**Практична робота 06-1.**  **Методологія розробки програмного забезпечення. Застосування методології RUP для ітеративної розробки**

**Мета роботи**: засвоєння навичок визначення комплекту UML-моделей для конкретних етапів життєвого циклу програмного забезпечення при ітеративної розробки за RUP – методологією.

**Завдання:**

1. Ознайомитися з теоретичною частиною
2. Для власного проекту визначте перелік UML- моделей через які ви будете надавати аспекти моделювання свого майбутнього програмного продукту з погляду використання ітераційної моделі та методології RUP.
3. Результати оформити у вигляді таблиці:

|  |  |  |  |
| --- | --- | --- | --- |
| <Назва проектуємого програмного продукту > | | | |
| №№ | Вид UML- моделі | Призначення та проектний аспект, що подається | Назва циклу ЖЦ ПЗ та циклу за методологію RUP із визначенням входження до ітерації |
|  |  |  |  |

Якщо UML-модель використовується на декількох циклах ЖЦ ПЗ або циклах за методологію RUP, ці визначення надати окремими рядками в їх логічній послідовності.

1. По закінченню практичну роботу потрібно здати на перевірку викладачеві, надіславши електронною поштою на адресу [t.i.lumpova@gmail.com](mailto:t.i.lumpova@gmail.com). Якщо викладач знаходить помилки чи неточності, він може повернути роботу на доопрацювання.

Файл з роботою повинен мати назву в такому форматі:

**OPI <Номер групи><Номер лекції / лабораторної /практичної> [літера позначення типу роботи L – лекція, R – лабораторна, Р - практична]<Прізвище англійською>**.. Наприклад, OPI**3104-1P**buts.doc.

Не копіюйте фрагментів з різних інформаційних джерел, подумайте і викладіть свою точку зору. При наявності робіт-"близнюків" відповідь буде зараховуватися першому за часом надсилання.

Тему в заголовку листа записати

**OPI<Номер групи>-ПР<Номер практичної>-<Прізвище >**

**Строк виконання цієї роботи ІПЗ-31 – 04.06.2024**

**ІПЗ-32 – 05.06.2024**

Всі запитання, що виникнуть, надсилайте на електронну адресу викладача, Тему в заголовку листа записати

**OPI <Номер групи>-Запитання-<Прізвище >**.

**КОНТРОЛЬНІ ПИТАННЯ:**

1. Дайте визначення проекту. Які ознаки має проект?
2. В чому полягає мета проекту?
3. Які основні функції проектного аналізу?
4. Наведіть класифікацію проектів.
5. На які групи можна поділити проекти за видами?
6. Визначте основні методології процесів розробки ІТ проектів.
7. Що потрібно, щоб програмний проект був успішним?

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

На сьогоднішній день практично всі провідні компанії - розробники технологій і програмних продуктів (IBM, Oracle, Borland) мають розвинені технології створення ПЗ, які створювалися як власними силами, так і за рахунок придбання продуктів і технологій, створених невеликими спеціалізованими компаніями. Одна з найбільш досконалих технологій, яка претендує на роль світового корпоративного стандарту - Rational Unified Process (RUP). RUP являє собою програмний продукт, розроблений компанією Rational Software (www.rational.com), яка нині входить до складу IBM.

Rational Unified Process (RUP) – одна з кращих методологій розробки програмного забезпечення, створена в компанії Rational Software, допомагає створювати складні програмні системи, ґрунтуючись на індустріальних методах розробки. Одним з основних засад, на які спирається RUP, є процес створення моделей за допомогою уніфікованої мови моделювання (UML).

RUP значною мірою відповідає стандартам та нормативним документам, пов'язаним з процесами ЖЦ ПЗ і оцінкою технологічної зрілості організацій-розробників (ISO 12207, ISO 9000, CMM та ін). Основним принципом RUP є ітераційний і інкрементний (нарощуваний) підхід до створення ПЗ. У відповідності з ним розробка системи виконується у вигляді декількох короткострокових міні-проектів фіксованої тривалості (від 2 до 6 тижнів), так званих ітерації. Кожна ітерація включає свої власні етапи аналізу вимог, проектування, реалізації, тестування, інтеграції і завершується створенням дієвої системи.

***Ітераційний цикл RUP***

Ітераційний цикл ґрунтується на постійному розширенні і доповненні системи в процесі декількох ітерацій з періодичним зворотним зв'язком і адаптацією додаються модулі до існуючого ядра системи. Система постійно розростається крок за кроком, тому такий підхід називають ітераційним і інкрементним.

