Контекст статьи

В первой статье серии, «Моделирование SOA: Часть 1. Идентификация сервиса», рассказывалось о принципах идентификации сервисов, отвечающих бизнес-требованиям. Сначала мы определили, какие бизнес-задачи должны быть решены и какие цели достигнуты для реализации бизнес-миссии. Затем мы смоделировали бизнес-операции и бизнес-процессы, необходимые для выполнения задач и достижения целей. После этого мы рассмотрели бизнес-процесс как кооперацию сервисов, представляющую собой контракт о требованиях к сервису, который должен выполняться нашим будущим решением. Впоследствии мы использовали этот контракт о требованиях как вспомогательное средство для идентификации сервисов и потенциальных взаимоотношений между ними. Благодаря этому мы получили формальный механизм идентификации значимых для бизнеса сервисов с привязкой к бизнес-целям и бизнес-задачам, для выполнения которых они предназначены.

В предыдущей статье мы также рассмотрели возможность максимизации потенциала SOA-решения путем идентификации значимых для бизнеса сервисов. Мы спроектировали топологию сервиса с учетом бизнес-требований и снова ассоциировали сервисы с кооперациями сервисов, представляющими собой контракт о требованиях к сервису, которые должны удовлетворяться решением.

В этой статье мы продолжим определение SOA-решения путем моделирования подробной спецификации каждого сервиса. Спецификации будут описывать соглашения между потребителями и изготовителями сервиса. Такие соглашения включают информацию о предоставляемых и запрашиваемых интерфейсах, ролях, которые эти интерфейсы выполняют в спецификации сервисов, а также правилах и протоколах, определяющих взаимодействие ролей.

Описание спецификации сервиса

Теперь можно приступить к моделированию деталей спецификации сервиса. В спецификации необходимо определить все, что нужно потенциальному потребителю, чтобы он мог решить, нужен ли ему этот сервис, а также предоставить информацию о том, как использовать сервис. Кроме того, спецификация должна определять все, что должен знать поставщик сервиса для его успешного использования. Таким образом, спецификация сервиса является промежуточным звеном, или соглашением, между потребностями потребителя и функциями, предоставляемыми поставщиками.

В идеале эта информация должна размещаться в одном месте. Это значительно облегчит поиск ресурсов для сервисов многократного использования в спецификации репозиториев и получение всей необходимой информации, при этом не придется перемещаться между множеством различных документов или искать связанные элементы. Как минимум, спецификация включает следующую информацию:

Имя сервиса, отражающее его назначение;

Предоставляемые и запрашиваемые интерфейсы, при помощи которых определяются функциональные средства, предоставляемые сервисом и средства, запрашиваемые его потребителями. Примечание: речь здесь идет не о способе реализации сервиса, а о взаимодействии между потребителями и поставщиками сервиса;

Все протоколы, определяющие правила или порядок использования функциональных средств;

Ограничения, являющиеся отражением задач, которые должен решать сервис при успешном применении, а также критериев оценки успешного применения.

Характеристики, которые, по мнению потребителя, должны быть предоставлены поставщиком, например, стоимость, доступность, производительность, размер, соответствие задаче, информация о конкурентах и т. д.;

Политики использования сервиса, например политики обеспечения безопасности и объем транзакций для поддержания безопасности и целостности или для восстановления из состояния неработоспособности до состояния успешного выполнения данного сервиса или любого запрашиваемого сервиса.

Как и во всех остальных статьях данной серии, мы воспользуемся имеющимися инструментами IBM® Rational® и IBM® WebSphere®для создания артефактов решения и их взаимной привязки, благодаря чему мы сможем проверить соответствие нашего решения требованиям и более эффективно управлять изменениями. В таблице 1 представлен процесс, который мы будем использовать для разработки примера, а также инструменты, которые мы будем применять для создания артефактов. Кроме того, мы адаптируем универсальный язык моделирования (UML) к моделированию сервисов, добавив профиль IBM® Software Services Profile к UML-моделям в IBM® Rational® Software Architect.

