Столыпинский вестник

Научная статья

Original article

УДК 004.4

ЭФФЕКТИВНОСТЬ ИСПОЛЬЗОВАНИЯ АРХИТЕКТУРНОГО ПОДХОДА ПРИ РАЗРАБОТКЕ ПРОГРАММНОГО ПРИЛОЖЕНИЯ

THE EFFECTIVENESS OF USING AN ARCHITECTURAL APPROACH IN THE DEVELOPMENT OF A SOFTWARE APPLICATION

Егорян В.В., Студент бакалавриата 3 курс, МИРЭА-Российский технологический университет (РТУ МИРЭА), 119454, Россия, г. Москва, проспект Вернадского, 78, Институт информационных технологий, Россия, г. Москва

Калугин А.В., Студент бакалавриата 3 курс, МИРЭА-Российский технологический университет (РТУ МИРЭА), 119454, Россия, г. Москва, проспект Вернадского, 78, Институт информационных технологий, Россия, г. Москва

Egoryan V.V., Undergraduate student 3rd year, MIREA-Russian Technological University (RTU MIREA), 119454, Russia, Moscow, Vernadsky Avenue, 78, Institute of Information Technology, Russia, Moscow

Kalugin A.V., Undergraduate student 3rd year, MIREA-Russian Technological University (RTU MIREA), 119454, Russia, Moscow, Vernadsky Avenue, 78, Institute of Information Technology, Russia, Moscow

Аннотация: На сегодняшний день архитектурный подход при разработке программного приложения не имеет эталонного шаблона или методики проектирования. Данный подход является результатом слияния науки системного подхода и искусства. Искусство в данном употреблении подразумевает собой возможность видоизменения существующих шаблонов архитектуры приложений и данных.

Abstract: To date, the architectural approach to developing a software application has no reference pattern or design methodology. This approach is the result of a merging of systems approach science and art. The art in this usage is the ability to modify existing application and data architecture patterns.

Ключевые слова: Архитектура, приложения, данные, подход.

Keywords: Architecture, application, data, approach.

Настоящая статья посвящена теме эффективности использования архитектурного подхода при разработке приложений и данных.

Понятие архитектуры обеспечивает структурное хранение данных, исключая риск возникновения утечек данных и несанкционированную обработку, подразумевает наличие шаблонов и методик, которые гарантируют бесперебойное и устойчивое функционирование приложения.

Целью исследования является изучение эффективности архитектурного подхода при разработке приложений.

Объектом исследования являются аспекты архитектуры приложений и данных.

Предмет исследования

Архитектурный подход и его эффективность при разработке программного приложения.

Практическая значимость исследования

Использование различных шаблонов архитектур позволяет оптимизировать пути взаимодействия между разработчиками приложения,

облегчить разработку программного приложения, разделить обязанности в команде разработчиков.

Матрица Джона Захмана

Так как у каждого разрабатываемого ИТ-продукта есть определенные задачи и преследуемые цели в результате проектирования архитектуры приложения получаются самые различные модели, в каждой из которых внимание может быть уделено конкретному слою или модулю информационной системы. Ярким примером может являться многоуровневая модель архитектуры приложений и данных, в которой количество и каналы передачи данных может быть подобно лабиринту.

Джон Захман был одним из первых ИТ-консультантов, кто совершил попытку создания систематизированного подхода к проектированию приложений и даных. В основе методики Джона Захмана заложена матрица моделирования архитектуры. В рамках матрицы Захмана архитектура предприятия рассматривается, как набор описательных моделей, которые в свою очередь имеют возможность развития в течении определенного периода времени [3]. Пример матрицы Захмана представлен на Рисунке 1.

