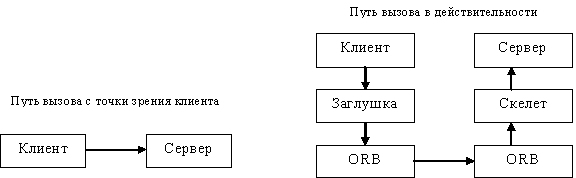
**Лекция 6**

Основные концепции CORBA, обсуждавшиеся до сих пор, должны быть близки тем разработчикам, кто хорошо знаком с шаблонами проектирования. Заглушки и скелеты являются посредниками (объектами, управляющими доступом к другим объектам). Сервисы, возвращающие объекты, являются *фабриками* (объектами, создающими объекты, но перекладывающими их реализацию на производные классы). В CORBA фабрики и посредники могут быть обнаружены повсюду. Те классы и объекты, которые применяют паттерны проектирования, скрывают от разработчиков детали реализации. Паттерны проектирования унифицируют проектные решения и терминологию. Изучение паттернов проектирования помогает разработчикам повторно использовать проекты, компонент, сервисы и структуры.

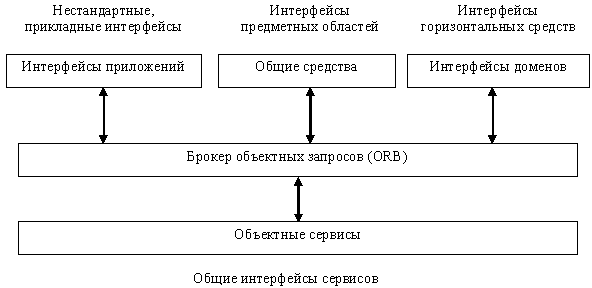
Брокер объектных запросов ( ORB ) - основной компонент CORBA. Все CORBA -совместимые объекты должны иметь брокер объектных запросов между ними и всеми, кто к ним обращается ([рис. 7.1](https://www.intuit.ru/studies/courses/633/489/lecture/11082?page=1#image.7.1)). Брокер объектных запросов должен иметься в CORBA -совместимой распределенной системе.



**Рис. 7.1.** Путь вызова от клиента к удаленному объекту

ORB можно рассматривать как коммутационную плату (или коммуникационную шину) распределенных систем. Что касается коммуникационной шины, то суть здесь заключается в том, что все объекты, использующие брокер объектных запросов, могут взаимодействовать с любыми другими объектами, подключенными через брокер объектных запросов. В чем преимущество использования брокеров объектных запросов? Брокеры объектных запросов представляют собой гибкую конструкцию, но какие все же они решают проблемы? На оба эти вопроса консорциум OMG дает ответ в Object Management Architecture (архитектуре управления объектами).

Типичная программная система призвана удовлетворять различные потребности. Унаследованные системы (то есть любые системы, разработанные и установленные ранее) обычно являются изолированными, сосредоточенными, и не предназначены для совместного использования информационных ресурсов. Они решают узкую задачу, даже если решение требует больших объемов данных. Со временем, по мере модификации систем, их модернизация обходится все дороже. За последние 30-40 лет эти системы стали практически несовместимы. Развитие аппаратных и программных средств, изменение способов ведения бизнеса приводили к появлению множества несовместимых систем, пока системные интеграторы не занялись проблемой их связи. Переход от частных решений к интеграции лишь подчеркнул необходимость осуществлять интеграцию не за счет дублирования усилий, а за счет новых решений. Консорциум OMG, вместе с компаниями-производителями, являющимися членами OMG, разработал Object Management Architecture ([рис. 7.2](https://www.intuit.ru/studies/courses/633/489/lecture/11082?page=1#image.7.2)).



