Занятие 5. Сервис-ориентированная архитектура.

XML и SOA

Архитектура SOA основывается на открытых стандартах и поддерживает платформенно-независимую бизнес-интеграцию, но она нуждается в общей технологии представления данных, 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:
    - Платформенно-независимым.
    - Пригодным для использования в Интернете.
    - 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:
    - Сервисы, доступные через интерфейс web-сервиса, такие как список имен методов и сообщений-атрибутов.
    - Тип данных сообщений.
    - Адрес сервиса, используемый для его вызова.

Реестры сервисов

Реестр сервисов представляет собой каталог сервисов, доступных в системе SOA. Он содержит физическое месторасположение сервисов, версии и их срок действия, а также

документацию по сервисам. Реестр сервисов является одним из основных строительных блоков архитектуры SOA. Его роль описывается ниже:

- Реестр сервисов реализует 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 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 (governance) сервисами и стратегий. Дополнительная информация о руководстве и важности SOA приводится далее в данном руководстве.

Что такое бизнес-процесс?

Термин *бизнес-процесс* часто употребляется в среде SOA. Ниже приведены два определения бизнес-процесса:

Цитата из статьи "Бизнес-процессы и поток работ в мире web-сервисов" (developerWorks, январь 2003):

"Бизнес-процесс модет быть определен как набор взаимосвязанных задач, относящихся к деятельности, имеющей функциональные границы. Бизнес-процессы имеют начальные и конечные точки и являются повторяемыми ".

Примером бизнес-процесса является выдача служебного пропуска. Для инициации процесса вы предоставляете свидетельство о рождении, резюме, а также фотографию. Затем заводится личное дело, проводится проверка и после этого выдается пропуск.

В парадигме SOA бизнес-процесс управляет потоком сервисов. Бизнес-процесс управляет потоком событий, вызывает и координирует сервисы и создает контекст для их взаимодействия. Бизнес-процесс, будучи отделенным от реализации сервисов, заботится о ходе деятельности. Такое разделение задач позволяет больше сконцентрироваться на создании процесса и облегчает изменение процесса в соответствии с новыми требованиями.

Элементы бизнес-процесса

Возможно, лучше определить бизнес-процесс в понятиях составляющих его элементов; это позволяет взглянуть на бизнес-процесс с технической точки зрения.

- **Входные данные (input)** - информация, необходимая процессу для формирования результата. В примере с пропуском входными данными могут быть ваши резюме, свидетельство о рождении и фотография.

- **Выходные данные (output)** - все данные и информация, сгенерированные процессом. Выходные данные представляют собой бизнес-цели и показатели, необходимые для бизнес-деятельности. В примере с пропуском выходными данными могли бы быть личное дело и готовый пропуск, а также показатели работы процесса.

- **События (events)** - уведомления о возникновении чего-либо важного, например, визуальная индикация. Они могут возникать до, во время и после выполнения процесса. В примере с пропуском событием могло бы быть предоставление нового документа, который ранее отсутствовал и который должен быть включен в личное дело.

- **Подпроцесс (subprocess)** - более мелкий процесс или этап в рамках процесса. Подпроцесс используется тогда, когда невозможно представить объем работы одним набором действий. Он имеет те же элементы, что и процесс. В примере с пропуском подпроцессом могло бы быть исследование вашего досье и получение результатов.

- **Действие (activity)** - наименьший элемент работы в процессе. В нашем примере действием могло бы быть создание нового личного дела для человека, получающего пропуск.

- **Показатели производительности (performance metrics)** - атрибуты, представляющие эффективность процесса для определения его соответствия необходимой производительности. Эти показатели помогают определить производительность и сравнить ее с требующимися значениями. Они также выделяют потенциальные области совершенствования процесса, реализуя в конечном итоге цикл улучшений, обещанный архитектурой SOA. В примере с пропуском эти показатели могли бы определять, выполнение какой части процесса занимает больше всего времени либо приводит к достижению пика загрузки. Они помогают дальнейшему совершенствованию процесса.

Как SOA управляет транзакциями?

Поскольку в процесс вовлечено множество действий, бизнес-транзакции, возникающие в среде SOA, могут быть очень сложными. Причиной этого является природа сервисов долго выполняющихся процессов в контексте SOA, которые часто являются асинхронными, не сохраняющими состояния, распределенными и непрозрачными.

