**Практична робота №14. Управління програмними проектами. Методологія RUP**

**На період** карантину в дистанційній формі навчання на надані запитання потрібно надати письмові відповіді, надіславши їх на електронну адресу викладача. Файл надавати з іменем у форматі

**PI<Номер групи><Номер лекції / практичної / лабораторної>[-<Номер завдання>][літера позначення типу роботи L – лекція, P –практична, R – лабораторна]<Прізвищеанглійською>**. Наприклад, **PI3104L**buts.doc. Відповіді на запитання повинні бути не довгими і змістовними. Не копіюйте фрагментів з різних інформаційних джерел, подумайте і викладіть свою точку зору. При наявності відповідей-"близнюків" відповідь буде зараховуватися першому за часом надсилання.

**Строк виконання цієї роботи 24.05.2020**

**Мета:**

1. Розглянути основні етапи та артефакти процесу розробки ПЗ за методологією RUP (Rational Unified Process)

2. Згідно стандарту RUP розробити специфікацію системних вимог (СВ) в вигляді структурованого текстового опису для наданої предметної області.

**Завдання:**

1. Опрацювати теоретичні відомості. Перевірити засвоєння Вами матеріалу на контрольних запитаннях. Прочитати методичні вказівки.
2. На основі прикладу розробити текстовий опис основних СВ для своєї для проекту, над яким працювали на попередніх лабораторних та практичних роботах.
3. По закінченню лабораторну роботу потрібно здати на перевірку викладачеві, надіславши електронною поштою. Якщо викладач знаходить помилки чи неточності, він може повернути роботу на доопрацювання.

**Контрольні питання для самоперевірки**

1. Для чого була розроблена методологія RUP?
2. З яких фаз складається загальний ЖЦ системи методології RUP?
3. Що за термінологією RUP визначають терміни "технологічний процес", "дисципліна", "артефакт"?
4. В чому переваги RUP?
5. В чому недоліки RUP?

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

**Rational Unified Process (RUP)** є результатом об єднання підходів **Rational Approach** та **Objectory Process**, яке відбулося після злиття у 1995 році компаній **Rational Software** і **Objectory AB** (створеної Іваром Якобсоном). Можна також стверджувати, що RUP поєднує в собі найбільш позитивні риси інших методологій проектування, зокрема таких як каскадна та спіральна моделі життєвого циклу розробки ПЗ.

**Методологія RUP** описує абстрактний загальний процес, на основі якого організація або проектна команда повинні створити спеціалізований процес, орієнтований на її по-треби. RUP включає інтегрований пакет методик, технологій та програмних засобів. Сутність роботи в рамках RUP – це створення й супровід моделей на базі Unified Modeling Language (UML).

В основі RUP лежать наступні принципи:

- рання ідентифікація і безперервне усунення основних ризиків;

- концентрація на виконанні вимог замовника до виконуваної програми (аналіз і побудова моделі варіантів використання);

- очікування змін у вимогах, проектних рішеннях і реалізації;

- компонентна архітектура, що реалізовується і тестується на ранніх стадіях проекту;

- постійне забезпечення якості на всіх етапах розробки ПЗ;

- робота над ПЗ у згуртованій команді, ключова роль в якій належить архітекторам.

Основним принципом RUP є принцип ітеративної розробки (iterative development), в рамках якої розробка ведеться у вигляді короткочасних міні- проектів фіксованої тривалості (наприклад по 4 тижні), які називаються ітераціями (iteration). Кожна ітерація складається із власних фаз аналізу вимог, проектування, реалізації, тестування, інтеграції та створенням робочої системи (рис.1).