Пересмотр идентификации сервисов

Давайте начнем с пересмотра бизнес-требований и идентифицированных нами сервисов, которые должны обеспечивать их выполнение (об этом подробно рассказывалось в статье «Моделирование SOA: Часть 1. Идентификация сервиса»). На рисунке 1 бизнес-требования показаны в виде кооперации ролей в бизнесе, ответственностей ролей и правил взаимодействия ролей.

Кооперация сервисов представляет собой контракт для требований, созданный на основе бизнес процесса и определяющий функции, которые должно обеспечивать решение сервиса. Это архитектурно-нейтральная, и, тем не менее, формальная спецификация требований, которая не должна излишне ограничивать SOA-решение. Под понятием «архитектурно-нейтральная» мы подразумеваем следующее: контракт для требований определяет, какие именно функции должно выполнять решение, но не определяет, как оно будет их выполнять. показана сводная схема идентифицированных спецификаций сервисов, которые будут придавать законченный вид решению и использовать зависимости, показывающие, как по мнению разработчика следует их использовать.

На этом разговор об идентификации сервисов и их связи с бизнес-требованиями можно считать законченным. Далее в статье объясняется, как моделировать детали спецификации сервисов. Спецификации сервисов представляют собой уточнение интерфейсов, показанных на рисунке 2. Они определяют многие из деталей, перечисленных в общем описании.

Когда интерфейсы будут готовы, нам все еще не будет известно, какие участники сервиса предоставляют или запрашивают сервисы, описываемые этими интерфейсами, как реализуются функции сервисов, которые, возможно, используются другими сервисами. Об этом мы поговорим в следующей статье, посвященной реализации сервисов.

Типы спецификаций сервисов

Спецификация сервиса должна предоставлять следующую информацию:

Имя сервиса, отражающее назначение сервиса или функцию, которую он выполняет;

Предоставляемые и запрашиваемые интерфейсы, описывающие функциональные возможности сервиса. Каждая функциональная возможность содержит следующие компоненты:

Имя, которое часто представляет собой глагольную фразу, объясняющую, что делает данная функциональная возможность;

Все обязательные или дополнительные операции ввода и вывода данных;

Все необходимые предпосылки, которые должны выполнить потребители до использования функциональной возможности;

Все постусловия, которые потребители вправе ожидать, а поставщики обязаны предоставить после успешного использования функциональной возможности;

Все исключения или условия отказа, которые могут иметь место в том случае, если функциональную возможность по каким-либо причинам невозможно будет предоставить даже при выполнении всех предпосылок.

Все протоколы связи или правила, определяющие, когда или в каком порядке можно использовать функциональные возможности;

Все функции, которые, по расчетам поставщика, должны быть обеспечены потребителями для использования сервиса или для взаимодействия с ним;

Требования, которые должен выполнять любой изготовитель сервиса в процессе предоставления сервиса;

Ограничения, являющиеся отражением задач, которые должен решать сервис при успешном применении, а также критериев оценки успешного применения;

Характеристики сервиса, которые, по мнению потребителя, должны предоставить поставщики, а именно: Стоимость, доступность, производительность, размер, соответствие задаче, информация о конкурентах и т. д.;

Политики использования сервиса, например политики обеспечения безопасности и объем транзакций для поддержания безопасности и целостности или для восстановления из состояния неработоспособности до состояния успешного выполнения данного сервиса или любого требуемого сервиса.

Понятно, что это довольно объемная информация, в данной статье не предполагается ее изучение в полном объеме. В частности, мы не будем рассматривать характеристики или политики сервисов. Это достаточно сложная тема, которая требует отдельной статьи. Кроме того, характеристики и политики могут быть специфичными для конкретного поставщика сервиса, а не только для интерфейса конкретного сервиса. Вместо этого мы сосредоточимся на изучении основ, необходимых для определения и использования сервисов.

