Распределенные информационно-аналитические системы

Лекция № 14.

Распределенная система объектов CORBA

Профессор кафедры КБ-2: д.т.н. Шатовкин Р.Р.

Учебные цели:

Изучить основы научных знаний по основным понятиям CORBA; технологии CORBA; архитектурой CORBA; созданию CORBA-систем.

Учебные вопросы:

1. Основные понятия CORBA.

2. Технология CORBA.

3. Архитектура CORBA.

4. Создание CORBA-систем.

1. Основные понятия CORBA

CORBA (Common Object Request Broker Architecture – общая архитектура брокера объектных запросов) – это технология разработки распределенных приложений, ориентированная на интеграцию распределенных изолированных систем.

В начале 1990-х г.г. ночным кошмаром были проблемы обеспечения возможности общения программ, выполняемых на разных машинах, особенно, если использовались разные аппаратные средства, операционные системы и языки программирования: либо программисты использовали сокеты и сами реализовывали весь стек протоколов, либо их программы вовсе не взаимодействовали. (Другие ранние средства промежуточного программного обеспечения были ограничены средой операционной системы и не подходили для использования в неоднородных средах)

Для решения данной проблемы в 1989 г. был создан консорциум **OMG** (Object Management Group), основной задачей которого стала разработка и продвижение объектно-ориентированных технологий и стандартов. Это некоммерческое объединение, разрабатывающее стандарты для создания корпоративных платформо-независимых приложений.

Концептуальной инфраструктурой, на которой базируются все спецификации OMG, является архитектура **OMA** (Object Management Architecture). В состав OMA входят разнообразные стандартизованные или в настоящий момент стандартизируемые OMG службы, сервисы, программные образцы и шаблоны (CORBA services, horizontal and vertical CORBA facilities); язык описания интерфейсов распределенных объектов **IDL** (Interface Definition Language), формат которого не зависит от языка программирования; стандартизованные или стандартизируемые отображения IDL на языки программирования; объектная модель CORBA.

Главной особенностью CORBA является использование компонента **ORB** (Object Request Broker – брокер объектных запросов) – программы-транслятора межобъектного взаимодействия; работая на клиенте и на сервере, передает объектные запросы между ними для создания экземпляров объектов и вызова их методов. Данный компонент формирует «мост» между приложением и инфраструктурой CORBA (рисунок 1).

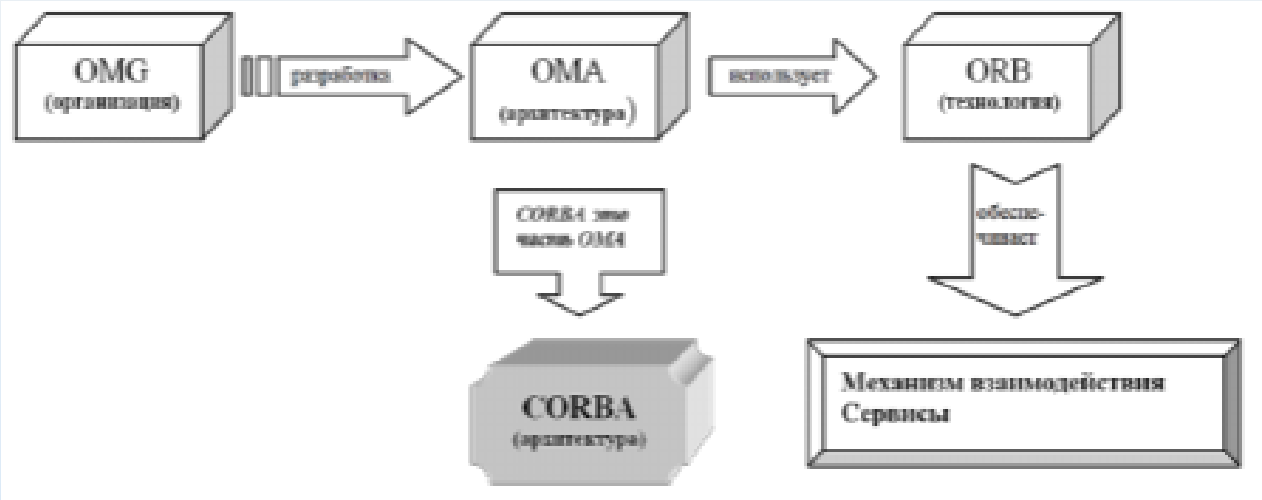

Рисунок 1 – Основные понятия технологии CORBA

В 1997 г. консорциум OMG опубликовал спецификацию CORBA 2.0. В ней определялись стандартный протокол и отображение для языка C++, а в 1998 г. было определено отображение для Java. В результате разработчики получили инструментальное средство, позволяющее им относительно легко создавать неоднородные распределенные приложения.

CORBA быстро завоевала популярность, и с использованием этой технологии был создан ряд критически важных приложений.

Некоторые основные понятия CORBA:

BOA (Basic Object Adapter) – стандарт объектного адаптера до CORBA 2.2 (недостаточно полно специфицированный).

CORBA-объект – виртуальное понятие: нечто, посылающее запросы к другим CORBA-объектам – серверным объектам и получающее запросы от других CORBA-объектов – клиентов.

IIOP (Internet Inter-ORB Protocol) – протокол передачи объектных запросов по TCP/IP.

IOR (Interoperable Object Reference) – ссылка на объект, уникальная в пределах сервера (как правило, содержит идентификатор объекта как составную часть).

