ЛЕКЦІЯ 1-1. Архітектура програмного забезпечення

1. Поняття архітектури ПЗ. Цілі архітектури. Принципи проектування архітектури.
2. Проектування архітектури. Компоненти. Складання плану реалізації моделі предметної області програмного забезпечення. Варіанти складання моделей проектів.
3. Класифікація архітектури.
4. Типи архітектури і їх моделі. Архітектурні шаблони і стилі. Поєднання архітектурних стилів. Компонентна архітектура.
5. Багатошарова архітектура. N-рівнева/3-рівнева архітектура. Сервісно-орієнтована архітектура.
6. Розробка і оцінка архітектури на основі сценаріїв.
7. Статичні та динамічні діаграми при проектуванні архітектури ПЗ.

(жовтим маркером на самостійну роботу)

1. Поняття архітектури ПЗ. Цілі архітектури. Принципи проектування архітектури.

"Архітектура" і "Інженерія", як види людської діяльності, існували задовго до появи комп'ютерних технологій. Насамперед, ці види діяльності пов'язував процес створення проекту - прототипу, прообразу передбачуваного або можливого об'єкта. Іншими словами проектування містить у своєму складі поняття "архітектура" і "Інженерія", а проектування програмного забезпечення деяким відрізняється в цьому сенсі від проектування, наприклад, будівель і споруд. Тенденції розвитку будівельної архітектури останніх *десятиліть пов'язані з максимальною функціональністю проектованих об'єктів*. Архітектурне проектування ПЗ також переслідує аналогічну мету.

Згідно енциклопедії «Вікіпедія», ***архітектура програмного забезпечення***– це представлення системи програмного забезпечення, що дає інформацію про компоненти складають систему, про взаємозв'язки між цими компонентами і правилах, що регламентують ці взаємозв'язки, яке призначене для ефективної розробки проекту такої системи.

Проектування програмного забезпечення, у свою чергу, передбачає вироблення властивостей системи на основі аналізу постановки задачі (моделей предметної області (Domain Design) і вимог до ПЗ), а також досвіду проектувальника.

Автори книги "*Порождающее программирование: методы, инструменты, применение*" К. Чарнецький і У. Айзенекер [1] визначають ***проектування архітектури ПЗ*** як "високорівневе проектування, метою якого є створення гнучкої структури, що задовольняє всім основним вимогам і передбачає деяку ступінь свободи реалізації.

Як правило, з тих деталей, які менш інших схильні до змін, формується «скелет». При цьому всі інші деталі робляться якомога більш гнучкими, з тим, щоб

згодом їх можна було без праці оновити. Втім, зміни іноді вносяться навіть у скелет".

Слайд 6.

Архітектура **-** це базова організація системи, втілена в її компонентах, їхніх відносинах між собою і з оточенням, а також принципи, що визначають проектування та розвиток системи.

Архітектура - це набір значимих рішень з приводу організації системи програмного забезпечення, набір структурних елементів і їх інтерфейсів, за допомогою яких компонується система, разом з їх поведінкою, обумовленим у взаємодії між цими елементами, компонування елементів в поступово укрупняющися підсистеми, а також стиль архітектури який направляє цю організацію - елементи та їхні інтерфейси, взаємодії та компоновку.

Архітектура програми або комп'ютерної системи - це структура або структури системи, які включають елементи програми, видимі ззовні властивості цих елементів і зв'язки між ними.

Слайд 7.

Архітектура ПЗ – це артефакт, що представляє собою результат процесу розробки програмного забезпечення. Елементи архітектури ПЗ і моделі їх з'єднання призначені для задоволення вимог до проектованих системам. У проекті архітектури ПЗ повинні бути враховані функціональні та нефункціональні вимоги до ефективності, витривалості, розширюваності, відмовостійкості, продуктивності, можливості повторного використання, а також адаптування розробляється ПО.

Архітектурний проект ПЗ, дозволяє оперативно визначити, наскільки даний програмний продукт відповідає пропонованим до нього вимогам.

Слайд 8.

Метою архітектурного проектування предметної області є наступні артефакти:

