Лекция 4.
Концепции СОА

SOA - Service-Oriented Architecture

Проблемы интеграции в современных системах обработки данных и исторические пути их решения [3]

Интеграция программных средств - задача, возможно, наиболее сложная, поскольку именно программные средства обеспечивают интеграцию всех остальных составляющих.

На протяжении всей истории информационных технологий одним из передних фронтов была борьба за повторную используемость (reusability) программного кода. Первым средством, реализующим идею повторной используемости, были процедуры и функции - сначала встроенные, а затем оформляемые как модули. Модуль представляет собой единицу программного кода с хорошо описанным интерфейсом, отдельно компилирующуюся и отдельно хранящуюся. Интеграция модулей в единое приложение обеспечивается механизмами вызова процедур в языках программирования и механизмами сборки в загрузчиках и редакторах связей. Разработка модулей, реализующих типовые технологические и прикладные функции, значительно облегчает труд программиста, уменьшает объем кодирования и ускоряет процесс разработки и внедрения программного приложения, что является немаловажным конкурентным преимуществом. Развитые библиотеки модулей, предоставляющие почти исчерпывающий набор средств для решения определенных классов технологических или прикладных задач, получили название каркасов (framework). Применение каркасов в идеале превращает создание нового приложения из программирования в сборку программы из готовых элементов каркаса.

Следующим значительным этапом борьбе за повторную используемость явилась компонентная архитектура. Компонент - это модуль, реализующий стандартный интерфейс. Стандартизация интерфейса обеспечивает взаимозаменяемость компонентов и, что самое важное, взаимодействие компонента с промежуточным программным обеспечением - каркасами, инструментальными средствами разработки, контейнерами. Именно последнее обстоятельство кардинальным образом меняет процессы разработки и функционирования приложений, значительно ускоряя их создание и внедрение. Контейнер представляет собой некоторое промежуточное программное обеспечение, которое поддерживает выполнение компонентов. Говорят, что, реализуя стандартный интерфейс, компонент заключает контракт с контейнером: компонент «обязуется» обеспечивать некоторое определенное поведение (выполняет стандартные методы), а контейнер «за это» предоставляет ему низкоуровневую поддержку, такую как управление жизненным циклом компонента (определенную последовательность вызова его

методов), безопасность, прозрачную для компонентов связь между ними, управление транзакциями и т.д., и т.п. Выполнение контейнером низкоуровневых технологических функций избавляет программиста от их кодирования и позволяет ему сосредоточить свои усилия на прикладной задаче. Особую значимость контрактные отношения между компонентами и контейнером приобретают в технологиях распределенных приложений. Части распределенного приложении, выполняющиеся в различных узлах распределенной системы, являются компонентами, и их функционирование и взаимодействие поддерживается соответствующим контейнером. Интеграционную функцию, таким образом, выполняет контейнер, поддерживающий тот или иной тип контракта. Наиболее развитыми платформами распределенных приложений являются Microsoft .NET и Java 2 Enterprise Edition (J2EE). Последняя базируется на открытых стандартах и не зависит от аппаратной платформы и операционной среды. Основным производителем средств технологии .NET является, естественно, фирма Microsoft. Что же касается средств технологии J2EE, то в силу открытости ее стандартов, спектр производителей этих средств весьма широк. Лидерами являются фирмы Oracle, IBM, Sun, Red Hat, многие средства этой технологии распространяются свободно или имеют открытый код.

