УДК 004.021
UDC 004.021
DOI: 10.33744/2308-6645-2024-1-58-183-189

АРХІТЕКТУРНІ ШАБЛОНИ І СТИЛІ. ПОЄДНАННЯ АРХІТЕКТУРНИХ СТИЛІВ

Ходаков Д.В., кандидат технічних наук, Національний авіаційний університет, Київ, Україна, daniil.khodakov@npp.nau.edu.ua, orcid.org/0009-0000-7970-151X

ARCHITECTURAL TEMPLATES AND STYLES. COMBINATION OF ARCHITECTURAL STYLES

Khodakov D.V., Ph.D. in Technical Science, National aviation university, Kyiv, Ukraine, daniil.khodakov@npp.nau.edu.ua, orcid.org/0009-0000-7970-151X

Постановка проблеми.

Дослідження архітектури програмного забезпечення намагається визначити як найкраще розбити систему на частини, як ці частини визначають та взаємодіють одна з одною, як між ними передається інформація, як ці частини розвиваються поодинці і як все вищеописане найкраще записати використовуючи формальну чи неформальну нотацію.

Аналіз останніх досліджень та публікацій.

Область комп'ютерних наук з моменту свого утворення зіткнулася з проблемами, пов'язаними зі складністю програмних систем. Розглядаючи дослідження, методи та засоби в області проектування, раніше проблеми складності вирішувалися розробниками шляхом правильного вибору структур даних, розробки алгоритмів і розмежуванню повноважень [3]. Хоча термін «архітектура програмного забезпечення» є відносно новим для індустрії розробки ПЗ, фундаментальні принципи цієї області невпорядковано застосовувалися піонерами розробки ПЗ починаючи з середини 1980-х. Перші спроби зрозуміти і пояснити програмну архітектуру системи були повні неточностей і страждали від нестачі організованості, часто це була просто діаграма з блоків, з'єднаних лініями. В 1990-ті роки спостерігається спроба визначити і систематизувати основні аспекти даної дисципліни [1]. Початковий набір шаблонів проєктування, архітектурних стилів, найкращих практик та мов опису були розроблені протягом цього часу [1,2]. Основною ідеєю дисципліни архітектури ПЗ є ідея зниження складності системи шляхом абстракції і розмежування повноважень. Досі немає згоди щодо чіткого визначення терміну «архітектура програмного забезпечення».

Будучи, наразі, дисципліною без чітких правил про «правильний» шлях створення системи, проєктування архітектури ПЗ є поєднанням науки і мистецтва. Аспект мистецтва полягає в тому, що будь-яка комерційна система передбачає наявність застосування або місії. Ключові цілі, які має система, описуються з допомогою сценаріїв як нефункціональні вимоги до системи, також відомі як атрибути якості, які визначають, як буде поводитись система. Атрибути якості системи включають в себе відмовостійкість, збереження зворотної сумісності, розширюваність, надійність, супроводжуваність, доступність, безпеку, зручність використання, а також інші якості [3]. З точки зору користувача програмної архітектури, програмна архітектура дає напрям руху і вирішення завдань, пов'язаних зі спеціальністю кожного такого користувача, наприклад, зацікавленої особи, розробника ПЗ, Служба технічної підтримки, спеціаліста по супроводу, спеціаліста по розгортанню ПЗ, тестера, а також кінцевих користувачів. У цьому сенсі архітектура програмного забезпечення насправді об'єднує різні точки зору на систему. Можливість системно об'єднати кілька різних точок зору при проектуванні системи є аргументом на захист необхідності і доцільності створення архітектури ще до етапу розробки ПЗ.

Головна ідея архітектури полягає у тому, щоб знизити складність сприйняття системи внаслідок розмежування повноважень та створення чіткої структури [2]. Архітектура та дизайн програмного забезпечення дозволяють створити чітку структуру, за якою зручно працювати програмістам. Від її якості залежить, наскільки просто проходитиме обслуговування ПЗ, його зміни, доповнення та підтримка.

Для безлічі завдань розробки архітектури застосовуються патерни чи, як їх ще називають, шаблони. Вони дозволяють вирішувати однотипні завдання швидшим способом, що позитивно впливає на час розробки та спрощує багато процесів. Паттерн описує певне завдання, і коли воно знову виникає у певному контексті, його застосування дозволяє "не винаходити велосипед", а скористатися готовим рішенням.

