Распределенные вычислительные системы

Агентные технологии, мобильные агенты, CORBA

Программные Агенты

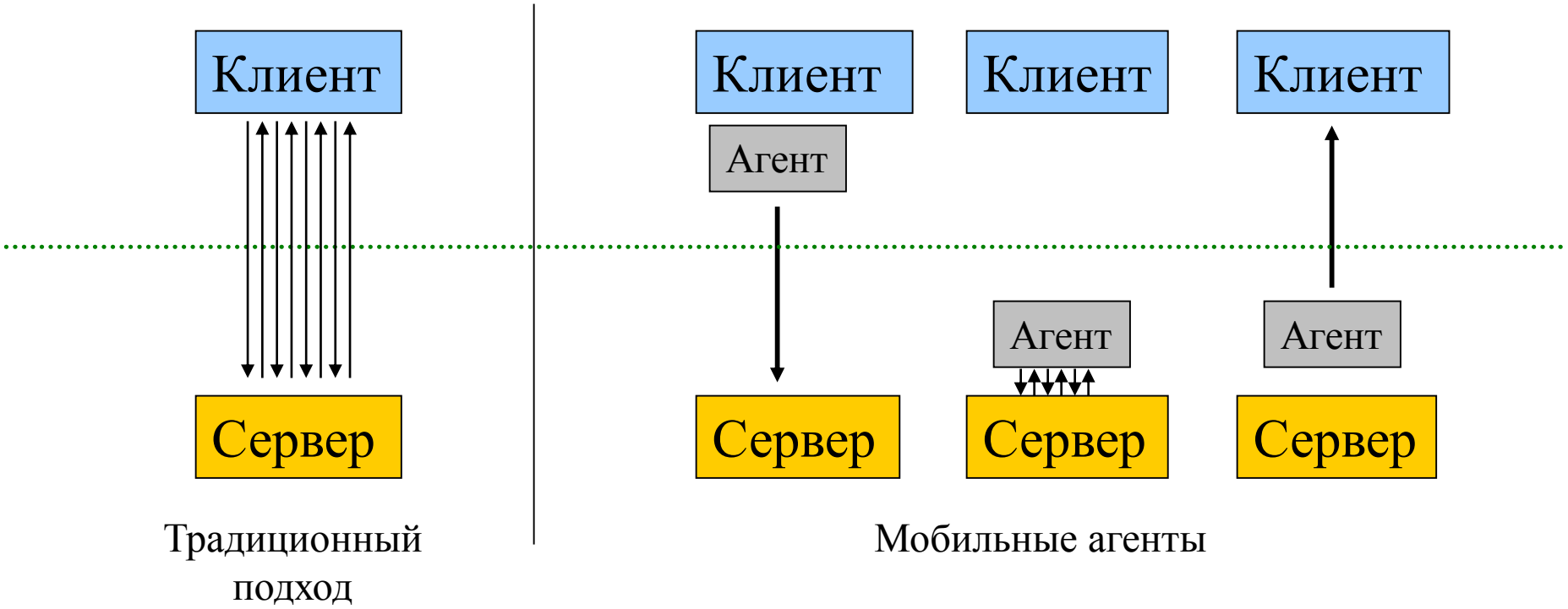
Программный Агент

- Агенты – это скорее парадигма, а не конкретная технология.
- **Программный агент** – автономный процесс, способный реагировать на среду исполнения и вызывать изменения в среде исполнения, возможно, в кооперации с пользователями или другими агентами.
- Программные агенты обладают некоторыми знаниями об окружающем мире, которые позволяют им самим решать проблемы без вмешательства пользователя.

Многоагентная система

- **Многоагентная система** (*МАС, англ. Multi-agent system*) — это система, образованная несколькими взаимодействующими агентами.
- Многоагентные системы могут быть использованы для решения таких проблем, которые сложно или невозможно решить с помощью одного агента или монолитной системы.

Клиент-сервер VS Агенты



Приемущества мобильных агентов

- Уменьшение объема данных передаваемых по сети
- Возможность автономного исполнения
  - Малое время взаимодействия On-Line
  - Малые требования к энергии
  - Поддержка мобильных устройств

Примеры использования агентских систем

- мобильные вычисления;
- задачи управления информацией;
- поиск информации;
- отбор (обработка) информации;
- мониторинг данных;
- универсальный доступ к данным.

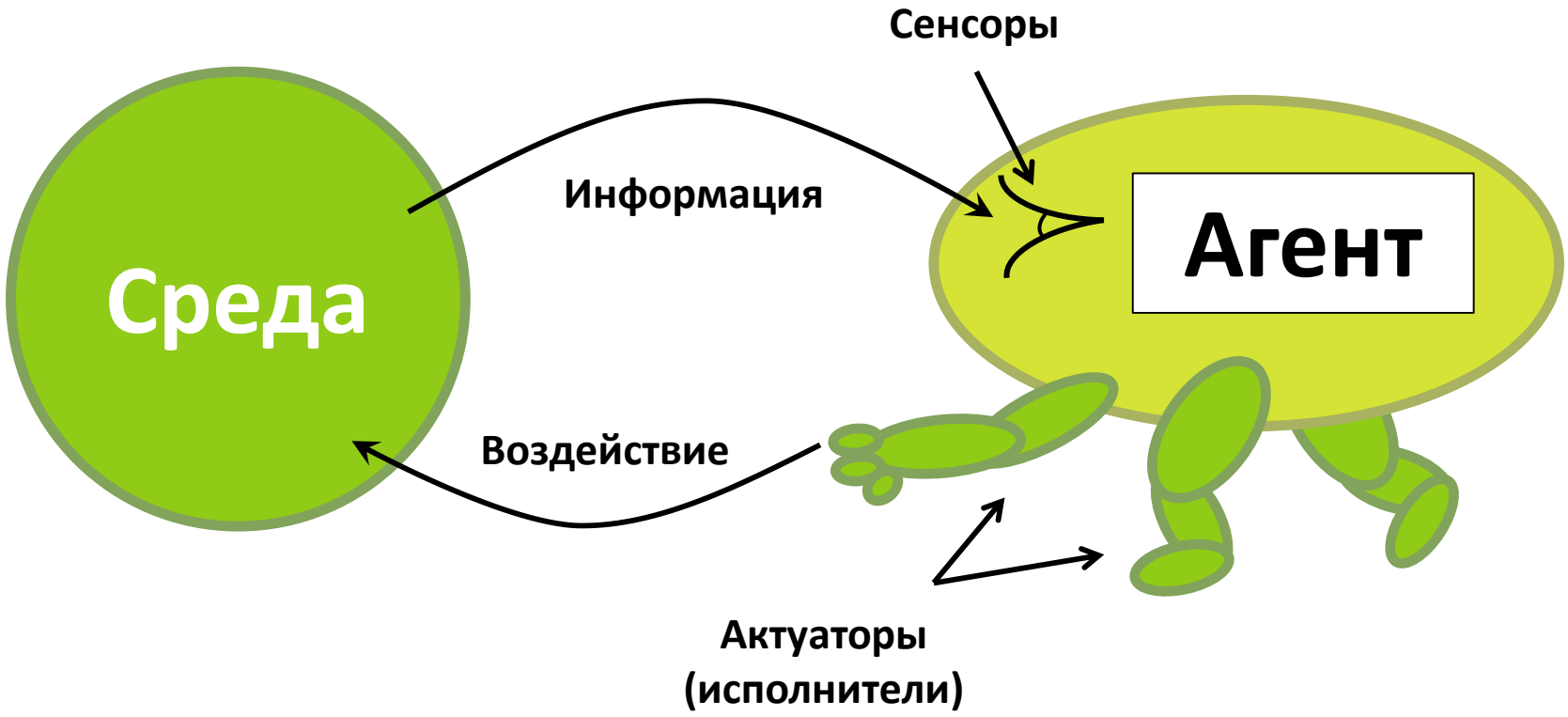
Основные характеристики агента

- **Автономность:** каждый агент является независимой программной системой (хотя бы частично)
- **Ограниченность представления:** ни у одного из агентов нет представления о всей системе
- **Децентрализация:** нет агентов, управляющих всей системой