**Рис. 7.2.** Эталонная модель архитектуры управления объектами (Object Management Architecture)

Object Management Architecture ( OMA ) - это эталонная архитектура распределенных систем, основанная на концепции брокера объектных запросов. Используя концепции объектно-ориентированных технологий, OMA определяет к использованию рабочее пространство, где объекты, по определению являющиеся общедоступными, открыты для использования любым другим объектом или сервисом посредством объектного брокера. Объектный брокер - это прозрачный коммуникационный механизм, обеспечивающий надежный обмен сообщениями между объектами независимо от их местоположения. OMA определяет абстракцию, которая скрывает тот факт, что различные системы применяют разные языки программирования или несовместимые версии одного и того же языка.

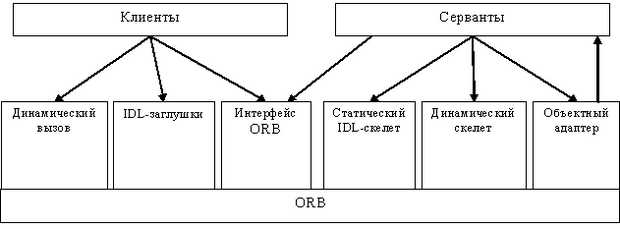
CORBA определяет правила функционирования брокера объектных запросов в условиях использования разных языков программирования. OMA определяет полиморфное рабочее пространство общих сервисов, которое выглядит однородным извне (со стороны API ), но может быть разнородным внутри.

Брокеры объектных запросов могут быть реализованы одним из двух способов: в виде библиотек или как процессы-демоны. Ни для клиента, ни для серванта средства реализации не имеют значения. Конструкция объекта брокера объектных запросов скрывает лежащую в ее основе реализацию. Обычно клиент использует брокер объектных запросов на основе библиотеки, а сервер использует ORB в виде процесса-демона. Это деталь реализации, решение о которой принимают администраторы систем. С позиции активного процесса ничего не меняется.

Брокер объектных запросов играет в OMA основную роль. Теперь мы рассмотрим, как клиент и сервант воспринимают брокер объектных запросов. Клиент взаимодействует с брокером объектных запросов одним из трех способов: посредством статической заглушки (создаваемой IDL -компилятором), динамического интерфейса (применяя API динамических вызовов CORBA ) или API брокера объектных запросов. Концептуально брокер объектных запросов взаимодействует с сервантом тремя способами: посредством статического скелета, динамического интерфейса или объектного адаптера серванта (что выглядит так, как если бы брокер объектных запросов взаимодействовал с сервантом напрямую). Сервант, обращающийся к другому серванту, становится клиентом и функционирует как клиент.

Наиболее прямолинейным способом взаимодействия с брокером объектных запросов является использование статических заглушек и скелетов. Они содержат код, необходимый для связи, и предоставляют возможность статического контроля типов на основе их IDL -описания. Динамические вызовы (и со стороны клиента, и со стороны серванта) требуют больших издержек, но являются более гибкими, так как позволяют разработчикам программно управлять вызовами удаленных объектов.

Та опосредованность, которая придает CORBA ее силу, в случае прямых вызовов брокера объектных запросов теряется, и восстановить ее в процессе реализации системы бывает трудно. Клиент и сервант взаимодействуют с брокером объектных запросов, чтобы получить доступ к определенного вида операциям, осуществить которые можно только через брокер объектных запросов, - операциям с объектными ссылками и доступом к репозиториям интерфейсов и реализаций (двум информационным хранилищам метаданных). Низкоуровневые решения такого рода в CORBA требуются только при реализации средств поддержки инфраструктуры, таких как драйверы и мосты. [Рис. 7.3](https://www.intuit.ru/studies/courses/633/489/lecture/11082?page=1#image.7.3) иллюстрирует взаимодействие с объектным брокером.



