Лекція № 6

Тема лекції:

Архітектура комп’ютерних систем . \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ .

**План лекції**

6.1. Поняття архітектури комп’ютерних систем

6.2. Типи архітектури

6.3. Мікроархітектура і макроархітектура

6.4. Ахітектурний підхід в моделюванні комп’ютерних систем.

6.5. Значення програмного забезпечення в моделюванні КС.

6.6. Функціональні компоненти комп’ютерних систем.

6.7. Платформені архітектури комп’ютерних систем.

**Література**

1. *Авдеев В.А.* Компьютерное моделирование цифровых устройств / В.А. Авдеев. - М.: ДМК, 2012. - 360 c.

2. *Агравал Г.П.* Системы автоматического управления: теория, применение, моделирование в MATLAB: Учебное пособие / Г.П. Агравал. - СПб.: Лань, 2013. - 208 c.

3. *Айвазян С.А.* Моделирование производственного потенциала на основе концепции стохастической границы: Методология, результаты эмпирического анализа. Оценка интеллект / С.А. Айвазян, М.Ю. Афанасьев. - М.: Красанд, 2015. - 352 c.

4. *Акаев А.А.* От эпохи великой дивергенции к эпохе великой конвергенции: Математическое моделирование и прогнозирование долгосроч. технологич. и экономич. развития / А.А. Акаев. - М.: Ленанд, 2015. - 352 c.

5. *Акаев А.А.* Моделирование и прогнозирование глобального, регионального и национального развития / А.А. Акаев, А.В. Коротаев, Г.Г. Малинецкий. - М.: КД Либроком, 2012. - 488 c.

6. *Акопов А.С.* Имитационное моделирование: Учебник и практикум для академического бакалавриата / А.С. Акопов. - Люберцы: Юрайт, 2016. - 389 c.

7. *Александров А. Ю.* Математическое моделирование и исследование устойчивости биологических сообществ: Учебное пособие / А.Ю. Александров, А. Платонов. - СПб.: Лань, 2016. - 272 c.

8. *Алексеев Г.В.* Численное экономико-математическое моделирование и оптимизация / Г.В. Алексеев. - СПб.: Гиорд, 2014. - 272 c.

9. *Алямовский А.А.* SolidWorks. Компьютерное моделирование в инженерной практике / А.А. Алямовский. - СПб.: BHV, 2006. - 800 c.

10. *Афонин В.В.* Моделирование систем: учебно-практическое пособие / В.В. Афонин С.А. Федосин. - М.: Интуит, 2016. - 231 c.

11. *Бархатов В.П.* Компьютерное моделирование в системе Mathсad. Учебное пособие / В.П. Бархатов. - М.: Финансы и статистика, 2006. - 144 c.

12. *Беннинг Ш.* Финансовое моделирование с использованием Excel / Ш. Беннинг. - М.: Вильямс, 2016. - 592 c.

13. *Благодаров А.В.* Моделирование и синтез оптимальной структуры сети Ethernet / А.В. Благодаров, А.Н. Пылькин, Д.М. Скуднев. - М.: РиС, 2014. - 112 c.

14. *Боев В.* Моделирование систем. Инструментальные средства GPSS World. / В. Боев. - СПб.: BHV, 2012. - 368 c.

15. *Бочкарев А.А.* Планирование и моделирование цепи поставок / А.А. Бочкарев. - М.: Альфа-Пресс, 2008. - 192 c.

16. *Варчук Т.В.* Виктимологическое моделирование в криминологиии и практике предупреждения преступности: Монография: Учебник / Т.В. Варчук. - М.: ЮНИТИ, 2012. - 239 c.

17. *Варчук Т.В.* Виктимологическое моделирование в криминологиии и практике предупреждения преступности: Монография. / Т.В. Варчук. - М.: ЮНИТИ, 2014. - 239 c.

18. *Васильева Л.Н.* Моделирование микроэкономических процессов и систем: Учебник / Л.Н. Васильева, Е.А. Деева. - М.: КноРус, 2012. - 392 c.

19. *Волков В.Ю.* Адаптивные и инвариантные алгоритмы обнаружения объектов на изображениях и их моделирование в Matlab: Учебное пособие / В.Ю. Волков. - СПб.: Лань, 2014. - 192 c.

20. *Володин В.Я.* LTspice: компьютерное моделирование электронных схем / В.Я. Володин. - СПб.: BHV, 2010. - 400 c.

21. *Воробьев С.Н.* Статистическое моделирование информационных систем: Учебное пособиеЧ. 1 / С.Н. Воробьев. - СПб.: ГУАП, 2010. - 152 c.

22. *Вьюненко Л.Ф.* Имитационное моделирование: Учебник и практикум для академического бакалавриата / Л.Ф. Вьюненко, М.В. Михайлов, Т.Н. Первозванская. - Люберцы: Юрайт, 2016. - 283 c.

23. *Гиберт В.В.* Моделирование будущего / В.В. Гиберт. - СПб.: ИГ Весь, 2016. - 320 c.  
24. *Глущенко П.В.* Техническая диагностика: моделирование в диагностировании и прогнозировании состояния технических объектов / П.В. Глущенко. - М.: Вузовская книга, 2013. - 248 c.

25. *Голубева Н.В.* Математическое моделирование систем и процессов: Учебное пособие / Н.В. Голубева. - СПб.: Лань, 2013. - 192 c.

26. *Горлач Б.А.* Математическое моделирование. Построение моделей и численная реализация / Б.А. Горлач, В.Г. Шахов. - СПб.: Лань, 2016. - 292 c.

27. *Гусева, Е.Н.* Экономико-математическое моделирование: Учебное пособие / Е.Н. Гусева. - М.: Флинта, 2008. - 216 c.

28. *Дамаскин Б.Б.* Мультиагентное моделирование в среде NetLogo: Учебное пособие / Б.Б. Дамаскин, О.А. Петрий, Г.А. Цирлина. - СПб.: Лань, 2015. - 176 c.  
29. *Даньков В.В.* Моделирование процессов и систем: Учебное пособие / В.В. Даньков, М.М. Скрипниченко, С.Ф. Логинова и др. - СПб.: Лань, 2015. - 288 c.  
30. *Душкин А.В.* Моделирование систем управления и информационно-технического обеспечения: Учебное пособие для вузов / А.В. Душкин, В.И. Новосельцев, В.И. Сумин. - М.: рис, 2015. - 192 c.

