# Распределенные вычислительные системы

## Лекция №6: Определение местонахождения распределенных объектов

Алексей В. Бурдаков, к.т.н.
burdakov@usa.net

## План лекций

| Дата | Тема |
|---|---|
| 06.09 | 1. Вводная лекция: распределенные системы |
| 13.09 | 2. Эволюция распределенных технологий, метамодель |
| 20.09 | 3. Принципы ПО среднего слоя |
| 27.09 | 4. Стандарты OMG, Архитектура CORBA, ORB |
| 04.10 | 5. COM, Java/RMI |
| 11.10 | РК1 |
| 18.10 | 6. Определение местонахождения распределенных объектов |
| 25.10 | 7. Долговременное хранение распределенных объектов |
| 01.11 | 8. Распределенные объектные транзакции |
| 08.11 | 9. Безопасность |
| 15.11 | 10. Расширенное взаимодействие |
| 22.11 | РК2 |
| 29.11 | Вакантно |
| 06.12 | Вакантно |
| 13.12 | Зачет/Экзамен |

6-2

# Мотивация

- Важное свойство распределенной системы – прозрачность расположения
- Пример системы с зашитым IP-адресом:
  - Невозможно перенести сервер на другой узел
- Объектная ссылка – универсальный механизм адресации объектов
- Каким образом получить объектную ссылку?
  - Именование (нахождение объектов по логическим именам - «белые страницы»)
  - Трейдинг (нахождение объектов по функциональным и нефункциональным (качественным) характеристикам – «желтые страницы»)

6-3

# План лекции

- Именование объектов
- Объектный трейдинг

6-4

## Общие принципы именования

- ОО ПО среднего слоя использует объектные ссылки для адресации серверных объектов
- Необходим способ получения таких ссылок
- Имя (последовательность строк) могло бы быть привязано к объектной ссылке
- По имени могла бы быть получена ссылка

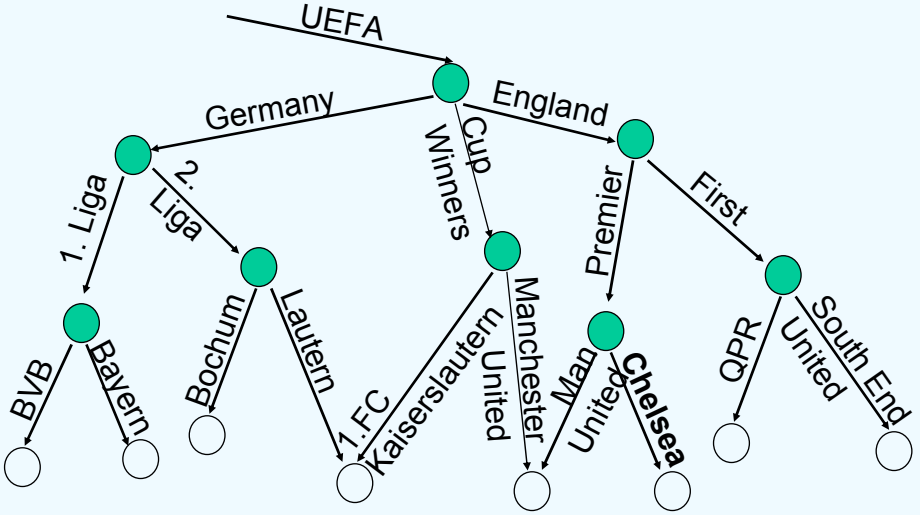
6-5

## Общие принципы именования

- В распределенной системе может быть множество распределенных объектов
- Серверные объекты могут иметь несколько имен
- Это приводит к большому объему связок имя-ссылка в системе
- Пространство имен д.б. организовано иерархически для того, чтобы избежать:
  - Конфликты имен
  - Деградацию производительности при привязке и получении ссылок
- Иерархия м.б. достигнута с помощью контекстов

6-6

# Контексты именования




6-7

# Общие принципы именования: композитные имена

- Имена составлены из возможно более чем одного компонента
- Используется для описания навигации через несколько контекстов именования
- Пример:
  (“UEFA”, “England”, “Premier”, “Chelsea”)

