довкілля та модернізацію транспортної інфраструктури. Тільки завдяки спільним зусиллям можна забезпечити безпеку та стійкість транспортного сектору у майбутньому.

Література

1. Сталий розвиток для України [Електронний ресурс]. Режим доступу: https://sd4ua.org/golovni-temi-stalogo-rozvitku/transport/. Дата доступу: 15.04.2024

2. Що таке кібербезпека? [Електронний ресурс]. Режим доступу: https://www.microsoft.com/ uk-ua/security/business/security-101/what-is-cybersecurity. Дата доступу: 15.04.2024

УДК 005.8

ВИКОРИСТАННЯ ГНУЧКИХ МЕТОДІВ У ПРОЄКТНІЙ ДІЯЛЬНОСТІ З РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Петренко Ю.А., Бугаєвський М.С.

Харківський національний автомобільно-дорожній університет, Харків

Анотація. Вивчено передумови, які призвели до застосування послідовних та гнучких методів розробки програмного забезпечення. Проаналізовано основні аспекти, переваги та недоліки використання гнучких методологій у проєктній діяльності при розробці програмного забезпечення.

Ключові слова: agile, scrum, kanban, waterfall, розробка програмного забезпечення, управління проєктами.

Людство стало розробляти програмне забезпечення відносно недавно - розробка великих програмних систем почалася всього близько 50 років тому. Цілком природно, що ранні підходи до розробки були мало формалізовані і представляли собою процес, який прийнято називати Code-and-Fix (кодування і виправлення).

При такому підході розробка програмного забезпечення починається безпосередньо з кодування (без попереднього планування, аналізу вимог і проєктування). Після цього знайдені в коді проблеми (дефекти, невідповідність вимогам тощо) виправляються шляхом внесення множинних змін в код. Тому після деякої кількості таких змін система стає заплутаною, її складно підтримувати і розширювати.

Згодом стало зрозуміло, що для створення великих стійких програмних систем потрібні більш продумані й формальні підходи. У пошуках вирішення проблеми увагу було звернуто на більш зрілі на той час галузі людської діяльності, пов'язані зі складним виробництвом - перш за все, на системотехніку (systems engineering), а також інші інженерні дисципліни, такі як проєктування і будівництво мостів, споруд тощо.

У результаті спроби використовувати перевірені в інших областях інженерні методи для розробки програмного забезпечення з'явилася нова дисципліна - програмна інженерія (software engineering), а як фактичні стандарти на довгі роки утвердилися так звані інженерні методології розробки програмного забезпечення. Ці методології також часто називають заснованими на плані (plan-driven), тому що в їх основі лежить припущення про те, що процес розробки програмного забезпечення є детермінованим інженерним процесом, який можна спланувати від початку і до кінця та виконати відповідно до плану, використовуючи формальні інженерні підходи.

В якості основного варіанту побудови життєвого циклу використовувався водоспадний життєвий цикл (waterfall), що передбачає одноразовий прохід по фазах аналізу вимог, проєктування, кодування, тестування тощо. Таким чином, перший фундаментальний зсув у області методологій розробки представляв собою перехід від хаосу підходу code-and-fix до строго формалізованого підходу, прийнятого в інженерних методиках.

Навіть у 60-70-х роках минулого століття існували окремі проєкти, в яких використовувались інші варіанти життєвого циклу (як різні модифікації водоспадного, так і ітераційні). Однак такі проєкти були, скоріше, винятком із правил, і в рамках цього дослідження ми визначаємо, перш за все, основні й масові напрямки та тенденції.

Починаючи з найперших програмних проєктів і по нинішній день, розробка програмного забезпечення була й залишається малопередбачуваною та далеко не завжди успішною справою.

Значний відсоток проєктів зі створення програмного забезпечення, як і раніше, завершується з перевищенням бюджету, термінів, а створені в результаті програми часто не до кінця відповідають вимогам користувачів або приносять мало реальної користі бізнесу. Перераховані проблеми є основними проявами так званої кризи програмного забезпечення.