Web-сервисы являются отличным представлением сервисов в среде SOA. Будучи самодостаточными (согласно требованиям SOA), они являются ограниченными с точки зрения межсервисных транзакций. Поскольку сервис находится на вершине транзакции, а область действия транзакции ограничена действиями, выполняемыми логикой сервиса, нет необходимости в реализации функциональности межсервисных транзакций, а сами транзакции могут управляться любой закрытой технологией (компонентной, традиционной или какой-либо другой), которая инкапсулирована в сервисе. Но с ростом числа сервисов увеличивается необходимость распространения транзакций на несколько сервисов.

Для решения проблемы транзакций был разработан ряд спецификаций web-сервисов. К ним относятся:

- **WS-Coordination**. Позволяет зарегистрированным процессам принимать участие в создании общего контекста, ответственного за хранение текущих данных и распространяемой между ними информации. Координация обработки существующих транзакций, потоков работ и других систем осуществляется интегрированной средой. Это позволяет скрыть проприетарные протоколы и работать в гетерогенной среде. Этот протокол обеспечивает инфраструктуру для других протоколов, таких как WS-AtomicTransaction или WS-BusinessActivity.

- **WS-AtomicTransaction**. Используется в краткосрочных распределенных действиях. Предоставляет три типа протоколов, которые могут использоваться с интегрированной средой WS-Coordination для реализации транзакций с двухфазной фиксацией типа ACID (транзакций, поддерживающих атомарность, согласованность, изоляцию и устойчивость).

- **WS-BusinessActivity**. Этот протокол используется с долго работающими транзакциями.

Основы архитектуры SOA

Теперь рассмотрим некоторые более сложные технические аспекты, такие как роль корпоративной сервисной шины (enterprise service bus - ESB), бизнес-процессы и их хореография (choreography), а также роль web-сервисов.

Составляющие базовой архитектуры SOA

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

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.

Рисунок 1. Базовая архитектура SOA

Ниже приведены определения некоторых понятий, используемых в данном разделе:

- **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.

Роль ESB в архитектуре SOA

Enterprise Service Bus (сервисная шина предприятия) — подход к построению распределѐнных корпоративных информационных систем. Обычно включает в себя промежуточное ПО, которое обеспечивает взаимосвязь между различными приложениями по различным протоколам взаимодействия.

Существует некоторое разногласие, что именно считать ESB — архитектуру или программное обеспечение. Обе точки зрения имеют право на существование.

Архитектура ESB заключается во взаимодействии всех приложений через единую точку, которая, при необходимости, обеспечивает транзакции, преобразование данных, сохранность обращений. Данный подход обеспечивает большую гибкость, простоту масштабирования и переноса. При замене одного приложения подключенного к шине нет необходимости перенастраивать остальные.

Конкретные реализации ESB содержат в себе адаптеры для соединения с другим ПО.

Среди популярных можно назвать SAP NetWeaver XI/PI (Exchange Infrastructure/Process Integration) от SAP AG, BizTalk от Microsoft, WebSphere от IBM, JBoss — опенсорсный продукт, поддерживаемый RedHat.

На EclipseCon 2009 было объявлено о выходе первой версии Eclipse Swordfish ESB.

ESB играет важную роль в архитектуре SOA. По сути, она предоставляет магистральную сеть и инфраструктуру для соединения провайдеров и потребителей сервисов.

Роли ESB в информационной системе:

1. Предоставляет интеграционную инфраструктуру, соответствующую принципам SOA:
    1.1. Устанавливает явные независимые от реализации интерфейсы для организации слабого связывания.
    1.2. Использует коммуникационные протоколы, независимые от расположения взаимодействующих сторон.
    1.3. Способствует определению сервисов, инкапсулирующих повторно используемые бизнес-функции.
2. Предоставляет средства для управления инфраструктурой сервисов.
3. Функционирует в распределенной гетерогенной среде через поддержку синхронных и асинхронных взаимодействий, а также использование стандартных интерфейсов.
4. Централизует управление и распределяет обработку.
5. Реализует защиту и обеспечение качества сервиса в проектах SOA.

Недостатки ESB:

- Требует достаточно больших трудозатрат и специфических знаний для реализации, при этом сама по себе (без дальнейшей реализации SOA) практически не приносит ощутимой пользы для бизнеса;
- По сравнению с простейшей (точка-точка) интеграцией между системами, вносит задержки, связанные с преобразованием XML сообщений.
- Требует тщательного продумывания и контроля над версионностью сообщений, в противном случае может увеличить связность систем друг с другом (при недостаточной унификации сообщений);

Оркестровка и хореография

ИТ организаций должны адаптироваться под требования клиентов и условия рынка. Существующие языки описания бизнес-процессов не поддерживают напрямую веб-сервисы, что заставляет организации разрабатывать собственные проприетарные протоколы компоновки сервисов. Стандарты оркестровки и хореографии позволяют решать эти задачи.

http://searchsoa.techtarget.com. SOA orchestration and choreography

Организация OMG определяет оркестровку как «моделирование направленных, внутренних бизнес-процессов», а хореографию как «спецификацию взаимодействий между автономными процессами».