Java IDL – не вполне корректное название реализации CORBA для Java (содержит не только компилятор IDL).

Java-IDL компилятор – компилятор описаний IDL в классы-заглушки и вспомогательные классы Java.

POA (Portable Object Adapter) – стандарт объектного адаптера начиная с CORBA 2.2 (достаточно полно специфицирован, является платформенно-независимым).

Smart Agent – административная утилита, осуществляет поиск объектов в домене и балансировку нагрузки.

Активация CORBA-объекта – запуск существующего CORBA-объекта для обработки клиентских запросов (в зависимости от политик объектного адаптера предполагает создание сервантов, занесение в карту активных объектов, и т.д.).

Виртуальный домен – один или несколько компьютеров, логически объединенных для выполнения некоторой задачи.

Временный (transient) CORBA-объект – объект, который уничтожается с завершением активировавшего его потока.

Деактивация CORBA-объекта – останов CORBA-объекта (разрыв связки между объектом и сервантом, в общем случае без разрушения объекта).

Демон активации объектов (Object Activation Daemon, OAD) – демон, отслеживающий входящие запросы и активизирующий нужные объекты-серверы.

Идентификатор объекта (Object ID) – уникальное имя объекта внутри его объектного адаптера.

Инкарнация серванта – связывание серванта с CORBA-объектом для обработки клиентского запроса.

Карта активных объектов (Active Object Map) – таблица объектного адаптера, в которой он ведет реестр активных CORBA-объектов и связанных с ними сервантов (первые представлены в карте своими идентификаторами).

Менеджер сервантов – элемент технологии CORBA, один из способов управлять связками объект-сервант, предоставляет подходящий сервант для объекта.

Объектный адаптер – элемент технологии CORBA, отображающий понятие программно-реализованных сервантов на концепцию CORBA-объектов; в его обязанности входит: создание CORBA-объектов и их объектных ссылок; демультиплексирование запросов на каждый серверный CORBA-объект; перенаправление запросы к соответствующему серванту, который обеспечивает реализацию серверного CORBA-объекта; активация и деактивация CORBA-объектов (соответственно, инкарнация и эфемеризация соответствующие серванты).

Связывание языка программирования – правила трансляции IDL-описаний в код на данном языке; эти правила определены OMG.

Сервант – физическая реализация CORBA-объекта; серверная программа, написанная на каком-либо из языков программирования и выполняющая CORBA-объект.

Сервис именования (Naming Service) – CORBA-объект, который позволяет обнаружить другие объекты по имени. Может быть устойчивым (запоминать ссылки и имена после остановки) и временным (не запоминать).

Скелетон – заготовка для серванта, генерируемая IDL-компилятором.

Устойчивый (persistent) CORBA-объект – объект, который может существовать дольше, чем активировавший его поток.

Эфемеризация серванта – разрушение связки CORBA-объект – сервант.

2. Технология CORBA

Технология CORBA создана для поддержки разработки и развертывания сложных объектно-ориентированных прикладных систем.

CORBA является механизмом в программном обеспечении для осуществления интеграции изолированных систем, который дает возможность программам, написанным на разных языках программирования, работающих в разных узлах сети, взаимодействовать друг с другом так же просто, как если бы они находились в адресном пространстве одного процесса.

Основные компоненты, составляющие ядро архитектуры CORBA, представлены на рисунке 2.

Рисунок 2 – Ядро архитектуры CORBA

Спецификация CORBA предписывает объединение программного кода в объект, который должен содержать информацию о функциональности кода и интерфейсах доступа. Готовые объекты могут вызываться из других программ (или объектов спецификации CORBA), расположенных в сети.

CORBA использует язык описания интерфейсов IDL для определения интерфейсов взаимодействия объектов с внешним миром, она описывает правила отображения из IDL в язык, используемый разработчиком CORBA-объекта.

Стандартизованы отображения для Ada, C, C++, Java, Python, COBOL, Lisp, PL/1 и Smalltalk. Также существуют нестандартные отображения на языки Perl, Visual Basic, Ruby и Tcl, реализованные средствами ORB, написанными для этих языков.

Для преобразования описания интерфейса на языке IDL на требуемый язык программирования используется специальный компилятор. В дальнейшем построенный с его помощью программный код может быть преобразован любым стандартным компилятором в исполняемый код.

Компонент ORB поддерживает удаленное взаимодействие с другими ORB, а также обеспечивает управление удаленными объектами, включая учет количества ссылок и времени жизни объекта.

Для обеспечения взаимодействия между ORB используется протокол GIOP (General Inter-ORB Protocol – общий протокол для коммуникации между ORB). Наиболее распространенной реализацией данного протокола является протокол IIOP, обеспечивающий отображение сообщений GIOP на стек протоколов TCP/IP.

Схема взаимодействия объектов ОМА представлена на рисунке 3.

Рисунок 3 – Схема взаимодействия объектов ОМА

Изначально, технология CORBA ориентирована на предоставление готовой проблемно-ориентированной инфраструктуры для создания РВС в рамках определенной проблемной области. Для этого, в состав CORBA включают набор стандартных объектных сервисов и общих средств. Спецификация CORBA предусматривает также ряд стандартизованных сервисов (CORBA Services) и общих средств (Common Facilities).

Сервисы представляют собой обычные CORBA-объекты со стандартизованными (и написанными на IDL) интерфейсами. К таким сервисам относится, например, сервис имен NameService, сервис сообщений, позволяющий CORBA-объектам обмениваться сообщениями, сервис транзакций, позволяющий CORBA-объектам организовывать транзакции.

