Организация синхронного взаимодействия для интеграции информационных систем

Тимакин Олег Анатольевич
доцент кафедры Экономической кибернетики
Экономического факультета ЮФУ
г. Ростов-на-Дону
E-mail: toa555@mail.ru

Интеграция приложений — это стратегический подход к объединению информационных систем, который обеспечивает возможность обмена информацией и поддержания распределенных бизнес-процессов. Интеграция информационных систем дает предприятию такие несомненные конкурентные преимущества, как:

· ведение бизнеса в режиме реального времени с использованием событийно-управляемых сценариев;

· владение достоверной, полной и своевременно полученной информацией.

Задача интеграции — обеспечить эффективный, надежный и безопасный обмен данными между различными программными продуктами, изначально не предназначенными для совместной работы. Как правило, требования бизнеса эволюционируют быстрее, чем способы их поддержки информационными технологиями.

Основные бизнес-выгоды, которые предприятие может получить в результате успешной реализации интеграционного проекта:

· улучшение качества поддержки и обслуживания клиентов;

· автоматизация бизнес-процессов;

· уменьшение производственного цикла;

· сокращение количества ошибок обработки данных;

· прозрачность процессов;

· уменьшение стоимости транзакций;

· оптимизация логистических процессов;

· более тесное взаимодействие с бизнес-партнерами;

· быстрое внедрение новых бизнес-сервисов;

· сохранение инвестиций в информационные технологии.

Одной из моделей взаимодействия интеграционных систем является метод синхронного взаимодействия. Взаимодействие ИС может быть описано в рамках модели «клиент-сервер». В данном контексте под клиентом будем понимать приложение, которое отправляет запрос на обработку. Приложение, отвечающее на запрос, будем называть сервером. В большинстве случаев одно и то же приложение в зависимости от ситуации может выступать в роли как клиента, так и сервера.

Взаимодействие между приложениями может быть синхронным или асинхронным. Синхронным называется такое взаимодействие, при котором приложение, выступающее в роли клиента, отослав запрос, блокируется и может продолжать работу только после получения ответа от приложения, выступающего в роли сервера. Иногда такой вид взаимодействия называют блокирующим. Схема синхронного взаимодействия представлена ниже на рисунке 1.

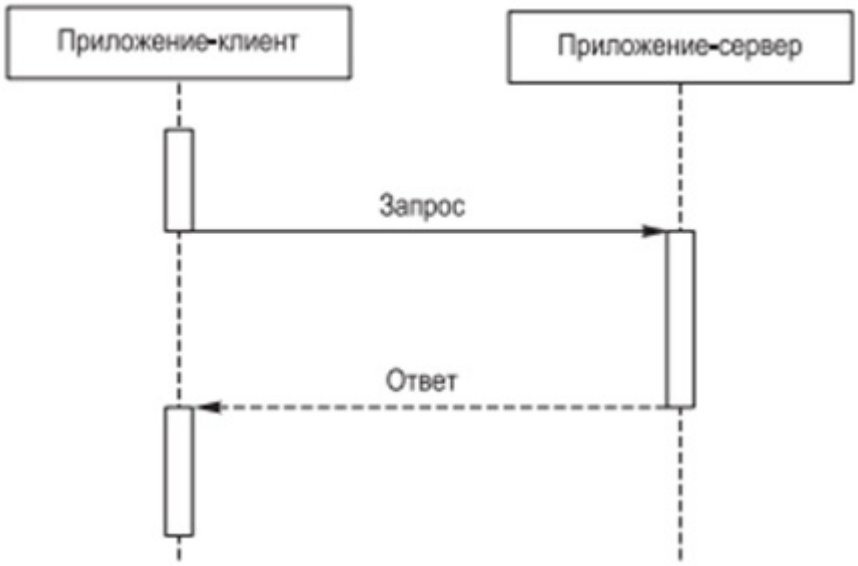

Рисунок 1 — Схема синхронного взаимодействия

Обычное обращение к функции или методу объекта с помощью передачи управления по стеку вызовов является примером синхронного взаимодействия.

Синхронное взаимодействие достаточно легко организовать, и оно является гораздо более простым для понимания. Человеческое сознание обладает единственным «потоком управления», представленным в виде фокуса внимания, и поэтому человеку проще понимать процессы, которые разворачиваются последовательно, один за другим, поскольку внимание не рассеивается при наблюдении за процессами, происходящими одновременно. Код программы клиентского компонента, которая описывает синхронное взаимодействие систем, устроен более простым способом — одна из его частей, которая отвечает за обработку ответа сервера, располагается сразу же после части, в которой составляется запрос. В силу того, что синхронные взаимодействия проще по своей организации, они используются гораздо чаще асинхронного взаимодействия.