6-8

## Общие принципы именования: поведение сервера именования

- Привязки имен к ссылкам управляются серверами имен
- Не для каждого серверного объекта нужно имя
- Серверные объекты могут иметь несколько имен
- Серверы имен должны долговременно хранить привязку
- Эффективное получение ссылок по именам
- Серверы именования м.б. распределенными

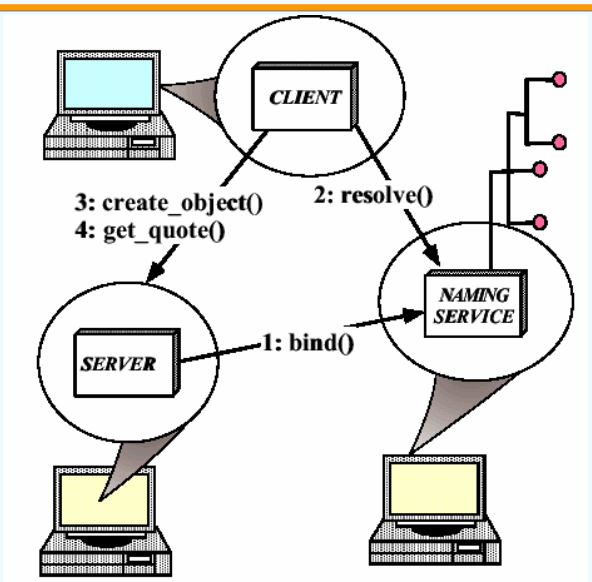
6-9

## Сервис именования CORBA

- Поддерживает привязку имен CORBA к объектным ссылкам
- Имена сгруппированы по контекстам именования
- Для одной объектной ссылки может быть определено несколько имен
- Не все объектные ссылки нуждаются в именах

6-10

# Архитектура сервиса имен




6-11

# Имена CORBA

- Имена составляются из простых имен
- Простые имена представляют собой пары «значение-тип»
- Атрибут «значение» используется для разрешения имен
- Атрибут «тип» (kind) используется для предоставления информации о роли объекта

6-12

## IDL типы для имен

```
module CosNaming {
 typedef string Istring;

 struct NameComponent {
   Istring id;
   Istring kind;
 };
 typedef sequence <NameComponent>
 Name;
 ...
};
```

6-13

## Интерфейсы IDL

- Сервис именования определен двумя IDL интерфейсами
  - NamingContext определяет привязки имен к объектам и выдает информацию по привязке
  - BindingIterator определяет набор операций для итераций по набору имен, определенных в контексте именования

6-14

## Интерфейс NamingContext

```
interface NamingContext {
    void bind(in Name n, in Object obj)
           raises (NotFound, ...);
    Object resolve(in Name n)
         raises(NotFound,CannotProceed,...);
    void unbind (in Name n)
         raises (NotFound, CannotProceed...);
    NamingContext new_context();
    NamingContext bind_new_context(in Name n)
                  raises (NotFound, ...)
    void list(in unsigned long how_many,
              out BindingList bl,
              out BindingIterator bi);
};
```

6-15

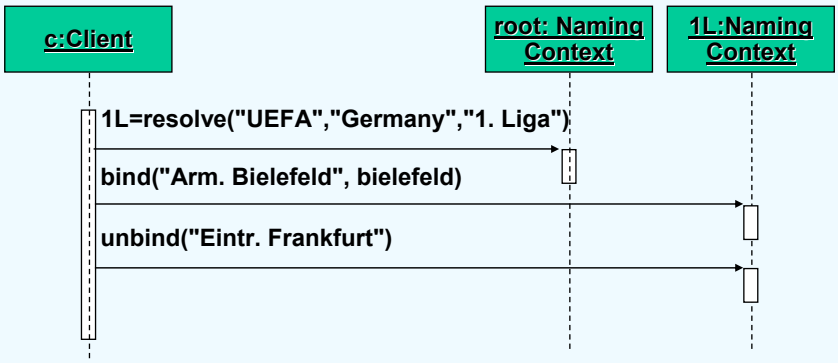
## Интерфейс BindingIterator

```
interface BindingIterator {
  boolean next_one(out Binding b);
  boolean next_n(in unsigned long how_many,
                 out BindingList bl);
  void destroy();
}
```