Виклад основного матеріалу.

Головне завдання та роль архітектури полягає в тому, щоб запобігти негативним наслідкам після розробки програми.

На основі наявного досвіду дослідниками і практиками розробки ПЗ вироблено деяку множину типових архітектур [2, 4]. 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.

Архітектура та проектування може мати атрибут, іменований стилем. Стиль дозволяє надати певну однаковість, визначається за допомогою вищеописаних шаблонів, а також конкретних з'єднувачів та компонентів. Ключова роль стилю – створити зрозумілу та цілісну архітектуру.

Стиль – це певний набір принципів, який дозволяє використовувати шаблони та зводити структуру до єдиного та простого для сприйняття виду. Це шлях до спільного розуміння та мови. Зокрема, якщо архітектуру розробляє одна команда розробників, а надалі зміни до функціоналу вносить інша команда фахівців, стиль – це зв'язуюча ланка, яка дозволяє зрозуміти структуру. Архітектурні стилі поділяються на п’ять груп (рис. 1).

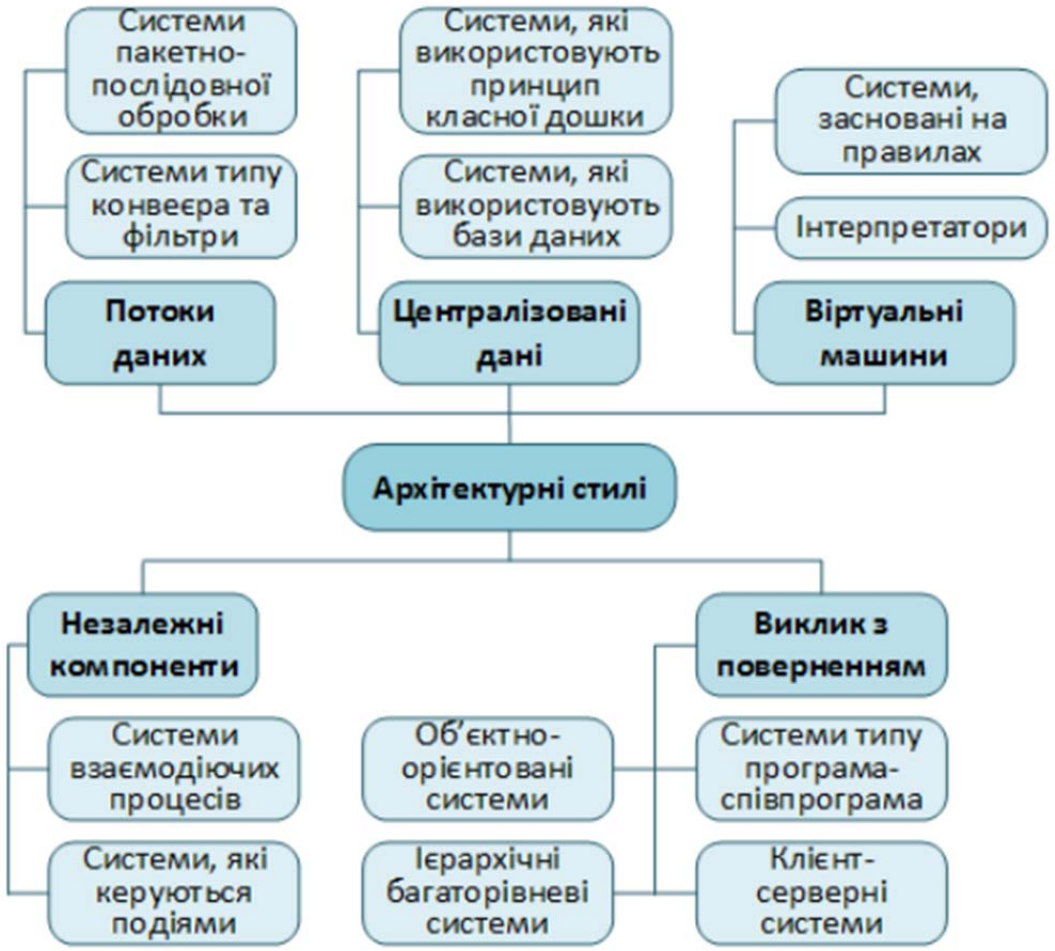

Рисунок 1 – Класифікація архітектурних стилів
Figure 1 – Classification of architectural styles

Архітектура та дизайн програмного забезпечення рідко обмежуються одним стилем. Як правило, використовується поєднання, яке створює повноцінну систему. Кожен стиль описується у технічній документації, щоб команда розробників змогла передати дані. Крім того, вимоги до системи безпеки зобов'язують використовувати різні стилі, зокрема із застосуванням багаторівневої архітектури.