Однако вместе с этим синхронное взаимодействие ведет к существенным затратам времени на ожидание ответа. Время, затраченное на ожидание ответа, зачастую можно использовать более целесообразно — при ожидании ответа на один запрос, клиент мог бы параллельно выполнить другую работу, обрабатывать другие запросы, которые не зависимы от еще не пришедшего результата. Так как все распределенные системы состоят из достаточно большого количества уровней, через которые проходят практически все взаимодействия, общее снижение производительности, связанное с синхронностью взаимодействий, оказывается очень значительным.

Одним из самых популярных и исторически первым практически универсальным методом реализации синхронного взаимодействия в распределенных системах является ***удаленный вызов процедур*** (Remote Procedure Call, RPC). Его модификация для объектно-ориентированной среды именуется ***удаленным вызовом способов*** (Remote Method Invocation, RMI).

Удаленный вызов процедур описывает как метод организации взаимодействия между компонентами, так и методику разработки данных компонентов (Рисунок 2).

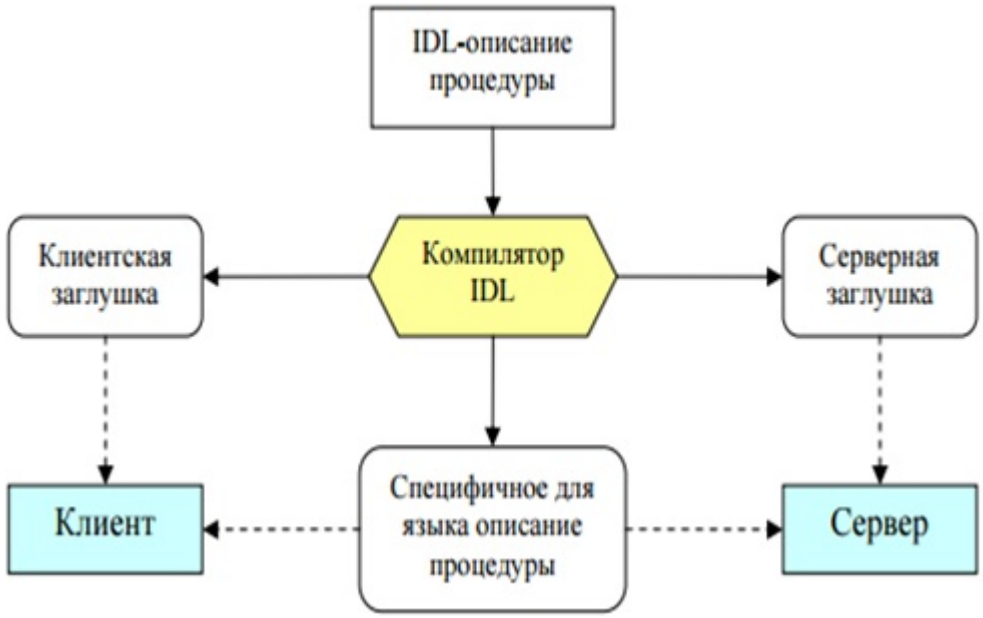

Рисунок 2 — Схема разработки компонентов, взаимодействующих с помощью RPC

Первым этапом разработки является разработка интерфейс процедур, которые будут использоваться для удаленного вызова. Это делается при помощи ***языка определения интерфейсов*** (Interface Definition Language, IDL). Данным языком может выступать либо специальный, либо обыденный язык программирования. Но с ограничениями, которые определяются возможностью передачи вызовов на удаленную машину. Спецификация процедуры для удаленных вызовов формируется компилятором IDL в описание данной процедуры на языках программирования, на которых будут разрабатываться сервер и клиент (например, файлы-заголовки на языках C/C++), и два дополнительных компонента — ***заглушки клиентские и серверные*** (client stub и server stub).

Клиентская заглушка представляет собой элемент, который размещается на той же машине, где находится клиент. Удаленный вызов процедуры клиентом функционирует как обыденный, локальный вызов определенной функции в клиентской заглушке. При обработке данного вызова клиентская заглушка выполняет нижеописанные действия:

1. Определение физического местонахождения в системе сервера, для которого предназначен данный вызов. Это шаг именуется ***привязкой*** (binding) к серверу. Результат данного

этапа — адрес машины, на которую необходимо передать вызов.

2. Вызов процедуры, затем упаковка ее аргументов в сообщение в некотором формате, который понятен серверной заглушке. Этот шаг именуется ***маршалингом*** (marshaling).

