МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ

КАФЕДРА РАДІОФІЗИКИ ТА

КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА ПЕДАГОГІКИ

ПЛАН – КОНСПЕКТ

лекції з курсу:
"Технології створення програмних продуктів"

на тему:
"Життєвий цикл і процеси розробки програмного забезпечення",
проведеної для курсу ФеІ-3
19 лютого 2015 року

Склала

студентка групи ФЕІ-51м

Литвин В. В.

Перевірили:

доц. Злобін Г. Г.
доц. Заячук Ю. Д.

Львів 2015

1. Мета:

навчальна (формування у студентів наукових знань про найпоширеніші моделі життєвого циклу програмних продуктів та основи програмних вимог; критичного ставлення до проблем проектування та конструювання програмного забезпечення; ціннісних уявлень про якість, тестування та супровід програмного забезпечення);

розвивальна (розвивати логічне мислення та вміння вирішувати системні завдання на абстрактних моделях з подальшою реалізацією цих моделей на практиці; удосконалювати вміння аналізувати проблему на рівні архітектури і покращувати спроектовану модель, використовуючи найефективнішу модель життєвого циклу у відповідності до особливостей системи, що проектується);

виховна (виховувати правила поведінки у колективі; сприяти взаємодії між студентами).

2. Методи, прийоми, засоби:

розповідь, діалог, порівняння, аналогія, наведення доказів, дискусія, побудова та аналіз графіків, кейс-метод, узагальнення, міркування, повідомлення.

3. Наочність:

схеми на дошці, графіки, презентація на проекторі.

4. Основні питання лекції:

1. Моделі життєвого циклу для створення програмних продуктів.
2. Основи програмних вимог.
3. Проектування програмного забезпечення.

5. Рекомендована література:

Основна:

1. Бабенко Л.П., Лавріщева К.М. Основи програмної інженерії. — К.:Знання, 2001. — 269 с.

2. S.Guckenheimer, J.Perez, Software Engineering with Microsoft Visual Studio Team System, Addison-Wesley Professional, 2006.

3. В.В. Липаев, Программная инженерия.Методологические основы, ТЕИС, 2006.

4. SWEBOK 2004 /http://www.computer.org/portal/web/swebok/htmlformat

5. Державний стандарт України. Основні напрямки оцінювання та відбору CASE-інструментів. ДСТУ 3919–1999. 2000 р.

Додаткова:

1. Трофимов С.А. CASE-технологии. Практическая работа в Rational Rose. — М.:Бином, 2002. — 284 с.

2. Ф. Брукс Мифический человеко-месяц или как создаются программные системы, Символ-Плюс, 2006.

3. Крачтен Ф. Введение в Rational Unified Process. - М.: Издат. Дом “Вильямс”, 2002. - 240 с.

6. Основні питання, що розглядаються під час лекції:

Хід лекції:

1. Вступна частина.

(Привітання).

На попередній лекції було розглянуто такі ключові поняття технології створення програмних продуктів, як “програма”, “програмний засіб(ПЗ)”, “правильність” та “надійність ПЗ”. Також було проаналізовано процес обробки даних. Сьогодні ми детальніше розглянемо процес проектування програмного забезпечення та основи програмних вимог. Також ми проаналізуємо основні моделі життєвого циклу, які використовуються для створення програмних продуктів.

(Повідомлення теми лекції).

Знання міжнародних та українських стандартів зі створення програмних продуктів є дуже важливими для вміння правильно вибирати модель життєвого циклу власної системи, а такі вміння є необхідними для висококласних спеціалістів. Крім того сьогоднішня лекція є важливою для розуміння всіх етапів

розробки програмних продуктів, які ми будемо розглядати на наступних заняттях.

2. Основна частина :

1. Моделі життєвого циклу для створення програмних продуктів.

Модель життєвого циклу (ЖЦ) - це схема виконання робіт і задач в рамках процесів, що забезпечують розробку, експлуатацію і супровід програмного продукту, та відображає життя програмної системи, починаючи від формулювання вимог до неї до припинення її використання.

(Студентам дається час занотувати визначення.)

Історично ця схема робіт містить:

- розробку вимог або технічного завдання;
- розробку системи або технічного проекту;
- програмування або робоче проектування;
- пробну експлуатацію;
- супровід і поліпшення;
- зняття з експлуатації.