6-16

# Сценарии именования: привязка

- Привязать команду Bielefeid к первой лиги, а команду Frankfurt - отвязать




6-17

# Начало работы с сервисом имен

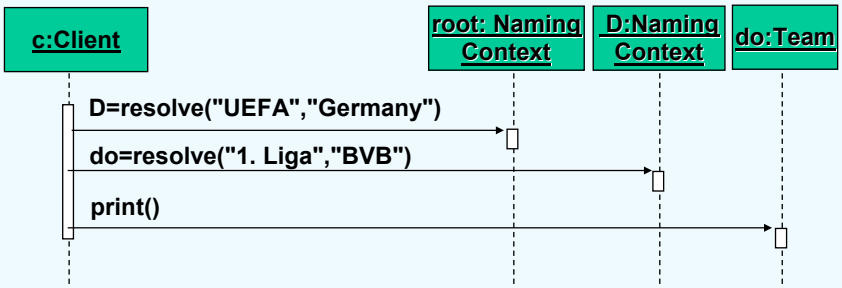
- Как получить корневой контекст именования?
- Интерфейс ORB обеспечивает для инициализации следующее:

```
module CORBA {
 interface ORB {
 typedef string ObjectId;
 typedef sequence <ObjectId> ObjectIdList;
 exception InvalidName{};
 ObjectIdList list_initial_services();
 Object resolve_initial_references
          (in ObjectId identifier)
raises(InvalidName);
 }
}
```

6-18

# Сценарий именования: разрешение

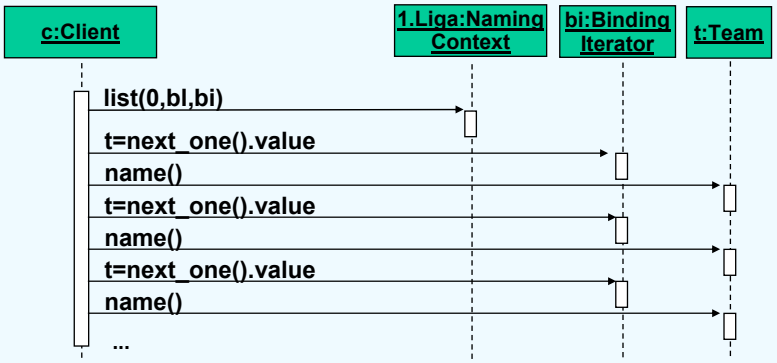
- Распечатать состав Дортмундской Боруссии




6-19

# Сценарий именования: итерации

- Распечатать все команды 1-й немецкой лиги




6-20

## Именование в COM: клички

- Moniker – кличка
- Поддержка привязки и разрешения имен
- Пространство имен м.б. иерархически структурировано

6-21

## Именование в COM: интерфейс IMoniker

```
interface IMoniker : IPersistStream {
 HRESULT BindToObject([in] IBindCtx *pbc,
          [in, unique] IMoniker *pmkToLeft,
          [in] REFIID riid,
          [out, iid_is(riid)] void **ppv);
          ...
}
```

6-22

## Создание мониторов: IParseDisplayName

- Для того, чтобы быть поименованым, серверный объект должен реализовывать интерфейс IParseDisplayName
- Создает объект-монитор разбирая внешнее (текстовое) имя, называемое отображаемым именем

```
interface IParseDisplayName {
HRESULT MkParseDisplayName([in] IBindCtx *pbc,
         [in,string] const OLECHAR *pwszName,
         [out] ULONG *ppchEaten
         [out] IMoniker **ppmk);
}
```