На вибір архітектурного стилю впливає безліч факторів, включно із обмеженням інфраструктури середовища розробки, стеком технологій, досвідченістю розробників. Насамперед стиль визнача-

ється шляхом опрацювання виконуваних процесів. Враховується можливість подальшого масштабування програми та особливості впровадження її на підприємство клієнта.

Існують чотири основні типові інтеграційні підходи:

1) інтеграція на рівні даних;

2) інтеграція на рівні бізнес-функцій і бізнес-об'єктів;

3) інтеграція на рівні бізнес-процесів;

4) портали.

Інтеграція на рівні даних (Information- Oriented Integration), наявність в системах баз даних, для роботи з якими необхідно розробити єдиний програмний інтерфейс.

До основних технологічних рішень даного підходу відносяться [4]:

– системи реплікації даних;

– федеративні бази даних;

використання API для доступу до EPR- системам.

Реплікація є процесом синхронізації даних між різними джерелами. Необхідність у цьому виникає в момент зміни блоку інформації в розподілених системах зберігання для гарантії коректності і несуперечності даних, які використовуються в усіх модулях або додатках інформаційної системи. Зазвичай функції реплікації покладають на проміжне програмне забезпечення.

Федеративні бази даних (Federated Database Systems) надають єдиний інтерфейс до розподілених даних. Це забезпечує інтеграцію множини автономних даних, які можуть бути фізично розташовані на різних пристроях в мережі [1]. Такі бази даних прийнято називати віртуальними.

Використання API для доступу до ERP-систем покликане спростити механізми обміну інформацією між одними додатками і програмним забезпеченням, призначеним для управління функціонуванням виробничих інформаційних систем (ERP).

Інтеграція на рівні бізнес-функцій і бізнес об'єктів передбачає реалізацію спільно використовуваних служб (сервісів). Служба може бути набором функцій, використовуваному в декількох додатках. Набір служб і буде бізнес-функціями.

При використанні сервіс-орієнтованої архітектури, бізнес-функції можна розглядати як бізнес-сервіси, а при компонентному підході – бізнес-об'єктами (бізнес-компонентами).

Інтеграція на рівні бізнес-процесів розрізняється залежно від рівня інтеграції. При внутрішній інтеграції взаємодіє велика кількість сервісів, а при зовнішній інтеграції, в основному, два. Самі бізнес-процеси функціонують над виділеними службами, для управління якими існує спеціальний інтерпретована мова.

Портали можна вважати графічними інтерфейсами бізнес-процесів, оскільки вони призначені для персоналізованого доступу до інформації та консолідації даних з декількох джерел.

Головне призначення процесу інтеграції – об'єднання функцій додатків або модулів для надання нової функціональності.

При інтеграції додатків можна виділити два основних типи завдань:

– завдання інтеграції корпоративних додатків;

– завдання інтеграції додатків з різних інформаційних систем.

Для вирішення завдань першого типу застосовуються системи EAI (*Enterprise Application Integration*) (рис.2), які іноді називаються A2A (*Application-to-Application Integration*), а для вирішення завдань другого типу застосовуються системи B2B (*Business-to-Business Integration*). У деяких ситуаціях дуже складно визначити різницю між інтеграцією A2A і B2B, оскільки складність деяких рішень всередині інформаційних систем може перевищувати складність рішень для їх спільного функціонування.

Основні цілі інтеграції додатків можуть бути визначені таким чином:

– скоротити вартість використання сукупності додатків підприємства;

– підвищити швидкість виконання типових завдань чи забезпечити терміни виконання;

– поліпшити якість виконання завдань за допомогою формалізації процесів та мінімізації людського фактора як головного фактора помилок.

Забезпечення автоматизованого контролю проходження базових бізнес-процесів на підприємстві, інформаційна безпека при реалізації бізнес-процесів досягається засобами благополучної інтеграції корпоративних систем.

