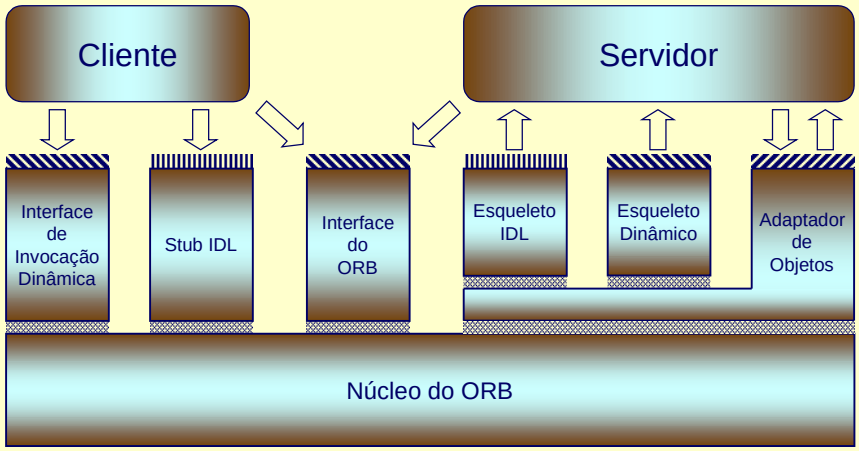
Revendo os Componentes de CORBA



Independe de ORB
Depende das definições IDL
Depende do adaptador
Interface proprietária

Adaptadores de Objetos (OAs)

- Conectam implementações de objetos ao ORB
  - Diferentes OAs podem suportar diferentes estilos de implementação de objetos
- Ativam implementações de objetos (servidores)
- Ativam os múltiplos objetos implementados por um servidor
- Encaminhmento de requisições de serviço
  - para implementações de objetos (servidores)
  - para um dos objetos implementados pelo servidor
- Geram *object references*
  - Para ter o status de “objeto CORBA”, o objeto deve ser registrado com o adaptador

O Basic Object Adapter (BOA)

- Presente na especificação CORBA até a revisão 2.1
- Problemas na especificação do BOA:
  - Aspectos importantes ficaram indefinidos
  - Muita ambiguidade
- CORBA 2.2 substituiu o BOA pelo POA, um adaptador cuidadosamente especificado
  - Mas por algum tempo o BOA continuou sendo o adaptador oferecido pela maioria dos ORBs
  - ORB com BOA → servidores não portáveis

O Portable Object Adapter (POA)

- Especificação aprovada em março de 1997
- Substitui o BOA
- Permite que servidores CORBA sejam escritos de modo portável
- Além de fechar os “buracos” da especificação do BOA, oferece muita funcionalidade adicional

Características do POA

- Identificadores de objetos automaticamente gerados ou especificados pelo usuário
- Ativação explícita ou por demanda
- Um servente → um ou mais objetos CORBA
- Aplicação tem total controle sobre o comportamento e sobre a existência dos objetos CORBA
- Serventes podem usar esqueletos estáticos ou dinâmicos

Identificador de Objeto

- Valor usado para identificar um objeto no contexto de um POA
  - Não é um identificador globalmente único!
- Pode ser determinado pela aplicação ou gerado pelo POA

```
typedef sequence<octet> ObjectID;
```

Objetos Transientes ou Persistentes

- Objeto transiente é um objeto CORBA com tempo de vida limitado pelo tempo de vida do processo servidor no qual o objeto foi criado
- Objeto persistente é um objeto CORBA com tempo de vida independente do tempo de vida de qualquer processo servidor

Ativação Explícita ou Sob Demanda

- Ativação explícita permite que o programador registre serventes chamando diretamente o POA
- Ativação sob demanda permite que uma aplicação registre “objetos ativadores” que receberão upcalls do POA
- “Serventes default” também são suportados