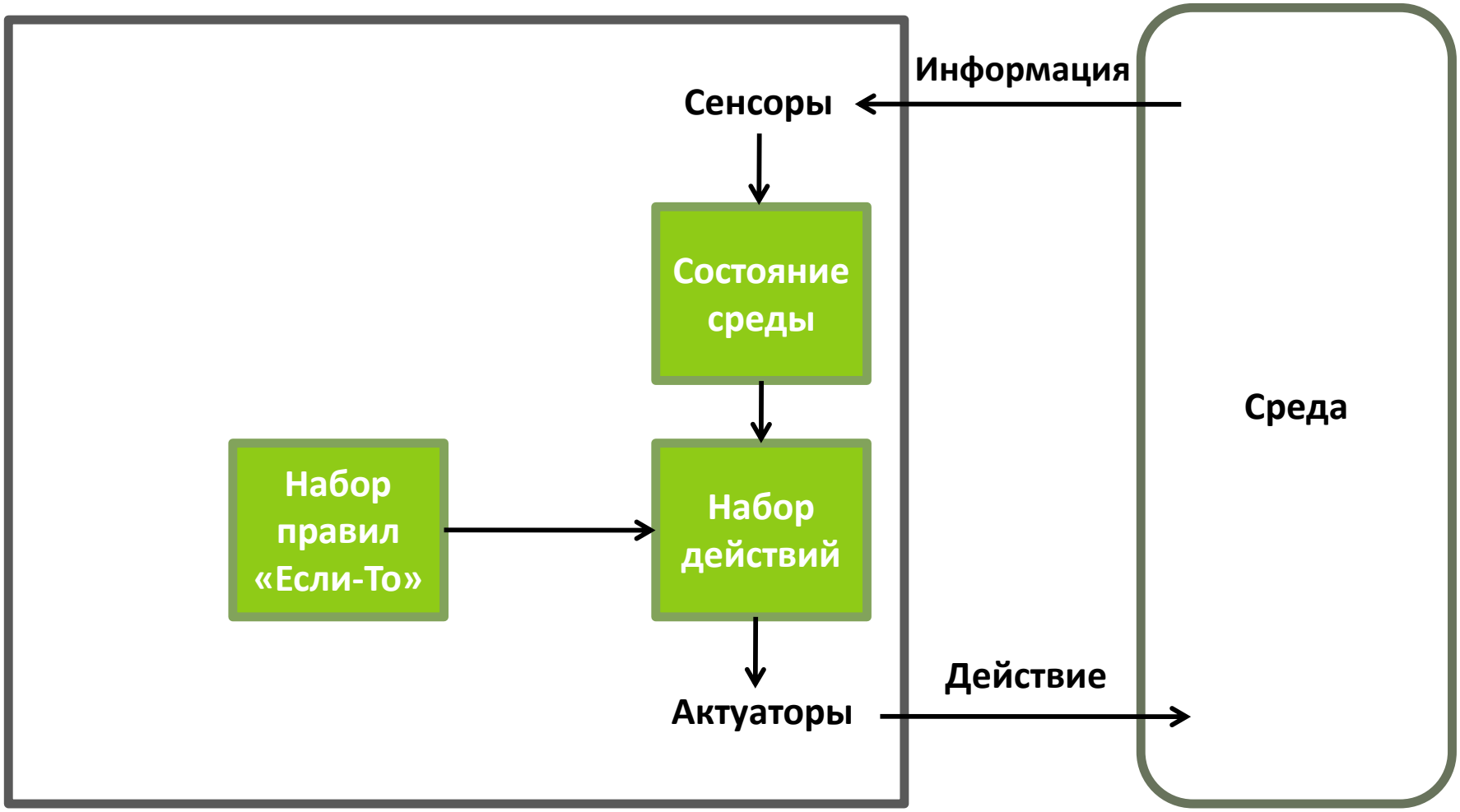
Автономный агент

- **Автономный агент** – это система, находящаяся внутри окружения и являющаяся его частью, воспринимающая это окружение (его сигналы) и воздействующая на окружение для выполнения собственной программы действий.

Автономный агент



Структура автономного агента



Свойства автономного агента

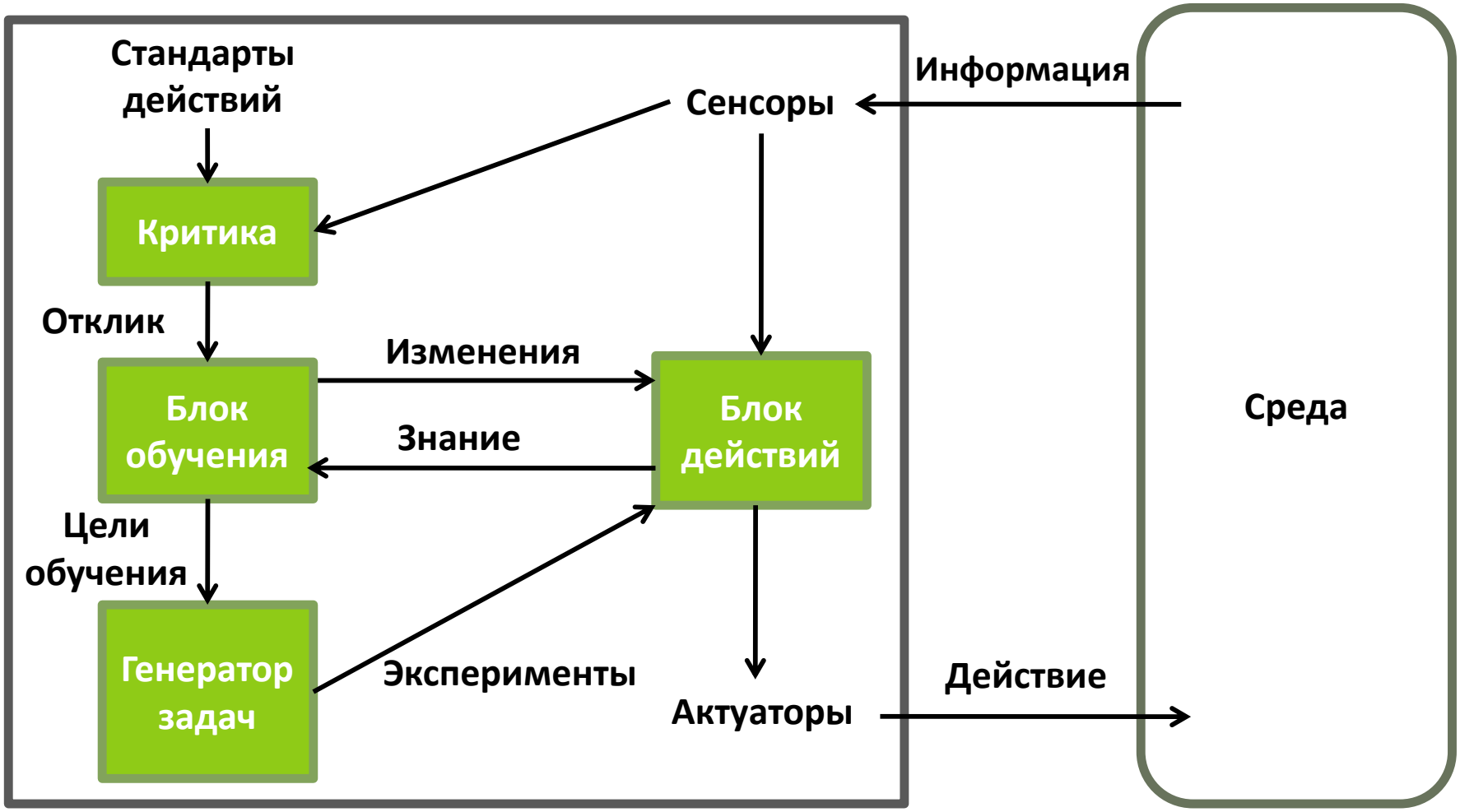
- **Реактивность** – агент временами отвечает на изменения в окружении.
- **Автономность** – агент является самоуправляющимся, сам контролирует свои действия.
- **Целенаправленность** – у агента имеется определенная цель и его поведение (воздействие на окружение) подчинено этой цели. Агент является управляющей системой, а не управляемым объектом.
- **Коммуникативность** – агент общается с другими агентами (включая людей), используя для этого некоторый язык.