Сьогодні основою формування нової моделі ЖЦ для конкретної прикладної системи є стандарт ISO/IEC 12207, який задає повний набір процесів (більше 40).

(Студентам демонструють схему основних процесів життєвого циклу програмних систем та схему допоміжних процесів життєвого циклу програмних систем.

До кожного процесу ЖЦ з обох схем наводиться його коротка характеристика та дається оцінка його ролі в ЖЦ та впливу на результат.)

Моделі життєвого циклу для розробки програмних систем:

В лекції розглядається чотири основних моделі:

1. Каскадна модель
2. Інкрементна модель
3. Спіральна модель
4. Еволюційна модель

(Для кожної з них подається розширена характеристика. Моделі поступово зображаються на дошці, в процесі пояснення кожного процесу він домальовується до загальної схеми. Коли всі процеси нанесені на схему проводиться аналіз її переваг та недоліків, а також наводяться приклади типового використання цієї моделі.

Студенти приймають активну участь в аналізі моделі та переваг її використання для ситуації запропонованої викладачем. Також розглядаються приклади, запропоновані студентами, по два для кожної моделі.)

1) Каскадна модель (waterflow model)

Особливість такої моделі полягає у фіксації послідовних процесів розроблення програмного продукту. В її основу покладена модель фабрики, де продукт проходить стадії від задуму до виробництва, потім його передають замовнику у вигляді готового виробу, де заміна не передбачена, хоча можна подати інший подібний виріб.

Недоліки цієї моделі:

– процес створення ПС не завжди вкладається в таку жорстку форму і послідовність дій;

– не враховуються змінювані потреби користувачів, нестабільні умови зовнішнього середовища, які впливають на зміни вимог до ПС під час її розроблення;

– значний розрив між часом внесення помилки, що призводить до суттєвої переробки ПС.

Переваги реалізації системи за допомогою каскадної моделі такі:

– всі завдання підсистем і системи реалізуються одночасно, завдяки чому не можна забути жодного завдання;

– повністю розроблену систему з документацією на неї легше

супроводжувати, тестувати, фіксувати помилки і вносити зміни не хаотично, а цілеспрямовано, починаючи з вимог.

Каскадну модель можна розглядати як модель ЖЦ, придатну для створення першої версії ПЗ з метою перевірки реалізованих в ній функцій. При супроводі і експлуатації можуть бути виявлені різноманітні помилки, виправлення яких спричинить повторне виконання всіх процесів, починаючи з уточнення вимог.

2) Інкрементна модель (Iterative and incremental development)

При застосуванні даної моделі необхідно враховувати наступні чинники ризику:

- вимоги, складені незрозуміло для реалізації;
- всі можливості системи потрібно реалізувати із самого початку;
- швидко міняються технології і вимоги до системи;
- обмеження в ресурсному забезпеченні можуть привести до затягування термінів здачі системи в експлуатацію.

Таку модель розробки доцільно використати, у випадках коли:

- бажано реалізувати деякі можливості системи швидко за рахунок створення проміжного продукту;
- система розділена на окремі складові частини структури, які можна уявляти як деякий проміжний продукт;
- можливе збільшення фінансування на розробку окремих частин системи.

3) Спіральна модель

Розробка ітераціями відображає об'єктивно існуючий спіральний цикл створення системи. Неповне завершення робіт на кожному етапі дозволяє переходити на наступний етап, не чекаючи повного завершення роботи на поточному. При ітеративному способі розробки відсутню роботу можна буде виконати на наступній ітерації. Головне ж завдання - щонайшвидше показати користувачам системи працездатний продукт, тим самим активізуючи процес уточнення і доповнення вимог.

Внесення змін орієнтоване на задоволення потреби користувачів одразу, як тільки буде встановлено, що створені артефакти або елементи документації не відповідають дійсному стану розробки.

Дана модель ЖЦ допускає аналіз продукту на витку розробки, його перевірку, оцінку правильності та прийняття рішення про перехід на наступний виток або повернення на попередній виток для доопрацювання на ньому проміжного продукту.