Оркестровка в бизнес-процессах – это серия действий в управляемом потоке работ, обычно имеющем одну линию выполнения.

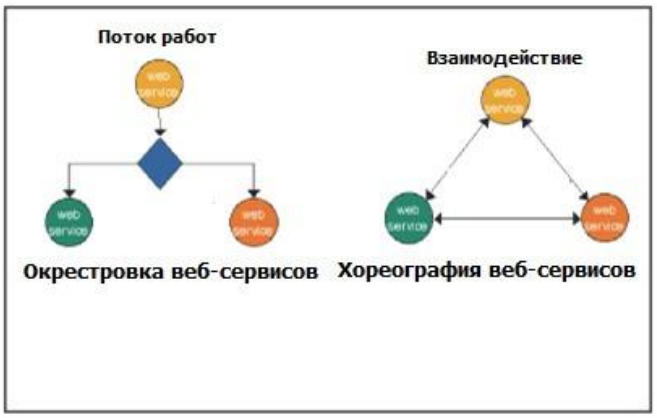
Хореография отражает видимый обмен сообщениями, правила взаимодействий и соглашения между двумя и более сервисами.

Ключевые элементы проектирования:

Для оркестровки: участник и его роль, переменные и свойства, определяющие взаимодействие участников, обработчики ошибок, события.

Для хореографии: структура сообщений, асинхронная и синхронная коммуникация сервисов, служебные сообщения.

Крис Пельтц – «Оркестровка и хореография веб-сервисов»



Стандарты WSCI (Web Service Choreography Interface) и BPEL4WS(Business Process Execution Language for Web Services) разработаны для облегчения взаимодействия веб-сервисов.

Оркестровка отличается от хореографии тем, что она описывает процесс, протекающий между сервисами, контролируемый основным участником. В хореографии нет участника, ведущего обмен сообщениями.

Технические требования для оркестровки и хореографии.

Определим требования к оркестровке и хореографии как к инфраструктуре СОА:

1. ***Гибкость***. Достигается разделением между логикой процесса и веб-сервисами. Достигается реализацией логики процесса с помощью оркестровки.
2. ***Простые и структурированные действия***. Язык оркестровки должен поддерживать действия как для обращения к другим веб-сервисам, так и для описания семантики процесса. Простое действие можно рассматривать как компонент, взаимодействующий с чем-то вне процесса, в то время как структурированное действие управляет общим выполнением процесса, специфицируя состав и порядок действий.
3. ***Рекурсивная композиция***. Отдельный бизнес-процесс может взаимодействовать с множеством веб-сервисов. Сам процесс может быть представлен как веб-сервис, для агрегации в процесс более высокого уровня.

Дополнительно, оркестровка и хореография предъявляют требования к целостности и стабильности взаимодействий. Они включают:

1. Хранение состояний и корреляция запросов. Способность хранить состояние между запросами веб-сервисов особенно важно, когда работа ведется с асинхронными сервисами. Язык и инфраструктура должны обеспечивать хранение данных и корреляцию запросов для построения диалогов более высокого уровня.
2. Обработка исключений и транзакции. Долго выполняемые сервисы должны обеспечивать транзакционную целостность и управление исключениями.

WSCI

WSCI определяет расширение WSDL для взаимодействия сервисов. Первоначально составленный в Sun, SAP, BEA и Intalio, он стал спецификацией W3C. Это язык хореографии, описывающий обмен сообщениями между сервисами и не определяющий выполнение бизнес-процесса.

Один интерфейс описывает только сообщение единственного участника обмена. Хореография включает набор интерфейсов, по одному на каждого участника. Нет контроллера, регулирующего обмен.