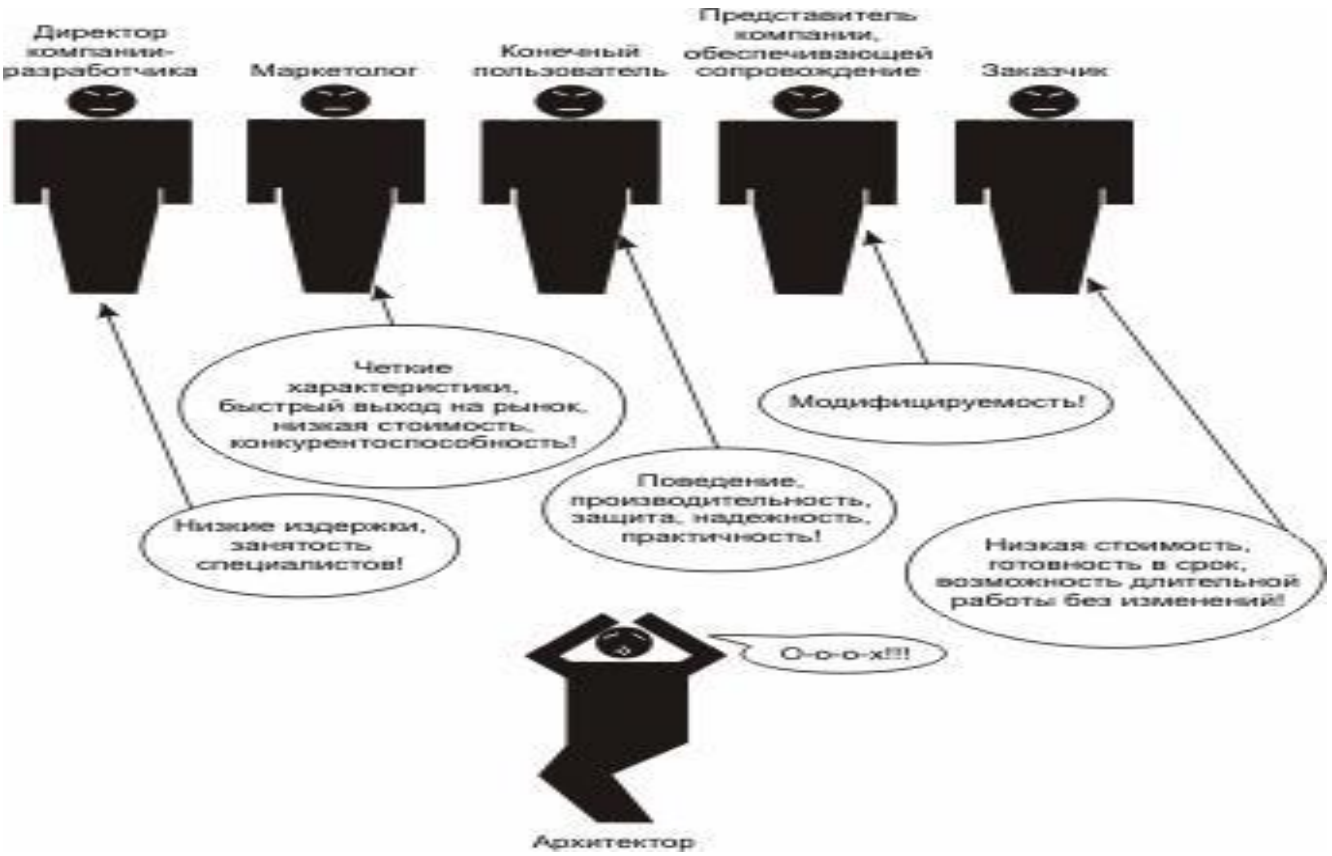
1. розробка архітектури множини (сімейства) систем, які входять до даної предметної області;
3. складання плану реалізації моделі предметної області;
4. реалізація моделі предметної області.

Слайд 9.

2. Проектування архітектури. Компоненти. Складання плану реалізації моделі предметної області програмного забезпечення. Варіанти складання моделей проектів.

2.1. ***Проектування архітектури***

Вплив на архітектуру ПЗ зацікавлених в системі осіб (на рис)



Слайд 10.

Вплив на архітектуру надає компанія-розробник. Компанії роблять прямі інвестиції в різні активи - зокрема в існуючі варіанти архітектури і засновані на них продукти. Кожен наступний проект у такому випадку мислиться як продовження ряду подібних систем, а в його кошторисі закладено активне повторне використання наявних коштів.

Дотримуючись своїм стратегічним завданням, компанії іноді роблять довгострокові інвестиції в інфраструктуру; в такому випадку передбачувана система мислиться як один із засобів фінансування та розширення цієї інфраструктури.

Певний вплив на програмну архітектуру надає організаційна структура компанії-розробника.

Слайд 11.

Вплив на архітектуру досвіду і навичок архітекторів.

Позитивний досвід - повторення його в наступних роботах.

Негативний досвід - відмова використовувати його в наступних роботах.

Архітектори люблять експериментувати з новими зразками (pattern) і методиками.

Слайд 12.

Вплив на архітектуру технічної бази

Підготовка та досвід архітектора проявляється, зокрема, в його роботі з технічною базою (technical environment).

До технічній базі можна віднести:

- методи роботи, прийняті в даній галузі
- прийоми програмної інженерії, поширені в професійному співтоваристві, в яке входить архітектор.

Слайд 13.

Варіативність факторів врливу на архитектуру

Слайд 14.

До складу архітектурного проекту ПЗ входять:

- опис елементів, з яких складається дана система;
- схеми взаємодій між цими елементами;
- документація зразків (patterns), на основі яких здійснюється їх компоновка;
- список і зміст обмежень (вимог), характерних для цих зразків.