Згідно RUP, ЖЦ ПЗ розбивається на окремі цикли, в кожному з яких створюється нове покоління продукту. Кожен цикл, у свою чергу, розбивається на чотири послідовні стадії:

• початкова стадія (inception);

• стадія розробки (elaboration);

• стадія конструювання (construction);

• стадія введення в дію (transition).

В основі методології RUP, як і багатьох інших програмних методологій, які об'єднують інженерні методи створення, лежить "покроковий підхід". Він визначає етапи життєвого циклу, контрольні точки, правила робіт для кожного етапу і, тим самим, впорядковує проектування та розробку ПЗ. Для кожного етапу життєвого циклу методологія задає:

- склад і послідовність робіт, а також правила їх виконання;

- розподіл повноважень серед учасників проекту (ролі);

- склад і шаблони формуються проміжних і підсумкових документів;

- порядок контролю та перевірки якості.

Методологія RUP дозволяє об'єднати проектну команду, надаючи в її розпорядження перевірені світовою практикою кращі підходи до розробки ІС. До них відносяться такі процеси життєвого циклу ПЗ, як управління проектами, бізнес-моделювання, управління вимогами, аналіз і проектування, тестування і контроль змін. Впровадження RUP в організації сприяє виробленню якісних внутрішньокорпоративних стандартів та підвищенню загальної культури розробки.

***Основні поняття RUP***

Статична структура - опис логічного групування елементів процесу (завдання, роль, артефакти) у головні дисципліни процесу.

Роль - область функціональних обов'язків компетенції та відповідальності, які повинна мати людина або група людей.

Завдання - одиниця роботи в RUP, яка може бути запропонована ролі для виконання. Час виконання коливається від кількох годин до декількох днів.

Категорії кроків завдання:

1. Кроки для роздумів (визначення суті завдання, формулювання результату).

2. Виконувані кроки (створення та модифікація деяких артефактів).

3. Перевірочні кроки (оцінка результату за певним критерієм).

Артефакт - обсяг інформації, створений, змінений або використаний процесом, що визначає сферу відповідальності і підлягає контролю версій. Артефакти можуть бути вихідними даними для рішення задачі, можуть створюватися роллю в процесі рішення задачі або бути результатом цього рішення.

Процес - набір частково впорядкованих кроків для досягнення мети. В програмному забезпеченні, ціль - необхідність побудови програмного продукту чи розширення існуючого. У технології, ціль полягає в тому, щоб розширити або збільшити модель процесу; відповідає діловому прецеденту використання в діловій розробці.

***Чотири ключові частини процесу*:**

1. Опис ролі ( «хто робить?")

2. Опис артефакту ( «що робить?)

3. Опис робочого процесу ( «коли робить?")

4. Опис завдання ( «як робить?")

Робочий процес - опис послідовності задач і їх взаємодії між ролями, що створюють видимий результат.

***Дисципліни RUP. Додаткові елементи процесу.***

*Дисципліна -* логічна група, що складається з ролей, завдань, артефактів та інших засобів управління процесом (концепцій, інструкцій, шаблонів).

*Дисципліна* - Високорівневий робочий процес. Дисципліна розділяється на деталі робочого процесу - робочі процеси всередині дисципліни.

***Дисципліни:***

1. Бізнес-моделювання - призначена для:

• Розуміння структури й поводження організації замовника, її поточних проблем, можливих змін бізнес-процесів;

• Визначення єдиного словника термінів і понять між замовниками, користувачами та розробниками;

• Для визначення деяких системних вимог.

2. Управління вимогами - призначена для:

* Визначення функцій системи між усіма зацікавленими особами (користувачами, розробниками, замовниками);
* Визначення меж системи;
* Розробка бази для технічного планування змісту ітерацій та визначення часу і вартості розробки системи;
* Визначення користувацького інтерфейсу системи.

3. Аналіз і проектування - призначена для:

• Побудови дизайну системи на підставі вимог до системи;

• Розробки архітектури системи;

• Підготовки дизайну впровадження системи.

4. Реалізація - призначена для:

• Визначення структури та організації вихідних кодів системи, розбитих на окремі рівні реалізації;

• Для реалізації класів і об'єктів в термінах компонентів (вихідні двійкові і виконавчі файли);

• Для модульного тестування розроблених компонентів;

• Для інтеграції результатів розробки окремих компонентів у працездатну систему.

5. Впровадження - призначена для:

• Для опису всіх дій, пов'язаних з доступністю кінцевого продукту для користувачів.

