РАДІОТЕХНІКА, ЕЛЕКТРОНІКА ТА ТЕЛЕКОМУНІКАЦІЇ

УДК 004.738.5

С.М. БУРБЕЛО, О.С. СТАРОДУБ, М.С. БОГДАНОВА
Вінницький національний технічний університет

ВИБІР ГНУЧКИХ МЕТОДІВ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розглянуто основні ідеї та принципи реалізації гнучких методів розробки програмного забезпечення, проведено їх варіантний аналіз, визначено перспективи використання методів.

Ключові слова: гнучкі методи розробки, методологія розробки, екстремальне програмування, функціонально-орієнтована розробка, скрам-метод.

S.M. BURBELO, O.S. STARODUB, I.V.KRUCHOK
Vinnytsya national technical university

TOWARDS FLEXIBLE CHOICE SOFTWARE DEVELOPMENT METHODS

Investigated basic ideas and principles of realization of flexible software development methods, main advantages, analyzed and shown perspectives of methods. Flexibility is important because the development of a new product naturally involves change from what came before it. Change can be expected in what the customer wants and how the customer might use the product, in how competitors might respond, and in the new technologies being applied in the product or in its manufacturing process. The more innovative a new product is, the more likely it is that the development team will have to make changes during development.

Flexible development counteracts the tendencies of many contemporary management approaches to plan a project completely at its outset and discourage change thereafter. These include Six Sigma, which aims to drive variation out of a process; lean, which acts to drive out waste; and traditional project management and phased development systems (including the popular Phase–gate model), which encourage upfront planning and following the plan. Although these methodologies have strengths, their side effect is encouraging rigidity in a process that needs flexibility to be effective, especially for truly innovative products.

Key words: flexible development methods, methodology development, extreme programming, feature-oriented development, scrum method.

Вступ

Розробка програмного забезпечення (ПЗ) є відносно молодою галуззю інженерних робіт, яка почала активно розвиватися у кінці минулого століття [1]. Тож природно, що ранні підходи до розробки програмних продуктів були мало формалізованими і базувалися на принципі Code-and-Fix (кодування і виправлення) [2]. Подальший розвиток методів та засобів програмування обумовив формалізацію підходів до створення програмних ресурсів з метою збільшення ефективності процесів програмування.

Впровадження гнучкої методології розробки ПЗ передбачало мінімізацію ризиків шляхом зведення процесу розробки програмного продукту до серії дискретних коротких циклів, поєднаних ітераційною архітектурою [2]. Такий підхід дозволив підвищити ефективність роботи програмістів та збільшити надійність розроблених програм [3].

У зв'язку з цим актуальною є задача аналізу гнучких методів розробки програмного забезпечення, визначення їх особливостей, переваг та перспектив використання.

Постановка завдання

Метою роботи є збільшення ефективності процесів програмування шляхом обгрунтованого вибору та впровадження гнучких методів розробки програмних продуктів.

Об'єктом дослідження постають процеси створення програмного забезпечення.

Під предметом дослідження розуміємо agile-методи як методи гнучкої розробки програм.

Головними задачами роботи вбачаємо визначення системи критеріїв оцінювання та проведення варіантного аналізу гнучких методів розробки програм, визначення перспектив використання розглянутих методів у процесі програмування.

Аналіз принципів гнучких методів розробки програм

Гнучка методологія розробки ПЗ (Agile software development) є досить молодою. Маніфест Agile був прийнятий у лютому 2001 року і підписаний представниками Extreme programming, Scrum, DSDM, Adaptive software development, Crystal Clear, Feature-driven development, Pragmatic programming [4].

Agile – це сімейство процесів розробки програмних продуктів, орієнтованих на використання ітеративної технології з динамічним формуванням і уточненням системи вимог на кожному витку ітерації [5]. Дискретизація локальних процесів передбачає організацію їх взаємодії в середині робочих груп, створених з кваліфікованих фахівців різного профілю.