Рис. 1 – Фази ітеративного процесу RUP

Згідно методології RUP загальний ЖЦ системи складається з декількох циклів-ітерацій, кожний з яких містить чотири фази: початок (Inception); розробка (Elaboration); побудова (Construction); впровадження (Transition). Для кожної фази методологія задає склад і послідовність робіт, а також правила їхнього виконання; розподіл повноважень серед учасників проекту (ролі); склад і шаблони формованих проміжних і підсумкових документів; порядок контролю та перевірки якості. Кожна фаза може бути розбита на етапи (ітерації), у результаті яких випускається версія для внутрішнього або зовнішнього використання. Проходження крізь ці чотири фази називається *циклом розробки*; кожний цикл завершується генерацією чергової версії системи. Якщо після цього робота над проектом не припиняється, то отриманий продукт продовжує розвиватися і знову пройде ті ж фази В кінці кожної ітерації (яка в ідеалі продовжується від 2 до 6 тижнів) проектна команда повинна досягти запланованих на дану ітерацію цілей, створити або доопрацювати проектні артефакти і отримати проміжну, але функціональну версію кінцевого продукту. Ітеративна розробка дозволяє швидко реагувати на зміну вимог, виявляти та усувати ризики на ранніх стадіях проекту, а також ефективно контролювати якість створюваного продукту.

В рамках RUP робота над проектом ПЗ включає 4 основні фази, дві перші з яких ми розглянемо більш детально:

1) **Початок (Inception)** – визначення мети, доцільності та технічної можливості виконання проекту розробки ПЗ (рис.2 ).



Рис. 2 – фаза Початку (Inception).

При цьому також мають бути визначеними

1. Коло осіб, які зацікавлені в цьому проекті та перелік їх основних СВ;

2. Словник (глосарій) проекту;

3. На підставі цього - загальна концепція побудови системи;

4. Перелік варіантів альтернативних системних архітектур (candidate architectures) для ПЗ;

5. Опис можливих методів тестування ПЗ, що розробляється (тобто план дій щодо визначення коректності побудованої ПЗ).

Ці проектні документи мають бути представлені у відповідній структурованій формі (див. п. "Текстовий формат специфікація СВ").

2) **Розвиток (Elaboration)** – формування детального проекту ПЗ. Для цього на підставі результатів першого етапу мають бути:

- визначено основний варіант системної архітектури ПЗ;

- детально розроблені всі основні прецеденти використання ПЗ (з використанням формалізованих нотацій – наприклад, UML, SADT, DFD, ER.

На цьому етапі проводиться ітеративна реалізація базової архітектури ПЗ, створення прототипів найбільш критичних окремих програмних компонентів специфікація всіх необхідних СВ, визначення більш реальних оцінок (кількісних метрик) для тестування ПЗ (рис.3).

RUP передбачає виконання різних видів діяльності, наприклад опис прецедентів в рамках визначених дисциплін (disciplines), які називаються технологічними процесами (workflow). *Дисципліна* – це набір видів діяльності та пов’язаних з ними артефактів в рамках одного етапу, наприклад аналізу вимог. *Артефакт* (artifact) – будь-який результат виконання певних дій по розробці ПЗ: текстова документація, занотована СВ, програмний код, графічне зображення, схема БД, тощо.



Рис. 3 – фаза Розвитку (Elaboration)

В рамках RUP існує 9 дисциплін (рис. 4), з них 6 основних:

1) Бізнес-моделювання (Business modeling) – моделювання об єктів предметної області, моделювання процесів. Бізнес-моделювання пояснює, як описати бачення організації, в якій буде розгортатись система і як використати це бачення для виділення процесу, ролей та обов'язків. Метою бізнес-моделювання є встановити глибше розуміння бізнесу, тобто, пояснити програмістам, структуру і динаміку цільової організації (клієнта), нинішні проблеми в організації, а також можливі удосконалення. Також забезпечити загальне розуміння цільової організації між клієнтами, кінцевими користувачами та розробниками.

2) Керування вимогами (Requirements) – аналіз вимог, визначення прецедентів та інших артефактів, що пояснюють вимоги, допомагають виявити запити зацікавлених осіб і перетворити їх в набір вимог, робочих продуктів, що осягають створювану систему й надають детальні вимоги до того, що система повинна робити.

3) Аналіз і Проектування (Analysis and Design) – визначення архітектури ПЗ, бази даних, конфігурації мережі, тощо. Метою аналізу і проектування, є показати, яким чином система буде реалізована. Ціллю є створення системи, яка:

* Виконує — в особливому середовищі реалізації — задачі та функції описані в описах прецедентів.
* Виконує всі свої вимоги.
* Легко змінити, коли змінюються функціональні вимоги.