В следующих разделах мы займемся уточнением всех идентифицированных ранее спецификаций сервисов, показанных на рисунке 2, в следующем порядке: мы начнем с самой простой

спецификации сервиса, не определяющей протоколы, затем рассмотрим спецификацию, которая предлагает простой протокол запросов/ответов, и наконец, более сложный сервис, который использует многофазный протокол и взаимодействие между потребителем и поставщиком.

Сервис планирования

Спецификация сервиса планирования, которая показана на рисунке 4, очень проста. Сервис предлагает две функциональных возможности: возможность отвечать на запросы о производственном планировании и возможность создавать расписание доставки. До сих пор, насколько нам известно, для этой ситуации не существует протокола, определяющего использование этих функциональных возможностей, поэтому обе функциональных возможности могут использоваться потребителем в любом порядке.

Спецификация сервиса планирования представляет собой простой UML-интерфейс, созданный в пакете производства. Он предоставляет две сервисных операции. Каждая из этих операций может иметь необходимые предпосылки и постусловия и, кроме того, порождать исключительные ситуации. Необходимо, чтобы параметры операций сервиса представляли собой либо данные сервиса (сообщения), либо примитивные типы. Это гарантирует, что параметры не будут делать предположений о том, как выполняется вызов — по ссылке или по значению, где размещаются данные сервиса (в каком адресном пространстве), работает ли потребитель или поставщик сервиса с копией данных или некоторым персистентным источником данных, и т. д. Все это необходимо для того, чтобы обеспечить отсутствие ограничений на место развертывания сервиса по отношению к другим сервисам. Об определении данных сервиса рассказывается в разделе Модель данных сервиса далее в этой статье.

Сервис доставки

Протокол сервиса доставки отличается меньшей сложностью. Потребитель, желающий организовать доставку продукции, запрашивает сервис доставки. Однако оператору доставки может потребоваться определенное время на то, чтобы определить, где находятся продукты, есть ли они на складе или их необходимо изготовить, и найти самый рентабельный вариант доставки. Следовательно, до получения расписания доставки может пройти некоторое время. Потребители, как правило, не хотят ждать, пока расписание будет составлено, потому что это могло бы либо помешать параллельному выполнению другой работы, либо привести к ненужной загрузке системных ресурсов длительными процессами.

Поэтому ИТ-архитекторы приняли решение использовать простой протокол запросов/ответов, или обратного вызова, между потребителем и поставщиком. Потребитель запрашивает доставку, а затем, через некоторое время, отвечает на запрос оператора доставки, чтобы получить готовое расписание. Чтобы создать модель этого протокола, нам необходимо определить роли изготовителя и потребителя, их ответственности, а также протоколы или правила их взаимодействия. Это очень важно, потому что оператор доставки не сможет отправить расписание, если не получит запрос на доставку.

Спецификация сервиса предоставляет информацию обо всем, что нужно знать о сервисе, включая требования, которым вы должны соответствовать, чтобы использовать сервис (такие требования иногда называют «соглашением об использовании» (см. статью Дэниелса (Daniels) и Чизмана (Cheesman)), и требования, которым должен удовлетворять реализатор сервиса (их иногда называют «соглашением о реализации»). Это почти та же информация, которая необходима для

сбора бизнес-требований; за исключением предметной области и уровня детализации. Это вполне понятно, потому что в спецификации в контракте о требованиях к сервису мы определяем, как взаимодействуют между собой потребитель и поставщик сервиса.

Спецификация ShippingService определяет две роли:

Роль «оператор доставки» — это роль поставщика сервиса. Данная роль отвечает за выполнение ответственностей, связанных с доставкой; эти ответственности определяются ее типом — интерфейс доставки;

Роль «оператор заказов» отвечает за обработку расписания доставки. Об этом говорит тип роли, ScheduleProcessing.