6-23

## Оценка

- COM имеет внутренние и внешние имена (клички и отображаемые имена), что усложняет именование
- Именование COM переплетено с другими частями спецификации COM (контейнеры)
- Именование COM не прозрачно для разработчиков серверных объектов, т.к. они должны реализовывать IParseDisplayName

6-24

# Именование в Java/RMI

- Упрощенная версия именования в CORBA
- Нет композитных имен
- Ограничения по безопасности: привязки имен не могут быть созданы с удаленных узлов
- На каждом узле д.б. реестр
- Различные реестры должны быть интегрированы в федеративные пространства имен

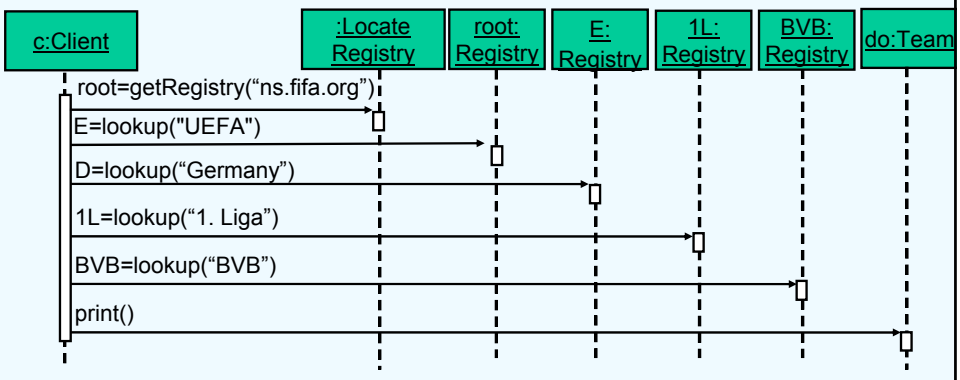
6-25

# Именование в Java/RMI: реестр

```
package java.rmi.registry;
public interface Registry extends java.rmi.Remote {
 public static final int REGISTRY_PORT = 1099;
 public java.rmi.Remote lookup(String name)
                  throws java.rmi.RemoteException,
                         java.rmi.NotBoundException,
                         java.rmi.AccessException;
 public void bind(String name, java.rmi.Remote obj)
                  throws java.rmi.RemoteException,
                         java.rmi.AlreadyBoundException,
                         java.rmi.AccessException;
 public void rebind(String name, java.rmi.Remote obj)
                  throws java.rmi.RemoteException,
                         java.rmi.AccessException;
 public void unbind(String name)
                  throws java.rmi.RemoteException,
                         java.rmi.NotBoundException,
                         java.rmi.AccessException;
 public String[] list()
                  throws java.rmi.RemoteException,
                         java.rmi.AccessException;
}
```

6-26

# Именование в Java/RMI: использование реестра




6-27

# Именование в Java/RMI: оценка

- Отсутствие иерархии в именовании увеличивает кол-во удаленных операций необходимых для разрешения имен
- Поиск корневого узла не всегда прозрачно в отношении размещения
- Ограничения безопасности уничтожает прозрачность размещения

6-28

## Именование: ограничения

- Клиент должен знать имя сервера
- Такой способ поиска нужных объектов не подходит, если клиент ищет сервис с определенными функциями и качеством, но не знает имя сервиса

6-29

## План лекции

- Именование объектов
- Объектный трейдинг

6-30

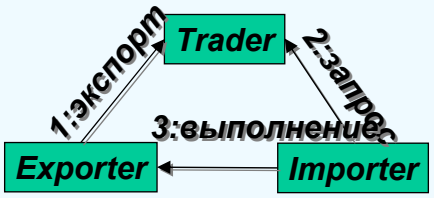
# Характеристики трейдер-сервиса

- Обнаружение объектов независимо от их физического расположения
- Именование не приемлемо когда клиент:
  - Не знает имя сервиса
  - Существует несколько сервисов с одинаковыми функциями
- Трейдер-сервис поддерживает обнаружение серверов по функциональности и качеству предоставляемого сервиса
- Именование ↔ Белые страницы
- Трейдер-сервис ↔ Желтые страницы

6-31

# Характеристики трейдер-сервиса

- Трейдер-сервис работает в качестве посредника между клиентом и сервером
- Позволяет клиенту менять перспективу с «кто?» на «что?»