Незважаючи на значні інтелектуальні зусилля, витрачені на пошук способів подолання кризи, досі так і не знайдено універсальне рішення. Гнучкі методи – це сучасна відповідь програмній індустрії на питання, як все-таки треба виконувати проєкти, щоб вони з більшою долею ймовірності завершувалися успіхом і приносили користь усім зацікавленим сторонам - і перш за все, замовнику і команді проєкту.

Починаючи розробку корпоративного програмного забезпечення, важливо розуміти, з яким стеком технологій можна підійти до вирішення завдань та які основні показники програмного продукту важливі для замовника.

Вибір нового програмного забезпечення великим підприємством є досить повільним процесом, коли понад 50% роботи виконується ще на етапі аналізу початкових вимог. Це пов'язано з основними викликами сучасного світу - високим ступенем автоматизації і швидкою адаптацією процесів і систем до потреб клієнтів і ринку. Співробітникам, які керують процесом розробки програмного забезпечення та беруть у ньому участь, необхідно приділяти увагу

ефективності, використовувати відповідні технології, стилі і методи управління для розробки великих програм. Саме завдяки ним можна координувати роботу різних підрозділів, від якості і швидкості взаємодії між якими залежить швидкість розробки програмних компонентів та їх відповідність поставленим вимогам, яка і визначає їх цінність для бізнесу.

Звідси й виникає необхідність у наявності підходу, притримуючись якого кожен член команди буде розуміти свою роль, свої завдання у роботі над проєктом та критерії оцінки фінального результату.

Для досягнення цієї мети може використовуватися гнучка методологія розробки програмного забезпечення для управління проєктами. Вона спрямована на застосування ітеративних підходів розробки, динамічну постановку вимог та забезпечення їхньої реалізації за допомогою взаємодії між членами самоорганізованих робочих груп, до складу яких можуть входити спеціалісти різних напрямів розробки та підтримки ПЗ. Найбільш розповсюдженими з тих, що відносяться до класу гнучких є декілька методологій розробки ПЗ: екстремальне програмування, DSDM, Scrum, Kanban [1].

Ці методології використовуються як основа для ефективної організації роботи груп невеликого розміру, які виконують однорідну роботу. Ціллю, яку намагаються досягнути гнучкі методології є розподіл загального об’єму роботи, необхідного для виконання проєкту, на дрібні етапи / ітерації (тривалість кожного декілька тижнів), що мінімізує велику кількість ризиків, найбільшим з яких є невдоволеність клієнта результатом роботи через те, що вимоги змінилися з часу початку проєкту.

Кожна ітерація схожа на невеликий програмний проєкт та включає всі етапи, які необхідно пройти для отримання атомарного приросту за функціональністю: планування, аналіз вимог, проєктування, програмування, тестування та документування. Тож завдяки цьому підходу після завершення кожного етапу клієнт матиме нову версію продукту / програми. Команда обов’язково має виконувати перегляд та переоцінку пріоритетів після завершення кожного етапу.

Життєвий цикл продукту при використанні гнучкої методології розробки представлений на рис. 1.

В основу комунікативних аспектів Agile-методів покладено особисте спілкування. У більшості випадків команда, яка притримується даного підходу розташовується в одному приміщенні, інколи до цих команд входить замовник проєкту або його представник (визначає вимоги до продукту, це може бути менеджер проєкту, бізнес-аналітик або клієнт). Команда включає в себе тестувальників, дизайнерів інтерфейсу, технічних спеціалістів та менеджерів.

Робочий продукт – це основна метрика, за якою вимірюється ефективність таких методів. При використанні в команді спілкування обличчя до обличчя, гнучкі методи дозволяють зменшити об’єм письмової документації порівняно з іншими методиками. Однак, ці методи отримали певну долю

критики в свій бік через те, що деякі спеціалісти вважають, що використання робить команду менш дисциплінованою [2].