Objetos CORBA e Seus Serventes

- Um POA pode exigir que cada objeto CORBA tenha seu próprio servente (que pode variar ao longo do tempo)
- Alternativamente, o POA pode permitir que um objeto servente encarne múltiplos objetos CORBA
- O programador escolhe a política que for mais conveniente

Comportamento e Existência dos Objetos

- As aplicações controlam completamente o estado e o comportamento dos objetos
  - O POA não mantém estado persistente para objetos CORBA
  - A aplicação tem a responsabilidade de manter o estado de seus objetos de modo persistente
- Aplicações determinam a existência (tempo de vida) dos objetos
  - Uma aplicação pode lançar exceções **`OBJECT_NOT_EXIST`**

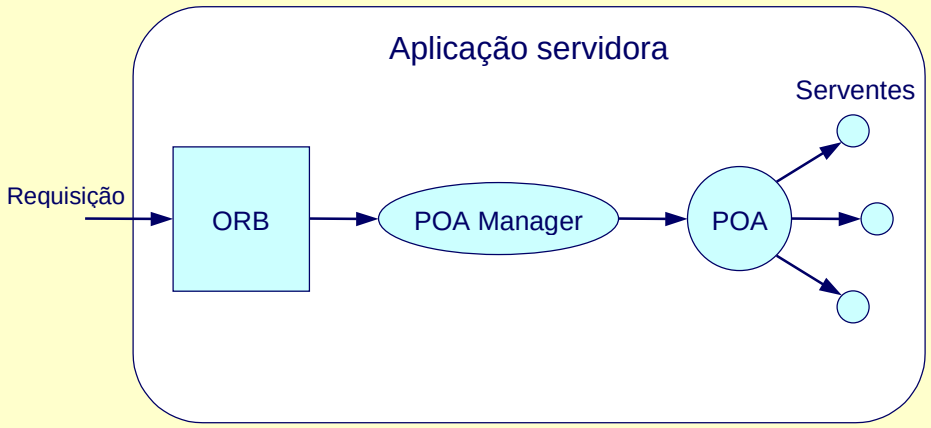
Esqueletos Estáticos ou Dinâmicos

- Serventes com esqueleto estático:
  - A ligação do servente ao esqueleto pode ser feita usando herança ou delegação (tie objects)
  - Padronização dos nomes dos esqueletos e dos tie objects
- Alternativamente, serventes podem usar esqueletos dinâmicos
  - Isso permite que o servente implemente uma interface ser ter conhecimento prévio (em tempo de compilação) de suas operações
    - Bridges usam esqueletos dinâmicos

Que ORBs Tem POA?

- Hoje a muitos ORBs já oferecem POA:
  - Visibroker 4.0 (Inprise)
  - Orbix 2000 (IONA)
  - ORBacus 4.0 (OOC)
  - TAO (open-source, Washington University)
  - JacORB (open-source, University of Berlin)
  - MICO (open-source, Univ. of Frankfurt)
  - omniORB (open-source, ATT Labs, Cambridge)
- Mas ainda há ORBs sem POA:
  - o do JDK (Sun), o ComponentBroker (IBM)

ORB, POA Manager, POA e Serventes



CORBA::Policy

```
module CORBA {
    typedef unsigned long PolicyType;

    interface Policy {
        readonly attribute PolicyType policy_type;

        Policy copy();
        void destroy();
    };
    typedef sequence<Policy> PolicyList;
    ...
};
```

Lifespan Policy

```
module PortableServer {
    ...
    enum LifespanPolicyValue {
        TRANSIENT,
        PERSISTENT
    };
    interface LifespanPolicy : CORBA::Policy {
        readonly attribute LifespanPolicyValue value;
    };
    ...
};
```

- Root POA: **TRANSIENT**
- Default: **TRANSIENT**

Id Assignment Policy

```
module PortableServer {
    ...
    enum IdAssignmentPolicyValue {
        USER_ID,
        SYSTEM_ID
    };
    interface IdAssignmentPolicy : CORBA::Policy {
        readonly attribute IdAssignmentPolicyValue value;
    };
    ...
};
```