- Та же самая идея:
  - биржевой брокер

6-32

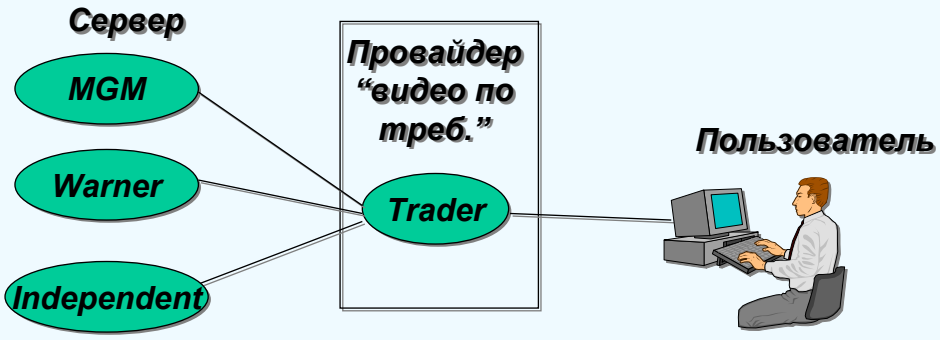
# Характеристики трейдер-сервиса

- Общий язык между клиентом и сервером:
  - Тип сервиса
  - Качество сервиса
- Сервер регистрирует сервис в трейдере
- Клиент запрашивает трейдер:
  - получить сервис определенного типа
  - сервис определенного качества
- Трейдер поддерживает:
  - сопоставление сервисов
  - «торговлю сервисами»

6-33

# Пример

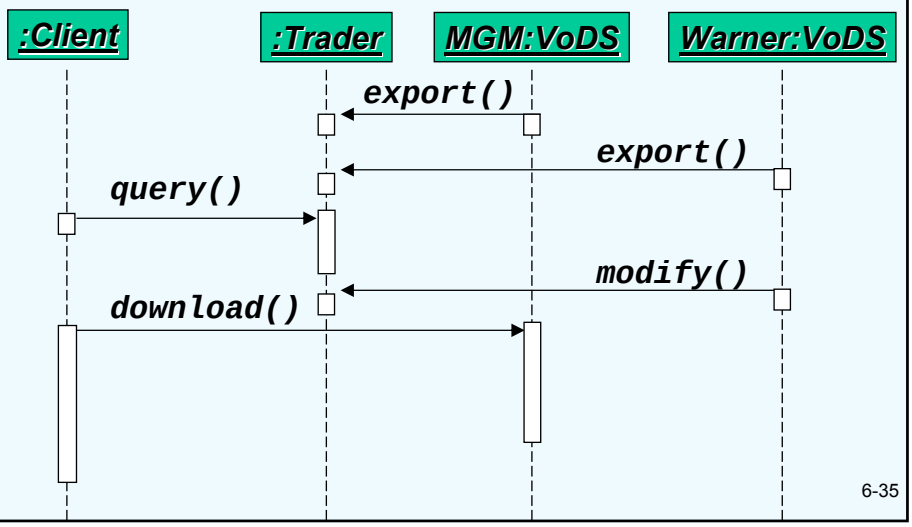
- Сервер «видео по требованию»




6-34

# Процесс «трейдинга»

- Сервер «видео по требованию»




6-35

# Определение типа сервиса

- Определение типа сервиса
  - функциональность, определенная сервисом
  - характеристика качества сервиса (QoS)
- Функциональность определяется типом объекта
- QoS определена свойствами:
  - имя свойства
  - тип свойства
  - значение свойства
  - режим работы
    - обязательный/не обязательный
    - только для чтения/модифицируемый

