглядати розробку ітерації проекту як вкладений ітеративний проект, так як у користувачів не виникає потреби в переході до дрібного релізу, "вкладеному" у більш загальний реліз. Проте іноді готують так звані “білди системи” - її варіанти, які поставляються в ході експлуатації релізу, наприклад, для швидкого виправлення помилок.

Визначаючи стратегії послідовного та ітеративного розвитку проекту, ми виходили з того, що контроль діяльності проекту - завдання, що вимагає спеціальних організаційних підходів. Загальною метою є створення програмного продукту в упорядкований процес, в рамках якого розвиток проекту можна зробити більш прогнозованим і ефективним. Для цього, зазвичай, створюється детальний опис процесу розробки системи, особливе місце в якому займає планування. Інакше кажучи, створюється метод, за допомогою якого передбачається побудова даної системи. В подальшому методи керування проектом узагальнюються, і в результаті досвід їх застосування перетворюється в методику, або, як зараз прийнято говорити, методологію. Нерідко при такому формуванні методики - методології фіксуються проміжні і випадкові рішення, що опинилися корисними через специфіку вдалих проектів. Поряд з усім корисним, ці рішення оголошуються обов'язковими, стандартними, їх потрібно застосовувати і тоді, коли вихідні передумови втрачені. Щоб слідувати таким методологіями, доводиться виконувати безліч різних приписів, що уповільнює темпи основних програмних робіт. Тому їх називають великоваговими або жорсткими.

Жорсткі методології привабливі для замовників програмних проектів, які завжди можуть перевірити, чи дійсно процес розробки впорядкований і результати відповідають планам. Природно пов'язувати цю можливість з організацією, яка отримує замовлення і забезпечує його менеджмент, оскільки крім побажань організацій – можливий адміністративний контроль.

В даний час створено стандарти зрілості процесів розробки програмного забезпечення. Серед них найбільш поширеною є пропозиція Інституту програмної інженерії при Університеті Карнегі-Меллона - так звана модель SW-CMM (Capability Maturity Model for Software). Цю модель можна вважати загальноприйнятою, оскільки на неї найчастіше орієнтуються замовники, зокрема з Міністерства оборони США, пропонуючи проекти тільки тим організаціям, які сертифіковані на рівнях зрілості “чотири” і “п’ять”. Зазначено, що модель CMM було сформовано при вагомому впливі практики військових замовлень з їх жорсткою процедурою контролю і звітності. Пропозиції CMM для визначення зрілості організації спираються на те, які процедури управління програмними проектами, відстеження їх розвитку та інші менеджерські методи прийняті в якості фірмового стандарту. Методології програмування, побудовані на базі цих пропозицій, часто розглядають як еталон жорсткості.

На противагу жорстким методологіям, останнім часом сформувався компромісний підхід, який об'єднано загальним терміном «agile development». На українську мову його переводять як швидкий розвиток, гнучкі методології (див. переклад статті М. Фаулера) і навіть "спритні технології"). У новому підході намагаються надати можливість полегшеної організованої роботи програмним колективам,

коли перевантаженість стандартизованого процесу перешкоджає ефективності. Вони претендують на те, що їх застосування можливо навіть тоді, коли не вдається досить точно представити проект на початку його створення. У цих випадках пропонується залучення замовника до формування завдань та коригування припущень протягом усього розвитку проекту. За його допомогою намагаються виявляти, найбільш точно і без зайвих бюрократичних процедур, актуальні потреби користувачів, що склалися в даний момент. В результаті з'являється можливість вказати саме ті вимоги, реалізація яких є необхідною і допускає максимально короткі терміни випуску релізу.

Всі методології швидкого розвитку орієнтуються на стратегію ітеративного нарощування можливостей системи, але з частковою відмовою від постулату незмінності апріорної архітектури. При цьому не тільки допускається, але навіть передбачається, що архітектура системи, а значить і програмний код, будуть змінюватися при переході від релізу до релізу. Всі вони надають розробникам значно більшу свободу, ніж, приміром, вимоги стандартів CMM. Але не слід думати, що цей підхід повністю скасовує жорсткі методології і, як визначають Боем і Тюрнер, необхідний баланс між свободою швидких методологій і дисципліною.

Більш детально охарактеризувати всі методології швидкого розвитку спільно є неможливим - занадто різні їх вихідні принципи, занадто залежні вони від специфіки колективів, які займаються розробками програмного забезпечення. На стратегічному рівні їх дійсно об'єднує так званий «Agile Manifesto», прийнятий ентузіастами в лютому 2001 року в містечку Сноуберд (США), який зводиться до чотирьох постулатів:

• індивідууми та їх взаємодія є важливішим від процесів та інструментів;

• працездатне програмне забезпечення є важливішим від ведення обширної документації;

• співпраця із замовником є важливішою від самого процесу укладення контракту;

• готовність до змін є важливішою за дотримання пунктів плану від слідування планом.