Нет необходимости в назначении таких ролей, как поставщик и потребитель. В потенциально долгосрочном диалоге возможны произвольные отклонения, которые могут способствовать вовлечению в него многих участников. Кроме того, о том, кто является потребителем и поставщиком сервиса, нетрудно догадаться по тому факту, что спецификация сервиса реализует предоставляемый интерфейс доставки и использует запрашиваемый интерфейс ScheduleProcessing.

Между ролями оператора доставки и оператора заказов мы видим соединительную линию. Она показывает, что протокол вызывает определенное взаимодействие ролей. Элемент «взаимодействие» requestShipping, который принадлежит к классу ShippingService, показывает, в чем именно заключается взаимодействие.

Взаимодействие ShippingService определяет поведенческие или динамические аспекты взаимодействия между ролями оператора доставки и оператора заказов. Оно показывает, что роль оператора заказов сначала отправляет сообщение requestShipping (или вызывает операцию requestShipping роли оператора доставки), а затем, иногда с некоторой задержкой по времени, отвечает на сообщение processSchedule, полученное от роли оператора доставки. Во взаимодействии участвуют два жизненных цикла: оператора заказов и оператора доставки. Эти экземпляры объектов представляют собой свойства ролей оператора заказов и оператора доставки в классе спецификации сервиса. То есть обмен сообщениями между описанными ролями происходит через соединительную линию между ними. Это простой шаблон запросов/ответов или внешнего вызова, обычный для многих протоколов сервисов.

Протокол ShippingService можно было бы определить с помощью любого поведения UML 2: деятельности, взаимодействия, конечного автомата, протокола конечного автомата или неясного поведения (кода). Какое из них выбрать, зависит от специалиста, создающего модель, стиля, который он предпочитает, или применимости предметной области задачи.

Сервис инвойсирования

Вычисление первоначальной и окончательной цены для счета требует использования несколько более сложного протокола для определения взаимодействия между ролями оператора заказов и оператора инвойсирования. Очевидно, что операцию initiatePriceCalculation необходимо вызывать раньше операции completePriceCalculation. После этого оператор заказов должен быть готов обработать окончательный счет.

Мы можем документировать описанный протокол при помощи абстрактного класса, который определяет роли оператора инвойсирования и заказов, их ответственности и протокол (диалог или правила), по которому они будут взаимодействовать. Это похоже на спецификацию ShippingService, за исключением того, что взаимодействие более сложное, чем обычный диалог запрос/ответ. Здесь мы имеем дело с последовательностью, в которой необходимо вызывать функциональные возможности, чтобы обеспечить корректное использование сервиса.

Данный протокол документирован в спецификации сервиса InvoicingService , которая показана на рисунке 6. Обратите внимание на то, что эта спецификация сервиса также реализует прецедент Invoicing. Прецеденты можно использовать для представления специфичных требований сервисов. Спецификация сервиса состоит из двух ролей: «оператор инвойсирования» и «оператор заказов». Эти роли имеют, соответственно, тип реализуемого интерфейса «Invoicing „и используемого интерфейса InvoiceProcessing . Данные интерфейсы инкапсулируют ответственности ролей (функциональные возможности или операции сервиса или требования). Деятельность InvoicingService в спецификации сервиса определяет протокол использования операций сервиса, реальное взаимодействие, которое может происходить между ролями „оператор заказов“ и „оператор инвойсирования“.

InvoicingService — это класс, который определяет протокол диалога, связи или правила взаимодействия между ролями оператора заказов и оператора счетов. Детали протокола фиксируются в поведении ownedBehavior данного класса, который используется для определения корректных шаблонов взаимодействия для использования этого сервиса. В этом случае протокол определяется в форме деятельности UML.

