УДК 681.3

ПРОБЛЕМЫ СОЗДАНИЯ И ПОСТРОЕНИЯ АРХИТЕКТУРЫ ВЕБ-ПРИЛОЖЕНИЙ

© 2018 В. Н. Кострова, А. В. Гусев

Воронежский государственный технический университет (г. Воронеж, Россия)
ОАО Сбербанк России (г. Воронеж, Россия)

В статье рассматриваются вопросы создания и построения архитектуры веб-приложений. Указаны основные принципы ее создания. Приведены основные рекомендуемые области применения архитектурных стилей.

Ключевые слова: сетевые технологии, архитектура, веб-приложение.

Архитектура ПО должна описывать использование или взаимодействие главных компонентов и элементов в приложении. Поиск подходящих алгоритмов обработки и структур данных или деталей реализации тех или иных компонентов – это вопросы проектирования. Архитектурные вопросы, и вопросы проектирования часто пересекаются. Наиболее рационально может быть комбинирование этих двух областей, а не их разграничение. Иногда решения, принимаемые по ПО, являются по своей сути архитектурными, в других же случаях решения в большей степени затрагивают оба вопроса.

При разработке архитектуры необходимо учесть такие вопросы, как:

– варианты использования приложения пользователем;

– развертывание и обслуживание ПО при эксплуатации;

– выдвинутые требования по безопасности, производительности, возможности параллельной обработки, интернационализации и конфигурации;

– гибкость модификации и удобство обслуживания в долгосрочной перспективе;

– влияние, которое может оказывать на ПО основные архитектурные направления в будущем после его развертывания или сейчас.

Цель архитектуры – это выявление технических и бизнес-требований, оказывающих влияние на структуру приложения. Качественная архитектура помогает снизить бизнес-риски, которые связаны с созданием технического решения. Также качественная структура имеет значительную гибкость, которая позволяет ей справиться с естественным развитием технологий в областях пользовательских сценариев и требований, оборудования, ПО. Архитектору необходимо учитывать совместный результат от принятых проектных решений, непременно присутствующих компромиссов между атрибутами качества (такими, как, например, безопасность и производительность) и компромиссов, необходимых для выполнения системных, пользовательских, и бизнес-требований. Хорошая архитектура должна скрывать детальную реализацию, но раскрывать структуру системы. В ней должны быть учтены и реализованы все возможные сценарии и варианты использования. Она должна в разумных пределах отвечать требованиям всех заинтересованных сторон и выполнять требования по качеству и функциональности.

При выборе архитектурных решений важно учитывать современные и будущие тенденции и тренды. При проектировании архитектуры необходимо принять во внимание, что дизайн будет изменятся со временем. Как правило, он изменяется и дополняется в процессе реализации, при тестировании на требования реального окружения. Создавая архитектуру, нужно ориентироваться на такие изменения, для возможности адаптации их к требованиям, которые не были известны в начале процесса проектирования.

Рекомендуется создавать открытую для изменений архитектуру и определять:

– фундаментальные части архитектуры, ошибки в реализации которых несут наибольшие риски;

– части, которые вероятнее всего подвергнутся модификации, и возможность от-

Кострова Вера Николаевна – Воронежский государственный технический университет, д. т. н., профессор, kostr_vn01u7@yandex.ru.
Гусев Алексей Витальевич – ОАО Сбербанк России, специалист, gggussev_6man20@yandex.ru.

ложить реализацию этих частей на более поздние этапы разработки;

– каким образом будут проверяться допущения;

– условия, при которых, вероятно придется изменять дизайн.

Основные принципы при разработке архитектуры:

– создание, для изменения (необходимо учесть, как со временем может потребоваться модернизировать приложение, для того что бы оно соответствовало вновь появившимся задачам и требованиям и предусмотреть гибкость приложения, для соответствия новым требованиям);

– создание модели для снижения рисков и анализа (рекомендовано использование средств проектирования, систем моделирования, таких как UML (Unified Modeling Language), средств визуализации для выявления требований и принятия архитектурных и проектных решений и анализа их последствий);

– использование моделей и визуализации для общения и обмена информации (это поможет ускорить процесс проектирования и разработки и упростить принятие решений о вносимых изменениях);

– выявление ключевых инженерных решений (выделение достаточного количества времени, в начале проекта, на ключевые инженерные решения, где наиболее вероятны ошибки, поможет создать более гибкий дизайн и в дальнейшем при внесении изменений с меньшей вероятностью потребует его полной переработки).