У будівництві мовою опису проекту є архітектурно-будівельні креслення і об'ємні моделі, а також текстові описи об'єктів, що зводяться і технологій їх зведення.

Слайд 15.

Ілюстративними засобами вираження характеристик ПЗ, в архітектурному проекті використовуються різні нотації:

- блок-схеми (схеми алгоритмів);
- ER-діаграми, UML-діаграми;
- DFD-діаграми;
- макети.

Кожна підсистема ПЗ, що складається із сукупності її ***компонентів*** і взаємодій між ними, повинна бути детально описана у відповідній частині проект з використанням цих нотацій. Оскільки така підсистема може виступати в якості складового елементу більш масштабної системи, в архітектурному проекті ПЗ обов'язково міститься докладний опис укрупнених частин системи за допомогою цих же засобів опису проекту.

Слайд 16.

Щодо вжитого терміна ***"компонент"***, Питер Илес (старший разработчик архитектуры информационных технологий, IBM) в статье "Что такое архитектура программного обеспечения?" [2] пише, що "... велика частина визначень архітектури не визначає терміну" компонент ", і стандарт IEEE 1471 – не виняток, оскільки навмисно залишає це поняття невизначеним, щоб воно відповідало певній множині тлумачень, можливих для конкретної галузі.

Компонент - абстрактна одиниця інструкцій програмного забезпечення і внутрішніх станів, яка забезпечує трансформацію даних через свої інтерфейси.

Компонент може бути:

- логічним або фізичним,
- технологічнонезалежним або технологічно-зв'язаним;
- крупно-або дрібногранульованим ...

Проектування взагалі, а також проектування ПЗ, ***є прикладним видом діяльності***. Оскільки в будь-якому з варіантів, проектування - це мистецтво створення того, чого немає в природі, архітектор (проектувальник) ПЗ повинен оволодіти мистецтвом проектування та самовираження, що дозволяє учасникам і замовникам проекту "будувати" необхідне ПЗ і управляти цим "будівництвом" і експлуатацією подальшою еволюцією системи.

На лабораторну роботу №1

2.2. Складання плану реалізації моделі предметної області програмного забезпечення

Реалізація моделі предметної області являє собою архітектурне моделювання проектованого об'єкта. План реалізації, складений архітектором ПЗ, регламентує способи отримання конкретних систем із загальної архітектури і компонентів. *Архітектурна частина проекту складання* ***моделі*** *предметної області* містить описи:

- інтерфейсу замовника для запуску конкретних підсистем;
- процесів складання компонентів;
- обробки запитів на зміни і розробку;
- вимірювань, супроводу та оптимізації бізнес-процесів.

Дана частина проекту описує збірку розроблення об'єкта з імовірним використанням автоматизованих засобів для *складання моделі*. Рівень автоматизації збирання залежить від безлічі факторів і пов'язаний як з програмно-технічною оснащеністю проекту, так і з наявністю достатнього рівня застосовуваних *артефактів* предметної області (у тому числі із застосуванням повторного коду).

Загалом можливі наступні ***варіанти складання моделей проектів***:

1) ***Збірка додатків з компонентів*** проводиться вручну.

   До складу проекту також входять:
   - *опису архітектури і компонентів;*
   - *реалізації предметно-орієнтованих мов;*
   - *керівництва по розміщенню графічних користувацьких інтерфейсів.*

2) Для ***складання компонентів*** застосовуються різноманітні інструментальні засоби:
   - засоби пошуку та перегляду компонентів,
   - описи застосування генераторів для автоматизації певних аспектів розробки додатків.

   До складу проекту також входять:

- *опису архітектури і компонентів,*
- *реалізації предметно-орієнтованих мов,*
- *реалізації керівництва по розміщенню графічних користувацьких інтерфейсів,*
- *генераторів і інфраструктури, за допомогою якої проводиться пошук,*
- *класифікація, поширення компонентів. Документація формується автоматично;*

3) ***Автоматичне складання моделі*** об'єкта із застосуванням інструментальних засобів для замовника (коштів породжує програмування), за допомогою яких формується запит на необхідне ПЗ.

Виробництво програми може бути досягнуто одним запитом, якщо в ньому не втримується частин, створюваних традиційними методами розробки.

До складу проекту також входять:

- *опису архітектури і компонентів;*
- *реалізації предметно-орієнтованих мов;*
- *керівництва по розміщенню графічних користувацьких інтерфейсів; генераторів і інфраструктури, за допомогою якої проводиться пошук;*
- *класифікація, поширення компонентів і тому подібні операції;*
- *організації процесів виробництва додатків.*

Вся документація також формується автоматично.

2.3. Реалізація моделі предметної області ПЗ

На даному етапі проводиться реалізація архітектури,компонентів і плану реалізації за допомогою методик, що містяться впроекті.