Agile Manifest формулює чотири основні ідеї методології [6]:

- виділення особистісного підходу (особистості та їх взаємодії є важливішими, ніж процеси та інструменти);
- забезпечення працездатності розробленого продукту (працююче програмне забезпечення є

важливішим, ніж забезпечення наявності повної документації);

- забезпечення конструктивного діалогу між розробником та замовником (співпраця із замовником є важливішою, ніж контрактні зобов'язання);
- гнучкість методу розробки з урахуванням динамічно оновлюваних вимог замовника (реакція на зміни є важливішою, ніж чітке дотримання початкового плану).

За маніфестом Agile можна виділити 12 принципів реалізації гнучких методів розробки ПЗ [4]:

- задоволення клієнта за рахунок ранньої та безперебійної поставки програмного забезпечення;
- вітання внесення змін до вимог навіть у кінці процесу виконання розробки з метою підвищення конкурентоспроможності кінцевого продукту;
- періодична поставка робочого програмного забезпечення за визначеним планом (щомісяця, щотижня чи частіше);
- забезпечення тісного (щоденного) конструктивного діалогу між замовником і розробником протягом всього терміну виконання проекту;
- мотивація виконавців проекту, забезпечення їх гідними умовами праці, підтримкою та довірою;
- рекомендований метод передачі інформації – особиста розмова ("обличчям до обличчя");
- орієнтація на забезпечення працездатності розроблюваного ПЗ, що вважається головним завданням розробки;
- спонсори, розробники та користувачі повинні мати можливість підтримувати постійний темп на невизначений термін;
- забезпечення виконання постійної вимоги щодо поліпшення технічної майстерності і реалізації зручного дизайну;
- забезпечення вимоги щодо простоти реалізації (заборона виконання зайвої роботи);
- орієнтація на реалізацію принципів самоорганізації в команді;
- постійна адаптація до зміни обставин і вимог до реалізації ПЗ.

Гнучкі методи розробки ПЗ відносяться до еволюційної стратегії. Процес створення програмних продуктів засобами Agile методології можна подати у вигляді спіральної моделі [3], яка презентує ітеративний підхід у процесі розробки ПЗ (рис.1).

Рис. 1. Модель розробки ПЗ гнучкими методами, де
1 – початковий збір вимог і планування проекту;
2 – виконання тієї ж роботи з урахуванням оновлених вимог розробника і замовника ПЗ на кожному витку ітерації;
3 – аналіз ризику на основі початкових умов;
4 – аналіз ризику з урахуванням реакції замовника на кожному витку ітерації, оптимізація процесу програмування шляхом скорочення надлишкових операцій;
5 – перехід до комплексної системи;
6 – проектування початкової моделі системи, розробка початкової версії ПЗ;
7 – конструювання ітераційних версій ПЗ вищого рівня;
8 – розробка кінцевого програмного продукту;
9 – ітераційне оцінювання проекту замовником та тестувальником, діалог між замовником та розробником, розповсюдження проміжних і кінцевої версії ПЗ.

Використання на етапі конструювання ПЗ засобів DSDM, що базуються на впровадження ітераційного та інкрементного підходу до створення моделі процесу програмування дозволяє деталізувати III квадрант моделі (рис. 1) за рахунок використання технології швидкої розробки програмних додатків – Rapid Application Development (RAD-технології) з активним залучанням замовника (користувача) на етапах планування, аналізу ризику та оцінювання проміжних і кінцевої версії ПЗ.

Використання методу Getting Real у межах гнучкої методології розробки ПЗ реалізує ітераційний підхід, коли на початковому етапі проектування створюється інтерфейс програмного продукту, а реалізація його програмних модулів відбувається на подальших витках ітерації у процесі створення проміжного і кінцевого ПЗ.

Таким чином, гнучкі методи розробки ПЗ дозволяють повне забезпечення виконання всіх вимог замовника. Орієнтація на забезпечення працездатності розроблюваного ПЗ підвищує надійність створених програмних продуктів.