6. Тестування - призначена для:

• Перевірки якості розроблюваного продукту за допомогою спеціальних процесів;

• Пошук і реєстрація дефектів;

• Демонстрація достовірності зроблених на етапі проектування та збору вимог припущень (архітектурних рішень).

7. Управління проектом - призначена для:

• Забезпечення розробки інструкцій з планування, підбору персоналу, виконання, і контролю проектів (моніторингу прогресу ітеративного проекту та метрик);

• Забезпечення управління ризиками.

8. Управління змінами - призначена для:

• Контролю змін;

• Підтримання цілісності продукту і складових його артефактів.

Додаткові елементи процесу:

1. Інструкції - правила, рекомендації для виконання завдань, кроків і поводження з артефактами.

2. Шаблони - для різних артефактів.

3. Інструкції з використання інструментальних засобів розробки.

4. Основні концепції, визначення та принципи (Concepts).

5. Roadmaps - графічні діаграми.

***Інструментарій для роботи за методологією RUP***

RUP спирається на інтегрований комплекс інструментальних засобів Rational Suite. Він існує в наступних варіантах:

• Rational Suite AnalystStudio - призначений для визначення та управління повним набором вимог до розроблюваної системи;

• Rational Suite DevelopmentStudio - призначений для проектування та реалізації;

• Rational Suite TestStudio - являє собою набір продуктів, призначених для автоматичного тестування додатків;

• Rational Suite Enterprise - забезпечує підтримку повного життєвого циклу ПЗ та призначений як для менеджерів проекту, так і окремих розробників, які виконують кілька функціональних ролей в команді розробників.

До складу Rational Suite, крім самої технології RUP як продукту, входять наступні компоненти:

• Rational Rose - засіб візуального моделювання (аналізу і проектування), який використовує мову UML;

• Rational XDE - засіб аналізу і проектування, інтегрований з платформами MS Visual Studio .NET і IBM WebSphere Studio Application Developer;

• Rational Requisite Pro - засіб керування вимогами, призначений для організації спільної роботи групи розробників. Воно дозволяє команді розробників створювати, структурувати, встановлювати пріоритети, відслідковувати, контролювати зміни вимог, що виникають на будь-якому етапі розробки компонентів програми;

• Rational Rapid Developer - засіб швидкої розробки додатків на платформі Java 2 Enterprise Edition;

• Rational ClearCase - засіб управління конфігурацією ПЗ;

• Rational SoDA - засіб автоматичної генерації проектної документації;

• Rational ClearQuest - засіб для управління змінами та відстеження дефектів в проекті на основі засобів e-mail і Web;

• Rational Quantify - засіб кількісного визначення вузьких місць, що впливають на загальну ефективність роботи програми;

• Rational Purify - засіб для локалізації важко виявлених помилок часу виконання програми;

• Rational PureCoverage - засіб ідентифікації ділянок коду, пропущених при тестуванні;

• Rational TestManager - засіб планування функціонального і навантажувального тестування;

• Rational Robot - засіб запису і відтворення тестових сценаріїв;

• Rational TestFactory - засіб тестування надійності;

• Rational Quality Architect - засіб генерації коду для тестування.

В основі RUP лежить методика проектування інформаційних систем на основі UML

***Використання моделей***

Уніфікований процес можна представити як суму різних видів діяльності компанії-розробника, необхідних для переведення вимог замовника в програмну систему. Систему, яка давала б «значимий результат» користувачам і виконувала б саме те, що вони очікують від системи. Тому процес управляється варіантами використання (Use Case) системи, або інакше – прецедентами.

Для реалізації вимог замовника у встановлені терміни, уніфікований процес поділяється на фази, які складаються з ітерацій, тому процес ще називають ітеративним і інкрементним. Кожна ітерація проходить цикл основних робіт і підводить розробників до кінцевої мети: створення програмної системи. В ході ітерацій створюються проміжні артефакти.

Вся розробка ПЗ розглядається в RUP як процес створення артефактів.

Одним з найцікавіших класів артефактів проекту є моделі, які дозволяють розробникам визначати, візуалізувати, конструювати і документувати артефакти програмних систем. Кожна модель є самодостатнім поглядом на розроблювану систему і призначена як для окреслення проблем, так і для пропозиції рішення. Самодостатність моделей означає, що аналітик або розробник може з конкретної моделі почерпнути всю необхідну йому інформацію, не звертаючись до інших джерел.