31. *Елизаров И.А.* Моделирование систем: Учебное пособие / И.А. Елизаров, Ю.Ф. Мартемьянов. - Ст. Оскол: ТНТ, 2013. - 136 c.

32. *Емельянов С.В.* Информационные технологии и вычислительные системы: Интернет технологии. Математическое моделирование. Системы управления. Компьютерная графика / С.В. Емельянов. - М.: Ленанд, 2012. - 96 c.

33. *Зайдель А.Н.* Математическое моделирование. Построение моделей и численная реализация: Учебное пособие / А.Н. Зайдель. - СПб.: Лань, 2016. - 304 c.

34. *Замятина О.М.* Вычислительные системы, сети и телекоммуникации. моделирование сетей.: Учебное пособие для магистратуры / О.М. Замятина. - Люберцы: Юрайт, 2016. - 159 c.

35. *Карпенков С.Х.* Моделирование микроэкономических процессов и систем / С.Х. Карпенков. - М.: КноРус, 2012. - 392 c.

36. *Королев А.Л.* Компьютерное моделирование. Лабораторный практикум / А.Л. Королев. - М.: Бином, 2015. - 296 c.

37. *Мартыненко С.А.* Управление потоками работ. Функциональное моделирование и основы управления проектами / С.А. Мартыненко. - СПб.: ГУАП, 2011. - 80 c.

38. *Нартова А.* PowerDesigner 15. Моделирование данных / А. Нартова. - М.: Лори, 2014. - 469 c.

39. *Николаев В.И.* Адаптивные и инвариантные алгоритмы обнаружения объектов на изображениях и их моделирование в Matlab: Учебное пособие / В.И. Николаев, Т.А. Бушина. - СПб.: Лань, 2014. - 192 c.

40. *Прокопов С.В.* Экономико-математическое моделирование в производствен-ном менеджменте Учебник. − К.: ИМСО, 2017. − 438 с.: ил. − Библиогр.: с. 435 – 437.

41. *Сафронова В.М.* Прогнозирование, проектирование и моделирование в социальной работе: Учебное пособие для студентов учреждений высшего профессионального образования / В.М. Сафронова. - М.: ИЦ Академия, 2011. - 240 c.  
42. *Сирота А.А.* Анализ и компьютерное моделирование информационных процессов и систем / Э.К. Алгазинов, А.А. Сирота; Под общ. ред. проф. д.т.н. Э.К. Алгазинов. - М.: ДИАЛОГ-МИФИ, 2009. - 416 c.

43. *Совертков П.И.* Занимательное компьютерное моделирование в элементарной математике / П.И. Совертков. - М.: Гелиос АРВ, 2004. - 384 c.

44. *Чернышев С.Л.* Фигурные числа: Моделирование и классификация сложных объектов / С.Л. Чернышев. - М.: Красанд, 2015. - 400 c.

44. *Чэнь М.* Валидация на системном уровне. Высокоуровневое моделирование и управление тестированием. / М. Чэнь, К. Цинь, Х.-М. Ку, П. Мишра. - М.: Техносфера, 2014. - 296 c.

46. *Шелухин О.И.* Моделирование информационных систем: Учебное пособие для вузов / О.И. Шелухин. - М.: рис. 2016. - 536 c.

**Додаткова**

1. *Боев В.Д.* Моделирование в среде anylogic : учебное пособие для вузов / В. Д. Боев. — Москва : Издательство Юрайт, 2019. — 298 с.
2. *Волкова В. Н.* Моделирование систем и процессов. Практикум : учебное пособие для академического бакалаврата / В. Н. Волкова [и др.] ; под редакцией В. Н. Волковой. — Москва : Издательство Юрайт, 2019. — 295 с.
3. *Волкова В. Н.* Моделирование систем и процессов : учебник для академического бакалавриата / В. Н. Волкова [и др.] ; под редакцией В. Н. Волковой, В. Н. Козлова. — Москва : Издательство Юрайт, 2019. — 450 с.
4. *Долганова О. И.* Моделирование бизнес-процессов: учебник и практикум для академического бакалавриата / О. И. Долганова, Е. В. Виноградова, А. М. Лобанова ; под редакцией О. И. Долгановой. — М.: Издательство Юрайт, 2019. — 289 с.
5. *Девятков В.В.* Имитационное моделирование: Учебное пособие / Н.Б. Кобелев, В.А. Половников, В.В. Девятков. - М.: КУРС, НИЦ ИНФРА-М, 2013. - 368 c.
6. *Жмудь В. А*. Моделирование замкнутых систем автоматического управления : учебное пособие для академического бакалаврата / В. А. Жмудь. — 2-е изд., испр. и доп. — М.: Издательство Юрайт, 2019. — 128 с.
7. *Пригарин С.М.* Статистическое моделирование многомерных гауссовских распределений: учебное пособие для вузов / С. М. Пригарин. — М.: Издательство Юрайт, 2019. — 83 с.
8. Советов, Б.Я. Моделирование систем: учебник для академического бакалавриата / Б. Я. Советов, С. А. Яковлев. — 7-е изд. — М.: Издательство Юрайт, 2019. — 343 с.
9. *Советов Б.Я.* Моделирование систем. Практикум : учебное пособие для бакалавров / Б. Я. Советов, С. А. Яковлев. — 4-е изд., перераб. и доп. — М.: Издательство Юрайт, 2019. — 295 с.
10. *Стельмашонок Е.В.* Моделирование процессов и систем : учебник и практикум для академического бакалавриата / под редакцией Е. В. Стельмашонок. — Москва : Издательство Юрайт, 2019. — 289 с.
11. ​ *Черткова, Е.А.*  Программная инженерия. Визуальное моделирование программных систем : учебник для академического бакалавриата / Е. А. Черткова. — 2-е изд., испр. и доп. — М.: Издательство Юрайт, 2019. — 147 с.

Зміст лекції

**6.1. Поняття архітектури комп’ютерних систем**

Рівень розвитку сучасних технологій настільки високий, що дозволяє моделювати компютерну систему будь-якого масштабу, складності і функціональності. Проте, враховуючи вимоги бізнесу, грунтовані на показниках різних бізнес-оцінок, виникають додаткові складнощі, дозвіл яких зводиться до забезпечення раціонального підходу до процесу проектування, реалізації і подальшої експлуатації комп’ютерних систем. Виходячи з цього, можна однозначно рахувати вибрану архітектуру одним з основних показників ефективності створюваної комп’ютерної системи, а, отже, і успішності бізнесу.