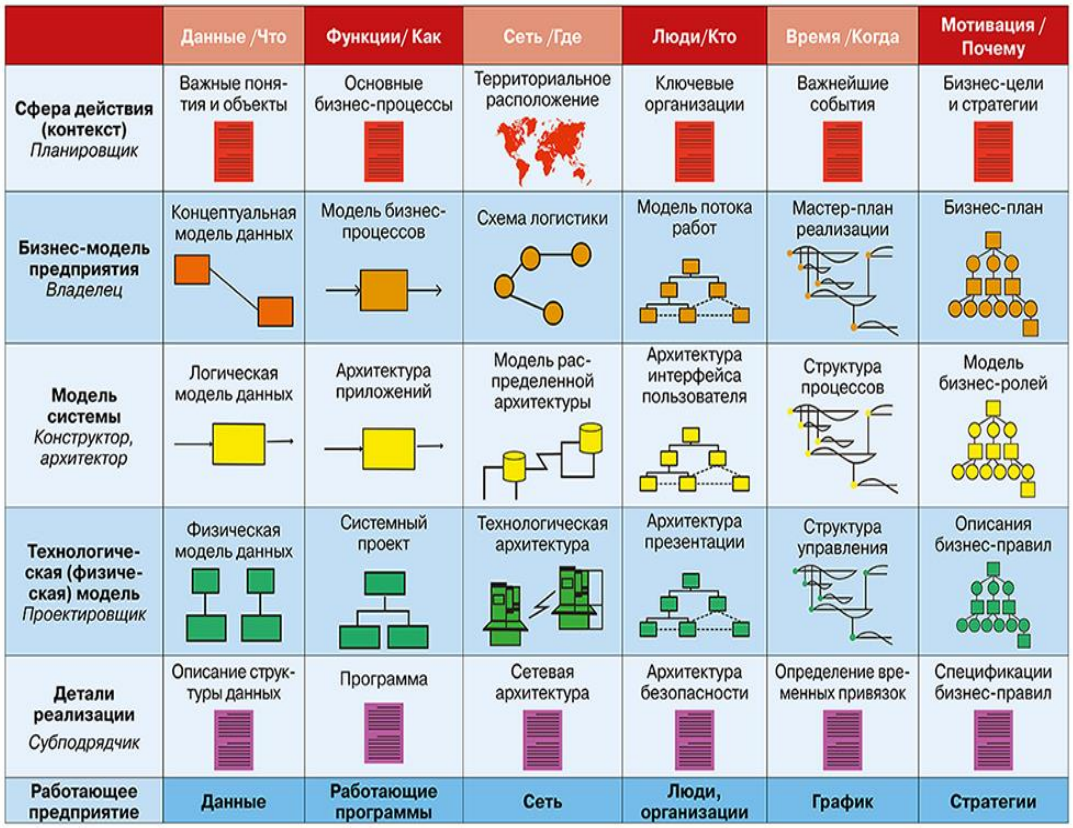

Рисунок 1 – Матрица Захмана

Методика Захмана решает проблему разобщенности и несогласованности данных. Джон Захман осветив проблему разобщенности поднял острую необходимость систематизированного подхода к разработке приложений, сервисов и ИТ-продуктов.

Для создания качественного ИТ-продукта и проектирования надежной архитектуры приложений и данных необходимо соблюдать принципы архитектурного подхода [4].

Принцип системности

Одним из основополагающих принципов разработки программных приложений является принцип системности. Он подразумевает четкое разграничение областей разработки и порядок разработки. Этот принцип обеспечивает разработать приложение постепенно и равномерно. При несоблюдении принципа могут произойти многочисленные ошибки во время разработки, которые потребуют начать разработку с начальной точки.

Принцип разделения задач

Принцип разделения задач так же является одним из основополагающих. Принцип подразумевает разделение программного обеспечения на отдельные компоненты, которые соответствуют выполняемым компонентами функциям. Так как форматируемые элементы отделены от функций форматирования, выбор элементов не должен зависеть от функций. На практике это определяет необходимость отдельно локализовать элементы бизнес-логики, функции пользовательского интерфейса и инфраструктуру приложения независимо друг от друга. Этот принцип обеспечивает легкость тестирования бизнес-модели приложения и простоту совершенствования бизнес-модели без тесной связи с более низкими уровнями сведениями реализации.