Слайд 17

3. Класифікація архітектури.

Вибір архітектури визначає спосіб реалізації вимог на високому рівні абстракції. Саме архітектура майже повністю визначає такі характеристики ПЗ як надійність, переносимість і зручність супроводу.

Архітектура значно впливає і на зручність використання і ефективність ПЗ, які визначаються також і реалізацією окремих компонентів. Значно менше вплив архітектури на функціональність – зазвичай для реалізації заданої функціональності можна використовувати *різні архітектури*.

Тому вибір між тією або іншою архітектурою визначається насамперед саме нефункціональними вимогами і необхідними властивостями ПЗ в аспектах зручності супроводу та переносимості.

При цьому **для побудови гарної архітектури** треба *враховувати можливі протиріччя між вимогами до різних характеристик і вміти вибирати компромісні рішення*, що дають прийнятні значення за всіма показниками.

Слайд 18

По-перше, для підвищення ефективності в загальному випадку вигідніше використовувати монолітні архітектури, в яких виділено невелике число компонентів (або єдиний компонент) – цим забезпечується економія як пам'яті, оскільки кожен компонент зазвичай має свої дані, а тут число компонентів мінімально, так і часу роботи, оскільки можливість оптимізувати роботу алгоритмів обробки даних є також, зазвичай, тільки в рамках одного компонента.

Слайд 19

З другого боку, для підвищення зручності супроводу, навпаки, краще розбивати систему на велике число окремих компонентів, з тим, щоб кожен з них вирішував свою невелику, але чітко визначену частину загальної задачі. При цьому, якщо виникають зміни у вимогах або проекті, їх зазвичай можна звести до зміни однієї-кількох таких підзадач, і, відповідно, змінювати тільки ті компоненти, що відповідають за вирішення цих підзадач.

Слайд 20

З третього боку, для підвищення надійності краще використовувати дублювання функцій, тобто зробити кілька компонентів відповідальними за вирішення однієї підзадачі. Причому, оскільки помилки в ПЗ найчастіше носять невипадковий характер (тобто вони повторювані, на відміну від апаратного забезпечення, де помилки пов'язані насамперед з випадковими змінами характеристик середовища і можуть бути подолані простим дублюванням компонентів, без зміни їх внутрішньої реалізації), краще використовувати різноматітні способи вирішення однієї і тієї ж задачі в різних компонентах.

Слайд 21

Список стандартів, що регламентують опис архітектури та проектну документацію взагалі, виглядає так:

- IEEE 1016-1998 Recommended Practice for Software Design Descriptions;
- IEEE 1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems.

Слайд 22

4. Типи архітектури і їх моделі. Зразки проектування. Архітектурні шаблони і стилі. Поєднання архітектурних стилів. Компонентна архітектура.

4.2. Зразки проектування та архітектурні стилі

На основі наявного досвіду дослідниками і практиками розробки ПЗ вироблено деякий безліч типових архітектур, знайомство з якими дозволяє не винаходити велосипед для вирішення досить відомих завдань. Подібні типові рішення на рівні архітектури називаються архітектурними стилями. Точніше, архітектурний стиль визначає набір типів компонентів системи і набір шаблонів їхнього взаємодій з передачі даних або управління. Різні архітектурні стилі підходять для вирішення різних завдань в плані забезпечення нефункціональних вимог, хоча одну і ту ж функціональність можна реалізувати, використовуючи різні стилі.

Слайд 23

Архітектурні стилі є зразками проектування на рівні архітектури. Зразок проектування (design pattern) –***це шаблон рішення*** часто зустрічається завдання проектування, який можна використовувати всякий раз, коли ця задача виникає. ***Зразки проектування*** поділяються залежно від масштабу рішень на архітектурні, що визначають можливу декомпозицію системи в цілому або великих підсистем, області відповідальності підсистем і правила їх взаємодії, проектні, що визначають шаблон взаємодій групи компонентів, зазвичай в рамках деякої підсистеми, для вирішення деякої загальної задачі проектування в повторюваному контексті, і ідіоми, що визначають спосіб використання мовних конструкцій для вирішення подібних завдань.

Приклади архітектурних стилів приведені в наступній таблиці.