Хотя компонентная архитектура обеспечивает быструю и удобную интеграцию компонентов, это относится только к компонентам, которые были разработаны в соответствии со спецификациями данной платформы. Однако современные требования рынка чрезвычайно ужесточают требования к срокам создания и внедрения систем и приложений информационных технологий. Начиная с середины 1990-х годов отмечается множество случаев, когда разработка и реализация проекта информационной системы «с нуля» заканчивались крахом по той причине, что, во-первых, затраты на разработку не окупались, во-вторых, к моменту завершения проекта условия, на основании которых были составлены спецификации проекта, существенно изменялись. Необходимость быстрого реагирования предприятий на изменяющиеся условия привели фирму IBM к концепции бизнеса по требованию. По определению IBM, бизнес по требованию (on demand business) - это предприятие, бизнес-процессы которого тесно интегрированы внутри всей компании, а также с бизнес-процессами партнеров, поставщиков, потребителей, что дает предприятию возможность гибко и быстро реагировать на изменяющиеся требования покупателей, конъюнктуру рынка и внешние воздействия. Информационные системы, поддерживающие бизнес по требованию, должны своей реактивностью соответствовать требованиям того бизнеса, который они обслуживают. Интеграция процессов внутри предприятия подразумевает, что отдельные процессы в предприятии уже автоматизированы, причем при этом могли использоваться разные языки программирования, операционные системы и платформы промежуточного программного обеспечения. Разработка всех или некоторой части приложений заново в соответствии с единой принятой платформой окажется слишком долгим и дорогим делом.

Предприятие должно в максимальной степени использовать имеющиеся активы информационных технологий, чтобы снизить затраты и уменьшить сроки внедрения. Также при разработке новых или модификации имеющихся приложений целесообразно в максимальной степени использовать уже имеющиеся компоненты. При интеграции же с информационными системами партнеров, поставщиков, потребителей и т.д. приведение всех компонентов системы к единой платформе просто невозможно. Необходимость решения этих проблем породила концепцию сервисно-ориентированной архитектуры.

Программные средства, реализующие компоненты бизнес-процессов, оформляются как сервисы. Сервис - это программный компонент, реализующий законченную функцию предоставления или обработки данных, переводя их из одного целостного состояния в другое, целостное же. Основным отличием сервиса от обычного компонента является стандартный и платформенно-независимый интерфейс. Клиент, обращающийся к сервису, не обязан ничего знать о подробностях реализации сервиса: на каком языке и в какой модели программирования он создан, на каких аппаратных средствах, в какой операционной среде, на какой платформе промежуточного программного обеспечения он выполняется. Сервисно-ориентированная архитектура, таким образом, позволяет компоновать бизнес-процессы из компонентов, выполняющихся на разных платформах (корпоративных бинов J2EE, компонентов .NET, отдельных приложений), представлять в виде сервисов и повторно использовать в новых бизнес-процессах унаследованные компоненты. В последнем случае сами унаследование компоненты не изменяются, но для них только делается оболочка, адаптирующая их интерфейсы к стандартам сервисов.

Обратите внимание: оформление процесса как сервиса не предъявляет каких-либо требований к разработке процесса, а только требования к его интерфейсу. Чтобы успешно выполнять свои интегрирующие функции, платформа, воплощающая сервисно-ориентированную архитектуру, должна соответствовать следующим требованиям:

- обеспечивать спецификации интерфейса, которые были бы приняты, если не всеми, то большинством разработчиков компонентов;
- использовать общепринятые протоколы для взаимодействия сервисов и клиентов;
- не использовать в интерфейсе какие-либо сложные и/или закрытые форматы представления информации;
- не требовать для своей поддержки дорогостоящего или ресурсоемкого программного обеспечения.

Определение СОА

Существует много взглядов на концепцию SOA. В наиболее простом виде СОА — это

подход к построению программных систем, который концентрируется на том, *как* программное обеспечение создается [1].

Ниже представлены распространенные примеры определения СОА. Они выбраны из разных источников и интересны тем, что иллюстрируют различные представления и взгляды на то, что такое СОА, однако все они вместе дают общий смысл:

- *Бизнес-определение*. Набор бизнес-методов, методов процесса, организационных методов, методов управления, предназначенных для уменьшения или устранения возможности невыполнения информационными технологиями своих функций и для колличественного изменения ценности IT для бизнеса в ходе создания гибкой бизнес-среды для получения преимуществ в конкурентной борьбе.
- *Еще одно бизнес-определение (предложено IBM).* СОА предлагает возможность гибкой работы с элементами бизнес-процессов и лежащей в их основе IT-инфраструктурой как с безопасными, стандартизованными компонентами (службами), которые можно использовать многократно и комбинировать при изменении приоритетов бизнеса.
- *Самое широкое техническое определение*. IT-архитектура в масштабе предприятия, которая обеспечивает слабые связи, многократное использование и взаимодействие между системами.
- *Довольно сложное техническое определение*. Архитектура приложений, в которой все функции и службы определены при помощи языка описаний и имеют интерфейсы вызова, обращения к которым осуществляются при выполнении бизнес-процессов. Каждое взаимодействие является независимым от всех прочих взаимодействий и внутренних протоколов, обеспечивающих соединение устройств. Поскольку интерфейсы независимы от платформы, клиент может использовать службу, применяя любое устройство, используя любую операционную систему и любой язык.
- *Определение — наименьший общий знаменатель.* Системная архитектура, в которой функции приложений создаются в форме компонентов (служб), которые имеют слабые связи и четко определены с целью совместимости, повышения гибкости и возможности многократного использования.
- *Самое узкое определение*. СОА — это синоним таких архитектур решений, которые используют технологии Web-служб, такие как SOAP, WSDL и UDDI.

Концепции СОА [2]

1. Сервис

Существует множество определений сервиса:

- Сервис - это функция, являющаяся четко определенной, самодостаточной и не зависящей от

контекста или состояния других сервисов.

- Сервис - это программный компонент, реализующий законченную функцию предоставления или обработки данных, переводя их из одного целостного состояния в другое, целостное же.

2. Концепция слабого связывания

В парадигме СОА концепция слабого связывания проявляет себя следующим образом:

- Оно помогает организовать уровень абстракции между производителями и потребителями сервисов.
- Оно способствует реализации гибкости в изменении реализации сервисов без воздействия на потребителей сервисов.
- В архитектуре SOA функциональность организуется как набор модульных повторно используемых общих сервисов. Эти сервисы имеют четко определенные интерфейсы, инкапсулирующие ключевые правила доступа к ним. Они также строятся без каких-либо допущений о том, кто будет использовать или потреблять эти сервисы. Таким образом, они слабо связаны с потребителями сервисов.

3. Открытые стандарты

Архитектура SOA основывается на открытых стандартах и поддерживает платформенно-независимую бизнес-интеграцию, но она нуждается в общей платформе, на которой будет базироваться ее инфраструктура. Эта инфраструктура должна поддерживаться всеми участвующими сторонами и, чтобы служить основой для взаимопонимания. В центре этой инфраструктуры находится технология XML. Тому есть целый ряд причин:

- XML является фундаментом практически всех стандартов Web-сервисов, в том числе XML Schema, SOAP, WSDL (Web Services Description Language) и UDDI (Universal Description, Discovery, and Integration). Эти стандарты опираются на основополагающую концепцию основанных на XML представлений - поддерживаемый во всем мире формат обмена информацией между провайдерами сервисов и инициаторами запросов в SOA.
- Использование XML решает проблему работы с различными форматами данных в различных приложениях, работающих на разных платформах.
- Преимущество XML заключается в простоте представления, являющегося по своей природе текстовым, гибким и расширяемым.

Примеры стандартов, основанных на XML и используемых в SOA:

- SOAP. Этот простой основанный на XML протокол позволяет приложениям обмениваться информацией по транспортным протоколам, таким как HTTP. Благодаря использованию

XML протокол SOAP является:

o Платформенно-независимым.

o Пригодным для использования в Интернете.

o Читабельным, структурированным и текстовым.

Благодаря всем этим преимуществам SOAP является рекомендованным и самым широко используемым коммуникационным протоколом для Web-сервисов. А так как Web-сервисы являются краеугольным камнем архитектуры SOA, этот протокол является также основным коммуникационным протоколом для основанных на SOA решениях.

- WSDL. Это документ, написанный на XML и описывающий Web-сервис. Он определяет месторасположение сервиса и отображаемые им операции (или методы), позволяющие обращаться к этому сервису. WSDL-файл описывает четыре главные вещи:

o Сервисы, доступные через интерфейс Web-сервиса, такие как список имен методов и сообщений-атрибутов.

o Тип данных сообщений.

o Информация о связывании для транспортного протокола, такого как HTTP и JMS.

o Адрес сервиса, используемый для его вызова.