Проектування дає в результаті модель проектування, а аналіз відповідно модель аналізу. Модель дизайну служить абстракцією вихідного коду; тобто модель дизайну працює «синькою», розміткою того як буде структурований та написаний вихідний код. Дизайн моделі складається проектування класів структурованих в пакети і підсистеми з чітко визначеними інтерфейсами, які представляють, що стане компонентами у реалізації. Він також містить опис того, як об'єкти цих сконструйованих класів співпрацюють для виконання прецедентів.

4) Реалізація (Implementation) – програмування системи, але не її розгортування; Метою реалізації є:

* Визначити організацію коду з точки зору реалізації підсистем, які організовані в шари.
* Реалізація класів та об'єктів у термінах компонентів (вихідних файлів, виконуваних файлів, та інших).

Системи реалізуються через реалізацію компонентів. Процес описує як повторно використати існуючі компоненти, чи реалізувати нові компоненти з чітко визначеними відповідальностями, роблячи систему легше підтримуваною і збільшуючи можливості для повторного використання.

5) Тестування (Test). Цілі тестування:

* Перевірити взаємодії між об'єктами.
* Перевірити належну інтеграцію всіх компонентів програмного забезпечення.
* Переконатися, що всі вимоги були правильно виконані.
* Визначити та переконатись що дефекти будуть розглянуті до розгортання програмного забезпечення.
* Переконатись, що всі дефекти виправлені, повторно перевірені та закриті.

RUP пропонує ітеративний підхід, а це означає, що тестування відбувається протягом всього проекту. Це дозволяє виявляти дефекти якомога раніше, що радикально знижує вартість виправлення дефекту. Тести проводяться за чотирма вимірами якості: *надійності, функціональності, продуктивності додатків і продуктивності системи*. Для кожного з цих вимірів критеріїв якості, процес описує як пройти життєвий цикл планування, проектування, виконання і оцінки тесту.

6) Розгортання (Deployment).

Метою розгортання є успішно створювати релізи продукту, та постачати програмне забезпечення для кінцевих користувачів. Дісципліна охоплює широке коло заходів, у тому числі виробництво зовнішніх версій програмного забезпечення, запаковування програмного забезпечення та бізнес-додатків, розповсюдження програмного забезпечення, встановлення програмного забезпечення та надання допомоги і підтримки для користувачів..

В рамках RUP також існує 3 допоміжні дисципліни:

1) Керування проектом (Project management);

2) Керування змінами (Change management);

3) Середовище (Environment) – інсталяції необхідних засобів, налагодження процесу для ПЗ.

Процес RUP – процес, керований на основі варіантів використання. Це означає, що як метод опису функціональних вимог до системи, а також як природна одиниця для подальшого планування і оцінки виконання робіт, застосовуються сценарії використання. Сценарії використання дозволяють легко виявляти реальні потреби майбутніх користувачів ПЗ і відстежувати повноту опису їх вимог. Крім того, використання завершених сценаріїв, як одиниць виміру прогресу, допомагає уникнути неадекватної оцінки ступеня ви-конання проекту виконавцем.



Рис. 4 – Фази та дисципліни RUP

Методологія RUP має наступні переваги.

RUP дозволяє компанії-розробникові налаштовувати весь процес розробки ПЗ. На відміну від більшості сучасних методологій або вимог до процесу розробки, орієнтованих на строго визначений рівень формалізації процесу (як правило, або дуже високий, або, навпаки, дуже низький), RUP дозволяє отримати саме той рівень формалізації, який необхідний в проекті.

При розробці ПЗ для державних підприємств і у низці інших ситуацій компанії-розробникові доводиться виконувати формальні вимоги вітчизняних і міжнародних стандартів (ГОСТ серій 19 і 34, ДСТУ, ISO/IEC 12207:1995 та ін.). В цьому випадку можна налаштувати RUP на досить високий рівень формалізації процесу. Більше того, при використанні інструментальних засобів IBM Rational нескладно розробити шаблони документів, які створюватимуться автоматично за допомогою IBM Rational SODA на основі стандартних моделей та артефактів і відповідати вимогам стандартів.