**Рис. 7.3.** Структура интерфейса запросов ORB

Концепции CORBA, рассматривавшиеся ранее, касались объектных адаптеров - объектов, которые располагаются между клиентом и сервером, чтобы управлять доступом к распределенному объекту. Объектный адаптер действует как "соединитель" между клиентом и кодом серванта, который выполняется при вызове операции (брокер объектных запросов располагается между клиентом и объектным адаптером). До появления CORBA 3.0 стандартным объектным адаптером был Basic Object Adapter (базовый объектный адаптер), или BOA. BOA упрощал соединение клиента и сервера. В CORBA 3.0 был определен другой объектный адаптер, названный Portable Object Adapter ( POA - переносимый объектный адаптер). POA заменил BOA как более предпочтительный. BOA не соответствовал требованиям, которые предъявляют Internet -приложения. В то время, когда OMG впервые специфицировал BOA, то, что сейчас рассматривается как стандартная функциональность (т.е. возможность использования средств CORBA разных производителей), тогда не воспринималось как первоочередная задача.

Так же как Java вытесняет различные технологии, спецификация CORBA заменила BOA на POA.

POA служит многим целям, включая возможность отделить доступ к серванту от самого серванта. Потребность клиента в обслуживании означает, что ему нужны конкретные услуги в конкретное время. Для удовлетворения этой потребности несколько компонентов осуществляют совместные действия. Во-первых, у клиента есть объектная ссылка, представленная CORBA -объектом. В объектной ссылке CORBA -объект содержит информацию о местонахождении создавшего данный объект объектного адаптера. Объектный адаптер анализирует клиентский вызов, принимая решение, какой объект (или сервант) может обработать данный вызов, и соответствующим образом завершает вызов (основываясь на различных опциях конфигурации, задаваемых при создании объектного адаптера). Если клиент удерживает полученную им при первом подключении к серванту объектную ссылку в течение длительного периода времени и не нуждается в обращениях к этому серванту, то такое ожидание никак не скажется на самом серванте. Пострадает масштабируемость, превращая CORBA в неэффективное с точки зрения системной интеграции решение. Благодаря отделению серванта от клинических обращений к сервису различные обслуживающие объекты (контролируемые с помощью времени жизни объектов и паттернов активизации) могут обрабатывать вызовы методов, обращенные к сервису в целом. Последнее утверждение заключает в себе целый ряд аспектов. Опосредованность, достигаемая использованием POA, означает прозрачное решение этих вопросов.

CORBA не ограничивается использованием в приложениях с заранее известными межмодульными интерфейсами.

Статические заглушки, использующие Static Invocation Interface ( SII ), имеют жестко запрограммированные объектные типы, чтобы осуществлять проверку типов во время компиляции, тогда как динамически заглушки, использующие Dynamic Invocation Interface ( DII ), осуществляют проверку типов во время выполнения.

Сервисы CORBA ( CORBAservices ) - это базовые сервисы, которые доступны всем объектам, подключенным к коммуникационной шине данного брокера объектных запросов. Так как брокер объектных запросов является ядром CORBA -системы, сервисы CORBA могут требовать для правильного функционирования его наличия. Всего имеется шестнадцать сервисов:

1. сервис имен ( Naming Service );
2. сервис управления событиями ( Event Management Service );
3. сервис жизненных циклов ( Life Cycle Service );
4. сервис устойчивых состояний ( Persistent State Service );
5. сервис транзакций ( Transaction Service );
6. сервис параллельного исполнения ( Concurrency Service );
7. сервис взаимоотношений ( Relationship Service );
8. сервис экспорта ( Externalization Service );
9. сервис запросов ( Query Service );
10. сервис лицензирования ( Licensing Service );
11. сервис управления ресурсами ( Property Service );
12. сервис времени ( Time Service );
13. сервис безопасности ( Security Service );
14. сервис уведомлений ( Notification Service );
15. сервис трейдинга ( Trader Service );
16. сервис коллекций ( Collections Service ).

Эти сервисы являются основными. Все сервисы CORBA имеют стандартные IDL -интерфейсы, которые описывают предлагаемые этими сервисами услуги. Назначение стандартных интерфейсов точно соответствует еще одной цели OMA: подключаемые сервисы должны иметь стандартные механизмы подключения.