Слайд 24-25

Таблиця 1.1 Стиль або зразок. Контекст використання. Приклади

Стиль або зразок	**Контекст використання**	**Приклади**
– Конвеєр обробки даних (data flow)	Система видає добре певні вихідні дані в результаті обробки добре певних вхідних, при цьому процес обробки не залежить від часу, застосовується багаторазово, однаково до будь даними на вході. Важливою властивістю є чітко визначена структура даних та підтримка можливості інтеграції з іншими системами	
– Пакетна обробка	Один висновок виробляється на основі читання деякого набору даних на вході, проміжні перетворення послідовні	Виконання тестів
– • Канали і фільтри	Потрібно забезпечити перетворення безперервних потоків даних, перетворення Інкрементальний, наступне може бути розпочато до закінчення попереднього, можливе додавання додаткових перетворень	Утиліти UNIX, компілятори
– Замкнутий цикл управління	Потрібно забезпечити постійне управління в умовах погано передбачуваних впливів оточення, особливо, якщо система повинна реагувати на зовнішні фізичні	Системи управління рухом

	події	
– Виклик-повернення (call-return)	Порядок виконання дій досить визначений, компонентам немає чого витрачати час на очікування звернення від інших	
– Процедурна декомпозиція	Дані незмінні, процедури роботи з ними можуть трохи змінюватися, можуть виникати нові	Основна схема побудови програм для мов C, Pascal, Ada
– Абстракние типи даних	Важливі можливості внесення змін та інтеграції з іншими системами, в системі багато даних, структура яких може мінятися	Бібліотеки компонентів
– Багаторівнева система	Важливі переносимість і можливість багаторазового використання, є природне розшарування системи на специфічні тільки для неї функції та функції загального характеру, специфічні для платформи	Протоколи (модель OSI і реальні)
– Незалежні компоненти	Можливо розпаралелювання роботи і ізпользованія декількох машин, система природно розбивається на слабо пов'язані невеликі компоненти, робота яких може бути організована майже незалежно	
– Клієнт-сервер	Завдання, які вирішуються природно розподіляються між ініціаторами та обработчиками запитів, можлива зміна зовнішнього представлення даних і способів їх обробки	Основна модель бізнес-додатків
– Розподілені об'єкти	Можливість використання розподіленої архітектури і численні дані з мінливою структурою	
– Інтерактивні системи	Необхідність досить швидко реагувати на дії користувача, мінливість користувальницького інтерфейсу	
– Дані-подання-обробник	Зміни в зовнішньому поданні досить вірогідні, одна і та ж інформація представляється по-різному в декількох місцях, система повинна швидко реагувати на зміни даних	Document-View в MFC (Microsoft Foundation Classes)
– Представлення-абстракція-управління	Інтерактивна система на основі агентів, що мають власні стани і користувальницький інтерфейс, можливе додавання нових агентів	
– Системи на основі сховища даних	Основні функції системи пов'язані зі зберіганням, обробкою і представленням великих кількостей даних	
– Репозиторій	Порядок роботи визначається потоком зовнішніх подій і цілком визначений	Средовище розробки і CASE-системи
– Дошка оголошень	Спосіб вирішення завдання в цілому невідомий або занадто трудомісткий, але відомі методи, частково вирішуючи задачу, композиція яких здатна видавати прийнятні результати, можливо додавання нових споживачів даних або обробників	Системи розпізнавання тексту

5. Багатошарова архітектура. N-рівнева/3-рівнева архітектура. Сервісно-орієнтована архітектура.

6. Розробка і оцінка архітектури на основі сценаріїв.

UML. Види діаграм UML

Для представлення архітектури (точніше різних вхідних в неї структур) зручно використовувати графічні мови. На даний момент найбільш опрацьованим і найбільш широко використовуваним з них є уніфікована мова моделювання (Unified Modeling Language, UML), хоча на високому рівні абстракції архітектуру системи зазвичай описують просто набором іменованих прямокутників, з'єднаних лініями і стрілками, що представляють можливі зв'язки.

Деякі такі мови закріплені у вигляді стандартів, наприклад

• IEEE 1320.1-1998 (R2004) Standard for Functional Modeling Language - Syntax and Semantics for IDEF0

Цей стандарт описує мову ієрархічних діаграм потоків даних.

• IEEE 1320.2-1998 (R2004) Standard for Conceptual Modeling Language - Syntax and Semantics for IDEF1X97 (IDEFobject)

Цей стандарт описує мова опису структурованих даних на основі понять сутності, зв'язку та атрибуту суті, використовуваний, зокрема, для моделювання баз даних.

UML пропонує використовувати для опису архітектури 8 видів діаграм. Не всяка діаграма на UML описує архітектуру - 9-й вид діаграм, діаграми варіантів використання, не відносяться до архітектурних уявленнями, крім того, і інші види діаграм можна використовувати для опису внутрішньої структури компонентів або сценаріїв дій користувачів і інших елементів, до архітектури не відносяться .

• Статичні структури, які відображують постійно присутні в системі сутності й зв'язки між ними, або сумарну інформацію про сутності і зв'язках, якої сутності і зв'язки, що існують в якийсь момент часу

о Діаграми класів. Показують типи сутностей системи, атрибути типів (поля та операції) і можливі зв'язки між ними, а також відносини типів між собою за спадкоємства.

Найбільш часто використовуваний вид діаграм.

о Діаграми об'єктів. Показують об'єкти системи та їх зв'язку, в деякому конкретному стані або сумарно.

Використовуються рідко.