6-36

## Пример: тип сервиса

```
typedef enum {VGA,SVGA,XGA} Resolution;
service video_on_demand {
  interface VideoServer;
  readonly mandatory property float fee;
  readonly mandatory property Resolution
 res;
  modifiable optional property float
 bandwidth;
}
```

6-37

## Определение ограничений

- Импортер сервиса определяет желаемые качества сервиса в качестве части запроса:
- Пример:

```
fee<10 AND res >=SGA AND bandwidth>=256
```

- Трейдер выдает в качестве ответа на запрос только те сервисы, которые удовлетворяют ограничениям

6-38

## Политики трейдинга

- В зависимости от ограничений и доступных сервисов, большой набор предложений может быть возвращен в качестве результата
- Политики трейдинга могут использоваться для ограничения количества найденных предложений
  - •Указание максимального числа результатов
  - •Ограничения по замещению сервисов
  - •Ограничения по модифицируемым свойствам

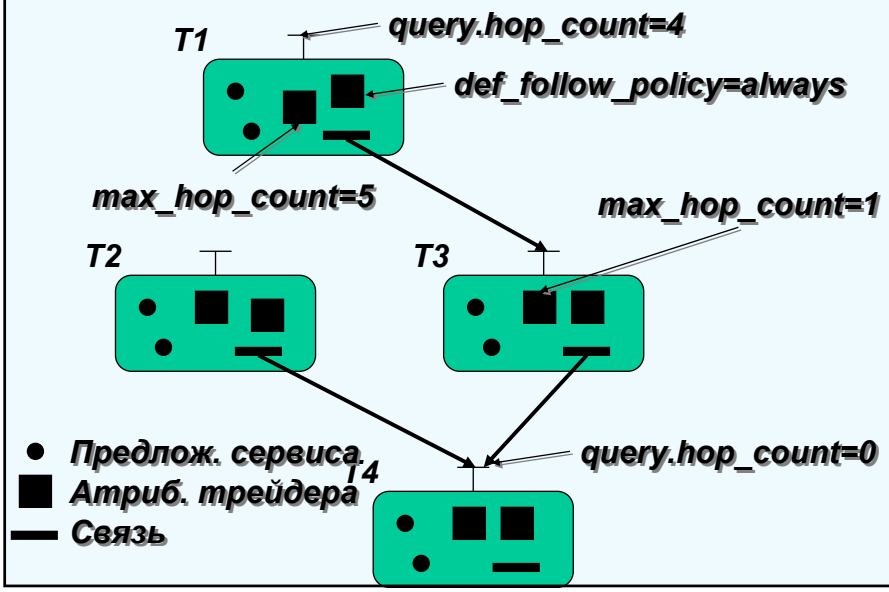
6-39

## Федерации трейдеров

- Масштабируемость приводит к необходимости создания федераций трейдеров
- Трейдер, участвующий в федерации:
  - •предлагает информацию о сервисах другим трейдерам
  - •запрашивает других трейдеров в случае если он не может удовлетворить запрос
- Проблемы
  - •непрерывный импорт
  - •дублирование предложений

6-40

# Граф процесса трейдинга




6-41

# Трейдинг и именование: ключевые моменты

- Распределенные объекты могут быть найдены с помощью именования и трейдинга
- Именование привязывает известные имена к объектным ссылкам и поддерживает разрешение имен для получения объектной ссылки
- Трейдинг поддерживает обнаружение объектов на основе их функциональности, которую они предлагают и на основе качества сервиса, которую они гарантируют

6-42

# Литература / Internet источники


- В. Эммерих ***Конструирование распределенных объектов***. - М.:Мир. - 2002.
- М.Р. Когаловский ***Энциклопедия технологий баз данных***. - М.: ФС. - 2002.
- Ю.А. Григорьев, А.Д. Плутенко. - ***Жизненный цикл проектирования распределенных баз данных***. - Благовещенск. - 1999.
- Э. Таненбаум, М. Ван Стеен ***Распределенные системы. Принципы и парадигмы.*** – СПб.: Питер, 2003. – 877 с.: ил. – (Серия «Классика computer science»)
- www.omg.org


6-43