**Практична робота №2 Заняття 2. Проектування програмного забезпечення за еволюційною моделлю**

**Тема:** Проектування програмного забезпечення за еволюційною моделлю

**Мета:** Навчитись проектувати гіпотетичне програмне забезпечення, спираючись на принципи, що закладені в спіральній моделі.

**КОНТРОЛЬНІ ПИТАННЯ:**

1. Коли вперше виникла концепція спіральної моделі?
2. Назвіть основні переваги спіральної моделі?
3. Назвіть основні недоліки спіральної моделі?
4. Яка область використання спіральної моделі?

**Завдання:** Ознайомтесь з основними принципами спіральної моделі, за темою та сформуйте детальний перелік робіт щодо розробки програмного забезпечення згідно етапів спіральної моделі для власного проекту. Кожний етап повинен бути супроводжений документацією (визначити перелік документів, необхідний для кожного етапу).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ітерація | Етап  (проектування, кодування, тестування) | Перелік робіт | Результат | Документ |
|  |  |  |  |  |

**Методичні вказівки до виконання роботи:**

1. Робота повинна бути виконана згідно критеріїв оформлення документації на аркушах формату А4.
2. Студентам перед виконання Практичної роботи потрібно ознайомитись із необхідним теоретичним матеріалом, потім обрати тему. В якості теми можна взяти тему попередніх Практичних робіт (власний проект) або змінити власну тему, узгодивши її з викладачем.
3. Спочатку рекомендується проговорити можливі варіанти виконання.
4. Якщо потрібні пояснення – можна звернутись до викладача. Викладач сам вирішує відповідати на питання чи надати підказки, де можна знайти відповідь.
5. Протягом виконання Практичної роботи студенти можуть використовувати конспекти або інші джерела інформації.
6. Роботу потрібно набирати на комп’ютері, що розташований в аудиторії або на власному ноутбукові, назвавши документ «Практична робота № ».
7. По закінченню Практичну роботу потрібно здати на перевірку викладачеві, роздрукувавши її або надіславши електронною поштою. Якщо викладач знаходить помилки чи неточності, він може повернути роботу на доопрацювання.

Оформлену відповідно до встановлених вимог роботу 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:

**KPZ <Номер групи><Номер лекції / Практичної> [літера позначення типу роботи L – лекція, R – Практична]<Прізвище англійською>**.. Наприклад, **KPZ4101R**buts.doc.

Тему в заголовку листа записати

**KPZ<Номер групи>-ЛР<Номер Практичної>-<Прізвище >**

**Строк виконання цієї роботи ІПЗ-41 –**

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

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

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

Найбільш відомим і поширеним варіантом еволюційної моделі є спіральна модель, що стала вже фактично самостійної моделлю, що має різні сценарії розвитку і деталізації.

Спіральна модель (рис. 1) була вперше сформульована Баррі Боемом в 1988 р. Відмінною особливістю цієї моделі є спеціальна увага ризикам, що впливає на організацію життєвого циклу. Головне досягнення спіральної моделі полягає в тому, що вона пропонує спектр можливостей адаптації вдалих аспектів існуючих моделей процесів життєвого циклу. Дана модель життєвого циклу характерна при розробці новаторських (нетипових) систем. На початку роботи над проектом у замовника і розробника немає чіткого бачення підсумкового продукту (вимоги не можуть бути чітко визначені) або стовідсоткової впевненості в успішній реалізації проекту (ризики дуже великі). В зв’язку з цим приймається рішення розробки системи по частинах з можливістю зміни вимог або відмови від її подальшого розвитку. Як видно з рис. 1, розвиток проекту може бути завершено не тільки після стадії впровадження, але і після стадії аналізу ризику.