В реальной системе не обязательно должны присутствовать все сервисы, их набор зависит от требуемой функциональности. На сегодня разработано всего 14 объектных сервисов.

Между объектными сервисами и общими средствами CORBA нет четкой границы.

Последние тоже представляют собой CORBA-объекты со стандартизованными интерфейсами.

Common Facilities делятся на горизонтальные (общие для всех прикладных областей) и вертикальные (для конкретной прикладной области). Например, разработаны Common Facilities для медицинских организаций, для ряда производств и т.п.

Процесс разработки приложения с использованием технологии CORBA состоит из следующих 4-х этапов:

1. Определение интерфейса на IDL.
2. Обработка IDL для создания кода заглушки и скелетона.
3. Создание кода реализации объекта (сервер).
4. Создание кода использования данного объекта (клиент).

Язык определения IDL позволяет независимо от используемого языка программирования создать универсальное описание интерфейса будущей системы.

Созданный на IDL код должен специальным компилятором преобразовываться в код интерфейса объекта на требуемом языке программирования. После чего, на клиенте автоматически генерируется заглушка, преобразующая вызовы методы данного интерфейса в обращения к ORB. На сервере программист на основе сгенерированного интерфейса создает собственную реализацию данного класса. Скелетон автоматизирует получение и обработку удаленного вызова методов, поступающих через ORB.

По сравнению с классическим клиент-серверным подходом, использование технологии CORBA для разработки распределенных приложений имеет следующие ***преимущества***:

- использование IDL для описания интерфейсов позволяет разрабатывать программные компоненты независимо от языка программирования и базовой операционной системы;

- поддержка богатой инфраструктуры распределенных объектов;

- прозрачность вызова удаленных объектов.

Однако программные решения на базе технологии CORBA редко выходят за рамки отдельных предприятий. Разработка крупномасштабных межорганизационных систем на базе технологии CORBA сопряжена со следующими трудностями:

- плохая совместимость различных реализаций технологии CORBA от различных поставщиков;

- проблемы взаимодействия узлов CORBA через Интернет;

- несогласованность многих архитектурных решений CORBA и отсутствие компонентной модели, которая могла бы значительно упростить разработку.

На смену технологии CORBA, пришли стандартизованные протоколы веб-сервисов, такие как XML, WSDL, SOAP и др. В настоящее время CORBA используется для реализации узкого круга приложений и является фактически нишевой технологией.

3. Архитектура CORBA

В данном учебном вопросе приводится краткий обзор архитектуры CORBA в том виде, как ее описание дается в спецификации OMG версии 3.0.

Требования этого документа могут в различной степени удовлетворяться фактическими реализациями брокера объектных запросов – ORB.

На рисунке 4 изображен запрос, посылаемый клиентом на реализацию объекта. Клиент – это сущность, которая хочет выполнить операцию с объектом, а реализация – это совокупность кода и данных, которые в действительности реализуют объект.

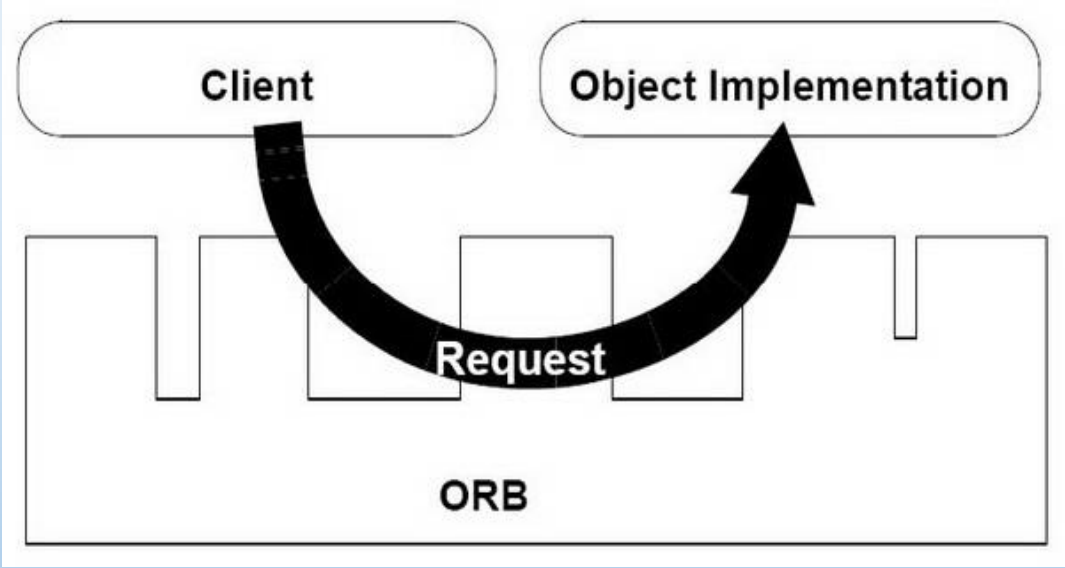

Рисунок 4 – Клиент посылает запрос реализации объекта

ORB отвечает за все механизмы, необходимые для поиска подходящей для запроса реализации объекта, подготовки реализации к получению запроса и передачи данных в процессе выполнения запроса. Интерфейс, видимый клиенту, совершенно независим от расположения реализации объекта, языка программирования, на котором она написана и любых других аспектов, не отраженных в спецификации интерфейса.