о Діаграми компонентів. Це компоненти у вузькому сенсі, компоненти «фізичного» уявлення системи - файли з вихідним кодом, динамічно підгружаємі бібліотеки, HTML-сторінки і ін. Вони визначають розбиття системи на набір сутностей, расматриваться як атомарні з погляду її збірки і конфігураційного управління.

Використовуються рідко.

о Діаграми розгортання. Показують прив'язку (у деякий момент часу або постійну) компонентів системи (під вузькому сенсі) до фізичних пристроїв - машинам, процесорам, принтерам, маршрутизаторів та ін.

Використовуються рідко.

• Динамічні структури, що описують відбуваються в системі процеси

о Діаграми діяльностей. Показують набір процесів-діяльностей і потоки даних, що передаються між ними, а також можливі їх синхронізації один з одним.

Використовуються досить часто.

о Діаграми сценаріїв. Показують можливі сценарії обміну повідомленнями / викликами в часі між різними компонентами системи (у широкому сенсі). Ці діаграми є підмножиною іншого графічного мови - мови діграмм послідовностей повідомлень (Message Sequence Charts, MSC).

Використовуються майже так само часто, як діаграми класів.

о Діаграми взаємодії. Показують ту ж інформацію, що і діаграми сценаріїв, але прив'язують обмін повідомленнями / викликами ні до часу, а до зв'язками між компонентами.

Використовуються рідко.

о Діаграми станів. Показують можливі стану окремих компонентів або системи в цілому, переходи між ними у відповідь на будь-які події і виконувані при цьому дії.

Використовуються досить часто.

При проектуванні архітектури системи на основі вимог, зафіксованих у вигляді варіантів використання, перші можливі кроки полягають у наступному.

1) Вибирається набір «основних» сценаріїв використання – найбільш істотних і часто використовуваних.

2) Визначаються, виходячи з досвіду проектувальників, обраного архітектурного стилю і вимог до переносимості та гнучкості, компоненти відповідають за певні дії – рішення певних підзадач – в рамках цих сценаріїв.

3) Сценарії розбиваються на послідовності обміну повідомленнями між отриманим компонентами.

4) При виникненні додаткових добре виділених підзадач, додаються нові компоненти, і сценарії уточнюються.

5) Для кожного компонента в результаті виділяється його інтерфейс - набір повідомлень, які він приймає від інших компонентів і посилає їм.

6) Розглядаються «неосновні» сценарії, які так само розбиваються на послідовності обміну повідомленнями з використанням, по можливості, уже визначених інтерфейсів.

7) Якщо інтерфейси недостатні – вони розширюються.

8) Якщо інтерфейс компонента дуже великий або компонент відповідає за занадто багато чого - він розбивається на більш дрібні.

9) Там, де це необхідно в силу вимог ефективності або зручності супроводу, кілька компонентів можуть бути об'єднані в один.

10) Все це робиться до тих пір, поки не виконаються наступні умови:

a) Всі сценарії використання реалізуються у вигляді послідовностей обміну повідомленнями між компонентами в рамках їх інтерфейсів.

b) Набір компонентів достатній для забезпечення всієї потрібної функціональності, досить зручний для супроводу і з точки зору переносимості і не викликає помітних проблем з ефективністю.

c) Кожен компонент має невеликий, чітко визначене коло вирішуваних завдань і чітко визначений, збалансований за розміром інтерфейсів.

На основі можливих сценаріїв використання або модифікації системи можливий також аналіз характеристик архітектури та оцінка її придатності або

порівняльний аналіз декількох архітектур. Це так званий **метод аналізу архітектури ПЗ (Software Architecture Analysis Method, SAAM)**.

Основні його кроки наступні.

1. Визначити набір сценаріїв дій користувачів або зовнішніх систем, або сценаріїв використання деяких можливостей, які можуть вже матися на вістеме або бути новими. Сценарії повинні бути значущими для конкретних зацікавлених осіб, будь то користувач, розробник, відповідальний за супровід, представник контролюючої організації тощо. Чим повніше набір сценаріїв, тим вищою буде якість аналізу. Можна оцінити частоту появи, важливість сценаріїв.

2. Визначити архітектуру (або декілька порівнюваних архітектур). Це має бути зроблено в зрозумілій всім учасникам оцінки формі.

3. Класифікувати сценарії. Для кожного сценарію з набору повинно бути визначено, чи підтримується він даної архітектурою або потрібно вносити до неї зміни, щоб цей сценарій став виконаємо. Сценарій може підтримуватися, тобто його виконання не зажадає внесення змін ні в один з компонентів, або ж не підтримуватися, якщо його виконання вимагає змін в описі поведінки одного або декількох компонентів або змін до їх інтерфейсах. Підтримка сценарію означає, що особа, зацікавлена в його виконання оцінює ступінь підтримки як достатню, а необхідні при цьому дії як досить зручні.