- Electronic Business using eXtensible Markup Language (ebXML). Язык ebXML является стандартным способом определения бизнес-транзакций, которые могут выполняться между различными бизнес-субъектами. ebXML определяет стандартные методы для обмена сообщениями, устанавливая торговые связи и регистрируя бизнес-процессы между компаниями.

4. Реестры сервисов

Реестр сервисов представляет собой каталог сервисов, доступных в системе СОА. Он содержит физическое месторасположение сервисов, версии и срок действия сервисов, документацию по сервисам и стратегии. Реестр сервисов является одним из основных строительных блоков архитектуры SOA. Роль реестра сервисов:

Реестр сервисов реализует обещанное архитектурой SOA слабое связывание. Храня месторасположения оконечных точек сервисов, он устраняет тесное связывание, приводящее к жесткой привязке потребителя к провайдеру. Он также облегчает потенциальные сложности замены одной реализации сервиса другой при необходимости.

Реестр сервисов является хорошо масштабируемым; он развивается в соответствии с развитием системы, которую обслуживает.

- Реестр сервисов позволяет системным аналитикам исследовать корпоративный портфель бизнес-сервисов. Исходя из этого они могут определить, какие сервисы доступны для

автоматизации процессов с целью удовлетворения актуальных бизнес-потребностей, а какие нет. Это в свою очередь позволяет узнать, что нужно реализовать и добавить в портфель, формируя каталог доступных сервисов.

- Реестр сервисов может выполнять функцию управления сервисами, обязывая подписывающиеся сервисы быть согласованными. Это помогает гарантировать целостность руководства сервисами и стратегий.
- Видимость доступных сервисов и их интерфейсов ускоряет разработку, способствует повторному использованию приложений, совершенствует руководство и улучшает бизнес-планирование и управление. Отсутствие реестра сервисов приводит к избыточности и неэффективности.
- Реестры сервисов помогают уменьшить время, затрачиваемое на обнаружение информации о сервисе.
- Без реестра, следящего за сервисами и их взаимоотношениями, COAA-среда не только утрачивает согласованность и контроль, но и приходит в состояние хаоса.

5. Бизнес-процессы

Определение бизнес-процесса было дано в первой лекции. В парадигме COA бизнес-процесс управляет потоком сервисов. Бизнес-процесс управляет потоком событий, вызывает и координирует сервисы и создает контекст для их взаимодействия. Бизнес-процесс представляет собой бизнес-абстракцию; будучи отделенным от реализации сервисов, бизнес-процесс заботится о ходе бизнес-деятельности. Такое разделение задач не только позволяет больше сконцентрироваться на создании процесса, но и облегчает изменение процесса в соответствии с новыми требованиями без необходимости изменения применяемых реализаций сервисов.

6. Ориентация на стандарты

Как правило, проекты COA всецело полагаются на стандарты и используют их из-за следующих основных преимуществ:

- Стандарты гарантируют возможность взаимодействия систем и партнеров.
- Использование стандартов ускоряет разработку и доставку посредством процессов и инструментальных средств.
- Стандарты улучшают управляемость и видимость информационных активов.
- Стандарты гарантируют качество сервиса (quality of service - QoS).
- Стандарты улучшают гибкость благодаря снижению зависимостей от конкретных реализаций.

Рассмотрим несколько примеров стандартов, применяемых в SOA, и покажем, как они

способствуют реализации обещанных этой архитектурой преимуществ.

WS-Security

Протокол WS-Security основан на добавлении к заголовку сообщения SOAP-расширений для хранения метаданных системы защиты, предназначенных для реализации механизмов поддержки целостности сообщения, конфиденциальности и аутентификации. Эти расширения дают универсальный механизм ассоциирования маркеров системы защиты с сообщением вместо фиксированного защитного механизма. Базовая платформа поддерживает различные механизмы защиты. Протокол является расширяемым.

BPEL4WS

Язык BPEL4WS (Business Process Execution Language for Web Services) определяется на сайте сообщества OASIS по BPEL следующим образом: «Этот протокол определяет модель и грамматику для описания поведения бизнес-процесса на основе взаимодействий между процессом и его партнерами. Он также определяет, как координируются взаимодействия между сервисом и его партнерами для достижения поставленной бизнес-цели, а также задает состояние и логику для этой координации».