Мабуть, ніхто не стане заперечувати ці положення. Але чи можна всерйоз говорити про відмову від того, що зазначено в правій частині кожного з тез маніфесту, якщо прагнути до виготовлення надійних програмних продуктів? Адже і автори так званих монументальних методологій в свій час теж прагнули поліпшити процес розробки програмного забезпечення! І чи не вийде так, що, прагнучи до полегшення процесу, доведеться вводити надмірно жорстку дисципліну, при якій тільки й можна буде відповідати маніфесту. Зрозуміло, у кожного підходу є межі застосування, і якщо не виходити за їх межі, то можна гарантувати відповідну якість. Успішність багатьох гнучких розробок вказує на те, що цей підхід досить змістовний.

З точки зору теорії діяльності, стратегічне розмежування між жорсткими та швидкими методологіями пов'язано з характером видів діяльності, на підтримку яких націлений кожен з підходів. Для жорстких методологій характерне прагнення забезпечити розробників рецептами, що не вимагають обговорень, тобто потрібно виконувати відомі приписи, дотримуватися регламентів, щоб відповідні операційні маршрути наближали до допустимих траєкторій. Іншими словами, жорсткі методології підтримують переважно такі діяльності, які можна назвати імперативними. У цій підтримці велику питому вагу має орієнтація на методи-припису, яким підпорядковуються засоби та інструменти як елементи діяльності. Тому тут значне місце приділяється плануванню, вимірам процесу з метою його відстеження, менеджменту як діяльності, що забезпечує не тільки корекцію неприпустимих траєкторій, але і прийняття рішень в не детермінованих випадках, тобто тоді, коли утруднений підбір відповідного рецепту для розробників.

На відміну від традиційних підходів швидкі методології орієнтуються на те, що діяльність з виробництва програмного забезпечення по своїй суті є переважно креативними, тобто такими, в яких від розробників вимагається не тільки розпізнавання ситуацій та застосування в них відомих методів, але і конструювання нових методів дій. А це інший, більш високий рівень знань і умінь. Як зазначає Фаулер, процес розробки програмного забезпечення постійно адаптується до мінливих вимог користувачів і є принципово непередбачуваний. Ця якість обумовлює креативність діяльності не тільки на початку проекту, але й протягом усього його розвитку. Непередбачуваність і креативність розробки програмного забезпечення вказує на те, що готових рецептів на всі випадки життя просто не вистачить, а тому серед елементів діяльності засоби та інструменти повинні мати більшу питому вагу, ніж методи.

У таблиці 1 подано порівняння жорстких і швидких стратегій в методологіях програмування.

З цього порівняння можна зробити висновок, що швидкий процес більше відповідає проектам, в яких вимагається до повної міри використовувати творчий потенціал співробітників. При жорстких методологіях серед інших показників цінності співробітника істотне місце займає здатність виконувати приписи, старанність, тоді як при швидких підходах - ініціативність, прагнення до взаємодопомоги. У жорсткому проекті замовник протиставлений виконавцям (одна з функцій менеджера безпосередньо пов'язана із забезпеченням взаємодії із замовником), а необхідною умовою швидкого проекту є тісна співпраця з замовником, як з членом команди виконавців.

Таким чином, вже на стратегічному рівні можна розмежувати сфери адекватного використання двох підходів. Це розмежування безпосередньо випливає з їхнього зіставлення. Зрозуміло, що у багатьох випадках можливі обидва варіанти організації проектної діяльності. Однак, пристосованість жорстких методологій до

Таблиця 1 – Порівняння жорстких і швидких стратегій в методологіях програмування

Жорсткі методології	Швидкі методології
Орієнтація на передбачувані процеси розробки програмного забезпечення з чітко визначеними цілями.	Усвідомлення того, що процеси розробки програмного забезпечення в принципі непередбачуваними.
Розпізнавання ситуацій і застосування готових методів.	Розпізнавання ситуацій та конструювання методів для роботи в них.
Планування з визначенням етапів: з обсягом робіт, з ресурсами, з термінами та рівнем якості робіт.	Дотримання балансу між параметрами проекту: обсягами робіт, ресурсами, термінами і рівнем якості робіт.
Замовник - суб'єкт, зовнішній по відношенню до проекту, що впливає на розробку тільки через надання ресурсів і контроль результатів, у тому числі, з поетапними термінами виконання проекту.	Замовник (його представник) - член команди розробників, наділений правом впливати на розробку; його головною метою є відстеження актуальності завдань, що вирішуються.
Рольовий розподіл праці працівників проекту.	Спільна діяльність співробітників і колективна відповідальність за результати.
Дисципліна і підпорядкування.	Самодисципліна і співробітництво.
Знеособлений процес, виконавці якого визначаються тільки за кваліфікаційними вимогами.	Процес, максимально враховує особисті якості виконавців.

імперативності вказує на можливість і виправданість впровадження технологій, тоді як творчого процесу швидкого підходу технології можуть сприяти лише як набір інструментів, причому далеко не завжди достатньо повний. Тут варто пояснити, що ми розуміємо під технологіями. Це розуміння відображає наступне визначення. Технологія як діяльність - це середовище підтримки виконання діяльності, що володіє засобами та інструментами, а також методами їх застосування. Неухильне дотримання цих методів гарантовано забезпечить виробництво програмних продуктів, тобто отримання з наданих ресурсів і матеріалів, продукту-результату, відповідного цілям, в необхідному обсязі, за певний час і з прийнятним рівнем якості.