Використання RUP з досить високим рівнем формалізації процесу також може допомогти компанії-розробникові виконувати вимоги сертифікації CMM (Capability Maturity Model). Оскільки RUP є добре документованим процесом, впровадження RUP допоможе компанії досягти рівнів 2 і 3 СММ. У ще більшій мірі RUP може допомогти при впровадженні CMMI (Capability Maturity Model Integration), оскільки CMMI більшою мірою відповідає сучасному ітераційному підхо-ду до розробки ПЗ.

З іншого боку, при використанні RUP в невеликих колективах для розробки не критичного ПЗ можна вибрати досить низький рівень формалізації, який не гальмуватиме розробку необхідністю формування і ретельного оформлення багатьох артефактів, що не роблять реального впливу на якість проектування та реалізації ПЗ.

До недоліків RUP можна віднести те, що широкі можливості налаштування не роблять впровадження RUP простим. Ефективному використанню RUP перешкоджає дуже багато складнощів. Часто послідовність фаз “ Початок – Розробка – Побудова – Впровадження ” помилково інтерпретують як фази “ Аналіз – Дизайн – Реалізація – Впровадження ” і це практично зводить нанівець всі переваги RUP. Для значної кількості людей виявляється незвичною концепція документування вимог у вигляді варіантів використання UML. В результаті специфікація вимог, створена проектною командою, може виявитися абсолютно невиразним документом, який ніхто не читатиме. Окрім того, для повноцінного впровадження RUP компанія-розробник повинна витратити значні кошти на навчання співробітників. При цьому спроба обійтися своїми силами швидше за все буде приречена на невдачу – необхідно шукати фахівця з відповідним досвідом або залучати консультантів.

Методологія RUP підходить, насамперед, для масштабних і довгострокових проектів.

**Текстовий формат специфікація СВ**

В контексті методології RUP системні вимоги (СВ) - це весь набір певних *ПРЕЦЕДЕНТІВ* використання ПЗ, тобто концептуальна модель функціонування системи та її оточення.

*прецедент*– це набір взаємопов’язаних *cцeнapiïв*, який описує використання ПЗ певним *aктopoм* для вирішення однієї із задач.

*актop -* сутність, яка має поведінку: напр., людина (користувач), окремий програмний компонент або інша програмна система.

*сцeнapiй) –* послідовність дій або взаємодій між актором та ПЗ.

Первинною формою опису прецедентів є текст, який має бути сформованим у процесі виявлення СВ (шляхом спілкування із майбутніми користувачами, експертами із предметної області тощо). Для подальшої формалізації (структуризації) текстового опису прецедентів існує декілька форм:

* *Cвoбoднa фopмa oписy* ***–*** неформальний стиль описання. Опис прецеденту займає кілька абзаців і охоплює різні можливі сценарії.
* *Стислий oпис* ***–*** анотація у вигляді одного абзацу. Вона описує тільки головний успішний сценарій.
* *Розгорнутий oпис* ***–*** при такому підході детально описуються всі шаги і варіанти розвитку сценарію.

Найбільш інформативним та корисним для подальшого використання СВ в проектуванні ПЗ є розгорнутий опис прецедентів. Він має наступний формат, який ми розглянемо на прикладі розробки ПЗ для оформлення продажів товару в деякому супермаркеті:

**ПРИКЛАД Опис розгорнутого сценарію за стандартом RUP функціональної схеми Веб-ресурсу “Експертне оцінювання на основі парних порівнянь” :**

**1. Зацікавлені особи прецеденту та їх вимоги:** в цій секції опису подається, що повинна зробити ПЗ для усіх можливих учасників процесу.

*Наприклад:*

1) Експерт: повинен точно і швидко заповнити дані щодо оцінювання заданої характеристики всіх потрібних об’єктів у таблиці основі парних порівнянь; надіслати готові результати оцінювання до бази даних.

2) Менеджер: повинен отримати показники рангів оцінювання об’єктів, що виконали всі експерти; визначає достовірність оцінювання шляхом визначення коефіцієнту конкордації.

Модульна структура програми зображена на рис.5.