3. Преобразование приобретенного сообщения в поток байтов (сериализация, serialization), затем он отсылается при помощи какого-либо протокола на машину, на которой помещен серверный компонент.

4. Распаковка сообщения после получения от сервера ответа, после этого оно возвращается клиенту в качестве результата работы процедуры.

В итоге для клиента удаленный вызов процедуры смотрится как обращение к обыкновенной функции.

На той же машине, где находится компонент-сервер, располагается и серверная заглушка. Заглушка выполняет операции, которые обратны по своей сущности к действиям клиентской заглушки — принимает сообщение, которое содержит аргументы вызова, распаковывает эти аргументы при помощи таких способов как ***десериализация*** и ***демаршалинг***, вызывает локально соответствующую функцию серверного компонента, получает результат от нее, упаковывает его и высылает по сети на машину-клиент. Таким образом обеспечивается отсутствие видимых серверу различий между удаленным вызовом некоторой его функции и ее же локальным вызовом.

После определения интерфейса процедур, вызываемых удаленно, становится возможным перейти к разработке сервера, который реализует данные процедуры, и клиента, который использует их для решения своих задач.

При удаленном вызове процедуры обращение клиента оформляется таким же способом, как вызов локальной функции и обрабатывается при помощи клиентской заглушкой. Она же определяет адрес устройства, на котором находится сервер, собирает данные вызова в сообщение и передает на серверную машину. На серверной машине серверная заглушка после получения сообщения распаковывает его, извлекает из него необходимые аргументы вызова, затем обращается к серверу с таким вызовом локально, дожидается от него результата, упаковывает результат в сообщение и, наконец, передает его обратно на клиентскую машину. Получив сообщение в ответ, клиентская заглушка распаковывает его и отправляет полученный ответ клиенту.

Эта же техника может найти свое применение и для реализации взаимодействия компонентов, которые работают в рамках различных процессов на одной машине (Рисунок 3).

Рисунок 3 — Схема реализации уделённого вызова процедуры

При организации удаленного вызова методов в объектно-ориентированной среде применяются такие же механизмы. Отличия в его реализации связаны с аспектами, которые описаны ниже:

· Аргументами удаленного вызова могут служить объекты. Следует отметить, что передача указателей в аргументах удаленного вызова процедур почти полностью запрещена — указатели имеют привязку к памяти данного процесса и не могут быть переданы в другой процесс.

· Один объект-сервер может реализовать несколько методов для удаленного обращения к ним. Для таких объектов формируются клиентские заглушки, которые имеют в своем интерфейсе все описанные методы. Вдобавок, информация о том, какой конкретно метод вызывается, должна упаковываться вместе с аргументами вызова и использоваться серверной заглушкой для обращения именно к данному методу. Серверная заглушка в контексте удаленного вызова способов иногда называется ***скелетоном*** (skeleton) или ***каркасом***.

При передаче объектов возможны следующие подходы, и все они используются на практике.

о Идентичность передаваемых объектов могут не иметь значения, например, если сервер использует их только как хранилище данных и получит точно идентичный результат при работе с их правильно построенными копиями. В этом случае определяются методы для сериализации и десериализации данных объекта, позволяющие сохранить их в виде потока байтов и восстановить объект с теми же данными на другой стороне.

о Идентичность передаваемого объекта может иметь большое значение, например, если сервер вызывает в нем методы, работа которых зависит от того, что это за объект. При этом используется специализированная ссылка на данный объект, которая позволяет обратиться к нему из другого процесса или с другой машины, т.е. также с помощью удаленного вызова методов.

Таким образом, интеграция информационных систем в настоящее время необходима бизнесу

для его дальнейшего успешного развития. Каждое предприятие выбирает способ интеграции информационных систем в зависимости от своих потребностей и ресурсов, и зачастую синхронное взаимодействие систем помогает их удовлетворить. Этот способ обладает как достоинством (простота организации), так и недостатком (затраты значительного количества времени при работе). Если же фирма считает, что данный вид взаимодействия не является подходящим для нее, то она сможет выбрать альтернативный способ — организация асинхронного взаимодействия информационных систем.

Список использованных источников

1. Морозова О.А. Интеграция корпоративных информационных систем: М80 учебное пособие. — М.: Финансовый университет, 2014. — 140 с.
2. В. В. Кулямин Технологии программирования. Компонентный подход. (http://panda.ispras.ru/~RedVerst/RedVerst/Lectures%20and%20training%20courses/Software%20De
3. Статья «Синхронное и асинхронное взаимодействие» (http://refereed.ru/ref_84ea9842f464980401b1e025f98c8412.html)