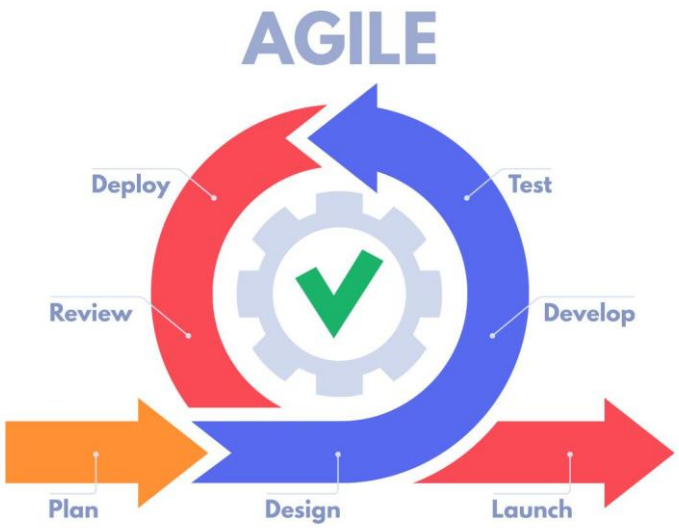

Рисунок 1 – Життєвий цикл продукту при використанні
гнучкої методології розробки

Agile – це сімейство процесів розробки, а не єдиний підхід у розробці ПЗ, і основні його принципи описуються у Agile Manifesto. Agile Manifesto не містить практичних порад, але відображає чотири основні ідеї та дванадцять принципів Agile.

Основні ідеї:

1. Люди та взаємодія набагато важливіші за процеси та інструменти.
2. Продукт, що працює, важливіший за документацію.
3. Співпраця із замовником є важливішою за обговорені умови контракту.
4. Готовність до змін важливіше за слідування початковому плану.

Принципи:

1. Задовольнити клієнта ранньою та безперебійною поставкою цінного програмного забезпечення.
2. Адекватно сприймати зміни вимог навіть у кінці розробки, що може підвищити конкурентну спроможність продукту.
3. Постійна поставка робочого ПЗ, з постійною частотою (кожен місяць / тиждень).
4. Тісне та щоденне спілкування замовника з розробниками протягом усієї тривалості проєкту.
5. Проєктом займаються мотивовані особистості, які забезпечені необхідними умовами роботи, підтримкою та довірою.
6. Рекомендований метод передачі інформації – особиста розмова (face-to-face).

7. Робоче ПЗ – найкращий вимірювач прогресу.

8. Спонсори, розробники та користувачі повинні мати можливість підтримувати постійний темп на невизначений термін.

9. Постійна увага сприяє покращенню технічної досконалості та зручному дизайну.

10. Простота – мистецтво не робити зайвої роботи.

11. Найкращі технічні вимоги, дизайн та архітектура створюються самоорганізованою командою.

12. Постійна адаптація до обставин, що змінюються. Команда зобов'язана систематично аналізувати можливі способи покращення ефективності та відповідно корегувати стиль своєї роботи [3].

До недоліків цього підходу можна віднести наступні важливі моменти.

Під час використання Agile-підходу дуже часто нехтують створенням плану дій, спрямованих на розвиток продукту на довгий період часу, та управлінням вимогами. Він надає можливість замовнику в будь-який момент, незважаючи на етап виконання, виставити нові вимоги, які можуть суперечити архітектурі продукту, який вже частково створено. Такий підхід інколи може призводити до катастрофічних наслідків: перероблення практично на кожній ітерації [4].

Вважають, що робота в Agile мотивує розробників вирішувати всі поставлені задачі достатньо простим та швидким способом. Таким чином, виконавці проєкту можуть не звертати увагу на правильність написання коду з урахуванням вимог використаної в основі платформи. І вже після початку використання розробленого продукту можуть виникнути конфліктні ситуації за технічною частиною, що буде свідчити про відсутність узгодження на етапах проєктування. Відповідно, такий підхід призводить до зниження якості продукту та накопичення дефектів.

Список літератури

1. Ashmore, S., & Runyan, K. (2014). *Introduction to Agile Methods.* Addison-Wesley Professional.

2. Moran, A. (2015). *Managing Agile: Strategy, Implementation, Organisation and People.* Springer.

3. *Manifesto for Agile Software Development*. Retrieved May 13, 2020, from Manifesto for Agile Software Development: https://agilemanifesto.org/

4. Кон, М. (2011). *Succeeding with Agile: Software Development Using Scrum*. Вильямс.