Интеллектуальный агент

- **Интеллектуальный агент** — это автономный агент, обладающий свойством обучаемости.

- **Обучаемость** – агент может корректировать свое поведение, основываясь на предыдущем опыте.

Структура интеллектуального агента



Мобильные агенты и многоагентные системы

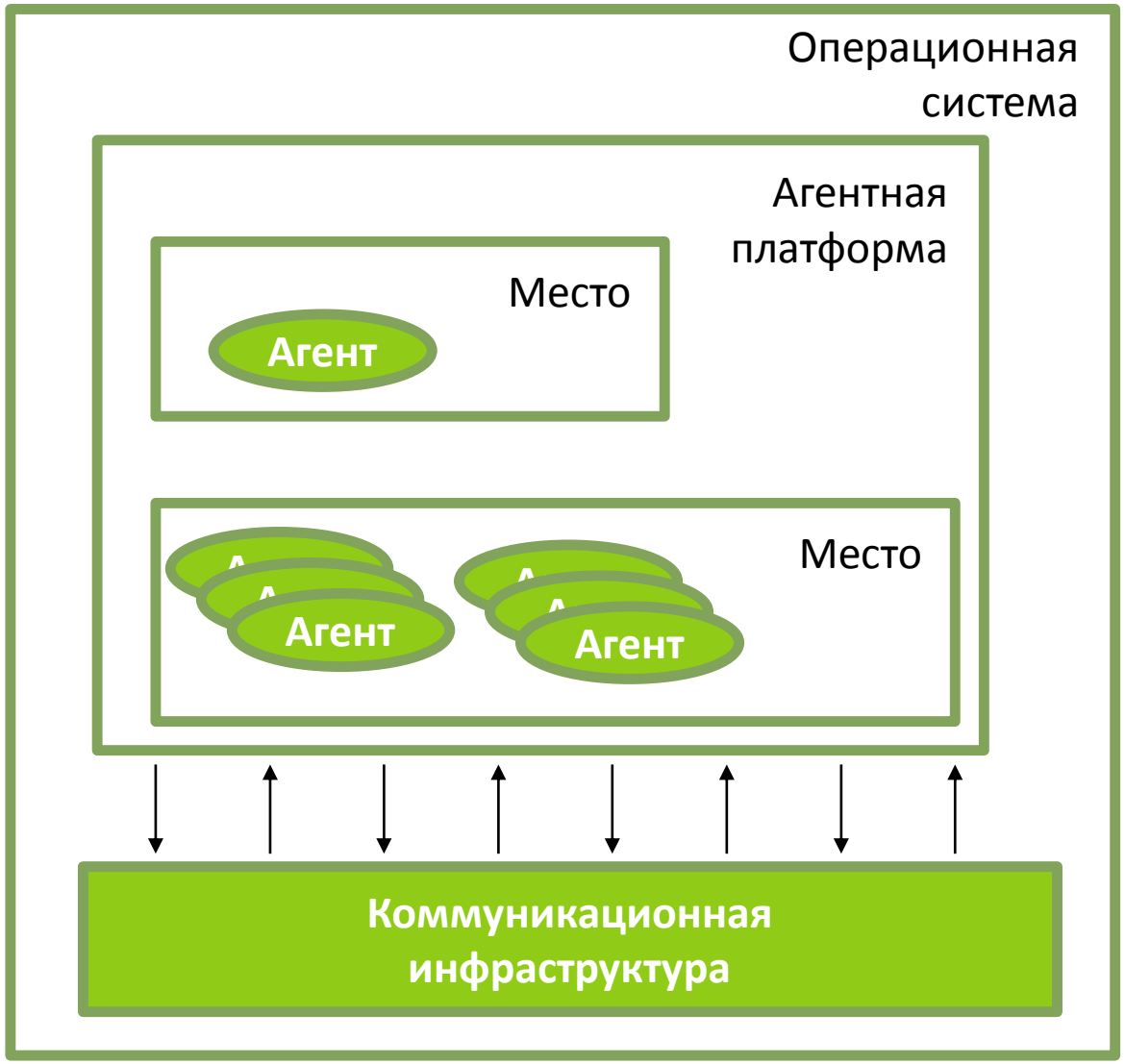
Агентная платформа

- **Агентная платформа** – программная оболочка, которая может создавать, интерпретировать, запускать, перемещать и уничтожать агенты.
- Как «воздух» для агента, обеспечивая ему среду для исполнения.

Возможности агентных платформ

- Агентная платформа предоставляет:
  - Среду исполнения агентов
  - Локационную информацию
  - Набор доступных ресурсов
  - Механизм аутентификации