Спочатку термін архітектура застосовувався в проектуванні і спорудженні різних споруд. Він визначав їх структуру, взаємозв'язки між складовими частинами, базові принципи їх організації і подальшого розвитку. З часом розуміння терміну "архітектура", стосовно технічних систем, дещо змінилося. У технічному аспекті можна розглядати архітектуру, як високо абстрактну модель, в якій відсутні подробиці реалізації.

Визначити поняття "Архітектура компютерної системи" можна безліччю способів. Це пов'язано:

1. З відсутністю загальноприйнятого визначення самої компютерної системи. Враховуючи складність структури, достатнім способом описати її можливо тільки при консолідації декількох точок зору, що у кожному конкретному випадку може призводити до різних результатів.

2. З різноманіттям трактувань самого терміну "архітектура".

В результаті, архітектуру компютерної системи можна описати як "концепцію, що визначає модель, структуру, виконувані функції і взаємозв'язок компонентів комп’ютерної системи". Процедура вибору архітектури для моделювання комп’ютерної системи, в ринкових умовах, зводиться до визначення вартості володіння нею. Вартість володіння компютерною системою складається з планових витрат і вартості ризиків. Планові витрати включають вартість технічного обслуговування, модернізації, зарплату обслуговуючого персоналу і так далі.

Сукупна вартість ризиків визначається з вартості усіх типів ризиків, їх вірогідності і матрицею відповідності між ними. Сама ж матриця відповідності визначається вибраною архітектурою інформаційної системи.

Можна виділити найбільш важливі типи ризиків :

- проектні риски (риски при створенні системи);

- риски розробки (помилки, недостатня оптимізація);

- технічні риски(простої, відмови, втрата даних);

- бізнес-риски (виникають із-за технічних ризиків і пов'язані з експлуатацією системи);

- невизначеності (пов'язані з варіативною бізнес-процесів і складаються з необхідності внесення змін до системи і неоптимальною процедурою функціонування);

- операційні (мають на увазі невиконання набору операцій, можуть виникати із-за технічних ризиків і бути ініціаторами бізнес-ризиків).

Концепція архітектури компютерної системи повинна формуватися ще на етапі техніко-економічного обгрунтування і вибиратися такій, щоб вартість володіння нею була мінімальною.

Для того, щоб конструктивно визначити архітектуру, необхідно відповісти на низку запитань:

1. Що робить система?

2. На які складені частина вона розділена?

3. Яким чином відбувається взаємодія цих частин?

4. Як і де ці частини розміщені?

Таким чином, можна вважати архітектуру комп’ютерної системи моделлю, що визначає вартість володіння через наявну в цій системі інфраструктуру.

**6.2. Типи архітектури**

Розглядаючи архітектуру великих організацій, прийнято використати поняття "Корпоративна архітектура". Її можна представити у вигляді сукупності декількох типів архітектури :

- бізнес архітектура(Business architecture);

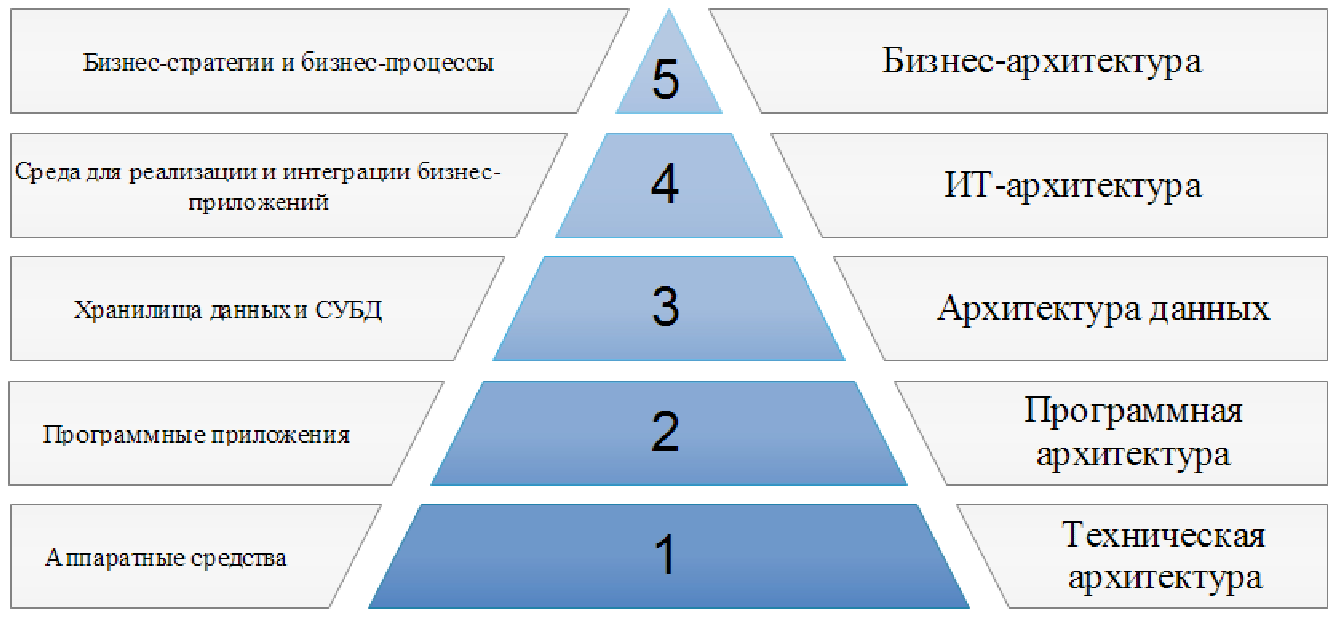
- ІТ-архітектура(Information Technology architecture);

- архітектура даних(Data architecture);

- програмна архітектура(Software architecture);

- технічна архітектура(Hardware architecture).

Модель корпоративної архітектури представлена на мал. 8.



**Технічна архітектура** є першим рівнем архітектури комп’ютерної системи. Вона описує усі апаратні засоби, що використовуються при виконанні заявленого набору функцій, а також включає засоби забезпечення мережевої взаємодії і надійності. У технічній архітектурі вказуються периферійні пристрої, мережеві комутатори і маршрутизатори, жорсткі диски, оперативна пам'ять, процесори, сполучні кабелі, джерела безперебійного живлення і тому подібне.