***Переваги моделі***:

* Дозволяє швидше показати користувачам системи працездатний продукт, тим самим, активізуючи процес уточнення і доповнення вимог;
* Допускає зміну вимог при розробці інформаційної системи, що характерно для більшості розробок, у тому числі і типових;
* Забезпечує більшу гнучкість в управлінні проектом;
* Дозволяє отримати більш надійну і стійку систему. По мірі розвитку системи помилки і слабкі місця виявляються і виправляються на кожній ітерації;
* Дозволяє удосконалювати процес розробки – аналіз, проведений в кожній ітерації, дозволяє проводити оцінку того, що має бути змінено в організації розробки, і поліпшити її на наступній ітерації;
* Зменшуються ризики замовника. Замовник може з мінімальними для себе фінансовими втратами завершити розвиток неперспективного проекту.



Рисунок 1 – Оригінальна спіральна модель життєвого циклу (далі–ЖЦ), розроблена Боемом

***Недоліки моделі:***

* Збільшується невизначеність у розробника в перспективах розвитку проекту. Цей недолік випливає з попереднього достоїнства моделі;
* Ускладнені операції тимчасового і ресурсного планування всього проекту в цілому. Для вирішення цієї проблеми необхідно ввести тимчасові обмеження на кожну із стадій життєвого циклу. Перехід здійснюється відповідно до плану, навіть якщо не вся запланована робота виконана. План складається на основі статистичних даних, отриманих у попередніх проектах та особистого досвіду розробників.



Рисунок 2 - Модель ЖЦ еволюційного прототипування

Згідно Джону Коннеллу (Connell) і Лінде Шафер (Shafer), еволюційним прискореним прототипом є «дуже піддатлива модифікації та розширенню робоча модель передбачуваної системи, не обов’язково представляє собою всі властивості системи, завдяки якій користувачі даного додатка отримують фізичне уявлення про ключові частинах системи до її безпосередньої реалізації; це – легко створювана, без праці піддатлива модифікації, максимально розширювана, частково задана робоча модель основних аспектів передбачуваної системи «.

Бернард Боаріо (Bernard Boar) визначив прототип як «метод, призначений для визначення вимог, при якому потреби користувача витягуються, представляються і розробляються за допомогою побудови робочої моделі кінцевої системи – швидко і в необхідному контексті».

Стало класикою твір Фреда Брукса (Fred Brook) під назвою «Легендарний людина-місяць» (The Mythical Man-Month) сьогодні настільки ж актуально, як і в 1975 році. Технології радикально змінили світ, але багато недоліків менеджменту програмних проектів як і раніше ті ж. Десятки років тому Брукс сказав: «У більшості проектів перша побудована система навряд чи придатна до вживання. Вона може бути занадто повільної, дуже об’ємною, незручною у використанні або володіти всіма трьома перерахованими недоліками. Немає іншого вибору, окрім як почати з самого початку, приклавши всі зусилля, і побудувати модернізовану версію, в якій вирішувалися б всі три проблеми …

У разі, коли в проекті використовується нова системна концепція чи нова технологія, розробник змушений побудувати систему, якою згодом не скористовуються, оскільки навіть при найкращому плануванні неможливо передбачати досягнення потрібного результату.

Отже, питання менеджменту полягає не в тому, створювати чи ні експериментальну систему, якою потім не скористаються. Ви в будь-якому випадку так і зробите. Єдине питання в тому, чи потрібно планувати створення продукту одноразового використання заздалегідь або обіцяти поставити його замовникам … «

Саме ця концепція побудови експериментальної, або прототипну системи призвела до виникнення «структурної», «еволюційної» моделі швидкого прототипування (RAD), і спіральної моделі. У своїй більш пізньої, в рівній мірі повної плідних ідей роботі під назвою «No Silver Bullet, the Essence and Accidents of Programming» Брукс вважає, що більшість помилок, що виникають при розробці ПЗ, все ж пов’язані з неправильним розумінням концепції системи, а не з синтаксисом або логікою. Розробка ПЗ завжди буде важким завданням, і ми ніколи не знайдемо чудодійну панацею або «срібну кулю». Він підкреслює позитивний момент у застосуванні методів швидкого прототипування: «Найважчою складовою процесу побудови програмної системи є прийняття однозначного рішення про те, що саме необхідно побудувати. Жодна з інших складових роботи над концепцією не представляє собою таку трудність, як визначення детальних технічних вимог, включаючи всі аспекти зіткнення продукту з людьми, машинами та іншими програмними системами. Жодна інша складова роботи в такому ступені не завдає шкоди отриманої в результаті системі, якщо вона виконана неправильно. Саме цю складову процесу розробки найважче виправити на більш пізніх етапах.