4. Оцінити сценарії. Для кожного непідтримуваного сценарію треба визначити необхідні зміни в архітектурі – внесення

нових компонентів, зміни в існуючих, зміни зв'язків і способів взаємодії. Якщо є можливість, варто оцінити трудомісткість внесення таких змін.

5. Виявити взаємодію сценаріїв. Визначити які компоненти потрібно змінювати для непідтримуваних сценаріїв, компоненти, які потрібно змінювати для підтримки декількох сценаріїв - такі сценарії називають взаємодіючими, оцінити ступінь смислової пов'язаності взаємодіючих сценаріїв.

Мала зв'язаність за змістом між взаємодіючими сценаріями означає, що компоненти, в яких вони взаємодіють, виконують слабо пов'язані між собою завдання і їх варто декомпозувати.

Компоненти, в яких взаємодіють багато сценаріїв також є можливими проблемними місцями.

6. Оцінити архітектуру в цілому (або порівняти кілька заданих архітектур). Для цього треба використовувати оцінки важливості сценаріїв і ступінь їх підтримки архітектурою.

7. Статичні та динамічні діаграми при проектуванні архітектури ПЗ.

Приклади діаграм UML для простої системи ведення банківських рахунків.

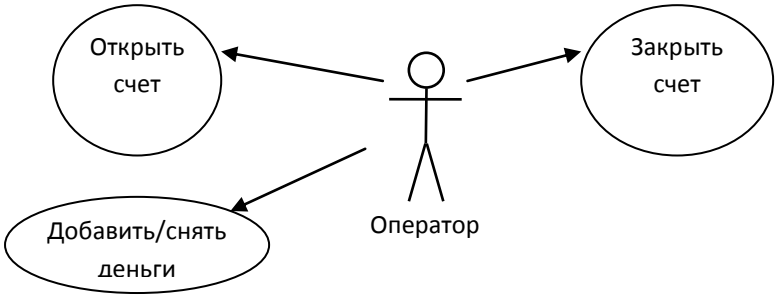


Рисунок 1. Діаграма варіантів використання (не архітектурна).