Проте, гнучкі методи не надають належного значення розробці супровідної документації, що обмежує експлуатаційний вік і універсальні характеристики подальшого багаторазового використання розроблених блоків ПЗ.

Варіантний аналіз Agile-методів розробки ПЗ

Головними гнучкими методами методології Agile є екстремальне програмування (XP), функціонально-орієнтована розробка (FDD) та Scrum-метод.

Екстремальне програмування (Extreme Programming) було запропоноване Кентом Беком у кінці 90-х років минулого століття [6]. Модель процесу програмування акумулює випуск дискретних версій ПЗ, які виходять так часто, наскільки це можливо. Кожна версія обов'язково включає нову функціональність алгоритму розв'язку задачі. На етапі генерації коду використовують наявні стандарти кодування (Coding

Standards), що дозволяє уніфікувати робочі функції та забезпечити подальше багаторазове використання розроблених і стандартних програмних компонент у процесі створення ПЗ за ітераційною RAD-моделлю роботи ПЗ (рис. 2).

Однак, у повному обсязі XP не була використана навіть її авторами [4]. Вона стала базовою філософією розробки ПЗ. XP широко впроваджувала нові принципи парного програмування, колективного володіння кодом та рефакторингу коду. Крім того, XP ініціалізувала ідею простого, не надлишкового дизайну проекту.

Метод функціонально-орієнтованої розробки – Feature Driven Development (FDD) є ітераційним гнучким методом методології Agile. FDD орієнтований на об'єднання популярних методик, спрямованих на забезпечення функціональності розроблюваного програмного продукту. Головним завданням FDD є розробка реальної працездатної версії ПЗ у визначені планом терміни. Модель FDD, подана на рис. 3, характеризує ітераційність та циклічність процесів розробки ПЗ.

Розробка моделі починається з високорівневого наскрізного аналізу кола наявних завдань і контексту системи, будується узагальнена модель програмного продукту. Подальша деталізація етапів створення ПЗ проводиться в інкрементних циклах моделі. Кожний інкремент забезпечує реалізацію проміжної версії ПЗ. Ітераційний цикл виконує умови базового інкремента.

FDD є зручним для розробки тих програмних продуктів, умови реалізації яких є строго визначеними замовником і враховуються на початку розробки загальної моделі та деталізуються на кожному інкрементному та ітераційному витку циклічного блоку реалізації визначених функцій.

Scrum є гнучким методом розробки програмного проекту невеликою (до 9 осіб) командою програмістів в умовах схематичної зміни (доповнення, модифікації) вимог до створюваного ПЗ. Процес розробки ПЗ є ітераційним і забезпечує можливість доповнення системи вимог на кожному ітераційному витку моделі (рис. 4).