Инкапсуляция отдельных частей программного приложения обеспечивает изоляцию элементов и модулей друг от друга. Этот принцип позволяет корректировать внутренние элементы без влияния на модули-участников совместной деятельности. Также инкапсуляция позволяет производить замену элементов на альтернативные без потенциальной возможности нанести вред информационной системе.

Принцип инверсии зависимостей

Принцип инверсии зависимостей способствует корректному созданию слабо связанных программных приложений, так как детали реализации могут реализовывать и описывать зависимости абстракции более высокого уровня. Пример инвертированных зависимостей на примере процессов компиляции и работы программы представлены на Рисунке 2.

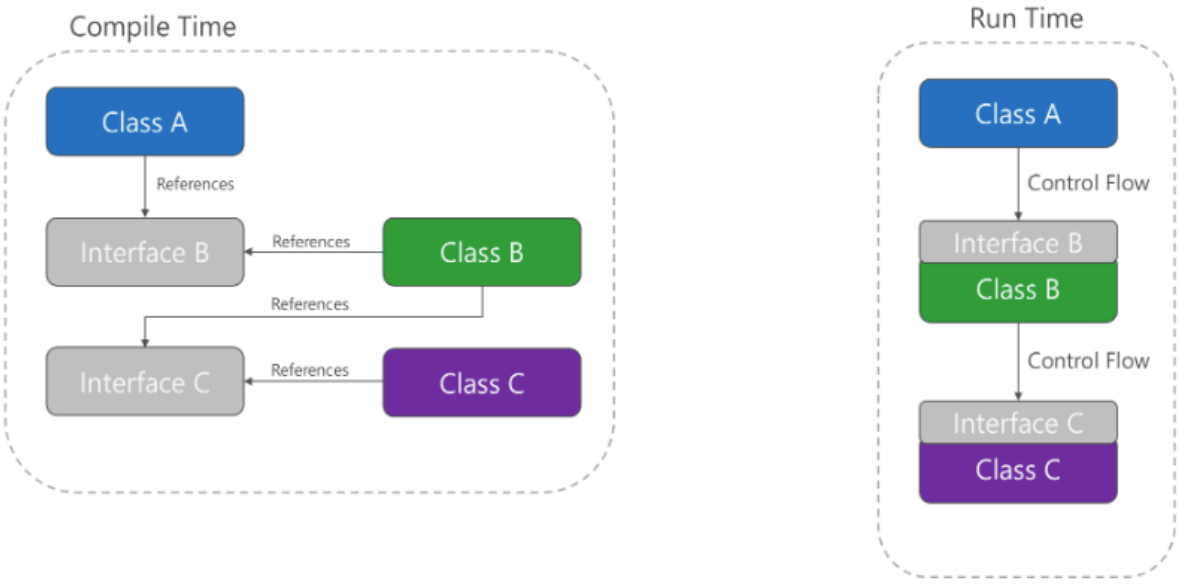

Рисунок 2 – Пример инвертированных зависимостей

В результате использования принципа инвертированных зависимостей достигается более высокий уровень тестируемости, удобства, обслуживания и модульности программного приложения.

Принцип единственной обязанности

Принцип единственной обязанности, который применяется к объектно-ориентированному проектированию возможен к рассмотрению, как независимый принцип проектирования архитектуры приложений и данных. Принцип единственной обязанности подразумевает наличие у объекта или элемента системы одной единственной обязанности внутри информационной системы и единственную причину для изменения. При соблюдении принципа добавление нового класса будет более безопаснее по сравнению с изменением существующих. Этот принцип лежит в основе микросервисной архитектуре приложений и данных. Соблюдение принципа способствует модульности и гибкости системы, а также легкость эксплуатации.

Принцип отсутствия повторений

Принцип отсутствия повторений подразумевает отказ от дублирования поведения отдельных элементов или модулей системы. Необходимо инкапсулировать логику поведения приложения в конструкции

программирования. Конструкция должна быть единственным исполнителем необходимого поведения для использования в различных частях приложения. Несоблюдение этого принципа может привести к несогласованности всей информационной системы или многочисленным ошибкам в работе программного приложения.