Протокол показывает, что оператор заказов должен сначала инициировать вычисление цены, и только после этого пытаться получить окончательный расчет цены. После этого оператор заказов должен быть готов ответить на запрос (в данном случае, обратный вызов) на обработку окончательного счета. Некоторые потребители, запрашивающие сервис инвойсирования, могут выполнять и другие действия помимо упомянутых трех, но протокол ограничивает определение последовательности этих специфических действий. Обратите внимание на то, что элемент ActivityPartitions в деятельности InvoicingService представляет роли или свойства в классе InvoicingService. Действие вызова операции, принадлежащее к какой-либо секции, показывает, что вызов осуществляется в роли, которую представляет данная секция (целевой входной точкой действия является роль, изображаемая секцией деятельности).

В данном случае имеет место только одно взаимодействие между оператором заказов и сервисом инвойсирования, поэтому класс спецификации сервиса имеет только одно поведение — ownedBehavor. В других ситуациях между потребителем и поставщиком может быть несколько взаимодействий, каждое из которых использует отдельный протокол. Спецификация сервиса будет содержать поведение ownedBehavior, определяющее шаблоны взаимодействия для каждого из этих диалогов.

На данный момент нам неизвестно, какой поставщик сервиса реализует InvoicingService . Мы также не знаем, какой потребитель сервиса мог бы его использовать. Нам известно только то, что должен сделать любой потребитель, чтобы использовать этот сервис, а также то, что должен делать любой поставщик для того, чтобы его реализовать.

Спецификация приобретения

И наконец, последняя спецификация определяет обработку заказов на приобретение (рисунок 7).

Так же, как и спецификация сервиса планирования, сервис приобретения представляет собой простой интерфейс, имеющий только одну операцию, которая предоставляет функцию обработки заказов на приобретение покупателю, который возвращает заполненный счет. В качестве побочного действия этой операции заказанные товары производятся (если нужно) и доставляются покупателю.

Интерфейс этого сервиса представляет собой функциональную возможность, определенную в исходном бизнес-процессе „Обработка заказов на приобретение“. Он также представляет сервис, идентифицированный и разработанный на основе бизнес-процесса.

Возвращаемся к топологии сервиса

На данный момент спецификации сервиса определены более полно, поэтому мы можем вернуться к топологии сервиса, показанной на рисунке 3. Вспомним, что на этом рисунке показано, как можно использовать экземпляры идентифицированных сервисов, чтобы обеспечить выполнение контракта о требованиях к сервису. Однако на этой диаграмме не слишком много информации о самих сервисах, о том, как они подключаются друг к другу или о том, какие протоколы используются в процессе их взаимодействий. Наша спецификация сервиса готова, теперь можно приступить к созданию новой диаграммы, показывающей, как участники, использующие эти сервисы, могут взаимодействовать, чтобы обеспечить выполнение описанных требований. Мы будем возвращаться к этой диаграмме в следующей статье серии, „Моделирование SOA: Часть 3. Реализация сервиса“, в которой она будет использоваться в качестве исходной точки для разработки окончательного интегрированного решения сервисов, описанных в данном примере. абстрактный компонент „Обработка заказов“, предоставляющий контекст для демонстрации другого представления топологии сервиса. Его элементы представляют участников процесса обработки заказов, которые будут предоставлять и/или запрашивать сервисы, необходимые для выполнения требований контракта „Обработка заказов на приобретение“. Мы не знаем точно, чем еще характеризуются эти элементы (они не имеют типов), но можем определить, что они должны делать, указав выполняемые ими роли в спецификации сервисов, определяющей, как они должны взаимодействовать и каким требованиям удовлетворять.

Линии, соединяющие элементы компонента „Обработка заказов“, показывают ожидаемые взаимодействия между этими подкомпонентами. Подписи над соединительными линиями соответствуют именам соответствующих им контрактов, которые в этом случае являются протоколами для соответствующих спецификаций сервисов. Они показывают, что эти элементы должны будут предоставлять и запрашивать интерфейсы, определяемые спецификацией сервиса, и использовать указанные интерфейсы в соответствии с протоколом спецификации сервиса. То, как именно это будет происходить, моделируется в описаниях реализации сервисов, о которой будет рассказано в следующей статье данной серии.