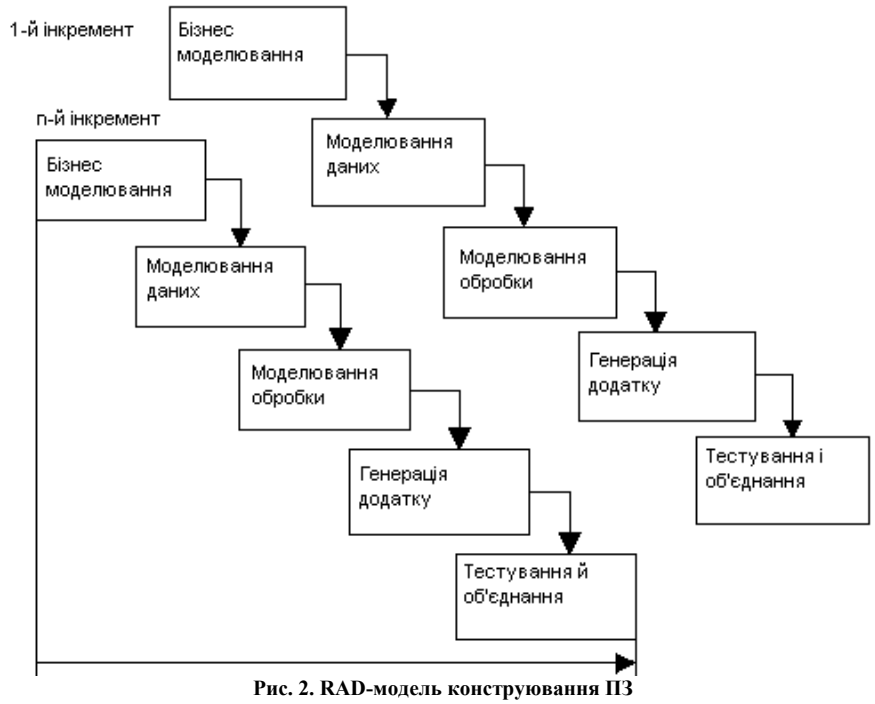

Рис. 2. RAD-модель конструювання ПЗ

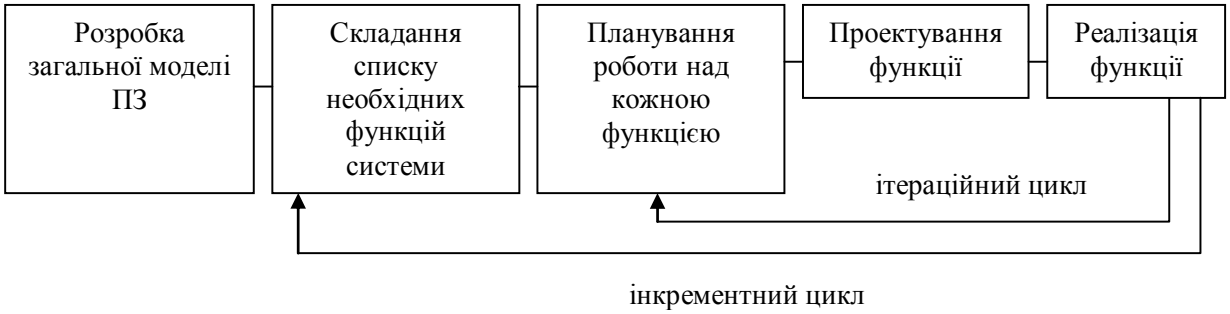

Рис. 3. Модель гнучкого методу функціонально-орієнтованої розробки ПЗ

Рис. 4. Схема методології Scrum

Недоліком використання Scrum-методу є характерна довгостроковість у розробці кінцевої версії ПЗ за рахунок обумовленої необхідності ітераційного обговорення результатів програмування із замовником на етапі формулювання вимог до програмного продукту. Проте тісний зв'язок розробників із користувачем забезпечує високу надійність та функціональність розроблюваного ПЗ.

До системи критеріїв оцінювання гнучких методів розробки ПЗ включимо рефакторинг як показник формування структури створюваного програмного продукту, принципи парного програмування, характер володіння кодом, міру участі замовника у процесі розробки, рівень інтеграції, можливість додавання функціональних властивостей продукту у процесі програмування. Результати порівняльного аналізу розглянутих гнучких методів програмування наведені в табл. 1.