Отже, найважливіша функція, яку виконує розробник клієнтських програм, полягає в ітеративном витяганні і уточненні вимог до продукту. Адже насправді клієнт не має уявлення про те, що саме він хоче отримати.

На даний момент часу є однією з найбільш обіцяних серед технологічних спроб, зосереджених на сутності, а не на труднощах вирішення проблеми розробки ПЗ, є розробка методів і засобів для прискореного прототипування систем як складової итеративной специфікації вимог «.

Уотт Хемфрі (Watts Humphrey), який відомий як натхненник створення моделі СММ, розробленої Інститутом SEI, підтримує Брукса в його підході до важливості вимог і їх розробки: «У більшості систем укладений основний принцип, який включає в себе більше, ніж незначне еволюційна зміна. Система може змінити саме експлуатаційне оточення. Оскільки користувачі можуть міркувати про те чи іншому явищі тільки в рамках відомого їм оточення, вимоги до таких систем завжди формулюються в рамках поточного оточення. Отже, ці вимоги неодмінно будуть неповними, неточними і оманливими. Головним завданням для системного розробника є винахід процесу розробки, за допомогою якого можна буде виявити, визначити і розробити реальні вимоги. Цього можна досягти тільки при максимальному включенні користувача в процес розробки і часто з допомогою періодичного тестування прототипів або версій, отриманих на ранніх етапах розробки. Виявляється, що такі процеси завжди займають більше часу, але незмінно наприкінці призводять до розробки кращої системи набагато швидше, ніж при використанні будь-якої іншої стратегії «.

***Переваги структурної еволюційної моделі швидкого прототипування***

При використанні структурної еволюційної моделі швидкого прототипування для прийнятного проекту проявляються такі переваги:

• кінцевий користувач може «побачити» системні вимоги в процесі їх збору командою розробників; таким чином, взаємодія замовника з системою починається на ранньому етапі розробки;

• виходячи з реакції замовників на демонстрації продукту, що розробляється, розробники отримують відомості про один або декількох аспектах поведінки системи, завдяки чому зводиться до мінімуму кількість неточностей у вимогах;

• знижується можливість виникнення плутанини, спотворення інформації або непорозумінь при визначенні системних вимог, що безсумнівно призводить до створення більш якісного кінцевого продукту;

• в процес розробки можна внести нові або несподівані вимоги користувача, що часом необхідно, так як реальність може відрізнятися від концептуальної моделі реальності;

• модель являє собою формальну специфікацію, втілену в робочу модель;

• модель дозволяє виконувати гнучке проектування і розробку, включаючи декілька ітерацій на всіх фазах життєвого циклу;

• при використанні моделі утворюються постійні, видимі ознаки прогресу у виконанні проекту, завдяки чому замовники відчувають себе впевнено;

• можливість виникнення розбіжностей при спілкуванні замовників з розробниками мінімізована;

• очікувану якість продукту визначається за активної участі користувача в процес на ранніх фазах розробки;

• можливість спостерігати ту чи іншу функцію в дії пробуджує очевидну необхідність у розробці функціональних додаткових можливостей;

• завдяки меншому об’єму доопрацювань зменшуються витрати на розробку;

• завдяки тому що проблема виявляється до залучення додаткових ресурсів скорочуються загальні витрати;

• забезпечується управління ризиками;

• документація сконцентрована на кінцевому продукті, а не на його розробці;

• беручи участь у процесі розробки протягом усього життєвого циклу, користувачі більшою мірою будуть задоволені отриманими результатами.