На рисунке 5 изображена структура ORB. Его интерфейсы показаны на рисунке штрихованными прямоугольниками, стрелки обозначают, вызывается ли брокер или сам выполняет вызов.

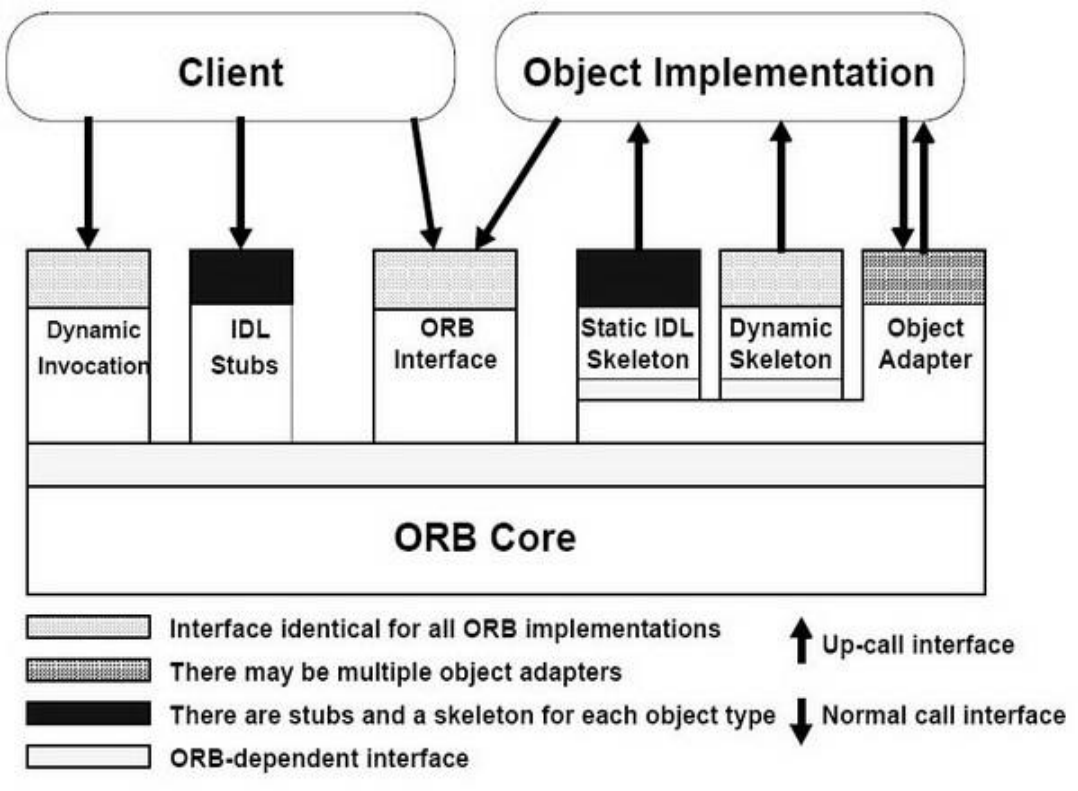

Рисунок 5 – Интерфейсы ORB

Чтобы сделать запрос, клиент может использовать ***динамический интерфейс вызова (Dynamic Invocation Interface – DII)***, один и тот же, вне зависимости от интерфейса целевого объекта, или ***IDL заглушку (stub)***, специфичную для интерфейса целевого объекта. Клиент также может напрямую взаимодействовать с брокером для получения некоторых функций. Реализация объекта получает запрос как вызов либо через автоматически сгенерированный ***IDL скелетон***, либо через ***динамический скелетон***. Реализация объекта может вызывать объектный адаптер или ORB во время выполнения запроса или в другое время.

Интерфейсы объектов могут быть описаны двумя способами.

Во-первых, статически, на языке описания интерфейсов IDL. Этот язык позволяет описывать типы объектов через предоставляемые ими операции и их параметры.

Во-вторых, интерфейсы могут быть добавлены в ***репозиторий интерфейсов***. Это специальный сервис, представляющий компоненты интерфейсов, как объекты, и предоставляющий доступ к этим компонентам во время выполнения.

Для выполнения запроса клиент должен иметь доступ к ***объектной ссылке (Interoperable Object Reference – IOR)***, знать тип объекта и ту операцию, которую он хочет выполнить. Клиент инициирует запрос, вызывая подпрограммы заглушки, специфичные для конкретного объекта, или создавая запрос динамически (рисунок 6).