- Root POA: **SYSTEM_ID**
- Default: **SYSTEM_ID**

Id Uniqueness Policy

```
module PortableServer {
    ...
    enum IdUniquenessPolicyValue {
        UNIQUE_ID,
        MULTIPLE_ID
    };
    interface IdUniquenessPolicy : CORBA::Policy {
        readonly attribute IdUniquenessPolicyValue value;
    };
    ...
};
```

- Root POA: **UNIQUE_ID**
- Default: **UNIQUE_ID**

Implicit Activation Policy

```
module PortableServer {
    ...
    enum ImplicitActivationPolicyValue {
        IMPLICIT_ACTIVATION,
        NO_IMPLICIT_ACTIVATION
    };
    interface ImplicitActivationPolicy : CORBA::Policy {
        readonly attribute
            ImplicitActivationPolicyValue value;
    };
    ...
};
```

- Root POA: **IMPLICIT_ACTIVATION**
- Default: **NO_IMPLICIT_ACTIVATION**

Request Processing Policy

```
module PortableServer {
    ...
    enum RequestProcessingPolicyValue {
        USE_ACTIVE_OBJECT_MAP_ONLY,
        USE_DEFAULT_SERVANT,
        USE_SERVANT_MANAGER
    };
    interface RequestProcessingPolicy : CORBA::Policy {
        readonly attribute
            RequestProcessingPolicyValue value;
    };
    ...
};
```

- Root POA: **USE_ACTIVE_OBJECT_MAP_ONLY**
- Default: **USE_ACTIVE_OBJECT_MAP_ONLY**

Servant Retention Policy

```
module PortableServer {
    ...
    enum ServantRetentionPolicyValue {
        RETAIN,
        NON_RETAIN
    };
    interface ServantRetentionPolicy : CORBA::Policy {
        readonly attribute
            ServantRetentionPolicyValue value;
    };
    ...
};
```

- Root POA: **RETAIN**
- Default: **RETAIN**

Thread Policy

```
module PortableServer {
    ...
    enum ThreadPolicyValue {
        ORB_CTRL_MODEL,
        SINGLE_THREAD_MODEL
    };
    interface ThreadPolicy : CORBA::Policy {
        readonly attribute ThreadPolicyValue value;
    };
    ...
};
```

- Root POA: **ORB_CTRL_MODEL**
- Default: **ORB_CTRL_MODEL**

Criação de Políticas

```
module PortableServer {
    interface POA {
        LifeSpanPolicy createLifeSpanPolicy(
                in LifeSpanPolicyValue value
            );
        IdAssignmentPolicy createIdAssignmentPolicy(
                in IdAssignmentPolicyValue value
            );
        IdUniquenessPolicy createIdUniquenessPolicy(
                in IdUniquenessPolicyValue value
            );
        ...
    };
    ...
};
```

Criação de POAs

```
module PortableServer {
    interface POA_Manager;
    exception AdapterAlreadyExists {};
    exception InvalidPolicy { unsigned short index; };

    interface POA {
        POA create_POA(
            in string adapter_name,
            in POA_MANAGER manager,
            in CORBA::PolicyList policies
        ) raises(AdapterAlreadyExists, InvalidPolicy);
        ...
    };
    ...
};
```


Criação de Objetos

```
module PortableServer {
    typedef sequence<octet> ObjectId;
    exception WrongPolicy {};

    interface POA {
        Object create_reference(
            in CORBA::RepositoryId intf
        ) raises(WrongPolicy);
        Object create_reference_with_id(
            in ObjectId oid,
            in CORBA::RepositoryId intf
        ) raises(WrongPolicy);
        ...
    };
};
```

Ativação de Objetos

```
module PortableServer {
    ...                 // definições de exceções...
    interface POA {
        ObjectId activate_object(in Servant p_servant)
            raises(ServantAlreadyActive, WrongPolicy);
        void activate_object_with_id(
            in ObjectId id, in Servant p_servant
        ) raises(
            ServantAlreadyActive,
            ObjectAlreadyActive,
            WrongPolicy
        );
        void deactivate_object(in ObjectId oid)
            raises(ObjectNotActive, WrongPolicy);
        ...
    };
```