Протокол BPEL4WS вводит необходимые методы для работы с исключительными ситуациями и сбоями, а также способы корректировки других зафиксированных процессов, которые могут потребовать отката в случае возникновения ошибок. Поскольку BPEL должен поддерживаться везде, он основывается на общепризнанном протоколе WSDL, который, в свою очередь, основан на технологии XML.

Базовая архитектура СОА [2]

Базовая архитектура SOA состоит из провайдера сервисов, сервиса и (необязательного) каталога сервисов (рис. 1). Для обмена информацией используется механизм обмена сообщениями типа «приложение к приложению».

Рисунок 1. Базовая архитектура СОА

- Провайдер сервиса. Предоставляет сервисы, контракт по активизации которых и месторасположение опубликованы.

- Потребитель сервиса. Потребляет сервисы, соответствующие его бизнес-потребностям и обнаруженные в каталоге сервисов.
- Каталог сервисов. Служит для публикации и ведения списка сервисов, доступных для потребителей.

Сходство между этой моделью и чистыми Web-сервисами совершенно очевидно, поскольку в обоих случаях применяется WSDL-документ, являющийся контрактом по активизации, хранящимся в каталоге сервисов, из которого этот сервис может быть запрошен и извлечен посредством механизма UDDI. Web-сервисы в действительности являются реализацией архитектуры SOA на самом базовом уровне.

В этой модели базовый сценарий таков. Сначала провайдер сервиса создает сервис, принимает решение открыть этот сервис и публикует его. Публикация выполняется путем отправки информации о сервисе в каталог сервисов. С другой стороны, инициатор запросов сервиса (service requester), нуждаясь в определенном сервисе, просматривает каталог сервисов в поисках того из них, который удовлетворяет необходимому критерию. После обнаружения такого сервиса и использования доступной в каталоге сервисов информации инициатор запросов сервиса может напрямую обратиться к провайдеру сервисов надлежащим способом для удовлетворения бизнес-потребности.

Эталлоная модель СОА от OASIS

Организация по стандартизации OASIS утвердила эталонную модель сервис-ориентированной архитектуры Reference Model for Service-Oriented Architecture 1.0 (SOA-RM). Модель призвана ввести четкую техническую терминологию SOA для разработчиков и архитекторов. Работавший над моделью технический комитет OASIS определяет SOA-RM как «абстрактную базу для понимания основных объектов и взаимосвязей между ними в сервис-ориентированной среде и для разработки согласованных стандартов и спецификаций, поддерживающих такую среду. Модель унифицирует концепции SOA и может быть использована архитекторами для разработки SOA или в обучении SOA». В комитете отмечают также, что SOA-RM не связана непосредственно ни с какими стандартами, технологиями или другими деталями конкретных реализаций. Цель модели — предоставить общую семантику, которая снимет все двусмысленности в различных реализациях SOA. Однако, по мнению аналитиков ZapThink, SOA-RM будет мало пригодна для разработки. Эталонная модель от OASIS, считают в ZapThink, поможет архитекторам координировать отдельные проекты по SOA в организации или планировать создание единой корпоративной архитектуры, но абстрактность ее концепций препятствует применению SOA-RM в конкретных реализациях сервис-ориентированной архитектуры.

Литература

1. Биберштейн Н., Боуз С., Джонс К., Фиаммант М., Ша Р. Компас в мире сервис-ориентированной архитектуры (SOA): ценность для бизнеса, планирование и план развития предприятия / Пер. с англ. - М.: КУДИЦ-ПРЕСС, 2007. - 256 с.
2. Мохаммед И. Мабрук. Краткие основы SOA [Элект. ресурс] Метод доступа: http://www.ibm.com/developerworks/ru/edu/ws-soa-ibmcertified/index.html
3. Технологии и средства консолидации информации: Учебное пособие. Деревянко А.С., Солощук М.Н. - Харьков: НТУ "ХПИ", 2008. - 432с.