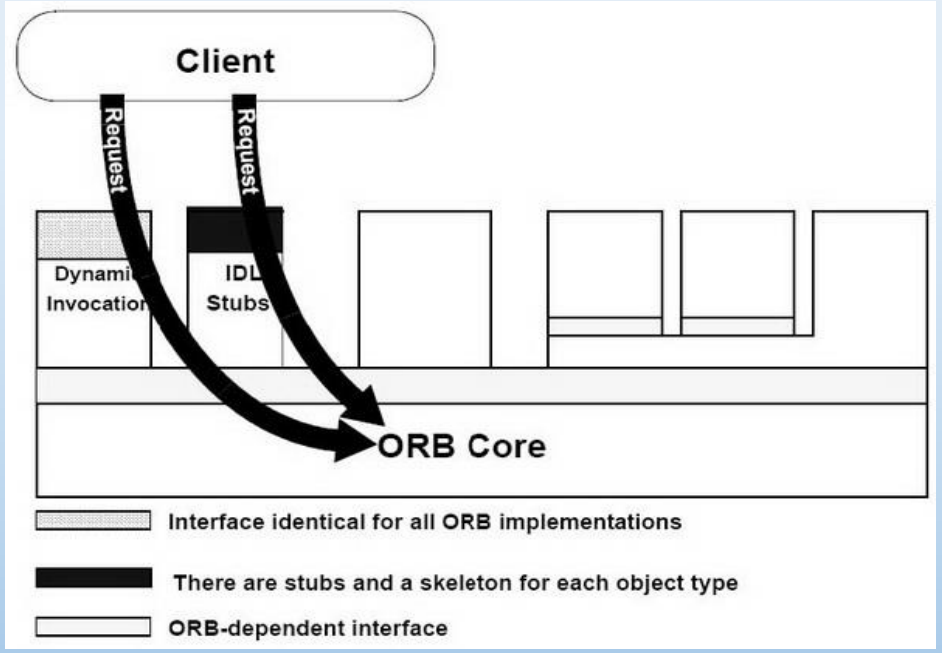

Рисунок 6 – Клиент выполняет запрос (динамически или через заглушку)

Динамический интерфейс вызова DII и интерфейс заглушки stub имеют одинаковую семантику, так что получатель сообщения не может определить, как был послан запрос. ORB находит подходящий код реализации объекта, пересылает ему параметры и отдает управление через IDL скелетон или динамический скелетон (рисунок 7).

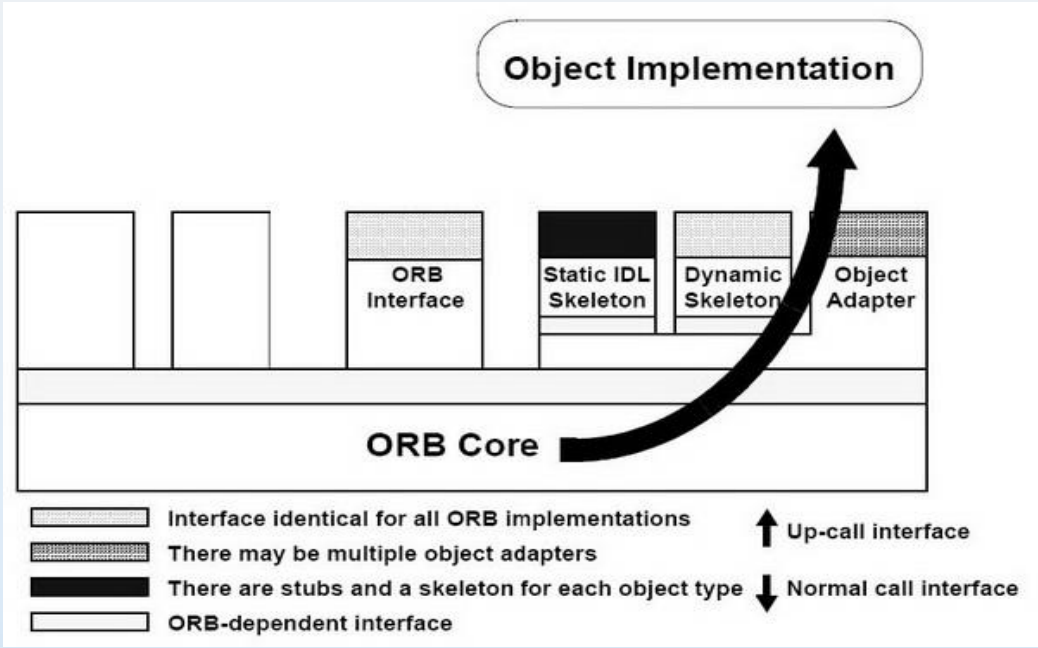

Рисунок 7 – Реализация объекта получает запрос

Скелетоны специфичны для конкретного интерфейса и объектного адаптера. Во время выполнения запроса реализация может пользоваться некоторыми сервисами ORB через объектный адаптер. Когда запрос выполнен, управление и значения результата возвращаются клиенту.

Реализация объекта может выбрать, какой объектный адаптер использовать, в зависимости от того, в каких сервисах она нуждается.

На рисунке 8 показано, как информация об интерфейсе и реализации становится доступной клиентам и реализациям объектов.