**Програмна архітектура** є сукупністю комп'ютерних програм, призначених для вирішення конкретних завдань. Цей тип архітектури потрібний для опису додатків, що входять до складу системи. На цьому рівні описують програмні інтерфейси, компоненти і поведінку.

**Архітектура даних** об'єднує в собі як фізичні сховища даних, так і засоби управління даними. Крім того, в неї входять логічні сховища даних, а при оріентованності даної компанії на роботу зі знаннями, може бути виділений окремий рівень - архітектура знань (Knowledge architecture). На цьому рівні описуються логічні і фізичні моделі даних, визначаються правила цілісності, складаються обмеження для даних.

Слід особливо виділити рівень комп’ютерної - архітектури, оскільки він є єднальним. На ньому формується базовий набір сервісів, які використовуються як на рівні програмної архітектури, так і на рівні архітектури даних. Якщо яка-небудь особливість функціонування для цих двох рівнів не була передбачена, то сильно зростає вірогідність збоїв в роботі, а, отже, втрат для бізнесу. В деяких випадках неможливо розділити ІТ-архітектуру і архітектуру окремого застосування. Таке можливе при високій мірі інтеграції додатків. Прикладом ІТ-архітектури може служити SharePoint від компанії Microsoft. Цей продукт надає сервіси для спільної роботи і зберігання інформації, що є дуже важливим аспектом функціонування будь-якої компанії. Його базові системні модулі відносяться до ІТ-архітектури, а призначені для користувача - до програмної. Основною функцією ІТ-архітектури є забезпечення функціонування важливих бізнес-застосувань для досягнення позначених бізнес-целей. Якщо деяку функцію потрібно відразу в декількох застосуваннях, то її слідує пер Якщо деяку функцію потрібно відразу в декількох застосуваннях, то її слід перенести на рівень ІТ-архітектури, тим самим підвищивши інтеграцію системи і понизити складність архітектури додатків.

Останнім в ієрархії є рівень **бізнес-архитектурі** або архітектура **бізнес-процесів.** На цьому рівні визначаються стратегії ведення бізнесу, способи управління, принципи організації і ключові процеси, що представляють для бізнесу величезну важливість.

**6.3. Мікроархітектура і макроархітектура**

Терміни мікроархітектура і макроархітектура більшою мірою застосовуються для опису програмних систем. У відповідність з розглянутою моделлю рівнів архітектури корпоративних інформаційних систем, мікроархітектуру можна віднести до рівнів програмної архітектури і архитектуры даних, а макро-архітектуру до рівня ІТ-архітектури.

Мікроархітектура описує внутрішній устрій конкретного компонента або підсистеми, а макроархітектура описує пристрій усією ИС, як сукупності її компонент або підсистем.

Складність програмних систем постійно збільшується. Це обумовлено зростанням об'єму передаваної і оброблюваної інформації, складністю самих завдань по обробці інформації і збільшенню кількості таких завдань. Без застосування якого-небудь архітектурного підходу при побудові складних систем, їх створення, обслуговування і модифікація, врешті-решт, стануть нерентабельними для бізнесу.

Для вирішення цієї проблеми можна використовувати такі методи моделювання як абстракції, декомпозиції, інкапсуля. Так, при розробці програмної системи, наприклад великої організації, що входить до складу інфраструктури, вона представляється у вигляді безлічі модулів, кожен з яких виконує визначену функцію, а всі разом вони виконують функції самої системи. В даному випадку, організація кожного модуля буде мікроархітектурою, а способи взаємодії між цими модулями у рамках системи - макроархітектурою.

Зменшення складності реалізації системи відбуватиметься за рахунок розбиття складних завдань на дещо простіших. В результаті, це може привести до появи великого набору простих завдань, на підставі якого можна буде реалізувати будь-кого, навіть дуже складної. У цьому полягає принцип об'єктно-орієнтованого програмування - створення конкретних класів об'єктів для вирішення конкретних завдань.

На жаль, не завжди сформована система класів зможе спростити процеси управління програмною системою. На приклад, підсистема виконує занадто великий спектр дій необхідих для різних процесів в організації і вихід її з ладу порушує функціонування відразу декількох з них.

Існують два принципи, що дозволяють оцінити взаємний вплив компонентів системи один на одного :

- low coupling(слабка зв'язаність);

- high cohesion(сильне зачеплення).

Принцип Low Coupling сприяє розподвлу функцій між компонентами системи так, щоб міра зв'язаності між ними стає низькою. Міра зв'язаності(coupling) - це міра взаємозалежності підсистем. Цей принцип пов'язаний з одним з основних принципів системного підходу, який вимагає мінімізації інформаційнних потоків між підсистемами.

Підсистема з низькою мірою зв'язаності (чи слабким зв'язуванням) має наступні властивості:

- мале число залежностей між підсистемами;

- слабка залежність однієї підсистеми від змін в іншій;

- висока міра повторного использова¬ния підсистем.

У свою чергу, принцип High Cohesion задає властивість сильного зачеплення усередині підсистеми. В результаті підсистеми виходять сфокусованими, керованими і зрозумілими. Зачеплення (функціональне зачеплення) - це міра зв'язаності і сфокусованої функцій підсистеми. Підсистема має високу міру зачеплення, якщо її функції тісно пов'язані між собою, і вона не виконує великих об'ємів роботи.

Підсистема з низькою мірою зачеплення виконує безліч різних функцій ніяк не пов'язаних між собою. Такі підсистеми створювати небажано, оскільки вони призводять до возникнове¬нию наступних проблем:

- трудність розуміння.

- складність при повторному використанні.

- складність підтримки.

- ненадійність, постійна схильність до змін.

Підсистеми з низькою мірою зачеплення не мають чіткого функціонального призначення і виконують занадто різнопланові функції, які можна легко розподілити між іншими підсистемами.

Слід зауважити, що зв'язаність є характеристикою системи цілком, а зв'язність характеризує окремо взяту підсистему. Зв'язаність (coupling) і зв'язність (cohesion) є загальносистемними характеристиками і можуть застосовуватися при проектуванні будь-яких систем.

**6.4. Ахітектурний підхід в моделюванні комп’ютерних систем.**

Процес проектування комп’ютерної системи тісним чином пов'язаний з її архітектурним описом, що відбито в деяких визначеннях терміну "архітектура".