При разработке архитектуры, необходимо оценить возможность использования инкрементного и итеративного подходов. Начиная с базовой архитектуры, и воссоздавая полную картину, далее прорабатывать различные варианты в процессе итеративного тестирования и доработки. Усложнение дизайна должно происходить постепенно, за счет многократных пересмотров, для того чтобы быть уверенным, в правильности принятых наиболее ключевых решений и лишь после этого сосредотачиваться на деталях. Наиболее частой ошибкой является переход к деталям без грамотно выработанных ключевых решений, из-за ошибок в допущениях или неспособности корректно оценить свою архитектуру.

Обобщенные принципы и шаблоны, используемые при создании и проектировании ПО, обычно называют архитектурными стилями или шаблонами. В литературе можно встретить следующее определение архитектурного стиля:

«Архитектурный стиль определяется как семейство систем, с точки зрения схемы организации структуры. Более конкретно, архитектурный стиль определяет совокупность компонентов и соединений, которые могут использоваться в экземплярах этого стиля, вместе с набором ограничений на их комбинирование. Они могут включать топологические ограничения архитектурных решений (например, отсутствие циклов). Другие ограничения, например, связанные с необходимостью обрабатывать семантику выполнения».

Компонентная архитектура, в которой описывается подход к разработке и проектированию системы с использованием методов проектирования ПО. В этом подходе разделение дизайна на отдельные логические или функциональные компоненты, относящиеся к четко определенным интерфейсам, содержащим методы, свойства и события, представляется особо важным и создает более высокий уровень абстракции, при сравнении с объектно-ориентированным стилем, также не концентрируется внимание на вопросах общего состояния или протоколах связи. Основные принципы – это использование компонентов, обладающих такими характеристиками, как замещаемость; возможность повторного использования; расширяемость; независимость от среды и контекста; независимость от других компонентов; инкапсуляция. Применяется чаще всего при создании компонентов пользовательского интерфейса; также при создании ресурсоемких компонентов доступ, к которым осуществляется не часто, а активация выполняется «на лету»; и для создания компонентов с очередью вызовов методов, которые могут асинхронно выполняться благодаря применению очереди сообщений, для пересылки и хранения. К преимуществам подхода можно отнести: простоту разработки; простоту развертывания; упрощение системы с технической точки зрения; меньшую стоимость разработки и обслуживания; возможность повторного использования.

Многоуровневая архитектура группирует связанную функциональность ПО в различных слоях, выстраивая их вертикально друг над другом, функциональность объединяется по общей ответственности или роли. При этом слои слабо связаны между собой и между ними осуществляют обмен данными. Слои приложения могут физиче-

ски располагаться на одном компьютере или на разных. Общие принципы: инкапсуляция; абстракция; возможность повторного использования слоя; высокая связанность внутри слоя; слабая связанность между слоями; четкое разделение функциональности. Примеры таких приложений: системы бухгалтерского учета, веб приложения или веб сайты, приложения, использующие централизованные сервера приложений для бизнес-логики.

N-уровневая/3-х уровневая архитектура – это стили развертывания, описывающая деление функциональности на сегменты, как и в многослойной архитектуре, но в данном случае эти сегменты могут располагаться на различных компьютерах, их называют уровнями. Как правило, для связи используются методы платформы, а не сообщений. К характеристикам N-уровневой архитектуры ПО можно отнести: сервисные компоненты и их распределенное развертывание (обеспечивает масштабируемость и доступность), эффективность и управляемость использования ресурсов функциональную декомпозицию приложения. Независимость уровней от остальных, за исключением тех, с которыми он непосредственно соприкасается: n-ному уровню необходимо только знать, как обработать запрос от n+1 уровня, как передать этот запрос на n–1 уровень (при наличии), и как обработать результаты запроса. Связь между уровнями, как правило, асинхронная, для лучшей масштабируемости. К преимуществам можно отнести: масштабируемость; гибкость; доступность; удобство поддержки. Примеры приложений: Веб-приложение с высокими требованиями к безопасности, насыщенный клиент.

Клиент/серверная архитектура дает описание распределенным системам, состоящим из отдельных сервера и клиента и сети, которая их соединяет. Примерами приложения могут быть: веб-приложение, выполняемое во внутренних сетях компании или в Интернет, настольные приложения, работающие с удаленными ресурсами. Основные преимущества: простота обслуживания, большая безопасность, централизованный доступ к данным, к минусам можно отнести сложность расширяемости, масштабирования и зависимость от центрального сервера.