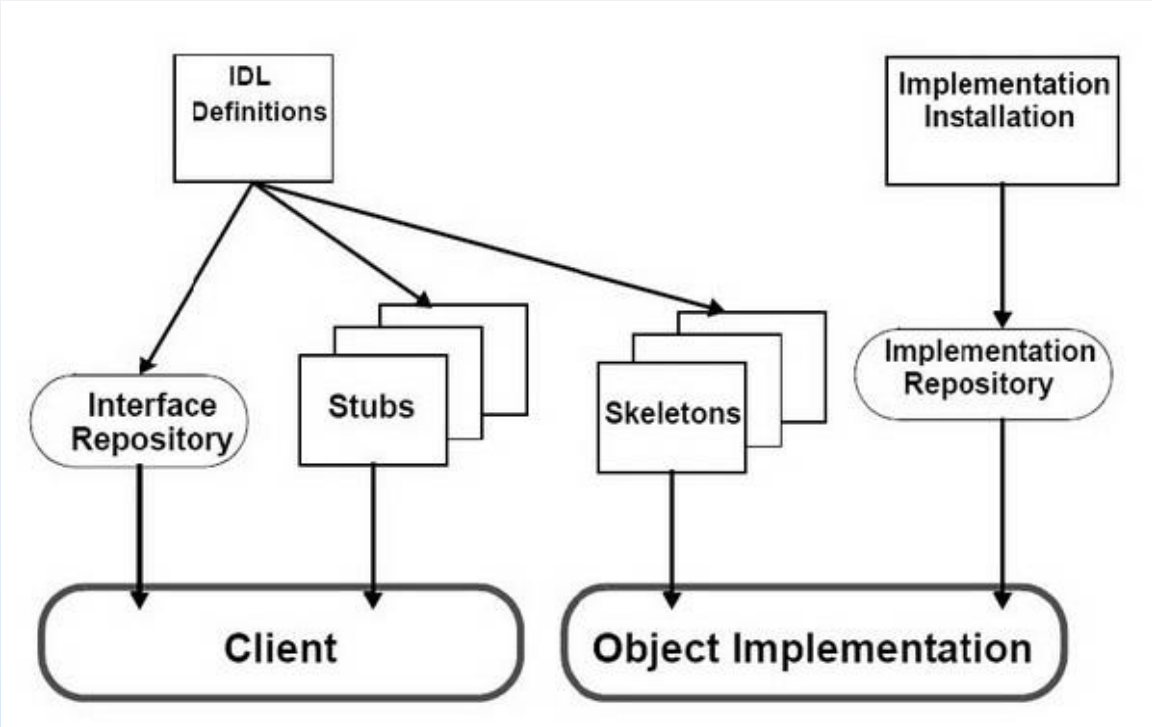

Рисунок 8 – Репозитории интерфейсов и реализаций

Интерфейсы описываются на IDL или с помощью репозитория интерфейсов. Их описания используются для генерации клиентских заглушек и скелетонов для реализации.

Информация о реализации объекта предоставляется во время инсталляции и хранится в репозитории реализации, а затем используется в процессе доставки запроса.

Брокер объектных запросов (ORB). Вместе с IDL-компилятором, репозиторями и различными объектными адаптерами, ORB предоставляет полный набор сервисов самым разным клиентам и объектным реализациям.

Ядро ORB – это часть брокера, обеспечивающая базовое представление объектов и передачу запросов. Технология CORBA может поддерживать различные объектные механизмы за счет компонентов, надстроенных над ядром и предоставляющих интерфейсы, которые позволяют скрыть различия между разными ядрами.

Клиенты. Клиент объекта имеет доступ к объектной ссылке и вызывает операции объекта. Клиент знает только логическую структуру объекта в соответствии с его интерфейсом и может наблюдать за поведением объекта через вызовы методов. Несмотря на то, что клиентом обычно считается программа или процесс, важно помнить, что понятие клиент может применяться только относительно определенного объекта. Например, реализация одного объекта может быть клиентом другого.

Клиенты обычно видят брокер через призму связывания с языком программирования. Клиенты являются максимально переносимыми и могут работать без изменения исходного кода с любым брокером, поддерживающим связывания с данным языком программирования и любым экземпляром объекта, реализующим данный интерфейс. Клиенты ничего не знают о реализации объекта, используемом ей объектном адаптере, а также о брокере, который осуществляет доступ к реализации.

Реализации объектов (Object implementation). Реализация предоставляет семантику объекта, определяя данные, хранимые в экземпляре объекта, и код его методов. Реализация может использовать другие объекты или программное обеспечения для выполнения функций объекта. В некоторых случаях, главное предназначение объекта – иметь побочные эффекты на что-то, не являющееся объектом. ORB может поддерживать множество видов реализаций объектов, включая разделенные серверы, библиотеки, «программа на метод», «инкапсулированное приложение», объектно-ориентированные базы данных и т.д.

Используя дополнительные объектные адаптеры, можно осуществлять поддержу абсолютно любого стиля реализации объектов. Вообще, реализации объектов не зависят от ORB и от того, как клиент вызывает объект. Реализации могут выбирать интерфейс брокера, меняя объектный адаптер.

Объектные ссылки (IOR). Объектная ссылка – это информация, необходимая для определения конкретного объекта внутри ORB. Как для клиента, так и для реализации объектная ссылка представляется так, как диктует связывание соответствующего языка программирования, таким образом, они изолированы от конкретного представления ссылки.

Объектная ссылка, переданная клиенту, действительна только на время жизни клиента. Различные брокеры должны предоставлять одно и то же представление объектной ссылки для данного языка программирования (это позволяет одной и той же программе получать доступ к объекту по ссылке независимо от используемого брокера). Кроме того, для удобства брокер может предоставлять другие способы доступа к объекту. Существует особенная объектная ссылка, не указывающая ни на один объект.

Язык описания интерфейсов (IDL). Появление в программировании того или иного языка обычно связано с возникновением и развитием некоей новой концепции. Так, одновременно с идеей объектно-ориентированной разработки (в ее современном варианте) был создан язык Smalltalk. Дальнейшее применение объектов и компонентов вкупе с внедрением последних в распределенные системы вызвало необходимость создания такого языка программирования, который бы позволил описать любой объект или компонент. И, что не менее важно, это описание должно быть одинаковым для любой платформы. Этим требованиям удовлетворяет язык описания интерфейсов IDL (Interface Definition Language).

Язык IDL определяет типы объектов путем спецификации их интерфейсов. Интерфейс состоит из списка операций и их параметров. Несмотря на то, что IDL предоставляет каркас для описания объектов, которыми манипулирует ORB, нет необходимости в том, что брокер имел доступ к исходному коду на IDL. Брокер может работать с эквивалентной информацией в виде заглушек подпрограмм и репозитория интерфейсов.