Можна виділити п'ять різних підходів до проектування:

- Календарний підхід.

- Підхід, за основу якого узятий процес управління вимогами.

- Підхід, грунтований на процесі розробки документації.

- Підхід, в основі якого лежить система управління якістю.

- Архітектурний підхід.

**Календарний підхід** має на увазі складлення графіку майбутніх робіт з їх поетапним виконанням. Слід зазначити, що ключові рішения приймаються на підставі локальних завдань і цілей кожного конкретного етапу розробки. Також при цьому підході практично не приділяється час на розробку документації, формуванню архітектури і процесів по внесенню різних змін. У довгостроковій перспективі через це зростає вартість володіння, розробленої таким чином системи. Такий стиль вважається морально застарілим, проте в деяких компаніях досі застосовується.

Підхід, за основу якого узято процес **управління вимогами**, велику частину часу усього процесу розробки виділяє на функциональні характеристики системи, а нефункціональні, такі як масштабованість, наприклад, практично не розглядаються. Усі рішення в ході проекту формуються виходячи з локальних цілей по реализації конкретного функціонала. Такий підхід може бути ефективний, якщо вимоги до разрабки системи визначені заздалегідь і не змінюються в процесі проектування. В якості недоліків можна виділити невідповідність стандарту якості ISO 9126 і нестабільність архітектур, що розробляються, оскільки кожна функція связуеться, що реалізовується, з одним або декількома компонентами. У зв'язку з цим, трудомісткість при додаванні до подобіних систем додаткових функцій зростає. Застосування цього підходу в довгостроковій перспективі є нераціональним, проте дозволить успішно управляти вимогами в рамках потреб функціональності.

При застосуванні підходу, грунтованого на процесі **розробки документації**, невиправдана велика кількість часу. шо витрачається на формування пакету документів, які часто не використовуються ні замовником, ні користувачем. Крім того, із-за нестачі часу страждає якість системи, що саме розробляється. Цей підхід використовується в урядових організаціях і великих компаніях.

Процес проектування, в основі якого лежит **система управління якістю**, включає велику кількість різнопланових заходів для відстеження найбільш значних для функціонування системи параметрів. Вибрані параметри спостерігаються на всіх стадіях розробки системи, причому, в деяких випадках, на шкоду іншим. В деяких випадках набір такого роду параметрів може бути зроблено не правильно, і оптимізація системи буде проведена помилково. У цьому полягає основний недолік подібного підходу. Окрім цього, при післяявища нових вимог дуже важко змінювати функціональність, а значить подальший розвиток системи, у тому числі і із-за архітектурних прорахунків, може бути неможливий. Такий підхід вважається консервативним, а його застосування доцільно при необхідності створити систему з екстремальними ха-рактеристиками.

**Архітектурний підхід** до проектування інформаційних систем можна вважати найбільш зрілим. Його ключовим аспектом є створення фреймворка, тобто каркаса, адаптація якого під потреби конкретної системи буде легка здійснена. Відповідно до цього, завдання проектування розбивається на дві: розробка багаторазово використовуваного каркаса і створення системи на його основі. Слід зазначити, що ці підзадачі можуть вирішуватися різними групами фахівців. При використанні каркасів з'являється можливість досить швидко змінювати функціональність системи за рахунок итеративности процесу проектування. Архітектурний підхід покликан ліквідувати недоліки, що виникли в процесі проектування, заснованому на управління вимогами.

З усіх розглянутих стилів проектування не можливо однозначно визначити кращий, оскільки кожна конкретна проектна група, кожна проектована система має свої особливості, на підставі яких слід вибрати той підхід, застосування якого дозволить вирішити поставлені задачі у встановлені терміни і у рамках виділеного бюджету.

**6.5. Значення програмного забезпечення в моделюванні КС.**

Користувачі сучасних комп’ютерних систем практично завжди взаємодіють з ними за допомогою спеціальних програмних модулів, від показників якості яких залежить рівень якості усієї інформаційної системи цілком.

Існує величезна кількість стандартів для створення правильної і надійної архітектури, а також для розробки і інтеграції програмних систем. Практично кожна технологія, існуюча у світі, має свій стандарт, описаний у відповідних нормативних документах. Застосування цих стандартів істотно більшуе шанси на успішне створення системи і її подальше безвідмовне функціонування, проте раціональність їх застосування повинна визначатися до моменту початку робіт, оскільки складність системи при їх інтеграції може істотно зрости.

Якістю програмного забезпечення можна вважати сукупність його характеристик, що відносяться до можливості задовольняти висловлені або такі, що маються на увазі потреби усіх зацікавлених осіб. Це визначення включене в стандарт ISO 9126, в якому також визначені і самі характеристики.

Можна виділити три аспекти якості:

1. Внутрішня якість(характеристики самого програмного забезпечення).

2. Зовнішня якість(поведінкові характеристики програмного забезпечення).

3. Контекстна якість(відчуття користувачів при різних контекстах вико рис-тання).

Керуючись цими аспектами, стандарт ISO 9126 виділяє шість характеристик якості програмного забезпечення:

1. Функціональність.

2. Надійність.

3. Продуктивність.

4. Зручність використання.

5. Зручність супроводу. 6.

Переносимість.

**Функціональність** має на увазі здатність ПЗ вирішувати завдання в певних умовах і підрозділяється на наступні підхарактеристики:

- функціональна придатність(suitability);

- здатність вирішувати потрібний набір завдань;

- точність(accuracy);

- здатність отримувати необхідні результати;

- здатність до взаємодії(interoperability);

- захищеність(security) - здатність запобігати неавторизованому доступу до даних і програм;

- відповідність стандартам і правилам(compliance) - відповідність програмного обеспече¬ния різним регламентуючим нормам.

**Надійність** (reliability) характеризується здатністю програмного забезпечення утримувати функціональність в заданих рамках за певних умов і підрозділяється на наступні підхарактеристики:

- зрілість(maturity) - величина, зворотна частоті відмов програмного забезпечення;

- стійкість до відмов(fault tolerance) - здатність утримувати певний рівень працездатності при різних відмовах і порушеннях правил взаємодії з оточенням;

- здатність до відновлення(recoverability) - здатність відновлювати необхідний рввень працездатності після відмови;

- відповідність стандартам надійності(reliability compliance).