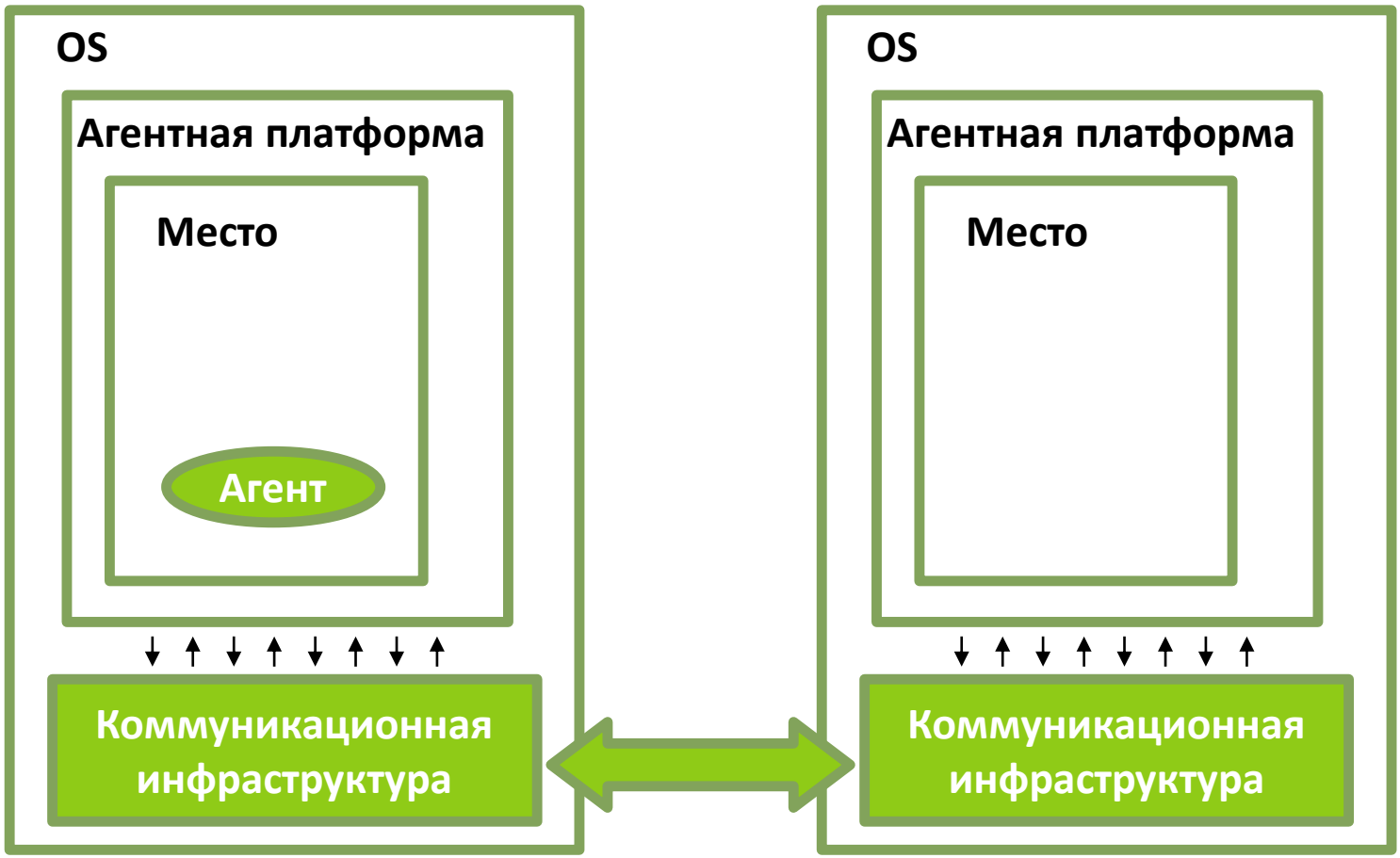
Агентная платформа



Мобильные агенты

- **Мобильные агенты** – агенты, обладающие свойством мобильности, т.е. возможностью транспортировки себя с одной вычислительной системы на другую.
- Мобильный агент может менять свое положение в окружении.

Связи между агентными платформами



Среды мобильных агентов

- Java
  - Jade и др.
- TCL
  - D’Agents, SMIA
- C/C++
  - Omniware

Коммерческие агентные системы

- Telescript/Odyssey - General Magic
- Voyager - ObjectSpace
- Aglets - IBM
- Concordia - Mitsubishi Electric ITA
- Jumping Beans – AdAstra

Преимущества Java

- Платформо-независимая
- Исполнение в рамках виртуальной машины
  - Важно для изолированного исполнения
- Адекватная система безопасности
  - Хотя все равно не идеальная!

Безопасность в системах мобильных агентов

Проблемы безопасности в мобильных системах

- Агенты могут нести личную информацию
  - например, агент в системе электронной коммерции может содержать номер кредитной карточки и паспортные данные
- Платформа должна предоставить защищенную среду для исполнения агента
- Агент может попытаться атаковать базовую среду и извлечь данные или заполучить иные ресурсы
- Сложности централизованного управления безопасностью

Классификация угроз в агентных системах

- **Агент атакует Хост**
  - Агент может украсть или модифицировать данные хоста
- **Хост атакует Агента**
  - Хост может украсть или модифицировать данные Агента, изменить его состояние или код
- **Злонамеренный агент атакует другого агента**
  - Злонамеренный агент может вмешаться в работу другого агента
- **Атака другими элементами**

Агент атакует хост: меры обеспечения безопасности

- **Традиционные средства**:
  - контроль уровня доступа,
  - песочницы,
  - аутентификация,
  - криптография
- **Анализ истории движения** агента: платформа может узнать о истории передвижения агента из лога и сделать вывод о его качестве

Хост атакует агента: меры обеспечения безопасности

- Традиционные средства **не работают**, т.к. хост должен иметь полную информацию о коде агента для его исполнения
- **Мобильная криптография**: функции и данные агента шифруются таким образом, что хост не может разобрать каким образом функции работают и извлечь код.
  - Недостаток: необходимость поиска схемы шифрования для произвольных функций, а также необходимость переноса ключа кадрирования.
- **Безопасное перемещение**: миграция только на определенные (доверенные) хосты.

Хост атакует агента: меры обеспечения безопасности (2)

- **Использование фиктивных данных**: в базе данных, системы, анализирующей работу агентов, хранится набор фиктивных данных, которые не изменяются в ходе нормальной работы агента.
- **Использование доверенного аппаратного обеспечения:** это могут быть смарткарты, интегрированные микросхемы, и т.п.

CORBA: объектно-ориентированные гетерогенные РВС

Ранние технологии построения распределенных систем

- Удаленный вызов процедур
  - Сильно завязан на базовом протоколе
  - Сильно завязан на семантике языка
  - Пропиретарные протоколы
- Ориентированы на гомогенные вычислительные среды
- Обычно ориентированы на связь 1 к 1
  - Не высоко масштабируемы

Эволюция РВС

- Объектно-ориентированный дизайн РВС
  - Связан с расцветом ОО-подхода
- Многозвенные архитектуры
- Внедрение “унаследованных” приложений
- Интеграция корпоративных приложений
- Работа на базе TCP/IP
- Интернет-вычисления

Object Management Group (OMG)

Object Management Group (OMG)

- OMG (Object Management Group) — консорциум, занимающийся разработкой и продвижением объектно-ориентированных технологий и стандартов.
  - Это некоммерческое объединение, разрабатывающее стандарты для создания корпоративных платформо-независимых приложений.
- В него входит более 800 крупнейших производителей ПО
- Не отвечает за разработку реализаций своих стандартов

Архитектура управления объектами

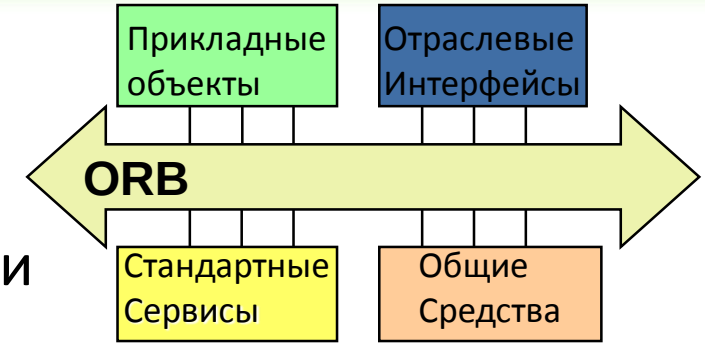
- Архитектура управления объектами (Object Management Architecture – OMA) – это две основные модели, на которых основывается технология OMG
- Core Object Model
  - Абстрактное определение объекта
  - Определяет как брокер запросов объектов (ORB или Object Request Broker) взаимодействует с PBC
- Reference Object Model
  - Архитектура инфраструктуры PBC

Core Object Model

- Переносимый дизайн
  - Взаимодействие с объектом на основе интерфейса
  - Независимый от платформы и языка доступ
- Обеспечение взаимодействия
  - Возможность инициировать операции других объектов независимо от их расположения, платформы, языка или реализации

Reference Object Model

- Определяет интерфейсы к инфраструктуре и сервисам удаленных объектов
- Руководство для разработчиков и поставщиков
- Определяет пять основных компонентов:
  - Брокер Объектных Запросов (ORB - Object Request Broker)
  - Стандартные сервисы *(Services)*
  - Общие средства *(Facilities)*
  - Отраслевые интерфейсы *(Domains)*
  - Прикладные объекты *(Application Objects)*



CORBA

Common Object Request Broker Architecture

Что такое CORBA?