IDL является средством, с помощью которого реализация объекта сообщает своим потенциальным клиентам, какие операции доступны и как они могут быть вызваны. Из IDL-описания CORBA-объект можно перевести на определенный язык программирования или в другую объектную систему.

Следует заметить, что IDL не только язык, но и инструмент, с помощью которого можно сохранять метаинформацию об объектах, то есть данные о том, как устроен объект.

Известно довольно много случаев, когда язык IDL используется для описания контрактов – технических параметров, позволяющих нескольким независимым группам работать одновременно над различными частями проекта. Хотя в основном исходные тексты на IDL служат своеобразным «сырьем», из которого специальные компиляторы IDL генерируют исходные тексты на одном из языков программирования высокого уровня. Такой процесс будем называть трансляцией. Типичный процесс создания распределенных объектных приложений состоит из описания объектов на IDL, их трансляции на какой-либо язык программирования, реализации объектов на данном языке и компиляции полученных исходных текстов в готовые для запуска модули.

Приведем список терминов, используемых при описании языка IDL:

- *модуль* – блок с заданным именем, объединяющий логически связанные конструкции языка IDL; в целом, модуль можно воспринимать как пакет (package) в языке Java или пространство имен (namespace) в языке C++;

- *интерфейс* – набор атрибутов и операций объекта, с помощью которого потребитель может обращаться к объекту;

- *операция* – сущность, которую вызывают для выполнения действий, связанных с функциональным назначением объекта; операцию можно сравнить с методом класса

- *атрибут* – сущность, описывающая какое-либо свойство объекта; атрибут эквивалентен паре операций, предназначенных для чтения и записи свойства класса.

Связывание языков программирования с IDL. Различные объектно-ориентированные и не объектно-ориентированные языки программирования могут получать доступ к CORBA-объектам по-разному. Для объектно-ориентированных языков, скорее всего, предпочтительно видеть CORBA-объекты как объекты языка программирования. И даже для не объектно-ориентированных языков скрытие фактического представления объектных ссылок и методов внутри брокера представляется удобным. Связывание того или иного языка программирования с IDL должно быть одинаковым для всех реализаций ORB.

Связывание языка включает определение специфичных для языка типов данных и интерфейсов процедур для доступа к объекту через ORB. Оно включает структуру интерфейса клиентской заглушки (для объектно-ориентированных языков не обязательно), интерфейс динамического вызова, скелетон реализации, объектные адаптеры и интерфейс для обращения напрямую к брокеру.

Связывание также определяет взаимодействие между вызовами объекта и потоками выполнения в клиенте и реализации. Самые распространенные связывания предоставляют синхронные вызовы, когда управление возвращается клиенту после завершения операции. Дополнительные связывания могут возвращать управления программе сразу после инициации вызова. В этом случае должны предоставляться дополнительные подпрограммы, зависящие от языка, осуществляющие синхронизацию потоков программы и вызова объекта.

Клиентские заглушки (client stubs). Обычно клиентские заглушки предоставляют доступ к операциям объекта, описанным на IDL, способом, ожидаемым для программиста, знакомого с IDL и связыванием конкретного языка программирования. Заглушки вызывают функции остальной части ORB, используя закрытые интерфейсы, которые могут быть оптимизированы для использования с конкретной реализацией ядра брокера.

Динамический интерфейс вызова (Dynamic invocation). Также доступен интерфейс, позволяющий создавать вызовы объекта динамически, то есть, вместо того, чтобы вызывать подпрограмму заглушки, специфичную для конкретного объекта, клиент может определить объект, который требуется вызвать, операцию, которую требуется выполнить, и набор параметров путем вызова (или последовательности вызовов) универсальной функции. Клиентский код должен предоставить информацию об операции, которую требуется выполнить, включая типы передаваемых параметров (их можно получить из репозитория интерфейсов или другого источника времени выполнения). Природа динамического интерфейса вызова может значительно различаться в зависимости от связывания.

Скелетон реализации (Server skeleton). Для каждого конкретного связывания языка программирования и, возможно, в зависимости от конкретного объектного адаптера, будет создан определенный интерфейс к методам, реализующим некоторый тип объектов. При этом реализация объекта предоставляет подпрограммы, удовлетворяющие интерфейсу, а ORB вызывает эти подпрограммы через скелетон.

Из существования скелетона не следует существование соответствующей клиентской заглушки: клиент может делать запросы и через динамический интерфейс вызова. Для некоторых объектных адаптеров скелетоны могут быть не нужны: например, в таких языках как Smalltalk есть возможность создавать реализации динамически.

Динамический интерфейс скелетона. Также доступен интерфейс, позволяющий управлять вызовами объектов динамически. Вместо того, чтобы обращаться к реализации объекта через скелетон, специфичный для определенной операции, можно обратиться к реализации через интерфейс, предоставляющий доступ к имени операции и ее параметрам так же, как клиентский динамический интерфейс вызова. Для определения параметров может быть использована как чисто статическая, так и динамическая (например, предоставленная репозиторием интерфейсов) информация. Реализация должна предоставить брокеру информацию обо всех параметрах операции, брокер, в свою очередь, предоставляет значения входных параметров операции. По завершении операции, код реализации предоставляет брокеру значения всех выходных параметров или исключения.

Динамические скелетоны могут быть вызваны как клиентскими заглушками, так и динамическим интерфейсом вызова на стороне клиента, причем результат должен быть одинаковым.