**Продуктивність** (efficiency) визначається здатністю програмного забезпечення за певних умовгарантувати необхідну працездатність по відношенню до ресурсів, що виділяються для цього. Можна також визначити, як відношення отримуваних результатів до витрачених ресурсів. Ця характеристика підрозділяється на наступні підхарактеристики:

- тимчасова ефективність(time behavior) - здатність програмного забезпечення отримувати необхідні результати і забезпечувати передачу необхідного об'єму даних за певний час;

- ефективність використання ресурсів(resource utilization) - здатність програмного обес-печение вирішувати необхідні завдання і використанням заданих об'ємів певних видів ресурсів;

- відповідність стандартам продуктивності(efficiency compliance).

**Зручність використання** (usability) характеризуется привабливістю для користувачів, зручністю в навчанні і використанні програмного забезпечення. У своєму складі також має ряд подха-рактеристик :

- зрозумілість(understandability) - величина зворотна зусиллям, витраченим користувачами, по усвідомленню застосовності програмного забезпечення для вирішення необхідних задач;

- зручність роботи(operability) - величина зворотна зусиллям, витраченим користувачами, для вирішення необхідних завдань за допомогою програмного забезпечення;

- зручність навчання(learnability) - величина зворотна зусиллям, витраченим користувачами, на процес навчання роботі з програмним забезпеченням;

- привабливість(attractiveness) - здатність програмного забезпечення бути привабливою для користувачів;

- відповідність стандартам зручності використання(usability compliance).

**Зручність супроводу** (maintainability) характеризується зручністю супроводу програмного забезпечення. Ця характеристика також включає ряд підхарактеристик :

- анализируемость(analyzability) характеризується зручністю проведення аналізу помилок, дефектів, недоліків, необходимостей внесення змін і їх можливих наслідків;

- зручність внесення змін(changeability) - величина зворотна трудовитратам на виконання необхідних змін;

- стабільність(stability) - величина обрат¬ная риску появи непередбачених наслідків при внесенні необхідних змін;

- зручність перевірки(testability) - величина зворотна необхідним трудовитратам на тестування і інші види перевірок досягнення передбачених результатів при внесенні змін;

- відповідність стандартам зручності супроводу(maintainability compliance).

**Переносимість** (portability) характеризується здатністю програмного забезпечення зберігати працездатність при зміні організаційних, апаратних і програмних аспектів оточення. Для цієї характеристики виділяються наступний підхарактеристики:

- адаптивність (adaptability) - здатність програмного забезпечення без здійснення непередбачених дій пристосовуватися до змін оточення;

- зручність встановлення (installability) - здатність програмного забезпечення встановлюватися в заздалегідь визначене оточення;

- здатність до співіснування (coexistence) - здатність програмного обеспече¬ния функціонувати в загальному оточенні з іншими програмами, розділяючи з ними ресурси;

- зручність заміни (replaceability) - можливість застосування програмного забезпечення замість вже використовуваного для вирішення тих же завдань, в тому ж оточенні;

- відповідність стандартам переносимості(portability compliance).

Усі вказані характеристики описують внутрішню і зовнішню якість програмного забезпечення. Для опису контекстної якості існує інший, зменшений набір характеристик :

- ефективність(effectiveness) - способность програмного забезпечення вирішувати призначені для користувача завдання із заданою точністю і в заданому контексті;

- продуктивність(productivity) - способность програмного забезпечення отримувати необхідні результати при використанні заздалегідь певної кількості ресурсів;

- безпека(safety) - здатність програмного забезпечення підтримувати необхідний низький рівень ризику нанесення збитку людям, бізнесу і довкіллю;

- задоволеність користувачів(satisfaction) - здатність програмного забезпечення при використанні в певному контексті приносити задоволення користувачам.

Керуючись розглянутими показниками можна значним чином збільшити якість програмних модулів, а, отже, і усієї інформаційної системи в цілому.

**6.6. Функціональні компоненти комп’ютерних систем.**

Враховуючи принцип декомпозиції, прийнято проектувати комп’ютерні системи з розділенням функціонального призначення їх компонентів, тобто створювати багаторівневе представлення.

Можна виділити три основні функциональні групи, призначені для вирішення разлних по сенсу завдань:

1. Взаємодія з користувачами.

2. Бізнес-логіка.

3. Управління ресурсами.

Реалізація такого функціонала відбувається за допомогою створення відповідної програмної системи. Така система також має багаторівневе представлення компонентів (рис. 9).

Програмная система

|  |  |  |
| --- | --- | --- |
| г | Компонент | \ |
| \ | предс тавления | / |
| г | Прикладной  компонент |  |
|  |  | ) |
| г | Компонент | \ |
| V | управления  ресурсами | У |

Рис. 9 - Компоненти программної системи

**Компонент представлення** служить для забезпечення взаємодії користувачів з програмою, тобто обробляє натиснення клавіш, руху різних контроллерів, здійснює виведення інформації - надає призначений для користувача інтерфейс.

**Прикладний компонент** є набором правив і алгоритмів реалізації функцій системи, реакцій на дії користувачів або внутрішніх події, обробки даних. Компонент управління ресурсами відповідає за зберігання, модифікацію, вибірку і видалення даних, пов'язаних з вирішуваною прикладною задачею.

Одним з найважливіших етапів проектування архітектури інформаційної системи є розподіл цих функціональних компонентів по вибраній платформеній архітектурі.

**6.7. Платформені архітектури комп’ютерних систем.**

Можна виділити три напрями розвитку платформеної архітектури:

1. Автономні.

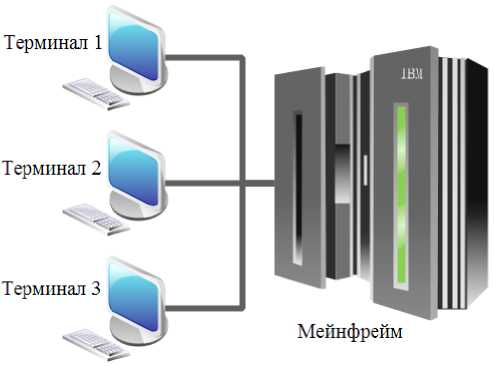
2. Централізовані.

3. Розподілені.

**Автономна архітектура** має на увазі наявність усіх функціональних компонентів системи на одному фізичному пристрої, наприклад, комп'ютері і не повинна мати зв'язків із зовнішнім середовищем. Прикладом таких систем можуть служити системні утиліти, текстові редактори і досить прості корпоративні програми. Слід відзначити, що в процесі побудови корпоративної комп’ютерної системи, як правило, не повинне формувати не пов'язаних вузлів або модулів. Їх поява може бути обумовлене визначеними вимогі до безпеки або надійності.