Основанная на шине сообщений архитектура описывает вариант использования программной системы, в которой отправляются и принимаются сообщения по одному или нескольким каналам связи, позволяя взаимодействовать приложениям без детальных знаний друг о друге. Взаимодействие реализуется путем передачи сообщений через шину, как правило, асинхронной. Типично использование маршрутизатора сообщений или шаблона Публикация/Подписка (Publish/Subscribe) и системы обмена сообщениями, такие как Очередь сообщений (Message Queuing). Шина позволяет обеспечить обработку: основанные на сообщениях взаимодействия; сложной логики обработки; изменение логики обработки; интеграцию с различными инфраструктурами. Основные преимущества: гибкость, расширяемость, слабое связывание, масштабируемость, простота приложений, невысокая сложность.

Сервисно-ориентированная архитектура (COA, SOA) дает возможность создавать приложения использующие программные сервисы и позволяет предоставлять функциональность ПО в виде набора сервисов. Сервисы слабо связаны благодаря использованию основанных на стандартах интерфейсов, которые можно вызвать опубликовать и обнаружить. COA позволяет упаковать бизнес-процессы в сервисы, поддерживающие возможность взаимодействия и использования различных форматов данных и протоколов. Основные принципы: Совместимость основана на политике, сервисы слабо связаны, автономны, совместно используют контракт, схему, но не класс, сервисы могут быть распределены. Преимущества: абстракция; возможность обнаружения и автоматического подключения через интерфейс, согласование предметных областей, рационализация, возможность взаимодействия.

Объектно-ориентированная архитектура, основана на разделение ответственности системы или приложения на пригодные для повторного использования объекты, каждый из которых содержит поведение и данные относящиеся к объекту. При данном подходе система рассматривается как набор взаимодействующих объектов, а не набор подпрограмм и процедурных команд. Основные принципы: наследование, инкапсуляция, композиция, абстракция, отделение, полиморфизм. Стиль используется для описания объектов, имеющих место в реальной жизни или для описания моделей поддерживающих сложные финансовые или научные операции. Основные плюсы: возможность повторного использования с помощью полиморфизма или абстракции, понятность,

улучшенная тестируемость, расширяемость, высокая связность.

Следует принять во внимание, что различные шаблоны могут использоваться в разных аспектах описания ПО. Некоторые из шаблонов применяются при описании схем развертывания, связи, другие при описании структуры или дизайна. Как правило, при разработке стандартного приложения применяется несколько различных архитектурных шаблонов.

Также особо рекомендуется сочетать архитектурные стили при построении web-приложений, так как это дает возможность эффективно разделить функциональность при применении многослойного архитектурного шаблона. Это дает возможность отделить бизнес-логику от логики доступа к данным. Трех и более уровневое развертывание приложения могут диктоваться требованиями безопасности компании. В этом случае можно разместить в пограничной сети, расположенной между внешней и внутренней сетью компании, уровень представления. В этом случае можно реализовать СОА и сделать связь между сервисом приложений и веб посредством обмена сообщениями. Или использовать в качестве модели взаимодействия на уровне представления такой шаблон как Модель-представление-контроллер (МПК) (Model-View-Controller (MVC).. Ниже приведена таблица, указывающая стили и основные области их применения (табл. 1).

Таблица

Основные рекомендуемые области применения архитектурных стилей

Архитектурный стиль	**Область применения**	**Примеры применения стиля**
Клиент/сервер	Развертывание ПО	1. Настольное приложение с графическим интерфейсом, обменивающееся данными с сетевыми сервисами данных. 2. Инструменты и утилиты для работы с удаленными системами. 3. Веб-приложение, выполняющееся в Интернет или внутренних сетях компании. 4. Приложения, выполняющие доступ к удаленным хранилищам данных (FTP, e-mail, и т. п.). 5. Серверы приложений, и др.
N-уровневая / 3-уровневая	Развертывание ПО	1. Типовое финансовое веб-приложение с высокими требованиями к безопасности. 2. Насыщенный клиент, где слой доступа данных и бизнес слой находятся на одном и более серверном уровне, а слой представления развернут на клиентских компьютерах.
Многослойная (многоуровневая архитектура	Структура ПО	1. Бизнес-приложения (line-of-business, LOB), например, системы управления заказчиками или бухгалтерским учетом. 2.Web-сайты и web-приложения компаний; Смарт или настольные клиенты компаний, использующие централизованные серверы приложений при размещении бизнес-логики.
Многослойная (многоуровневая архитектура	Структура ПО	2. Бизнес-приложения (line-of-business, LOB), например, системы управления заказчиками или бухгалтерским учетом. 3. Web-сайты и web-приложения компаний; Смарт или настольные клиенты компаний использующие централизованные серверы приложений при размещении бизнес-логики.
Компонентная архитектура	Структура ПО	1. Объектная модель распределенных программных компонентов (distributed component object model, DCOM) и объектная модель программных компонентов (component object model, COM) и в Windows. 2. Серверные компоненты Java (Enterprise JavaBeans, EJB) и общая архитектура брокера объектных запросов (Common Object Request Broker Architecture, CORBA) на других платформах.