Объектные адаптеры. Объектный адаптер предоставляет основной способ доступа к сервисам брокера со стороны реализации объекта. Предполагается, что будут существовать несколько общедоступных объектных адаптеров, интерфейсы которых подходят для определенных видов объектов. Сервисы, предоставляемые брокером через объектный адаптер, включают генерацию и интерпретацию объектных ссылок, вызов методов, безопасность взаимодействий, активацию и деактивацию объектов и их реализаций, сопоставление объектных ссылок реализациям и регистрацию реализаций.

Широкий диапазон уровней модульности, времен жизни, политик, стилей реализации и других свойств объектов делает невозможным предоставление ядром брокера единого интерфейса, удобного и эффективного для всех объектов. С помощью объектных адаптеров брокер может выделять группы реализаций объектов, имеющие схожие требования, и предоставлять интерфейсы, предназначенные для этих групп.

Интерфейс ORB. Это интерфейс, позволяющий обращаться напрямую к брокеру объектных запросов, он одинаков для всех брокеров и не зависит ни от интерфейса объекта, ни от объектного адаптера. Поскольку основная функциональность брокера предоставляются через объектный адаптер, заглушки, скелетоны или динамический вызов, только несколько операций могут запрашиваться напрямую. Эти операции полезны как клиентам, так и реализациям объектов.

Репозиторий интерфейсов. Репозиторий интерфейсов – это сервис, предоставляющий устойчивые объекты, отражающие IDL-информацию в форме, доступной во время выполнения. Информация из репозитория интерфейсов может быть использована брокером для осуществления запросов. Более того, используя информацию из репозитория, программа может найти объект, интерфейс которого был неизвестен во время компиляции программы, и, тем не менее, определить, какие операции могут выполняться объектом, и вызвать эти операции.

В дополнение к этой роли, репозиторий интерфейсов используется также для хранения дополнительной информации, связанной с интерфейсами объектов брокера. Например, отладочной информации, библиотек заглушек и скелетонов, и т.д.

Репозиторий реализаций. Репозиторий реализаций содержит информацию, которая позволяет брокеру находить и активировать реализации объектов. Большая часть информации в репозитории специфична для конкретного ORB и рабочей среды. Обычно, инсталляция реализаций и управление политиками, связанными с активацией и исполнением реализаций, выполняется через операции с репозиторием реализаций.

Репозиторий реализаций используется также для хранения дополнительной информации, связанной с реализациями объектов (отладочная информация, административный контроль, выделение ресурсов, безопасность и т.д.).

4. Создание CORBA-систем

Для того чтобы создать CORBA-систему, сначала необходимо установить и настроить соответствующий инструментарий. В этом разделе будет вкратце описаны основные действия на каждом этапе разработки, а затем в разделе с примерами эти действия будут рассмотрены на конкретном примере.

Инструменты и их конфигурирование. Среди наиболее популярных и доступных инструментов для создания CORBA-систем брокер для Java от Sun Microsystems, входящий в стандартную поставку Java, VisiBroker от Inprise/ Borland, WebLogic.

Порядок действия при создании CORBA-системы. Создавая CORBA-приложения, нужно помнить, что их модель отличается от модели традиционных монолитных программ и даже клиент-серверных систем, хотя с последними есть и нечто общее. Связку объектов CORBA и клиентов трудно назвать приложением как таковым. Подобные системы похожи на паутину, где все переплетено: клиент может в любую минуту стать сервером, и пользователь вряд ли узнает, с каким сервером объектов он работает в данный отрезок времени, а если проект выполнен грамотно, может даже и не заметить сбоя.

Типичная тактика действий программы, использующей технологию CORBA, такова: соединиться с нужным объектом, использовать его функции и отсоединиться от него. И таких атомарных циклов могут быть сотни.

Добиться хороших результатов в создании программ на основе CORBA можно, придерживаясь определенного порядка действий:

- объектно-ориентированный анализ и моделирование;
- описание и трансляция объектов;
- создание сервера;
- создание клиента;
- отладка объектов.

Объектно-ориентированный анализ и моделирование. CORBA – объектно-ориентированная технология, потому в первую очередь необходимо осуществить объектную декомпозицию и представить систему в виде взаимодействующих между собой классов.

Чтобы модель была понятна и разработчикам, нужно задокументировать ее. Построить IDL-описания по UML-модели поможет пакет Rational Rose.

Следует разработать порядок действий, в соответствии с которым будет создаваться реализация объектов.

Затем следует выделить в готовой модели атомарные объекты, не зависящие от других, они и станут кандидатами на первоочередное создание и отладку.

Неплохо подумать и о размещении объектов в сети, согласуясь с топологией последней.

В итоге образуется четкая последовательность инсталляции готового кода, определятся виртуальные домены.

Описание и трансляция интерфейсов. Готовая модель системы содержит классы, которые должны быть описаны с помощью языка IDL. Далее это описание можно транслировать с помощью IDL-компилятора в базовые исходные тексты на конкретном языке программирования (заглушки и скелетоны) или добавить IDL-описания в репозиторий интерфейсов.

Выводы

В ходе лекции рассмотрены следующие вопросы:

- основные понятия CORBA;

- технология CORBA;

- архитектура CORBA;

- создание CORBA-систем.

Задание на самостоятельную работу

1. Конспект лекций.

Вид и тема следующего занятия

Практическое занятие №14. URL Rewriting