B2B-система поєднує в собі рішення для постачальників і для покупців, інтегруючи їх у єдину систему на базі центрального порталу. Залежно від типу торгового майданчика слід враховувати низку важливих аспектів, необхідних для успішної роботи:

– доступність нових учасників;

– масштабована і надійна платформа (поява нових учасників чи інші причини не повинні впливати на функціонування майданчика;

– управління інформацією (використання якісної інформації, а також своєчасне її оновлення є ключовим моментом у досягненні успіху);

– можливості інтеграції (для зручності користувачів майданчик повинен містити всі види електронної комерції);

– забезпечення безпеки;

– аналітика;

– додаткові послуги (наприклад, аукціони або інші фінансові послуги).

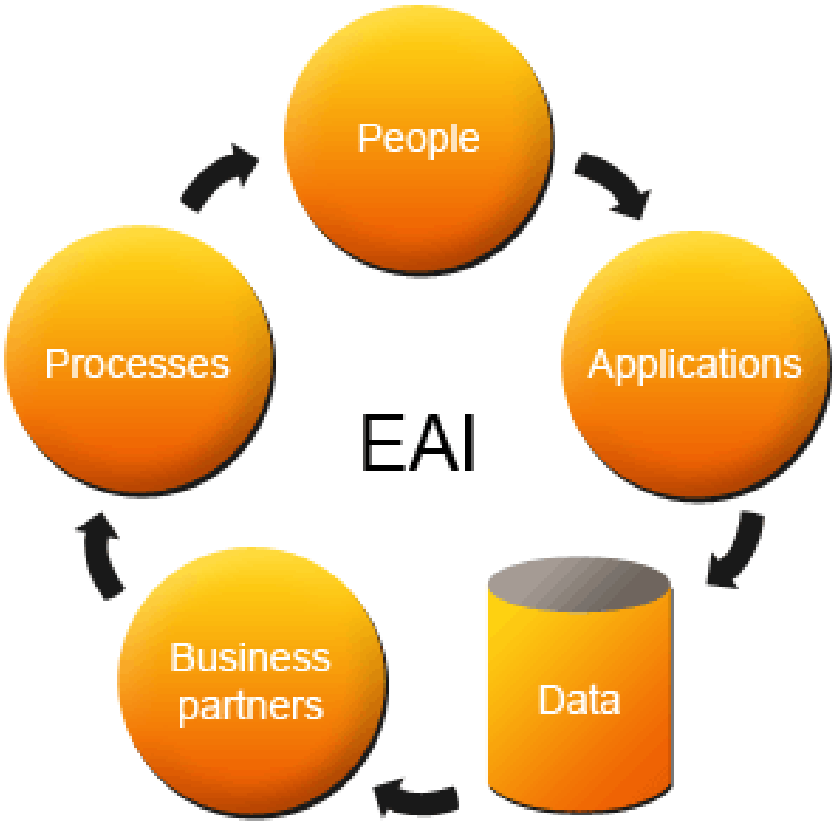

Рисунок 2 – Система Enterprise Application Integration
Figure 2 – Enterprise Application Integration system

Основним недоліком у даній системі ведення бізнесу все ж таки буде висока ціна за створення та підтримку B2B-майданчика.

Частина програмного забезпечення інфраструктури, що забезпечує конструкцію архітектури програмного забезпечення для надання базових послуг складних архітектур, називається технологія ESB (*Enterprise Service Bus*). EAI (*Enterprise Application Integration*) – це інтеграційна система, яка може бути використана для інтеграції набору комп'ютерних систем. EAI – це широке поняття, яке описує схеми інтеграції, а ESB – це технологія, яка забезпечує EAI.

Є деякі ключові відмінності між ESB і EAI. ESB – це частина інфраструктурного програмного забезпечення, яка допомагає розробникам розробляти служби та спілкуватися між службами через відповідні API, тоді як EAI – це інтеграційна структура для комп'ютерних програм у масштабах підприємства. Іншими словами, ESB діє як посередник між службами, тоді як EAI є моделлю концентратора для інтеграції. EAI – це концепція, яка описує всі типи шаблонів інтеграції, але ESB – це лише приклад технології, яка дає змогу EAI. Простіше кажучи, EAI – це концепція за кордоном, а ESB – це реалізація.