**Централізована архітектура** має на увазі виконання усіх необхідних завдань на спеціально відведенному вузлі, потужності якого вистачає, щоб задовольнити потреби усіх користувачів. Такий тип архітектури був популярний на зорі комп'ютерної техніки (70-і роки 20 віків), проте, і зараз являет-ся востре

виконання усіх необхідних завдань на спеціально отведен-ном вузлі, потужності якого вистачає, щоб удовлет-ворить потреби усіх користувачів. Такий тип архи-тектуры був популярний на зорі комп'ютерної техніки(70-і роки 20 віків), проте, і зараз являет-ся затребуваним. Компоненти системи в даному випадку розподіляються між обчислювальним вузлом, котрий зветься мейнфрейм (mainframe), і термінальною станцією, за якою працює користувач. Термінал містить компонент представлення, а мейнфрейм - прикладний компонент і компонент управління ресурсами. Слід зазначити, що термінал виступає виключно у вигляді облаштування вводу-виводу і не має інших функціональніх можливостей. Представлення централизованої архітектури показане на рис.10.



До переваг такої архітектури можна віднести:

- відсутність необхідності адміністрування робочих місць;

- легкість обслуговування і експлуатації системи, оскільки усі ресурси зосереджені в одному місці.

Недоліками подібної архітектури є:

- функціонування усієї системи повністю залежить від головного вузла(мейнфрейма);

- усі ресурси і програмні засоби виявляються колективними і не можуть бути змінені під потрібні конкретних користувачів.

Щоб позбавитися від останнього недоліку, в сучасних інформаційних системах застосовуються технології віртуалізації, завдяки яким стає можливим виділити кожному користувачеві необхідну кількість ресурсів і встановити необхідне програмне забезпечення.

Слід зауважити, що за допомогою технологій віртуалізації можна створити практично будь-яку архітектуру, використовуючи при цьому тільки ресурси мейнфрейма.

Поява і розвиток розподіленої архітектури пов'язана з інтенсивним розвитком технічних і програмних засобів. У цьому типі архітектури функціональні компоненти інформаційної системи розподіляються по наявних вузлах залежно від поставлених цілей і завдань. Можна виділити шість основних характеристик архітектури розподілених систем:

- спільне використання ресурсів (як апаратних, так і програмних);

- відкритість - можливість збільшення типів і кількості ресурсів;

- паралельність - можливість виконання декількох процесів на різних вузлах системи(при цьому вони можуть взаємодіяти);

- масштабованість - можливість додавати нові властивості і методи;

- відмовостійкість - здатність системи підтримувати часткову функціональ-ність за рахунок можливості дублювання інформації, апаратної і програмної складової.

До недоліків розподілених систем слід віднести:

- структурна складність;

- складно забезпечити достатній рівень безпеки;

- на управління системою потрібно велику кількість зусиль;

- непередбачувана реакція на зміни.

Усі з них пов'язані в першу чергу із складною структурою, різноплановим устаткуванням і складною системою розподілу прав доступу. Необхідно враховувати все з них, інакше розроблена інформаційна система не зможе функціонувати у рамках очікуваних параметрів.

Існують наступні види розподіленої архітектури:

- архітектура "файл-сервер";

- архітектура "клієнт-сервер";

- архітектура Web- додатків.

Файл-серверная архітектура має на увазі наявність виділеного мережевого ресурсу для зберігання даних. Такий ресурс називається "Файловим сервером". При такій архітектурі усі функціональні компоненти системи розташовані на користувач¬ском комп'ютері, який називається "клієнтом", а самі дані знаходяться на сервері.

Така організація системи має наступні переваги:

- розрахований на багато користувачів режим роботи з даними, що зберігаються на сервері;

- централізоване управління правами доступу до загальних даних;

- мала вартість розробки;

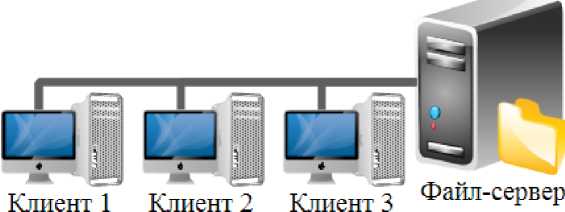
- висока швидкість розробки.

Недоліки файл-серверной архітектури :

- послідовний доступ до загальних даних і відсутність гарантії їх цілісності;

- продуктивність(залежить від продуктивності мережі, клієнта і сервера);

Класичне представлення файл-серверной архітектури представлене на рис.11.



Архітектура "клієнт-сервер" представляє собою мережеву інфра-структуру, в якій сервери є постачальниками певних сервісів (послуг), а клиент-скі комп'ютери виступають їх споживачами. Класичне представлення клієнт-серверної архітектури має на увазі наявність в мережі сервера і декількох підключених до нього клієнтів. У таких системах сервер, в основному, грає роль постачальника послуг з використання бази даних.

Ця архітектурна модель називається дворівневою(two - tier architecture). Двухрівнева архітектура представлена на рис.12.



Переваги цієї архітектури:

- підтримка розрахованої на багато користувачів работи;

- гарантія цілісності даних;

- наявність механізмів управління правами доступу до ресурсів сервера;

- можливість розподілу функцій між вузлами мережі.

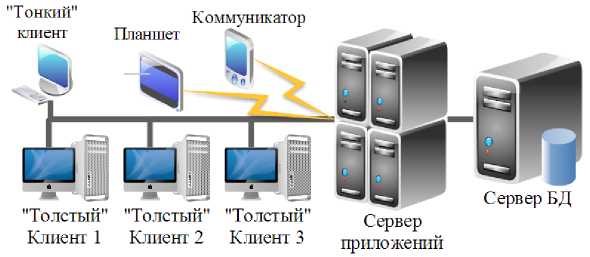
Недоліки:

- вихід з ладу сервера може спричинити непрацездатність усієї системи;

- потрібно високий рівень технічного персоналу;

- висока вартість устаткування.