Принцип независимости сохраняемости

Принцип независимости сохраняемости позволяет сохранять состояние бизнес-модели различными способами, увеличить степень связанности между типами, для которых требуется сохраняемость и применяемой для данных целей технологии. Во время эксплуатации системы способы сохранения имеют возможность изменения. Соблюдение принципа независимости сохраняемости обеспечивает увеличение гибкости приложения и повышение уровня безопасности данных программного приложения.

Принцип ограниченного контекста

Принцип ограниченного контекста является центральным при применении проблемно-ориентированного проектирования. Соблюдение принципа позволяет решить проблему сложности масштабных приложений и организовать систему путем разбиения на отдельные концептуальные модели. Каждый модуль в рамках принципа представляет собой контекст, который отделен от остальных, его развитие происходит независимо от других отделенных частей системы. Взаимодействие между ограниченными контекстами происходит посредством программных интерфейсов, благодаря этому элементы бизнес-логики имеют возможность реагирования на происходящие изменения. Принцип ограниченного контекста является основополагающим для микросервисной архитектуре приложений и данных [5]. Пример ограниченных контекстов в рамках архитектуры представлен на Рисунке 3.

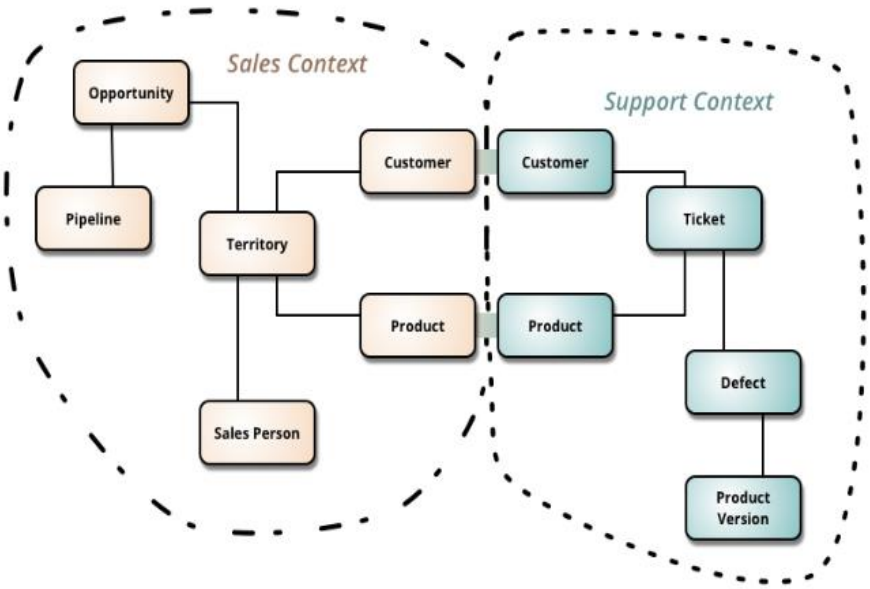

Рисунок 3 – Пример архитектуры с соблюдением принципа ограниченности контекстов

Архитектурный подход к разработке программных приложений играет одну из важнейших ролей при создании ИТ-продуктов. Главная задача подхода – предотвратить затруднения работы приложения на ранних этапах путем создания отдельной структурированной системы. Архитектура стандартизирует, регулирует все модули программного приложения, предопределяет структуру взаимосвязей, которые позволяют удобнее модернизировать и обслуживать приложение и данные. Архитектурный подход охватывает во внимание все важнейшие элементы приложения, обеспечивает высокую безопасность системы, масштабируемость, удобную эксплуатацию как для пользователей, так и для разработчиков, обеспечивает долговечность информационной системы. Существование систематизированного подхода внутри архитектурного обеспечивает распределить обязанности разработчиков разных областей программного приложения.