Common Object Request Broker Architecture — общая архитектура брокера объектных запросов. Включает в себя:

- архитектуру для совместимых распределенных вычислений
- протокол взаимодействия на основе Интернета (IIOP)
- методы отображения языков (OMG IDL)
- комплексные и многократно используемые сервисы

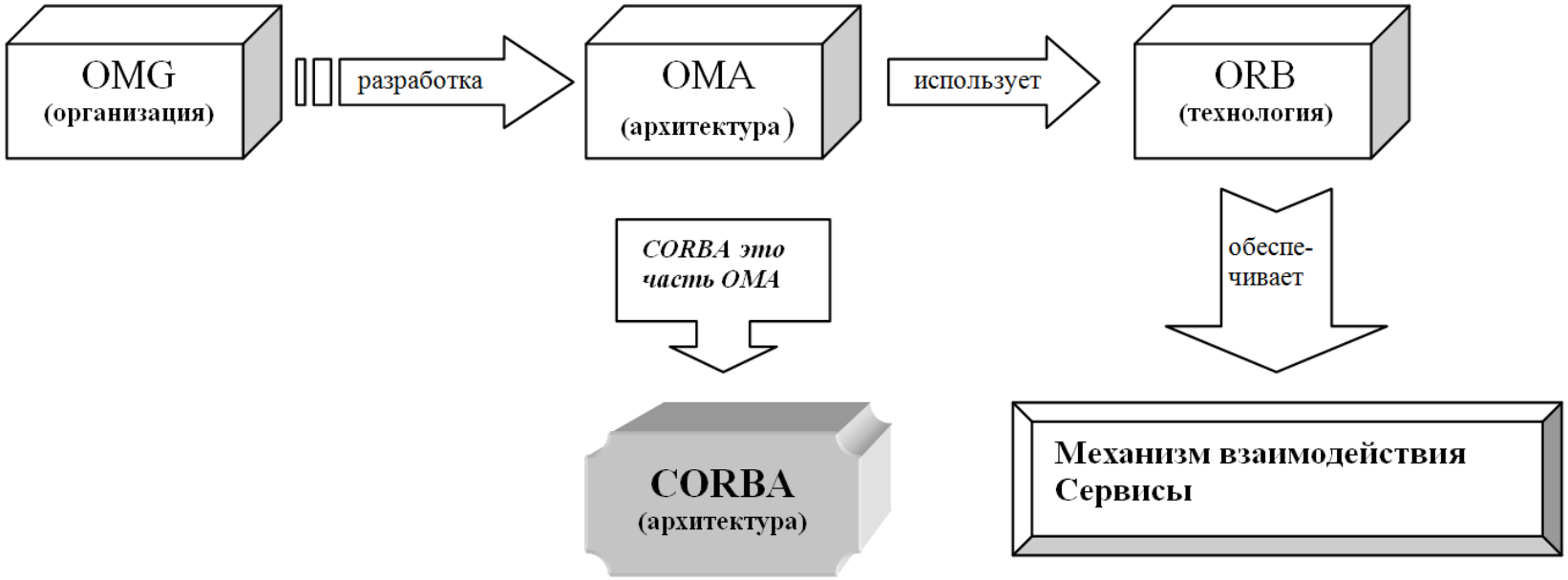
История CORBA

- Object Management Group основана в 1989
- CORBA 1.x (91-93)
  - Экспериментальная, сосредоточена на архитектуре, IDL
- CORBA 2.0 (1996)
  - Функциональная совместимость и интеграция с COM
- CORBA 2.1 (1997)
  - Безопасность, отображение языков

История CORBA (2)

- CORBA 2.2 (1998)
  - Совместимость с технологией DCOM, IDL/Java
- CORBA 2.4 (2001)
  - Улучшение QoS, асинхронный обмен сообщениями, поддержка работы в реальном времени
- Текущая версия: CORBA 3.2 (2011)
  - Улучшена поддержка Интерет (поддержка работы через Firewall)
  - Java RMI становится CORBA IIOP

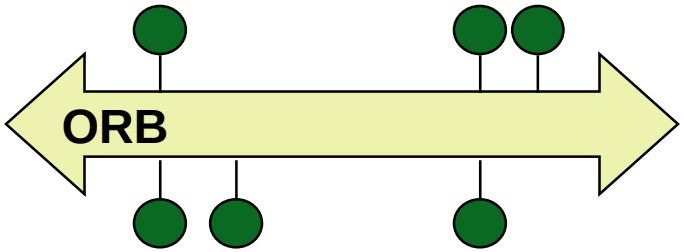
OMG, OMA, ORB и CORBA



Брокер Объектных Запросов (ORB)

- Обеспечивает прозрачное взаимодействие между удаленными объектами посредством запросов

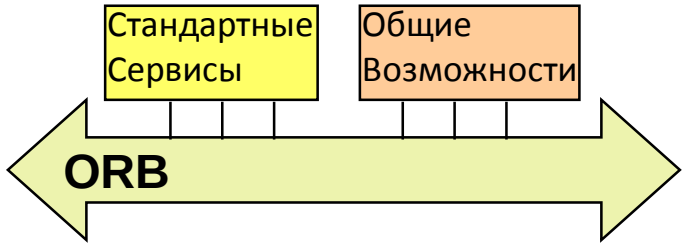
“... как телефонная станция, обеспечивая основной механизм для отправки и приема вызовов...”