Відмінність цієї моделі від каскадної полягає в можливості багато разів повертатися до процесу формулювання вимог і до повторної розробки версії системи з будь-якого процесу моделі.

Для програмного продукту така модель не дуже підходить з декількох причин. По-перше, висловлення вимог замовником носить суб'єктивний характер, вимоги можуть багаторазово уточнюватися протягом розробки ПС і навіть після завершення та випробовування, і часом може з'ясуватися, що замовник «хотів зовсім інше». По-друге, змінюються обставини та умови використання системи, тому загальновизнаним законом програмної інженерії є закон еволюції, який сформулюємо так: кожна діюча ПС з часом потребує внесення змін або виводиться з експлуатації.

При необхідності внесення змін до системи на кожному витку з метою отримання нової версії системи обов'язково вносяться зміни в заздалегідь зафіксовані вимоги, після чого повертаються на попередній виток спіралі для продовження реалізації нової версії системи з урахуванням усіх змін.

4) Еволюційна модель (RAD- Rapid Application Development)

У разі еволюційної моделі система послідовно розробляється з блоків конструкцій. На відміну від інкрементної моделі в еволюційній моделі вимоги встановлюються частково і уточнюються в кожному наступному проміжному блоці структури системи.

Використання еволюційної моделі припускає проведення дослідження предметної області для вивчення потреб її замовника і аналізу можливості застосування цієї моделі для реалізації. Модель використовується для розробки нескладних і некритичних систем, де головною вимогою є реалізація функцій системи. При цьому вимоги не можуть бути визначені відразу і повністю. Тому розробка системи здійснюється ітераційним шляхом її еволюційного розвитку з отриманням деякого варіанта системи–прототипу, на якому перевіряється реалізація вимог.

Розвитком цієї моделі є модель еволюційного прототипування в рамках усього ЖЦ розробки ПС.

При цьому підході враховуються такі чинники ризику:

- реалізація всіх функцій системи одночасно може привести до громіздкості;
- обмежені ресурси зайняті розробкою протягом тривалого часу.

Переваги застосування даної моделі ЖЦ:

- швидка реалізація деяких функціональних можливостей системи;
- у системі виділяються окремі частини для реалізації їх у вигляді прототипу;
- можливість збільшення фінансування системи;
- можливість зворотного зв’язку із замовником для уточнення вимог;
- спрощення внесення змін у зв’язку із заміною окремої функції.

Модель розвивається у напряму додавання нефункціональних вимог до системи, пов'язаних із захистом і безпекою даних, несанкціонованим доступом до них та ін.

2. Основи програмних вимог.

Вимоги (Requirements) - це властивості, якими має володіти ПЗ, щоб мати певну цінність для користувачів.

Вимоги віддзеркалюють потреби людей (замовників, користувачів, розробників), зацікавлених у створенні ПЗ. Замовник і розробник спільно проводять збір вимог, їх аналіз, переогляд, визначення необхідних обмежень і

документування.

Розрізняють вимоги до продукту і до процесу, а також функціональні та нефункціональні вимоги, системні вимоги.

У підрозділі розглядається теоретичний і інтелектуальний базис проектування – методи, принципи, засоби і методології, представлені областями в ядрі знань програмної інженерії SWEBOK.

Ядро знань SWEBOK – основний науково-технічний документ, що відображає знання та досвід багатьох іноземних і вітчизняних фахівців з програмної інженерії і узгоджується з регламентованими процесами ЖЦ стандарту ISO/IEC 12207.

Документ містить у собі опис 10 областей, кожна з яких представлена відповідно до прийнятої всіма учасниками формування ядра SWEBOK загальної схеми опису, що містить у собі визначення понятійного апарату, методів і засобів, а також інструментів підтримки інженерної діяльності. Стосовно кожної області визначено коло знань, які повинні практично використовуватися при виконанні процесів життєвого циклу.

(Студентам демонструють дві схеми: п'ять областей для розроблення ПС (головні) та п'ять областей, що забезпечують інженерію керування розробкою ПС (допоміжні організаційні області);

подається огляд кожної області, визначається її роль у проектуванні і реалізації програмних продуктів. У деяких підрозділах показаний зв'язок з положеннями відповідних стандартів, що регламентують і регулюють виконання процесів проектування програмних систем.)