Можно сделать вывод, что архитектурный подход разработки приложений и данных крайне важен для разработки программного приложения. Архитектурный подход играет ключевую роль для обеспечения качественной

разработки устойчивого программного обеспечения. Использование подхода обеспечивает надежность и высокое качество продукта.

Список используемых источников

1. **Ю.П. Сурмин.** Теория систем и системный анализ: учебное пособие / Ю.П. Сурмин. - Киев : МАУП, 2003. - 368 с. - (Бакалавр. Академический курс). - ISBN 966-608-290-X. - URL: https://studfile.net/preview/996456/ (дата обращения: 15.05.2022) - Режим доступа: Файловый архив студентов StudFiles.net. - Текст: электронный.
2. ITELON. Архитектура «Клиент-Сервер» : [сайт] - URL: https://itelon.ru/blog/arkhitektura-klient-server/ (дата обращения: 22.05.2022) - Текст: электронный.
3. Архитектура ИТ-решений. Объясняем матрицу Захмана : [сайт] - URL: https://mxsmirnov.com/2018/03/23/zachman/ (дата обращения: 22.05.2022) - Текст: электронный.
4. Microsoft. Архитектурные принципы. : [сайт] - URL: https://docs.microsoft.com/ru-ru/dotnet/architecture/modern-web-apps-azure/architectural-principles (дата обращения: 22.05.2022) – Текст: электронный.
5. martinFowler.com Bounded Context : [сайт] - URL: https://martinfowler.com/bliki/BoundedContext.html (дата обращения: 22.05.2022) - Текст: электронный.
6. Хабр. Лучшая архитектура для MVP: монолит, SOA, микросервисы. : [сайт] - URL: https://habr.com/ru/company/otus/blog/477930/ (дата обращения: 22.05.2022) - Текст: электронный.
7. **Роберт М.** Чистая архитектура. Искусство разработки программного обеспечения: книга / М. Роберт. - Санкт-Петербург : Питер, 2018. - 352 с. - ISBN 978-5-4461-0772-8. - Текст: непосредственный.

List of sources used

1. Yu.P. Surmin. Theory of systems and system analysis: a textbook / Yu.P. Surmin. - Kiev : IAPM, 2003. - 368 p. - (Bachelor. Academic course). - ISBN 966-608-

290-X. - URL: https://studfile.net/preview/996456 / (accessed: 05/15/2022) - Access mode: Students' File Archive StudFiles.net . - Text: electronic.

2. ITELON. Client-Server Architecture : [site] - URL: https://itelon.ru/blog/arkhitektura-klient-server / (accessed: 05/22/2022) - Text: electronic.
3. Architecture of IT solutions. We explain the Zahman matrix: [site] - URL: https://mxsmirnov.com/2018/03/23/zachman / (accessed: 05/22/2022) - Text: electronic.
4. Microsoft. Architectural principles. : [site] - URL: https://docs.microsoft.com/ru-ru/dotnet/architecture/modern-web-apps-azure/architectural-principles (accessed: 05/22/2022) – Text: electronic.
5. martinFowler.com Bound Context : [site] - URL: https://martinfowler.com/bliki/BoundedContext.html (accessed: 05/22/2022) - Text: electronic.
6. Habr. The best architecture for MVP: monolith, SOA, microservices. : [website] - URL: https://habr.com/ru/company/otus/blog/477930 / (accessed: 05/22/2022) - Text: electronic.
7. Robert M. Pure Architecture. The Art of software development: a book / M. Robert. - St. Petersburg : Peter, 2018. - 352 p. - ISBN 978-5-4461-0772-8. - Text: direct.

Для цитирования: Егорян В.В., Калугин А.В., ЭФФЕКТИВНОСТЬ ИСПОЛЬЗОВАНИЯ АРХИТЕКТУРНОГО ПОДХОДА ПРИ РАЗРАБОТКЕ ПРОГРАММНОГО ПРИЛОЖЕНИЯ// Научный сетевой журнал «Столыпинский вестник» №9/2022