Объектные Сервисы

- Набор общих сервисов и интерфейсов

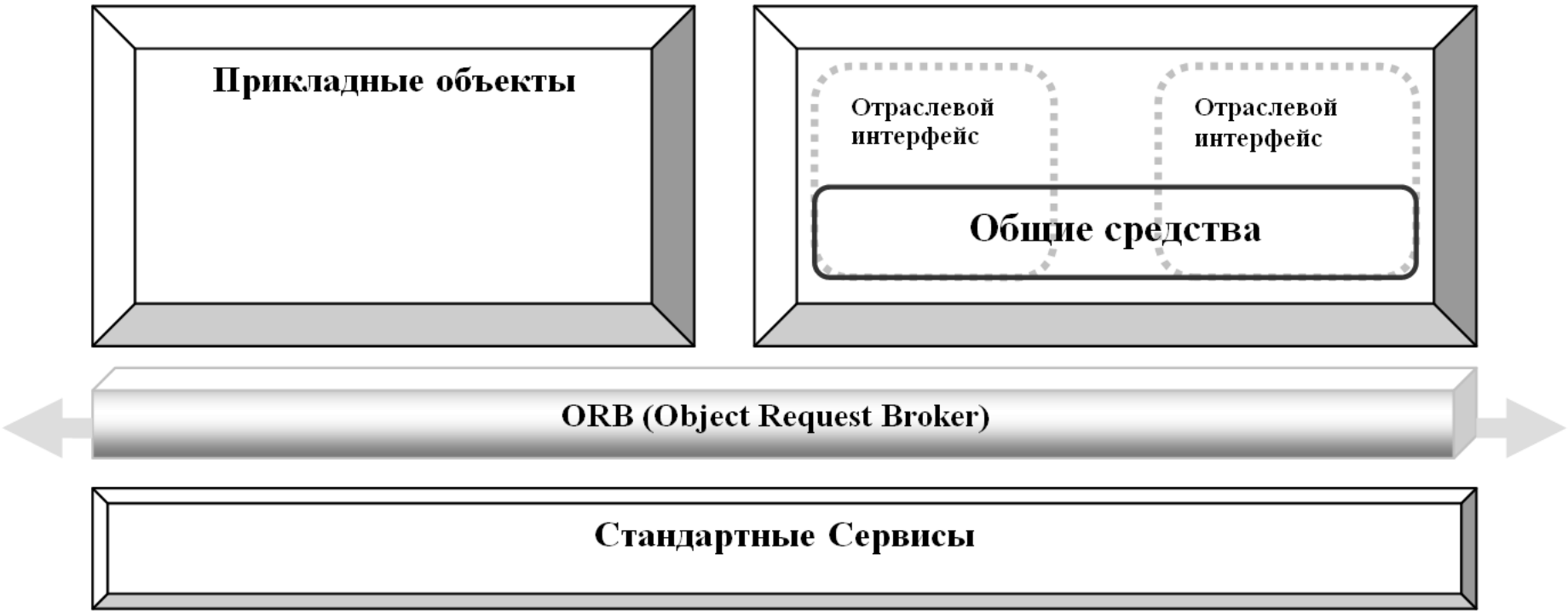
- Стандартные Сервисы
  - Поддержка функции для внедрения и использования объектов
  - например управление временем жизни и т.п.
- Общие Возможности
  - Сервисы, которыми многие приложения могут делиться, но не являются фундаментальными
  - Менеджер печати и т.п.



Отраслевые и прикладные объекты

- Отраслевые объекты
  - Вертикальные (от самых общих, до самых частных) стандарты, специфичные для определенной прикладной области (производство, телекоммуникации, финансы и т.п.)
- Прикладные объекты
  - Другие объекты, относящиеся к CORBA, не стандартизованные OMG

Схема взаимодействия объектов OMA

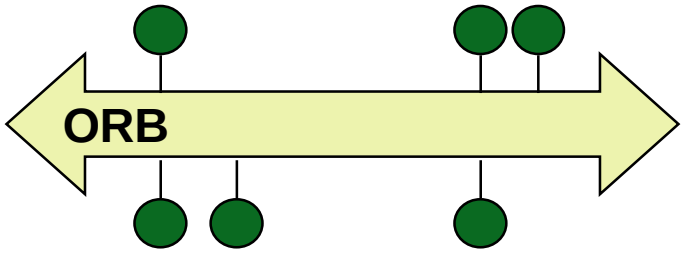


Брокер Объектных Запросов

Object Request Broker (ORB)

Что такое ORB?

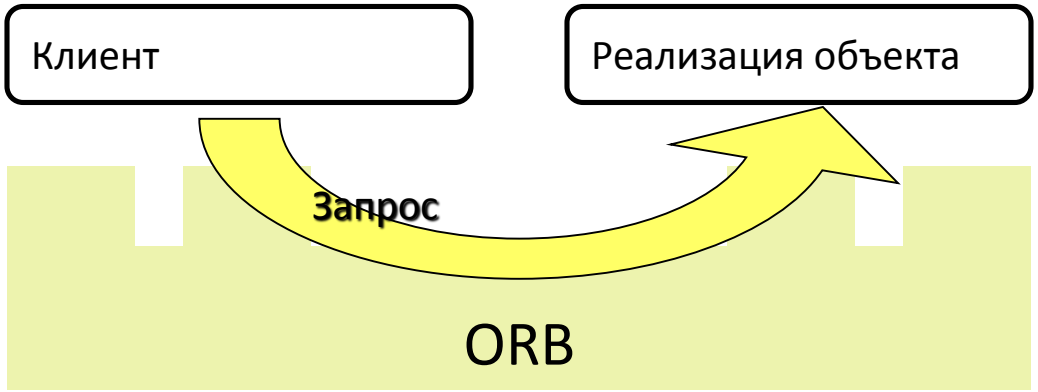
- Шина для работы распределенных объектов
- Скрывает механизмы передачи данных
  - Расположение
  - Вызов методов
  - Маршализация



- Язык описания интерфейсов (OMG IDL или Interface Definition Language) предоставляет независимую от языка семантику работы с объектами

Object Request Broker (ORB)

- Абстрагирует механизмы удаленных запросов и ответов
- Транспорт для удаленного вызова методов



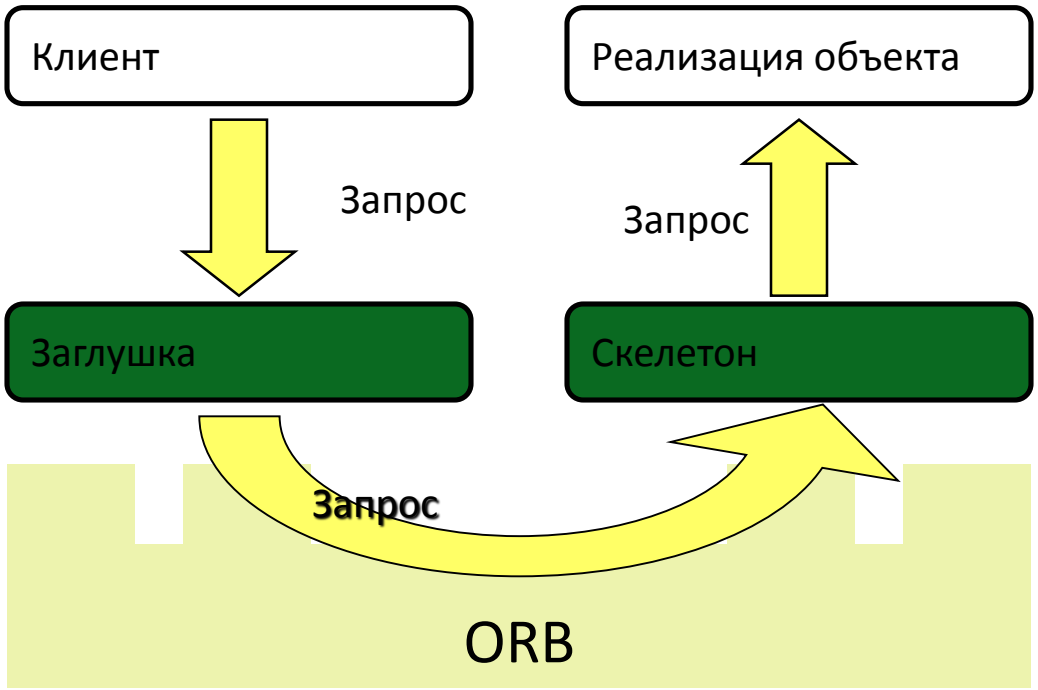
Использование ORB

- ORB реализован как синглтон
- Инициализация ORB
  - Вызов метода init()

```
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init();
```