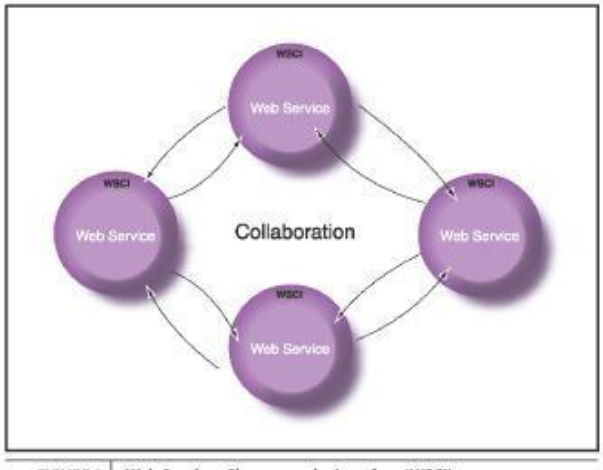


FIGURE 2 | Web Services Choreography Interface (WSCI)

Каждое взаимодействие является единицей работы, имеющей конкретную WSDL-спецификацию. WSDL описывает входные точки сервиса, WSCI описывает взаимодействие между WSDL-операциями.

WSCI может специфицировать запрос к сервису, внутреннему или внешнему.

BPEL4WS

BPEL4WS поддерживает как абстрактные бизнес-протоколы, так и выполняемые бизнес-процессы.

- ***Бизнес-протокол*** поддерживает публичный обмен сообщениями между участниками обмена. Его нельзя выполнить и он не определяет внутреннее выполнение процесса.
- ***Выполняемый процесс*** моделирует выполнение действий. Он обеспечивает оркестровку, в то время как бизнес-протоколы сфокусированы на хореографии.

Спецификация поддерживает простые действия для общения с веб-сервисами. Типичный сценарий заключается в приеме сообщения выполняемым процессом. Процесс запрашивает дополнительную информацию, вызывает внутренние сервисы и возвращает результаты.

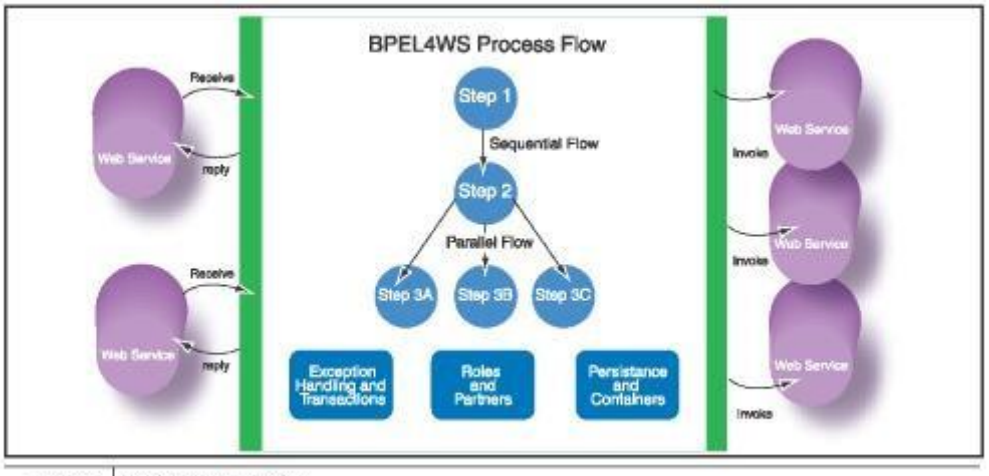


FIGURE 3 | BPEL4WS Process Flow

BPEL4WS поддерживает структурированные действия для построения бизнес-логики процесса. Переменные и партнеры – важные составляющие BPEL4WS

Процесс, определенный в BPEL4WS, состоит из:

- Действий (activities), которые являются отдельными бизнес-этапами внутри процесса. Действия могут быть простыми или состоять из других действий (структурированными).
- Ссылок на партнеров, которые определяют внешние сущности, взаимодействующие с процессом или, наоборот, использующие WSDL- интерфейсы.
- Переменных, хранящих сообщения, передаваемые между действиями, и, следовательно, представляющих состояние.
- Корреляционных наборов (correlation sets), использующихся для корреляции нескольких сообщений запросов сервиса и ответов с одним экземпляром бизнес-процесса. (К сервису могут обращаться различные бизнес-процессы, нельзя

смешивать результаты работы для разных бизнес-процессов).

- Обработчиков неисправностей (fault handlers), занимающихся исключительными ситуациями, которые могут возникнуть во время работы бизнес-процесса.
- Обработчиков событий (event handlers), принимающих и обрабатывающих сообщения параллельно с обычным выполнением процесса.
- Корректирующих обработчиков (compensation handlers), определяющих логику коррекции для отката действия или нескольких действий при возникновении исключительной ситуации.