У визначенні явно проглядається використання понять трикутника менеджменту проектів та елементів діяльності. Додатковим в ньому те, що воно фіксує детермінований процес виробництва, в якому немає місця творчості. Саме творчість перешкоджає гарантіям отримання очікуваного результату в задані терміни.

Коли говорять про детерміновану діяльність, зазвичай під цим розуміють, що відсутні непередбачені, а значить, і не забезпечені методами дій ситуації. У контексті поняття технології, крім того, слід виділити й інший аспект детермінізму – в'язку методів і цілей, якій підпорядковані задані засоби та інструменти. Коли детермінізм порушується, тобто доводиться вирішувати проблеми вибору або стикатися з непередбаченими ситуаціями, настає кінець технології, і розробка вступає в область ремісничого виробництва, яке, можливо, є більш привабливою (за рахунок творчої складової), і цілком може призводити до підвищення якості, але не гарантує а ні результату, а ні дотримання термінів.

Розробники жорстких методологій старанно і послідовно витісняли “недетермінізм” з виробництва програмного забезпечення. Там, де це не вдавалося, вони виставляли своєрідні огорожі: з одного боку, це типові прийоми, вказівки та рекомендації, а з іншого - перевірка результатів, контрольні заходи і багато іншого. Але, незважаючи на появу різноманітних методик, засобів та інструментів підтримки управління, які змушені застосовувати менеджери, особливого успіху досягти їм не вдалося. Причина - вже не раз згадувана мінливість вимог, і властива природі людини схильність допускати помилки на всіх рівнях роботи. Відділяти ситуації від можливих помилок, допомагати їх пошуку, використовуючи засоби автоматизації, здатні ліквідувати багато рутинних операцій. Тому можна говорити про деяку технологізацію. Але від цього креативний процес програмування не став подібний до того, що демонструє матеріальне виробництво або, наприклад, бухгалтерський облік.

Представлене визначення технології досить конструктивне в будь-якій ситуації. Якщо елементи діяльності визначено, то можна задавати цілком конкретні питання, що дають змогу з'ясувати технологічність. Але, на жаль, перевірка галузі розробки програмного забезпечення на практиці свідчать про те, що технологічність її діяльностей характеризується фрагментарністю: окремі види робіт, доведені навіть до рівня автоматизації комплексної технології, що охоплює весь процес, ніхто не пропонує. Швидше за все, складність виробництва програмного забезпечення, а також об'єктивна непередбаченість його, не дає можливості говорити про універсальність технологічного процесу на всі випадки життя.

Література

1 Храбатин Р.І. Систематизування математичних моделей систем управління у вигляді передавальних функцій / Р.І. Храбатин, Л.В. Саманів, М.В. Крихівський // Нафтогазова енергетика – 2011. – № 1(14). – С. 99-101.

2 Храбатин Р.І. Математичне моделювання регулювання тиску газу на компресорних станціях магістральних газопроводів / Р.І. Храбатин, Д.Ф. Тимків, М.В. Крихівський, Д.Д Матієшин // Шоста науково-практична конференція з міжнародною участю «математичне та імітаційне моделювання систем МОДС 2011. – Чернігів, 2011. – С. 170-171.

3 Скопин И.Н. Основы менеджмента программных проектов / И.Н. Скопин. – М. Открытые системы, 2007. – 157 с/

4 Перевозчикова О.Л. Сучасні інформаційні технології / О.Л. Перевозчикова. – К.: Інститут економіки та права "Крок", 2002. – 121 с.

5 Уокер Ройс. Управление проектами по созданию программного обеспечения / Уокер Ройс. – М.: Лори, 2002 – 424 с.

6 Боэм Б.У. Инженерное проектирование программного обеспечения / Б.У. Боэм. – М: Радио и связь, 1995. – 511 с.

7 Бабенко Л.П. Основи програмної інженерії / Л.П. Бабенко, К.М Лавріщева.. – К.: Знання, 2001. – 269 с.

8 Лешек А. Мацяшек. Анализ требований и проектирование систем. Разработка информационных систем с использованием UML / Лешек А. Мацяшек: пер. с англ. – М.: Вильямс, 2002. – 428 с.

9 Дин Леффингуэлл. Принципы работы с требованиями к программному обеспечению. Унифицированный поход / Дин Леффингуэлл, Дон Уидриг: пер. с англ. – М.: Вильямс, 2002. – 446 с.

Стаття надійшла до редакційної колегії 28.03.13
Рекомендована до друку професором ***Юрчишиним В.М.*** *(ІФНТУНГ, м. Івано-Франківськ) д-ром техн. наук* ***Лютаком І.З.*** *(Прикарпатський національний університет ім. В. Стефаника, м. Івано-Франківськ)*