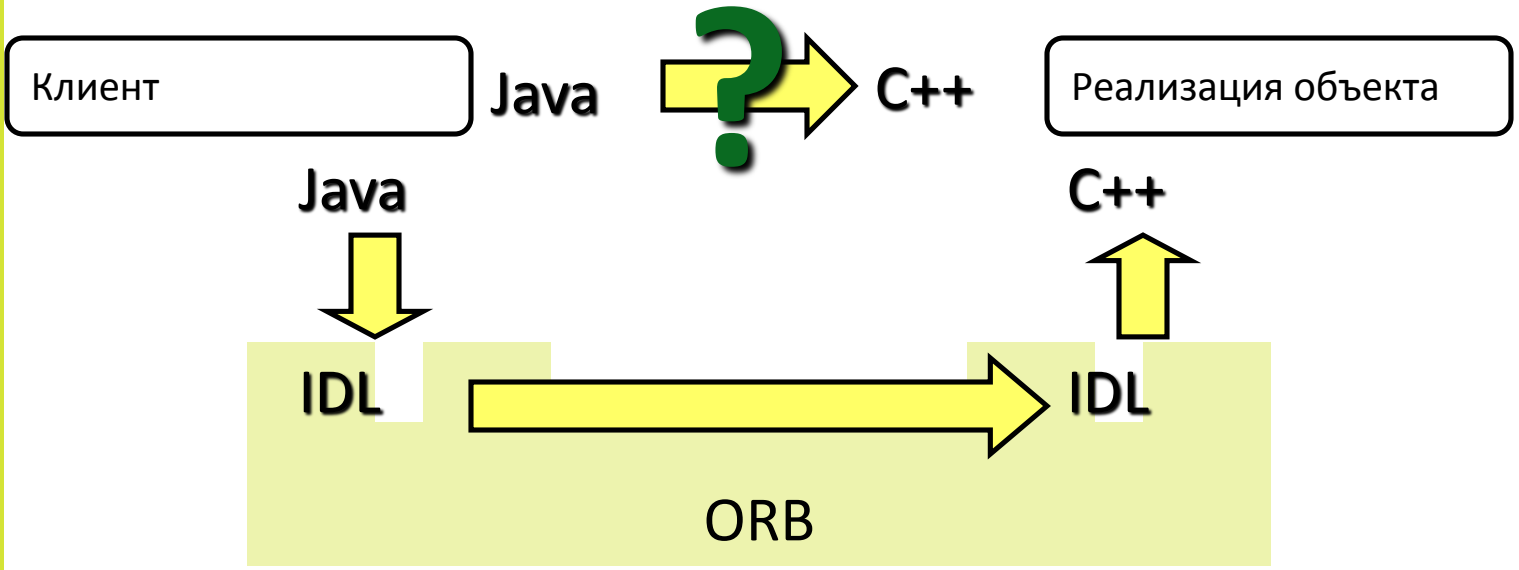
  - После инициализации производится регистрация объектов ORB посредством Object Adapter

Вызов удаленного объекта



Абстракция ORB

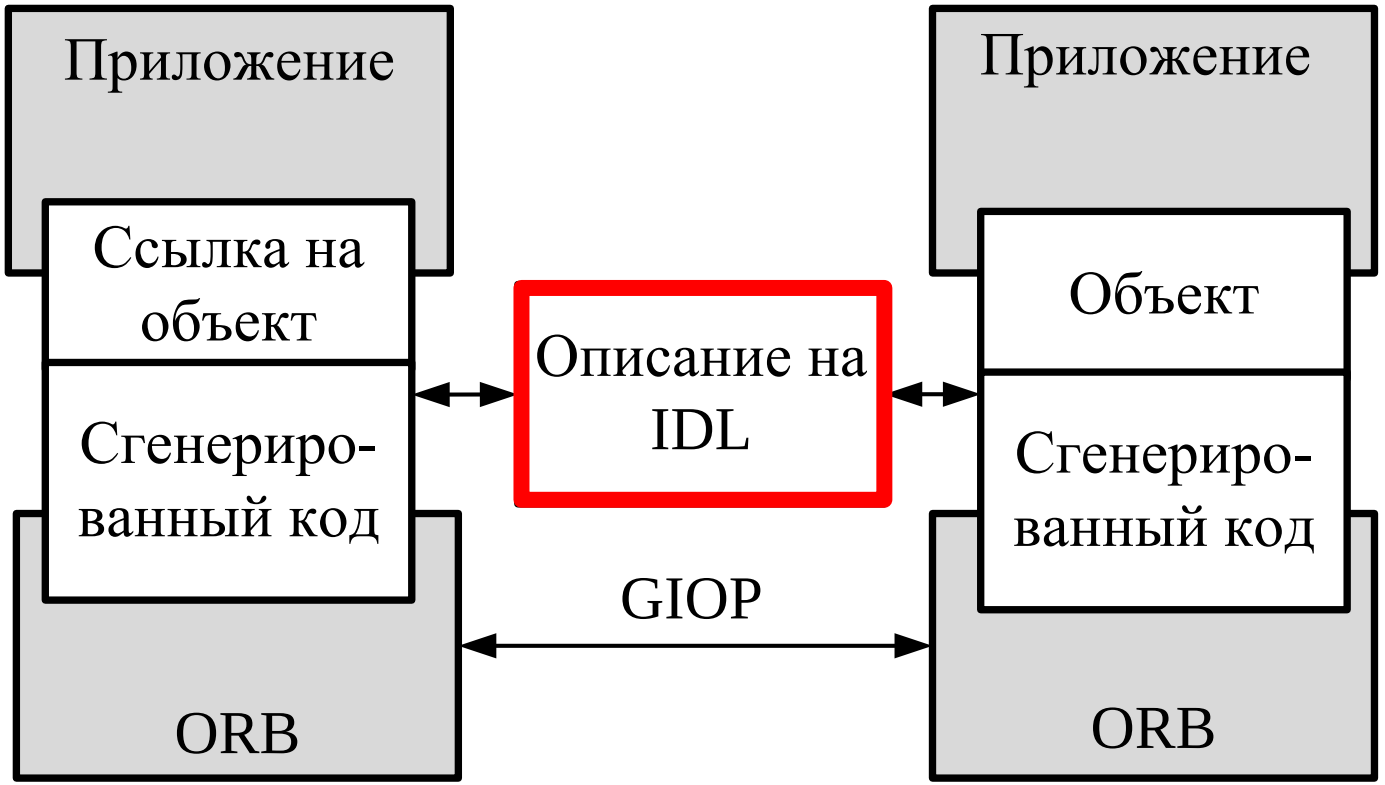
- ORB обеспечивает возможность взаимодействия систем, реализованных на базе различных платформ



Протокол GIOP

- GIOP (General Inter-ORB Protocol) – это протокол, обеспечивающий взаимодействие между ORB в распределенной вычислительной среде.
- Наиболее распространенная реализация этого протокола – IIOP (Internet Inter-ORB Protocol), который отображает сообщения GIOP на уровень TCP/IP.