***Визначення вимог***

Уніфікований процес – це процес, керований прецедентами, які відображають сценарії взаємодії користувачів. Для полегшення цього процесу аналітики використовують діаграми прецедентів.

Для деталізації конкретного прецеденту використовується діаграма активності (Activity Diagram).

Простота діаграми прецедентів дозволяє аналітикам легко спілкуватися з замовниками у процесі визначення вимог, виявляти обмеження, що накладаються на систему і на виконання окремих вимог, таких, наприклад, як час реакції системи, які в подальшому потрапляють в розділ нефункціональних вимог.

Також діаграма прецедентів може використовуватися для створення сценаріїв тестування, оскільки всі взаємодії користувача і системи вже визначені.

***Аналіз***

Після визначення вимог і контексту, в якому буде працювати система, настає черга аналізу отриманих даних. У процесі аналізу складається аналітична модель, яка підводить розробників до архітектури майбутньої системи. Аналітична модель – це погляд на систему зсередини, на відміну від моделі прецедентів, яка показує, як система буде виглядати зовні.

Ця модель дозволяє зрозуміти, як система повинна бути спроектована, які в ній повинні бути класи і як вони повинні взаємодіяти між собою. Основне її призначення - визначити напрям реалізації функціональності, виявленої на етапі збору вимог і зробити начерк архітектури системи.

На відміну від створюваної надалі моделі проектування, модель аналізу є більшою мірою концептуальною моделлю і тільки наближає розробників до класів реалізації. Ця модель не повинна мати можливих суперечностей, які можуть зустрітися в моделі прецедентів.

Для відображення моделі аналізу за допомогою UML використовується діаграма класів зі стереотипами (зразками поведінки) «граничний клас», «сутність», «управління», а для деталізації використовуються діаграми співробітництва (Collaboration). Стереотип «граничний клас» відображає клас, який взаємодіє із зовнішніми актантами, «сутність» – відображає класи, які є сховищами даних, а керування – класи, керуючі запитами до сутностей.

***Проектування***

Наступним етапом у процесі створення системи буде проектування, в ході якого на основі моделей, створених раніше, створюється модель проектування. Ця модель відображає фізичну реалізацію системи і описує створюваний продукт на рівні класів і компонентів. На відміну від моделі аналізу, модель проектування має явно виражену залежність від умов реалізації, застосовуваних мов програмування і компонентів.

Для створення моделі проектування використовуються цілий набір UML діаграм: діаграми класів, діаграми кооперації, діаграми взаємодії, діаграми активності.

Додатково у цьому робочому процесі може створюватися модель розгортання, яка реалізується на основі діаграми розгортання (Deployment Diagram). Це найпростіший тип діаграм призначений для моделювання розподілу пристроїв у мережі. Для відображення використовується всього два варіанти значків процесор і пристрій разом зі зв'язками між ними.

***Реалізація***

Основне завдання процесу реалізації – створення системи у вигляді компонентів – вихідних текстів програм, сценаріїв, двійкових файлів, виконуваних модулів і т. д. На цьому етапі створюється модель реалізації, яка описує те, як реалізуються елементи моделі проектування, які класи будуть включені в конкретні компоненти. Дана модель описує спосіб організації цих компонентів у відповідності з механізмами структурування і розбиття на модулі, прийнятими у вибраній середовищі програмування і представляється діаграми компонентів.

***Тестування***

В процесі тестування перевіряються результати реалізації. Для даного процесу створюється модель тестування, яка складається з тестових прикладів, процедур тестування, тестових компонентів, однак не має відображення на UML діаграми, але діаграми можуть використовуватися для підготовки процесу тестування.

Основні процеси методології RUP також розглядалися на лекції. Методологія досить велика, і містить рекомендації по веденню різних програмних проектів, від створення програм групою розробників в кілька людей, до розподілених програмних проектів, що об'єднують тисячі людей на різних континентах. Однак, незважаючи їх колосальну різницю, методи застосування моделей, створюваних за допомогою UML будуть одні і ті ж. Діаграми UML, що створюються на різних етапах розробки, невіддільні від інших артефактів програмного проекту і часто є сполучною ланкою між окремими процесами RUP.

Рішення про застосування конкретних діаграм залежить від поставленого в компанії процесу розробки, який, хоча і називається уніфікованим, але є динамічним, може бути поліпшеним і допрацьованим. Застосування UML разом з уніфікованим процесом дозволить отримати очікуваний результат, укластися у відведений бюджет, підвищити віддачу від учасників проекту і якість створюваного програмного продукту.