Мы также видим, что все эти элементы будут взаимодействовать особым образом, позволяющим обеспечить выполнение требований контракта „Обработка заказов на приобретение“. Таким образом, Обработка заказов объединяет два момента:

Потребители и поставщики (или участники) сервиса должны выполнять бизнес-требования;

Подключения и протоколы, которые показывают, как эти участники взаимодействуют для выполнения указанных требований.

Модель данных сервиса

Модель данных системы взаимоотношений с клиентами, Customer Relationship Management (CRM), определенная в пакете org::crm, определяет все данные, используемые всеми операциями сервиса в модели „Обработка заказов на приобретение“ в данном примере (рисунок 9). Пакет CRM представляет собой проект модели данных сервиса «Управление взаимоотношениями с клиентами“, которую можно повторно использовать в нескольких сервисах, даже если эти сервисы предоставляются разными организациями. В данной статье мы не рассматриваем вопросы обнаружения и нормализации данных сервиса, а также их отношения к персистентным сущностям или физическим источникам данных. Ее тема — описание данных сервиса и документирование модели.

Каждый тип данных <<message>> на рисунке 9 представляет данные сервиса. Данные сервиса — это данные, которые участвуют в обмене сообщениями между потребителями и поставщиками сервисов. Типы данных параметров для операций сервиса относятся либо к сообщениям, либо к примитивным типам.

Примечание:

сообщения данных сервиса и сообщения языка описания Web-сервисов Web Services Description Language (WSDL) — это не одно и то же. Операция сервиса может иметь любое количество входов и выходов с типами сообщений или примитивными типами. Проект операций сервиса может предусматривать использование общих входа, выхода и сообщений об ошибках, но это не является обязательным, в результате может возникнуть нежелательное сцепление данных штампа.

Сообщения представляют собой объекты передачи данных (объекты DTO), которые можно свободно использовать для обмена данными между адресными пространствами в распределенных средах. Поставщики и потребители сервисов не делают никаких предположений о том, где на самом деле размещаются данные и, следовательно, допускают, что имеют дело с копией некоторого представления реальных персистентных данных предметной области. Сообщения не имеют идентификаторов. Они являются ценными объектами, потому что их тождество определяется по значению их содержания, а не по идентификаторам.

Сообщения содержат данные, участвующие в обмене между потребителями и поставщиками сервиса в потенциально распределенной среде. Поставщики сервисов часто нуждаются в доступе к персистентным данным, чтобы использовать их в проектах своих реализаций. Взаимосвязь между данными сервиса и персистентными данными предметной области, используемыми в реализации сервиса, должна поддерживаться поставщиком сервиса и его реализацией функциональных возможностей сервиса. Часто данные сервиса представляют собой выборку и проекцию (или представление) данных предметной области. Однако способ извлечения данных сервиса из предметной области и обновления предметной области данными сервиса определяется реализацией сервиса. Объекты данных сервиса (объекты SDO) — это очень полезный механизм реализации сообщений с данными сервиса. Они также имеют функции управления наборами изменений и автоматической фиксации изменений в персистентных

хранилищах. Реализации участников сервиса могут использовать эти функции, однако нет необходимости решать их в модели.

Модель данных использует стереотип <<id>> для идентификации атрибутов, которые уникальным образом идентифицируют экземпляры содержащего их класса. К этой теме мы будем неоднократно обращаться при моделировании сервисов, потому что Web-сервисы и особенно язык исполнения бизнес-процессов для Web-сервисов (Business Process Execution Language for Web Services, BPEL) используют бизнес-данные для корреляции экземпляров или идентификации объекта по значению.