Основные элементы взаимодействия CORBA



Язык определения интерфейса

Interface Definition Language (IDL)

Язык определения интерфейса(IDL)

- Независимый от языка интерфейс
  - Реализовано отображение IDL на множество высокоуровневых языков программирования
- Основная парадигма
  - Обеспечить интерфейс для объектов, независим от их реализации

Отображения IDL

- Существуют методы отображения IDL на множество высокоуровневых языков

- C
- C++
- Smalltalk
- Lisp
- Ada ‘95
- COBOL
- Python
- Java

Пример определения IDL

```
// Модуль системы ценовых предложений
module QuoteSystem
{
  // Структура данных цены
  struct Quote
  {
    string value;
  }

  // Интерфейс сервера ценовых предложений
  interface QuoteServer
  {
    // Атрибут определяющий фондовую биржу
    string exchange;
    // Исключение «Неизвестный идентификатор»
    exception UnknownSymbolException { string message; };
    // Поиск предложения по идентификатору
    Quote getQuote (in string symbol)
                raises (UnknownSymbolException);
  };
};
```

Пакет QuoteSystem

Интерфейс объекта QuoteServer

член данных

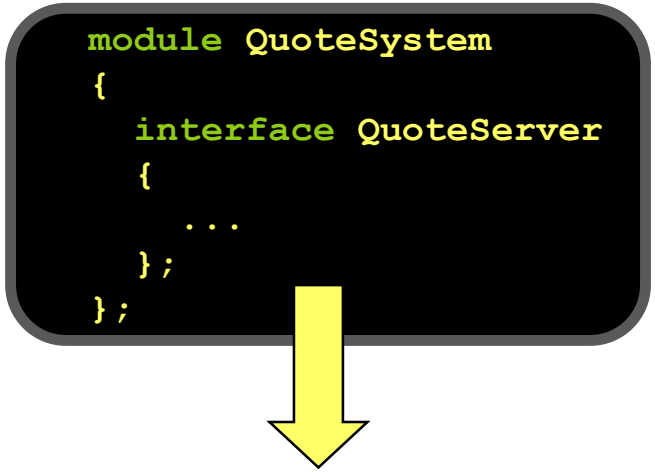
Возвращаемый тип

Аргумент (направление и тип)

заявляет, что метод формирует исключение

Модули и интерфейсы

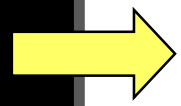
- Модуль
  - Отображается в пакет Java
  - Может содержать множество интерфейсов
- Интерфейс
  - Отображается в набор связанных классов и интерфейсов



Структуры

- Структуры
  - Отображаются в Java-классы
  - Разрабатываются для хранения информации

```
struct Quote
{
  string value;
}
```

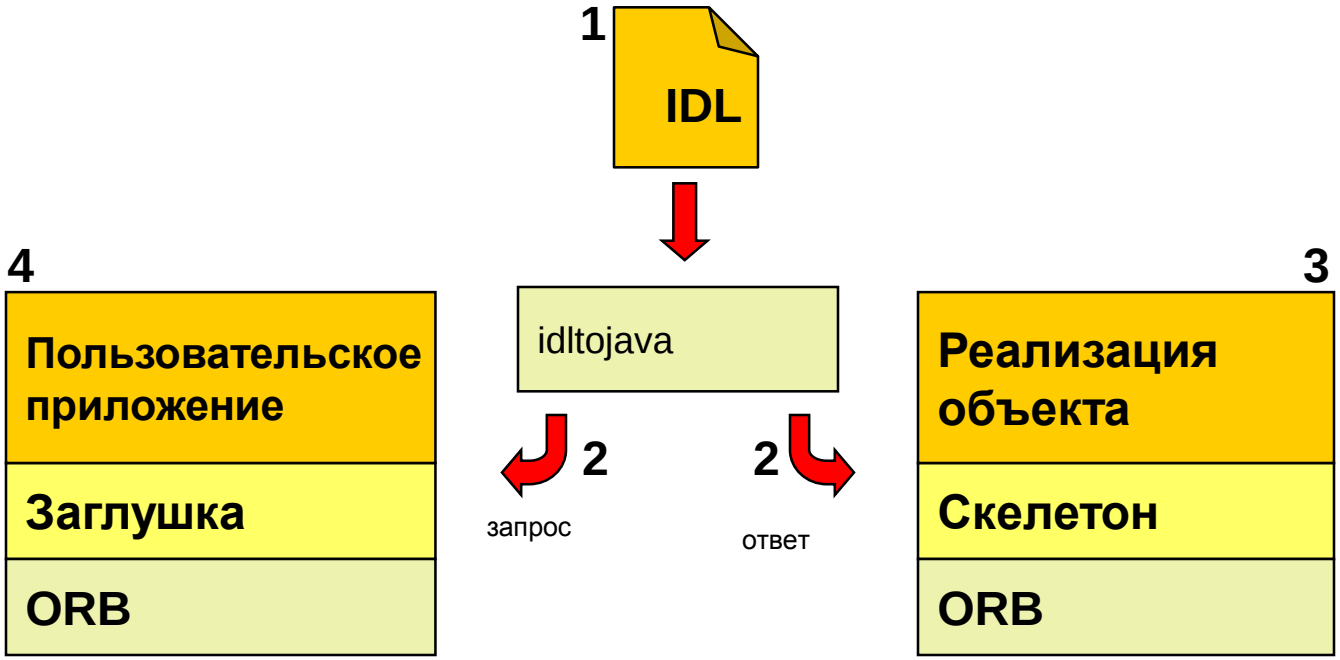


```
public final class Quote
{
  public String value;
}
```

Разработка объектов CORBA

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)

Процесс разработки



Процесс:

1. Создание файла IDL
2. Компиляция посредством idltojava (заглушка и скелетон создается автоматически)
3. Реализация объекта
4. Создание пользовательского приложения

Резюме

CORBA vs. RMI

CORBA	RMI
Платформо-независимая	Только для Java
Не зависит от протокола	Пропиретарный протокол (JRMP)
Независим от языка (IDL)	Java

Преимущества CORBA

- 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.

Проблемы CORBA

- плохая совместимость различных реализаций технологии CORBA от различных поставщиков;
- проблемы взаимодействия узлов CORBA через Интернет;
- отсутствие связи с World Wide Web;
- чрезмерная сложность разработки (более 200 строк определения интерфейсов можно было бы уложить в 30, т.к. 170 – не несут функциональной поддержки)
- Огромные объемы спецификаций (v 3.3 – 1150 страниц спецификаций)
- Коммерческие реализации CORBA обычно стоили несколько тысяч долларов за рабочее место, к чему во многих случаях добавлялась плата за каждую установленную копию приложения.
- несогласованность, отсутствие компонентной модели (появилась в 1999 г., была не проработанной, запутанной);
- Более подробно: http://citforum.ru/SE/middleware/corba_history/

Проблемы CORBA: отсутствие компонентной модели

- объектная модель CORBA возлагала на разработчика обязанность управления жизненным циклом CORBA-объектов: создание, активизация и уничтожение серверных объектов, а также "сборка мусора", возлагалась на программистов.
- Большой проблемой являлась задача сохранения состояния серверных объектов, особенно с учетом возможных сбоев в системе.
- Создание сложной системы немыслимо без обеспечения транзакционности на уровне объектов. Сервис Транзакций CORBA обеспечивал все необходимое для этого, но реализация некоторых интерфейсов – в первую очередь, интерфейса CosTransactions::Resource – возлагалась на программиста. Не было предусмотрено никаких стандартных механизмов взаимодействия с серверами баз данных с учетом наличия объектных транзакций.
- Не существовало универсального и простого в использовании подхода для управления взаимодействием объектов CORBA (порождения событий и их обработки).
- Отсутствие компонентов и контейнеров компонентов в CORBA привело к тому, что был сильно затруднен процесс создания, распространения и использования универсальных CORBA-компонентов.

Что вместо CORBA?

- Microsoft: COM, DCOM
  - Пропиретарный, завязан на Microsoft Windows
- Java: Enterprise JavaBean
- SOAP и Веб-сервисы