Визначається набір принципів інтеграції та забезпечується проміжне програмне забезпечення, що може складатися з поєднання технологій та послуг, які забезпечують інтеграцію декількох систем (рис.3). EAI займається зв'язком корпоративних програм, таких як управління ланцюгами поставок,

управління взаємовідносинами з клієнтами, інструменти BI (*Business Intelligence*), управління людськими ресурсами та охорона здоров'я, які зазвичай не подрібнюються між собою.

Таким чином, EAI може вирішити неефективність, спричинену відсутністю зв'язку між цими програмами. EAI може використовуватися в основному для трьох різних цілей. Вони являють собою інтеграцію даних для забезпечення узгодженості (також відомої як *Enterprise Information Integration* або EII), забезпечення незалежності постачальника та як загальний фасад для кластера програм.

Безпосередня короткострокова перевага підходу ESB полягає в тому, що він досягає такого ж загального ефекту, як підхід EAI (хаб-спиця), але при значно нижчій загальній вартості володіння. Ця економія реалізується не лише за рахунок зменшення витрат на апаратне та програмне забезпечення, але й за рахунок економії праці, яка реалізується завдяки використанню розподіленої та гнучкої структури.

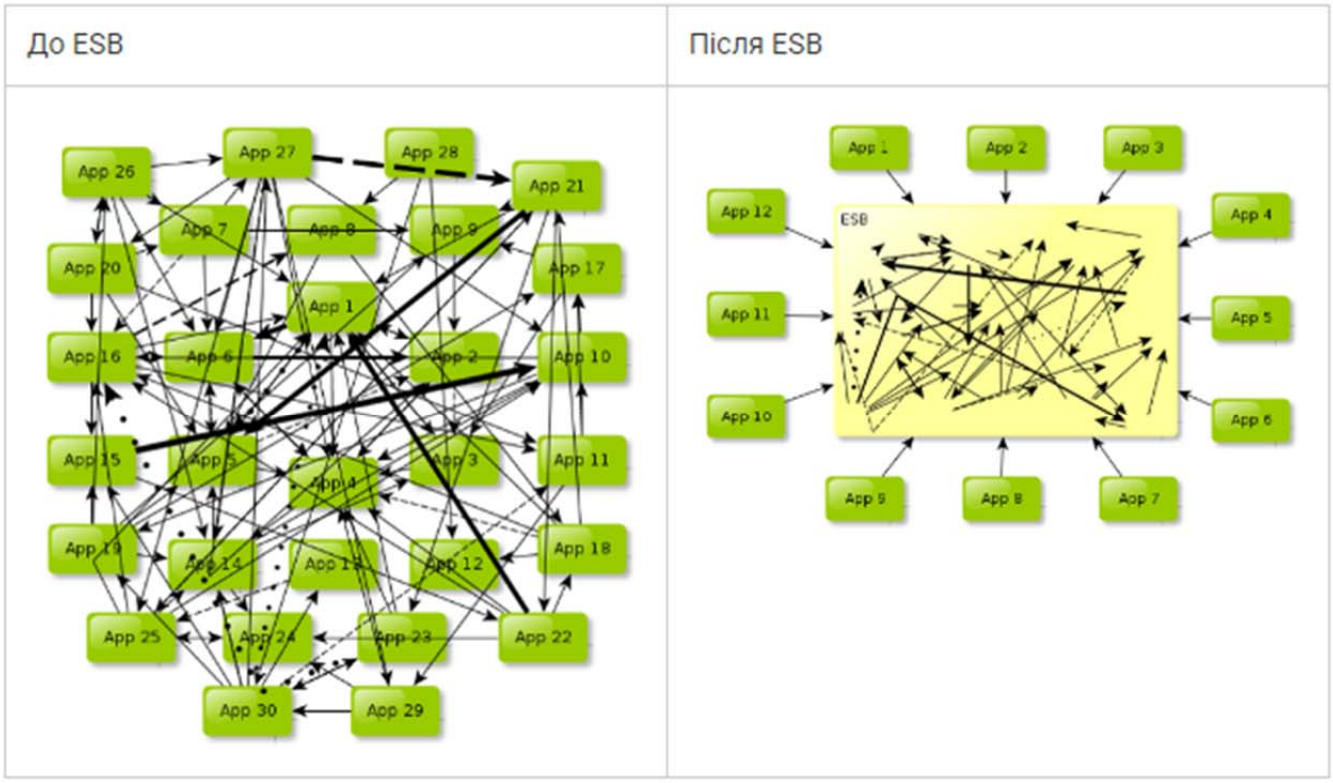

Рисунок 3 – Приклад використання технології ESB
Figure 3 – Example of using ESB technology

Висновки.