Таблиця 1

Порівняльний аналіз гнучких методів розробки програмних продуктів

Критерії оцінювання	**Гнучкі методи розробки ПЗ**		
	Екстремальне програмування	**Scrum**	**Функціонально-орієнтована розробка**
Рефакторинг	Поліпшення структури програми протягом процесу програмування		Структура програми обговорюється і затверджується на початку розробки проекту
Парне програмування	Так (один програмує, інший думає про наступні підходи або тестування)	Частково (кодом займається повністю вся команда)	Ні (кожен програміст відповідає за розробку окремих функцій)
Характер володіння кодом	Колективне володіння		Кожен програміст володіє лише частиною коду, яку розробляє
Участь замовника у процесі розробки	Замовник (або представник) входить до команди розробників на правах контролюючого органа або тестувальника	Після кожної ітерації замовнику презентують результати виконання програми	Замовник на початку процесу програмування повністю визначає необхідний функціонал програми, перевіряє результати роботи на кінцевому етапі тестування
Рівень інтеграції	Високий (безперервна інтеграція)		Невисокий (інтеграція можлива, але вся структура програми визначається заздалегідь)
Можливість додавання функціональних властивостей продукту у процесі програмування	Так		Бажано з усім функціоналом визначитися перед розробкою продукту

Висновки

Гнучкі (agile) методи розробки програмного продукту з'явилися як альтернатива формальним і великоваговим методологіям, забезпечуючи процес більшою ефективністю та зменшуючи обсяг рутинної роботи, пов'язаної з оформленням супровідної документації. Використання методу екстремального програмування, скрам-методу та функціонально-орієнтованої розробки дозволяє підвищення швидкості й

надійності процесів програмування та дотримання визначених термінів виконання проекту. Наслідком використання agile-методології є забезпечення повного виконання усіх без спотворень вимог замовника у процесі розробки програмного забезпечення за рахунок наявної можливості використання додаткових консультацій із замовником на кожному ітераційному витку моделі створення програмного продукту.

Література

1. Бек К. Дванадцять принципів Agile-розробки / Кент Бек, Майк Бідл, Арі ван Беннекум, Алістер Коберн, Уорд Каннінгем Джеймс, Мартін Фаулер, Греннінг Джим, Хайсміт Ендрю, Хант Рон, Джеффріс Джон, Керн Брайан, Марік Роберт, К. Мартін, Стів Меллор.: Ward Cunningham, 2001.

2. Якобсон А. Унифицированный процесс разработки программного обеспечения / А. Якобсон, Г. Буч, Дж. Рамбо.– СПб. : Питер, 2002. – 496 С.

3. Beck, K., Fowler, M. (2004). Planning extreme programming. Upper Saddle River, NJ: AddisonWesley. 160 p.

4. Cockburn, A. (2002). Agile software development. Boston, MA: Addison-Wesley. – 278 p.

5. Schwaber, K., & Beedle, M. (2008). Agile software development with scrum. Upper Saddle River, NJ: Prentice-Hall. – 158 p.

6. Palmer, S. R., Felsing, J. M. (2002). A practical guide to feature driven development. Upper Saddle River, NJ: Prentice-Hall. – 304 p.

References

1. Bek. K. Dvanadtsiat' pryntsypiv Agile-rozrobky / Kent Bek, Maik Bidl, Ari van Bennekum, Alister Kobern, Uord Kanningam Dzhaims, Martin Fauler, Hrenninh Dzhim, Haismit Andriu, Hant Ron, Dzheffris Dzhon, Kern Braian, Marik Robert, K. Martin, Stiv Mellor.: Ward Cunningham, 2001.

2. Yacobson, A. Unyfytsyrovannyi protsess razrabotki programnogo obespicheniia/ A.Yacobson, G. Buch, Dzh. Rambo. – SPb. : Piter, 2002. – 496 C.

3. Beck, K., Fowler, M. (2004). Planning extreme programming. Upper Saddle River, NJ: AddisonWesley. 160, the

4. Cockburn, A. (2002). Agile software development. Boston, MA: Addison-Wesley. – 278, the

5. Schwaber, K., & Beedle, M. (2008). Agile software development with scrum. Upper Saddle River, NJ: Prentice-Hall. – 158.

6. Palmer, S. R., Felsing, J. M. (2002). A practical guide to feature driven development. Upper Saddle River, NJ: Prentice-Hall. – 304.

Рецензія/Peer review : 13.5.2013 р. Надрукована/Printed :18.6.2013 р.
Рецензент: Перевозніков С.І.