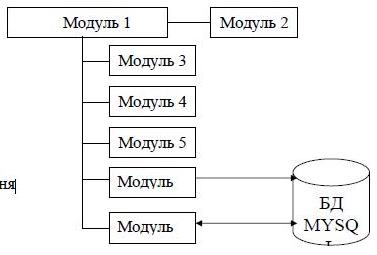


Рис. 5. Функціональна схема Веб-ресурсу “Експертне оцінювання на основі парних порівнянь”

|  |  |  |
| --- | --- | --- |
| №  з/п | Позначення | Призначення програмних модулів |
| 1 | Модуль1 | Головний модуль. Містить головну веб-сторінку та  забезпечує перехід з однієї веб-сторінки на іншу |
| 2 | Модуль2 | Модуль зворотного зв’язку |
| 3 | Модуль3 | Модуль інформаційної довідки 1 |
| 4 | Модуль4 | Модуль інформаційної довідки 2 |
| 5 | Модуль5 | Модуль інформаційної довідки 4 |
| 6 | Модуль6 | Модуль формування таблиці основі парних  порівнянь |
| 7 | Модуль7 | Модуль конкордації рангових оцінок експертів |

**2. Користувач ПЗ тобто основний актор цього прецеденту**: це менеджер, який приймає рішення за допомогою ПЗ, що має бути розроблена.

**3. Передумови прецеденту (preconditions):** це перелік **подій** які завжди повинні виконуватися до початку сценарію поточного прецеденту. Передумови не перевіряються у даному сценарії прецеденту, вони вважаються результатом успішного виконанням деякого іншого прецеденту.

*Наприклад*, це можуть бути наступні умови:

ПЗ має бути активним.

Експерти повинні успішно пройти процедуру аутентифікації в ПЗ.

**4. Основний успішний сценарій**: ця секція специфікації описує «сценарій успіху». Тобто, дії що призводять до успішного завершення подій в основному процесі.

*Наприклад*:

• ПЗ надсилає повідомлення експертам про необхідність проведення оцінювання.

• Експерт вводить бали оцінювання характеристики об’єктів з переліку менеджера.

• ПЗ надсилає повідомлення менеджеру про виконання оцінювання всіма експертами.

• Менеджер отримує інформацію для прийняття рішення після здійснення експертизи всіх експертів.

**5. Розширення основного сценарію або альтернативні потоки:** у цій секції вказуються всі інші можливі варіанти розвитку подій в ПЗ, які призводять до успішного завершення основного сценарію або альтернативні сценарії, що приводять до некоректного завершення прецеденту. При цьому після виконання обробки усіх можливих розширень прецеденту ПЗ повинна забезпечити повернення користувача в основний сценарій, якщо в ПЗ не передбачено альтернативний хід подій.

*Наприклад*, розширення основного сценарію:

***Неправильний ідентифікатор оцінювання*:**

Експерт повторно вводить коректний ідентифікатор (це є точка повернення в основній сценарій).

**6. Пост-умови (postconditions):** це перелік умов, які завжди повинні виконуватися у випадку успішного виконання основного сценарію (тобто коли задовольняються інтереси усіх зацікавлених осіб).

*Наприклад:*

1) Дані щодо оцінювання внесені та оброблені в таблиці попарного порівняння.

2) Транзакція успішно зафіксована в базі даних ПЗ.

3) Друк даних експертного оцінювання.

**7. Спеціальні системні вимоги:** ця секція опусу містить перелік не функціональних вимог, атрибутів, якості або певних обмежень в функціонуванні ПЗ, що є пов’язаними із даним прецедентом. *Наприклад,* це може бути:

1. Необхідно забезпечити 100% надійність обробки всіх транзакцій.

2. Потрібно забезпечити можливість локалізації інтерфейсу користувача ПЗ (тобто підтримка багатомовності діалогу з ПЗ).

**8. Список необхідних технологій та додаткових пристроїв**: типовим прикладом можуть бути технічні вимоги, які висувають зацікавлені особи для технологій вводу/виводу даних тощо.

*Наприклад:*

1) ПЗ має бути розроблена як Web-орієнтована система.

2) ПЗ повинно якісно відображатись на браузерах мобільних та планшетних пристроїв .