Головне завдання та роль архітектури полягає в тому, щоб запобігти негативним наслідкам після розробки програми. Якщо необхідно вносити редагування до функціоналу або інтерфейсу, зробити це без чіткої та зрозумілої архітектури вкрай важко. Імовірність зачепити пов'язані функції – вкрай висока.

Попри те, що архітектура та дизайн програмного забезпечення коштують компанії недешево, набагато дорожче коштуватиме подальше обслуговування програми та забезпечення працездатності функцій. Розробка архітектури ПЗ надає наступні переваги:

– повністю описує організацію програмного забезпечення та дозволяє бачити структуру кожного процесу;

– є єдиним та загальним базисом у разі розробки серії продуктів;

– знижує ризики та ймовірність помилок під час розробки;

– є формою першорядного виду готового робочого програмного забезпечення, яке можна відчувати як єдиний цілісний організм, тобто, ще на етапі розробки архітектури стає зрозуміло, чи працюватимуть ті чи інші функції;

– дозволяє повторно використовувати модулі, що значно прискорює процес розробки, знижує навантаження на код й робить застосунок швидшим;

– є єдиною системою для роботи в команді, ведення документації, містить термінологію, дозволяє команді розробників розуміти один одного й набагато швидше та ефективніше розробляти програмне забезпечення;

– дозволяє бачити можливості програми, оцінювати рівень масштабованості та методи подальшого розвитку системи.

Архітектура розробки програмного забезпечення виконує одну з ключових ролей під час розробки.

Для того, щоб побудувати правильну і надійну архітектуру і грамотно спроектувати інтеграцію програмних систем необхідно чітко слідувати сучасним стандартам в цих областях. Без цього велика ймовірність створити архітектуру, яка нездатна розвиватися і задовольняти зростаючим потребам користувачів IT. В якості законодавців стандартів у цій галузі виступають такі міжнародні організації як SEI (*Software Engineering Institute*), WWW (консорціум *World Wide Web*), OMG (*Object Management Group*), організація розробників Java - JCP (*Java Community Process*), IEEE (*Institute of Electrical and Electronics Engineers*) та інші.

ПЕРЕЛІК ПОСИЛАНЬ

1. Роберт Мартін, Чиста архітетура, Фабула, – К., 2019, – 368 с.
2. Neal Ford, Mark Richards, Fundamentals of Software Architecture: An Engineering Approach. A Comprehensive Guide to Patterns, Characteristics, and Best Practices, O'Reilly Media, – USA., 2020, – 400 p.
3. Лаврищева К. М., Підручник "Програмна інженерія". Режим доступу: https://web.archive.org/web/20120628103421/http://www.programsfactory.univ.kiev.ua/content/books/2
4. Architectural Styles and the Design of Network-based Software Architectures [Електронний ресурс] // UCI: [сайт]. [2000]. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

REFERENCES

1. Robert Martin, Clean Architecture, Fabyla, – K., 2019, – 368 p.
2. Neal Ford, Mark Richards, Fundamentals of Software Architecture: An Engineering Approach. A Comprehensive Guide to Patterns, Characteristics, and Best Practices, O'Reilly Media, – USA., 2020, – 400 p.
3. Лаврищева К. М., Підручник "Програмна інженерія". Режим доступу: https://web.archive.org/web/20120628103421/http://www.programsfactory.univ.kiev.ua/content/books/2
4. Architectural Styles and the Design of Network-based Software Architectures [Електронний ресурс] // UCI: [сайт]. [2000]. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

РЕФЕРАТ

Ходаков Д.В. Архітектурні шаблони і стилі. Поєднання архітектурних стилів / Д.В. Ходаков // Вісник Національного транспортного університету. Серія «Технічні науки». Науковий, науково-виробничий журнал. – К.: НТУ, 2024. – Вип. 1 (58).