Інженерія вимог до ПЗ - це процес аналізу та документування вимог до ПЗ, який полягає у перетворюванні запропонованих замовником вимог до системи в опис вимог до ПЗ, їх специфікація та верифікація.

Виявлення вимог - це процес добування інформації з різних джерел

замовника (договорів, матеріалів аналітиків щодо завдань і функцій системи тощо), проведення технічних заходів (співбесід, зборів та ін.) для формування ділових вимог на розробку. Вимоги узгоджуються з замовником і виконавцем.

Аналіз вимог - процес вивчення потреб і цілей користувачів, класифікація та перетворення їх до вимог системи, апаратури та програмного забезпечення, встановлення та вирішення конфліктів між вимогами, визначення пріоритетів, меж системи і принципів взаємодії із середовищем функціонування.

(Студентам дається час занотувати основні визначення.)

3. Проектування програмного забезпечення

Проектування ПЗ (Software design) - процес визначення архітектури, компонентів, інтерфейсів, інших характеристик системи і кінцевого результату.

Базова концепція проектування ПЗ - це методологія проектування архітектури за допомогою різних методів (об'єктного, компонентного та ін.), процесів ЖЦ (стандарт ISO/IEC12207) і технік: декомпозиція, абстракція, інкапсуляція тощо.

При проектуванні структури ПЗ використовується архітектурний стиль проектування, заснований на визначенні основних елементів структури - підсистем, компонентів і зв'язків між ними.

Шаблон (патерн) - це типовий конструктивний елемент ПЗ, який задає взаємодію об'єктів (компонентів) проектованої системи, визначення ролей та відповідальності виконавців. Основною мовою завдання цього елемента є UML.

Шаблон може бути:

- структурним, в якому визначаються типові композиції структур з об'єктів і класів діаграм класів, об'єктів, зв'язків та ін.;
- поведінковим, що визначає схеми взаємодії класів об'єктів і їх поведінка діаграмами активностей, взаємодії, потоків управління та ін.;
- креативним, що відображають типові схеми розподілу ролей примірників об'єктів діаграмами взаємодії, кооперації та ін.

(Наводяться приклади шаблонів проектування, відомі студентам з попередніх курсів, зокрема з курсів "Основи програмування та алгоритмічні мови", "Об'єктно-орієнтоване програмування", "Організація баз даних і знань".)

Результат проектування - архітектура та інфраструктура, що містить набір об'єктів, з яких можна формувати деякий конкретний вигляд архітектурної схеми для конкретного середовища виконання системи.

Логічна структура проектованої системи - це композиція об'єктів і готових програмних продуктів, що виконують відповідні функції системи. Композиція ґрунтується на наступних положеннях:

- кожна підсистема повинна відображати вимоги та спосіб їх реалізації;
- змінні функції виділяться в підсистеми так, щоб для них прогнозувалися зміни вимог та окремі об'єкти, пов'язані з актором;
- зв'язок об'єктів здійснюється через інтерфейс;
- кожна підсистема повинна виконувати мінімум послуг або функцій і мати фіксовану множину параметрів інтерфейсу.

(Студентам дається час занотувати основні визначення.)

3. Заключна частина.

Отож сьогодні ми детально розглянули поняття ЖЦ ПЗ, його основні процеси та чотири моделі ЖЦ. Також ми ознайомились із основами та стандартами формування вимог до програмного забезпечення. На наступній лекції ми продовжимо розглядати процеси розробки ПЗ, зокрема поговоримо про методології моделювання та проектування великих програмних систем, які базуються на моделях ЖЦ ПЗ.

(Студентам даються рекомендації стосовно поглибленого вивчення теми, перераховуються джерела для додаткової інформації по темі і оголошуються питання, які виносяться на самостійне опрацювання.

Оскільки тема є дуже обширною і водночас надзвичайно актуальною для майбутніх розробників ПЗ та для працівників ІТ сфери загалом і такою, що

стрімко розвивається, студентам також дуже коротко повідомляється про тенденції в розвитку моделей проектування ПЗ та джерела, де можна почерпнути більше інформації про них.

Студентам також нагадують тему наступного заняття і дають завдання до лабораторної роботи.)

(Подяка за увагу)