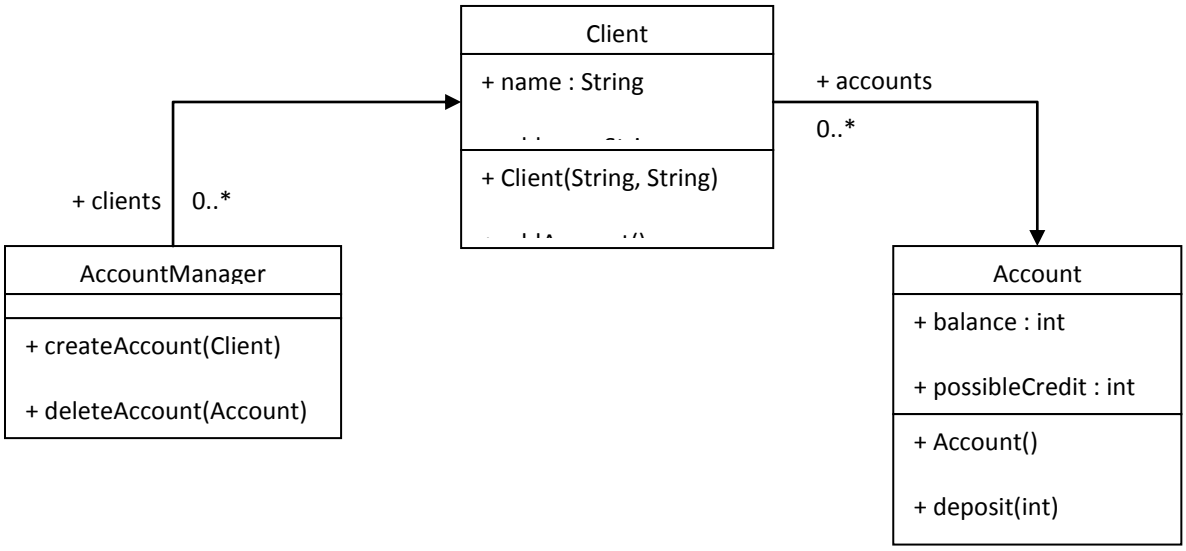


Рисунок 2. Діаграма класів

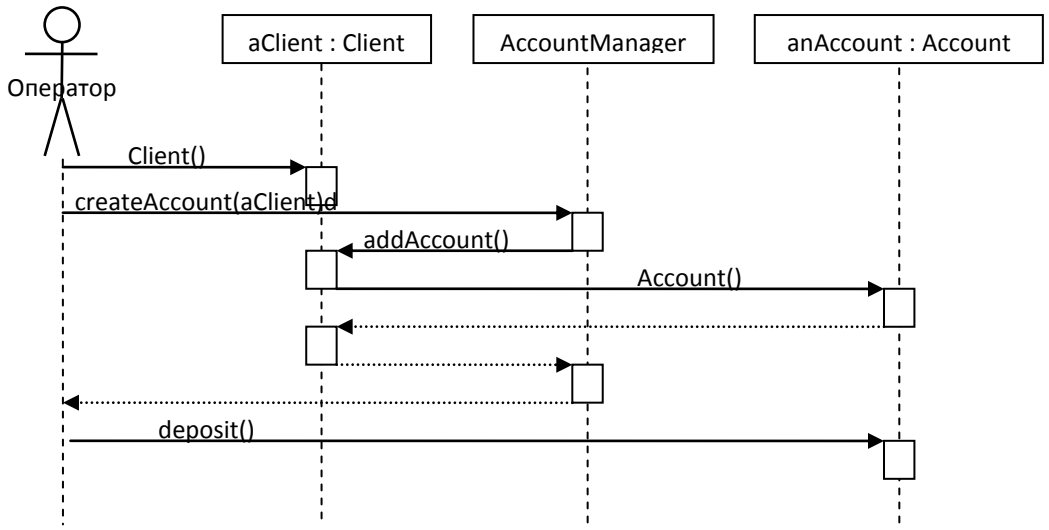


Рисунок 3. Діаграма сценаріювідткриття рахунку

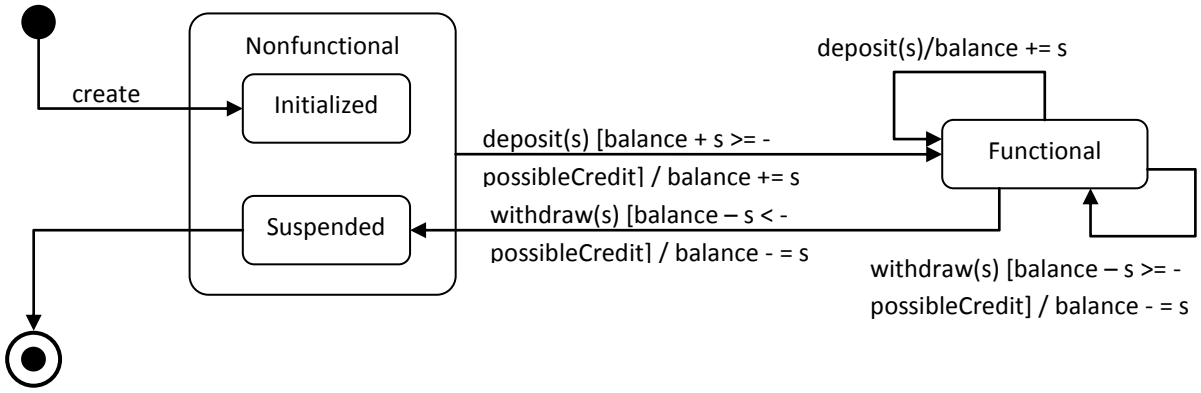


Рисунок 4. Діаграма станів рахунку

Запитання

1) Що таке архітектура ПО?
2) Які артефакти є метою проектування архітектури ПЗ "?
3) Яка роль архітектора при створенні ПЗ?
4) У якому порядку (черговості) виконуються процеси проектування ПО: проектування архітектури систем предметної області, складання плану реалізації моделі та реалізація моделі?
5) Назвіть склад архітектурної частини проекту розробки програмного забезпечення.
6) Яка роль архітектора при створенні ПЗ?

Питання для самостійної роботи

1. Принципи проектування архітектури.
2. Типи архітектури і їх моделі.
3. Багатошарова архітектура. N-рівнева/3-рівнева архітектура. Сервісно-орієнтована архітектура.

Література по темі «Архітектура ПО»

1. Чарнецки К., Айзенекер У. Порождающее программирование: методы, инструменты, применение. СПб.: Питер. 2005 .
2. Илес П. Что такое архитектура программного обеспечения? http://www.interface.ru/home.asp?artId=2116
3. Брукс Ф. Мифический человеко-месяц, или Как создаются программные системы, СПб.: Символ-Плюс, 2001.
4. И. Соммервилл. Инженерия программного обеспечения. Вильямс, 2002.
5. Э. Дж. Брауде. Технология разработки программного обеспечения. Питер, 2004.
6. М. Фаулер и др. Архитектура корпоративных программных приложений. Вильямс, 2004.
7. М. Фаулер, К. Скотт. UML в кратком изложении. М., Мир, 1999.
8. Г. Буч, Дж. Рамбо, А. Джекобсон. Язык UML.Руководство пользователя. М., ДМК, 2000.
9. Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. Приемы объектно-ориентированного проектирования. Паттерны проектирования. Питер-ДМК, 2001.
10. F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal. Pattern-Oriented Software Architecture. Wiley, 2002.
11. L. Bass, P. Clements, R. Kazman. Software Architecture in Practice. 2-nd edition, Addison-Wesley, 2003.