Service Oriented Architecture (SOA)

Перевод статьи: [Service Oriented Architecture (SOA)](https://herbertograca.com/2017/11/09/service-oriented-architecture-soa/) ([Herberto Graca](https://herbertograca.com/))

SOA существует с конца 1980-х годов и основывается на идеях таких подходов как CORBA, DCOM, DCE. Много было сказано о SOA, и есть несколько различных шаблонов её реализации, но, по сути, SOA фокусируется только на нескольких концепциях и не дает никакого рецепта их реализации:

* Совместимость пользовательских приложений;
* Повторное использование бизнес-сервисов;
* Независимость от технологического стэка;
* Автономность (независимое развитие, масштабирование, развертывание).

SOA – это набор архитектурных принципов, независимых от какой-либо технологии или продукта, как полиморфизм или инкапсуляция.

В этой статье я собираюсь осветить следующие шаблоны, связанные с SOA:

* CORBA – Common Object Request Broker Architecture
* Web Services
* Message Queue
* Enterprise Service Bus (ESB)
* Microservices

**CORBA – Common Object Request Broker Architecture**

В 1980-х годах, в связи с растущим использованием корпоративных сетей и клиент-серверной архитектуры, возникла потребность в общем способе взаимодействия приложений, которые были построены с использованием различных технологий и работали на разных компьютерах с разной ОС. CORBA была разработана для удовлетворения этой потребности. Это стандарт для распределенных вычислений, разработанный в 1980-х годах и достигший своей первой зрелой версии 1991 года.

Стандарт CORBA был реализован несколькими поставщиками и направлен на обеспечение:

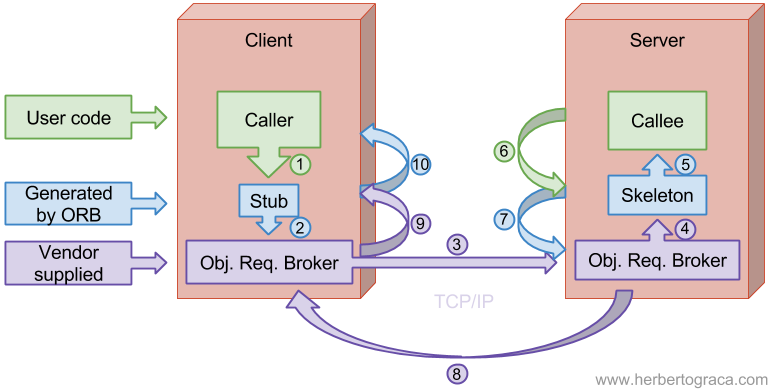
* Платформанезависимого RPC (Remote Procedure Call);
* Транзакций (также распределнных транзакций);
* Безопасности;
* Событии;
* Независимости от языка программирования;
* Независимости от ОС;
* Независимости от оборудования;
* Изоляции от детплей взаимодействия/передачи данных.

На данный момент стандарт CORBA все еще используется для распределенных вычислений. Например, он все еще является частью JAVA EE, несмотря на то, что будет упакован как отдельный модуль в JAVA 9 и далее.

Важно отметить, что я не считаю CORBA шаблоном SOA (хотя я считаю, что оба шаблона и CORBA и SOA подходят для организации распределенных вычислений). Я решил включить CORBA здесь, потому что я чувствую, что именно его недостатки привели к развитию SOA.

**Как стандарт COBRA работает**

Во-первых, нам нужно получить Object Request Broker (ORB), который соответствует спецификации CORBA. Он реализуется поставщиком и используется для создания заглушек (Stub’ов) и скелетов (Skeleton’ов) на языках клиентского кода. Используя ORB и определение интерфейса, выраженного с помощью IDL (аналогично WSDL), на стороне клиента мы генерируем Stub-классы реальных классов, которые могут быть вызваны удаленно. В тоже время на сервере мы генерируем Skeleton-классы, которые могут обрабатывать входящие запросы и вызывать методы целевого объекта.



1. Объект на стороне клиента (Caller) вызывает локальную процедуру, реализованную Stub’ом;
2. Stub проверяет вызов и создает сообщение запроса, которое передается ORB’y;
3. ORB на строне клиента отправляет сообщение по сети на сервер и блокирует текущий поток выполнения;
4. ORB на строне сервера получает сообщение запроса и создает экземпляр Skeleton’а;
5. Skeleton выполняет процедуру на вызываемом объекте (Callee);
6. Вызываемый объект (Callee) выполняет вычисление и возвращает результат;
7. Skeleton упаковывает выходные аргументы в ответное сообщение и передает его ORB’y;
8. ORB на стороне сервера отправляет сообщение по сети обратно клиенту;
9. ORB на строне клиента получает ответное сообщение, распаковывает его и доставляет Stub’у;
10. Stub передает выходные аргументы вызывающему объекту (Caller), освобождает поток выполнения, и вызывающий продолжает выполнение.

**Достоинства**:

* Независимость от технологического стека;
* Изоляции от детплей взаимодействия/передачи данных.

**Недостатки**:

* **Неопределнность расположения вызываемого объекта**: код клиента не знает, является ли вызов локальным или удаленным. Это звучит как хорошая вещь, однако, задержка и типы сбоев совершенно разные, в зависимости от типа вызова. И это делает невозможным для приложений выбор соответствующей стратегии для обработки вызова метода, и в конечном итоге может возникнуть ситуация, когда удаленные вызовы произодятся внутри цикла, что значительно замедляет всю систему.
* **Сложная, избыточная и неоднозначная спецификация**: она создавалась как месиво из нескольких существующих версий вендора, поэтому (в то время) она была неоднозначной и избыточной, что затрудняло ее реализацию.
* **Заблокированные каналы связи**: COBRA использовал определенные протоколы передачи данных через TCP/IP и определенные порты (или даже случайные порты). Но корпоративные правила безопасности и межсетевые экраны часто разрешают связь только через HTTP по порту 80, эффективно блокируя каналы связи, необходимые для CORBA.