Se quiser uma referência para o objeto CORBA ativado, use a operação **`id_to_reference`**

Servant Managers

- Dois tipos de **`ServantManager`**:
  - o **`ServantActivator`** é para um POA com tabela de objetos ativos (**`RETAIN`**)
  - o **`ServantLocator`** é para um POA sem tabela de objetos ativos (**`NON_RETAIN`**)

```
module PortableServer {
    exception ForwardRequest {
        Object forward_reference;
    };

    interface ServantManager {};
    ...
};
```

Servant Activator

```
module PortableServer {
    interface ServantActivator : ServantManager {
        Servant incarnate(
                    in ObjectId oid, in POA adapter
        ) raises(ForwardRequest);

        void etherealize(
                    in ObjectId oid, in POA adapter,
                    in Servant serv,
                    in boolean cleanup_in_progress,
                    in boolean remaining_activations
        );
    };
    ...
};
```


Servant Locator

```
module PortableServer {
    interface ServantLocator : ServantManager {
        native Cookie;
        Servant preinvoke (
                    in ObjectId oid, in POA adapter,
                    in CORBA::Identifier operation,
                    out Cookie the_cookie
        ) raises(ForwardRequest);
        void postinvoke (
                    in ObjectId oid, in POA adapter,
                    in CORBA::Identifier operation,
                    in Cookie the_cookie,in Servant serv
        );
    };
    ...
};
```

Registro do Servant Manager

```
module PortableServer {
    interface POA {
        ...
        ServantManager get_servant_manager()
            raises(WrongPolicy);

        void set_servant_manager(
                        in ServantManager serv_mgr
        ) raises(WrongPolicy);
    };
    ...
};
```


Default Servants

```
module PortableServer {
 interface POA {
        ...
        Servant get_servant()
            raises(NoServant, WrongPolicy);

        void set_servant(in ServantManager serv)
            raises(WrongPolicy);
    };
    ...
};
```

Obtenção do OID num Default Servant

```
module PortableServer {
    ...
    interface Current : CORBA::Current {
        exception NoContext {};

        POA get_POA
            raises(NoContext);
        ObjectId get_object_id()
            raises(NoContext);
    };
    ...
};
```


Mapeamento de Identidade

```
module PortableServer {
    interface POA {
        ObjectId servant_to_id(in Servant serv)
            raises(ServantNotActive, WrongPolicy);
        Object servant_to_reference(in Servant serv)
            raises(ServantNotActive, WrongPolicy);
        Servant reference_to_servant(in Object ref)
            raises(ObjectNotActive, WrongAdapter, WrongPolicy);
        ObjectId reference_to_id(in Object ref)
            raises(WrongAdapter, WrongPolicy);
        Servant id_to_servant(in ObjectId oid)
            raises(ObjectNotActive, WrongPolicy);
        Object id_to_reference(in ObjectId oid)
            raises(ObjectNotActive, WrongPolicy);
        ...
    };
```

Controle do Fluxo de Requisições

```
module PortableServer {
    interface POAManager {
        exception AdapterInactive {};
        enum state { HOLDING, ACTIVE, DISCARDING, INACTIVE };
        State get_state();
        void activate() raises(AdapterInactive);
        void hold_requests(in boolean wait_for_completion)
            raises(AdapterInactive);
        void discard_requests(in boolean wait_for_completion)
            raises(AdapterInactive);
        void deactivate(in boolean etherealize_objects,
                        in boolean wait_for_completion
             ) raises(AdapterInactive);
    };
    ...
};
```


Tratamento de Eventos

```
module CORBA {

    interface ORB {
        void run();
        void shutdown(in boolean wait_for_completion);
        boolean work_pending();
        void perform_work();
        ...
    };
    ...
};
```