Розглянуто дослідження в області проектування, раніше проблеми складності вирішувалися розробниками шляхом правильного вибору структур даних та розробки алгоритмів, частково розмежуванню повноважень. Проаналізовано архітектурні стилі, визначено набір типів компонентів системи і набір шаблонів взаємодій з передачі даних або управління. Для вирішення завдань першого типу застосовуються системи EAI (Enterprise Application Integration), які іноді називаються A2A (Application-to-Application Integration), а для вирішення завдань другого типу застосовуються системи B2B (Business-to-Business Integration). У деяких ситуаціях дуже складно визначити різницю між інтеграцією A2A і B2B, оскільки складність деяких рішень всередині інформаційних систем може перевищувати складність рішень для їх спільного функціонування. Проведено порівняння використання різних архітектурних стилів, які підходять для вирішення різних завдань в плані забезпечення нефункціональних вимог, хоча одну і ту ж функціональність можна реалізувати, використовуючи різні стилі. Запропоновано практичну модель поєднання, яка створює повноцінну систему. Безпосередня короткострокова перевага підходу ESB полягає в тому, що він досягає такого ж загального ефекту, як підхід EAI (хаб-спиця), але при значно нижчій загальній вартості володіння. Ця економія реалізується не лише за рахунок зменшення витрат на апаратне та програмне забезпечення, але й за рахунок економії праці, яка реалізується завдяки використанню розподіленої та гнучкої структури. Визначається набір принципів інтеграції та забезпечується проміжне програмне забезпечення, що може складатися з поєднання технологій та послуг, які забезпечують інтеграцію декількох систем. Архітектура розробки програмного забезпечення виконує одну з ключових ролей під час розробки.

КЛЮЧОВІ СЛОВА: АРХІТЕКТУРА ПЗ, ІНТЕГРАЦІЯ ПЗ, ПРОЕКТУВАННЯ ПЗ.

ABSTRACT

Khodakov D.V. Architectural templates and styles. Combination of architectural styles. Visnyk National Transport University. Series «Technical sciences». Scientific, scientific and industrial journal. – K.: NTU, 2024. – Issue 1 (58).

Research in the field of design is considered, previously the problems of complexity were solved by developers through the correct choice of data structures and the development of algorithms, partly by separation of powers. Architectural styles are analyzed, a set of types of system components and a set of patterns of data transfer or control interactions are defined. EAI (Enterprise Application Integration) systems, sometimes called A2A (Application-to-Application Integration), are used to solve the first type of tasks, and B2B (Business-to-Business Integration) systems are used to solve the second type of tasks. In some situations, it is very difficult to determine the difference between A2A and B2B integration, since the complexity of some solutions within information systems may exceed the complexity of solutions for their joint functioning. A comparison is made of the use of different architectural styles, which are suitable for solving different tasks in terms of ensuring non-functional requirements, although the same functionality can be implemented using different styles. A practical model of combination, which creates a full-fledged system, is proposed. The immediate short-term advantage of the ESB approach is that it achieves the same overall effect as the EAI (hub-and-spoke) approach, but at a significantly lower total cost of ownership. These savings are realized not only through reduced hardware and software costs, but also through labor savings realized through the use of a distributed and flexible structure. A set of integration principles is defined and middleware is provided, which can consist of a combination of technologies and services that enable the integration of multiple systems. Software development architecture plays one of the key roles during development.

KEYWORDS: SOFTWARE ARCHITECTURE, SOFTWARE INTEGRATION, SOFTWARE DESIGN.

АВТОР:

Ходаков Даніїл Віктович, кандидат технічних наук, Національний авіаційний університет, e-mail: daniil.khodakov@npp.nau.edu.ua, https://orcid.org/0009-0000-7970-151X, тел. +380444067098, Україна, 03058, м. Київ, проспект Любомира Гузара 1.

AUTHOR:

Khodakov Daniil Viktorovych, Ph.D. in Technical Science, National aviation university, e-mail: daniil.khodakov@npp.nau.edu.ua, https://orcid.org/0009-0000-7970-151X, ph. num.+380444067098, Ukraine, 03058, Kyiv, Lybomira Gyzara avenue 1.

РЕЦЕНЗЕНТИ:

Радішевський М.Ф., кандидат технічних наук, доцент кафедри інженерії програмного забезпечення, Національний авіаційний університет.

Куклінський М.В., кандидат технічних наук, доцент кафедри інженерії програмного забезпечення, Національний авіаційний університет.

REVIEWER:

Radishevskyi M.F., Candidate of Technical Sciences, Associate Professor of the Department of Software Engineering, National Aviation University.

Kuklinsky M.V., Candidate of Technical Sciences, Associate Professor of the Department of Software Engineering, National Aviation University.