При збільшенні масштабів системи може знадобитися заміна апаратної частини сервера і клієнтських машин. Проте, при збільшенні числа користувачів виникає необхідність сінхронізації версій великої кількості додатків. Для вирішення цієї проблеми використовують многорівневу архітектуру(три і більше рівнів). Частина загальних додатків застосувань переноситься на спеціально виділенний сервер, тим самим знижуються вимоги до продуктивності клієнтських машин. Клієнти з низькою обчислювальною потужністю називають "тонкими клієнтами", а з високою производительнос¬тью - "товстими клієнтами". При багатоланковій архітектурі з виділеним сервером додатків існує можливість використання портатив¬ных пристроїв. Багатоланкова архітектура показана на рис.13. Використання такого типу архітектури обумовлене високими вимогами додатка до ресурсів. У такому разі існує винести його на окремий сервер і, тим самим, знизити вимоги до продуктивності робочих станцій.

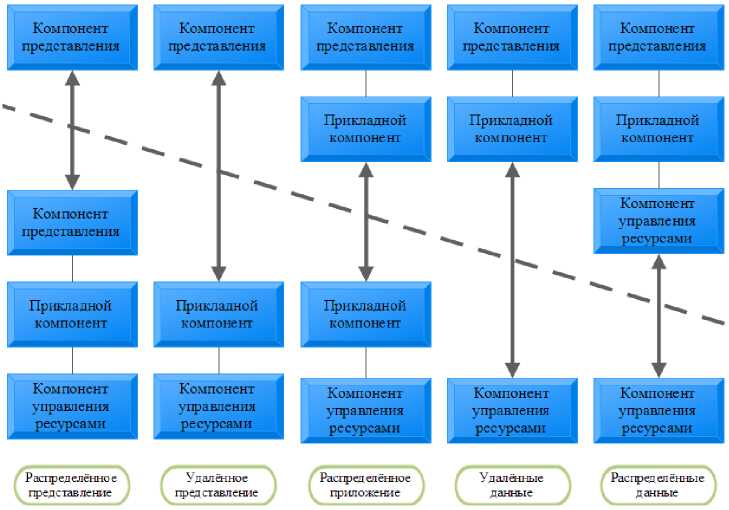


Грамотний підбір характеристик сервера додатків, сервера баз даних і клієнтських робітників станцій дозволить створити інформаційну систему з прийнятною вартістю володіння.

Слід зауважити, що розподіл функци-ональных компонентів системи при використанні клієнт-серверної архітектури може робитися декількома способами.

Архітектура Web- додатків або архитек¬туру Web- сервісів має на увазі надання деякого сервісу, доступного в мережі Internet, через спеціальне застосування.

Основою для надання таких послуг служать відкриті стандарти і протоколи SOAP, UDDI і WSDL. SOAP(Simple Object Access Protocol) визначає формат запитів до Web- сервісам. Дані між клієнтом і сервісом передаються в SOAP- конвертах(envelops). WSDL(Web Service Description Language) служить для опису інтерфейсу сервісу, що надається. Перед розгортанням web- застосування вимагається скласти



його опис, вказати адресу, список підтримуваних протоколів, перелік допустимих операцій, а так само формати запитів і відповідей. UDDI(Universal Description, Discovery and Integration) є протоколом пошуку Web- сервісів в мережі Internet. Пошук здійснюється по їх описах, які розташовані в спеціальному реєстрі.

Архітектура таких сервісів схожа по концепції з багатоланковою клієнт-серверною, проте, сервера додатків і баз даних розташовуються в мережі Internet.

Можна виділити три технології, які воз-можно використати для побудови розподіленої архітектури Web- сервісу :

1. EJB(Enterprise JavaBeans).

2. DCOM(Distributed Component Object Model).

3. CORBA(The Common Object Request Broker Architecture).

Ідеєю для створення EJB було бажання створити таку інфраструктуру, в якій було б лігко додавати і видаляти компоненти, при цьому змінювати функціональність сервера. EJB дозволяє разробнкам складати власні застосування із заздалегідь створених модулів. При цьому можлива їх зміна, що робить процес розробки гнучким і набагато швидшим. Ця технологія совмес¬тима з CORBA і Java API.

Взаємодія між клієнтів і сервером в даному випадку представляється як взаємодія EJB- об'єкту, генерованого спеціальним генератором, і EJB- компонента, написаного розробником. При необхідності викликати метод у EJB- компонента, що знаходиться на сервері, викликається однойменний метод EJB- об'єкту, розташованого на стороні клиента, який звязаний з необхідною компонентою і викликає необхідний метод.

Переваги EJB:

- просте і швидке створення;

- Java- оптимізація;

- кроссплатформенность;

- вбудована безпека.

Недоліки EJB:

- складність інтеграції з додатками;

- погана масштабованість;

- низька продуктивність;

- відсутність міжнародної стандартизації.

DCOM є розподіленнj. програмну архітектуру від компанії Microsoft. З її допомогою програмний компонент одного комп'ютера може передавати повідомлення програмного компонента іншого комп'ютера, причому з'єднання встановлюється автоматично. Для надійної роботи вимагається забезпечити захищене з'єднання між пов'язаними компонентами, а також створити систему перенаправления трафіку.

Переваги DCOM:

- незалежність від мови;

- динамічне знаходження об'єктів;

- масштабованість;

- відкритий стандарт;

Недоліки DCOM:

- складність реалізації;

- залежність від платформи;

- пошук через службу Active Directory;

- відсутність іменування сервісів через URL.

Технологія CORBA розглядає усі застосування в розподіленій системі як набір об'єктів. Об'єкти можуть одночасно виступати в ролі клієнта і сервера, викликаючи методи інших об'єктів і відповідаючи на їх виклики. Застосування цієї технології дозволяє будувати системи, що перевершують по складності і гнучкості системи з архітектурою клієнт-сервер(як дворівневою, так і трирівневою).

Переваги COBRA:

- незалежність від платформи;

- незалежність від мови;

- динамічні виклики;

- динамічне виявлення об'єктів;

- масштабованість;

- індустріальна підтримка.

Недоліки:

- відсутність іменування по URL;

- практично повна відсутність реалізації CORBA- сервісів;

При грамотному підході до побудови архітектури комп’ютерної системи може знадобитися використання відразу декількох з розглянутих технологій. Кожна з них реалізовуватиме певний функціонал, який може зажадати також декількох типів платформеної архітектури. У одній системі можуть працювати декілька файл-серверов, декілька серверів додатків і декілька серверів баз даних. Таким чином можна розподіляти навантаження в системі або групувати набори сервісів по виконуваних функціях.