Продолжение таблицы

Архитектурный стиль	Область применения	Примеры применения стиля
Объектно-ориентированная	Структура ПО	1. Создание приложения на базе реальных действий или объектов. 2. Инкапсуляция логики и данных в компоненты, для повторного использования. 3. Приложения со сложной бизнес-логикой в которых необходима динамическое поведение и абстракция.
Сервисно-ориентированная архитектура (SOA)	Связь	1. Приложения обеспечивающие совместное использование информации или выполняющие многоэтапные процессы (например, интернет-магазины или системы резервирования отелей). 2. Разработка составных приложений, объединяющих данные из большого количества источников. 3. Предоставление специализированных сервисов ил данных между различными компаниями.
Шина сообщений	Связь	1. Реализация решений, требующих взаимодействие с приложениями, находящимися в разных средах или внешними приложениями. 2. Подключение к шине несколько экземпляров одного приложения 3. Приложения с облачными сервисами Позволяет обрабатывать: 4. Интеграция с различными решениями 5. Изменение логики обработки 6. Сложная логика обработки 7. Основанное на сообщениях взаимодействие
Дизайн на основе предметной области	Предметная область	1. Работа со сложной предметной областью, для улучшения понимания и обмена информацией разработчиками 2. Представление проекта на понятном всем заинтересованным лицам языке 3. Сценарии обработки больших объемов сложных данных компании, с которыми возникают сложности при использовании других подходов

ЛИТЕРАТУРА

1. Ермолова В. В. Методика построения семантической объектной модели / В. В. Ермолова, Ю. П. Преображенский // Вестник Воронежского института высоких технологий. – 2012. – № 9. – С. 87-90.

2. Иванов М. С. Разработка алгоритма отсечения деревьев / М. С. Иванов, Ю. П. Преображенский // Вестник Воронежского института высоких технологий. – 2008.– № 3. – С. 031-032.

3. Львович И. Я. Особенности проектирования корпоративных компьютерных сетей / И. Я. Львович, А. П. Преображенский, О. Н. Чопоров // Оптимизация и моделирование в автоматизированных системах. Материалы всероссийской молодежной научной школы. Министерство образования и науки РФ, Воронежский государственный технический университет, Российский фонд фундаментальных исследований. – 2017. – С. 16-20.

4. Львович И. Я. Применение информационных технологий в медицинской сфере / И. Я. Львович, А. П. Преображенский, О. Н. Чопоров // Интеллектуальные информационные системы. Материалы всероссийской конференции с международным участием. – 2017. – С. 164-165.

5. Львович И. Я. Основы информатики / И. Я. Львович, Ю. П. Преображенский, В. В. Ермолова / учебное пособие, Воронеж, Издательство: Воронежский институт высоких технологий. – 2014. – 339 с.

6. Львович Я. Е. Проблемы построения корпоративных информационных систем на основе web-сервисов / Я. Е. Львович, И. Я. Львович, Н. В. Волкова // Вестник Воронежского государственного технического университета. – 2011. – Т. 7. – № 6. – С. 8-10.

7. Максимов И. Б. Принципы формирования автоматизированных рабочих мест / И. Б. Максимов // Вестник Воронежского института высоких технологий. – 2014. – № 12. – С. 130-135.

8. Максимов И. Б. Классификация автоматизированных рабочих мест / И. Б. Максимов // Вестник Воронежского института высоких технологий. – 2014. – № 2. – С. 127-129.

9. Мэн Ц. Анализ методов классификации информации в интернете при решении задач информационного поиска / Ц. Мэн // Моделирование, оптимизация и информационные технологии. – 2016. – № 2 (13). – С. 19.

10. Паневин Р. Ю. Реализация транслятора имитационно-семантического моделирования / Р. Ю. Паневин, Ю. П. Преображенский // Вестник Воронежского института высоких технологий. – 2009. – № 5. – С. 057-060.

THE PROBLEMS OF CREATION AND ARCHITECTURE FOR WEB-APPLICATIONS

© 2018 V. N. Kostrova, A. V. Gusev

Voronezh state technical university (Voronezh, Russia)
OJSC Sberbank of Russia (Voronezh, Russia)

The paper deals with the issues of creating and building the architecture of web applications. The basic principles of creation of such architecture are specified. The main recommended fields of application of architectural styles are given.

Key words: network technologies, architecture, web application.