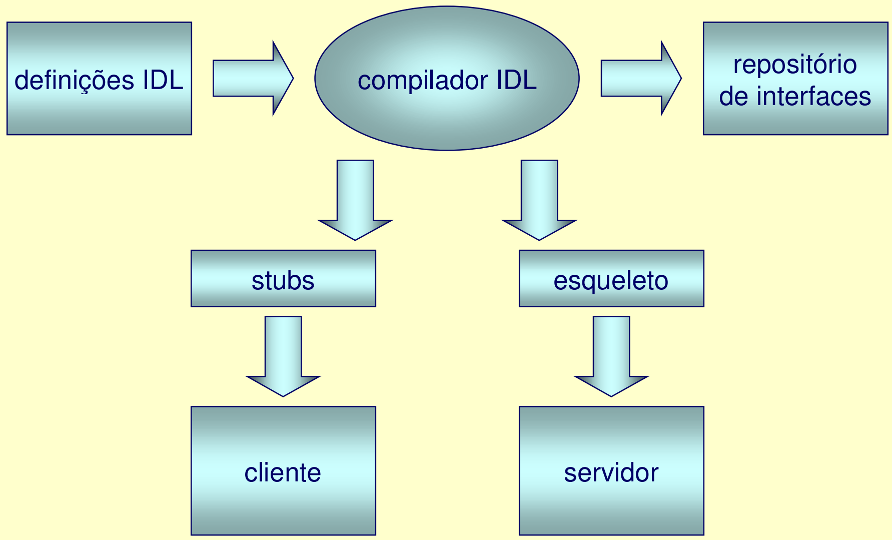
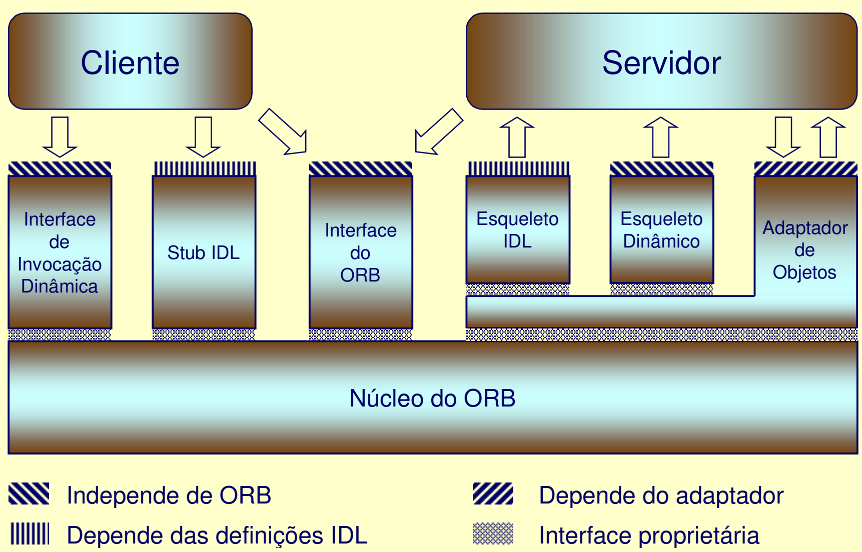
Geração de Stubs e Esqueletos



Revendo os Componentes de CORBA



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 o BOA ainda é o adaptador de objetos oferecido pela maioria dos ORBs
  - Por isso os servidores atuais não são 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

Voltando ao Nosso Exemplo

```
// IDL
module Stock {
    exception UnknownStock {
        string name;
    };
    interface Quoter {
        string name();
        long value(in string stock)
            raises(UnknownStock);
    };
};
```


Esqueleto C++ para a Interface Quoter

```
// Código C++ gerado pelo tradutor IDL
namespace POA_Stock {
    class Quoter: public virtual
                  PortableServer::ServantBase {
    public:
        virtual char* name()
            throw(CORBA::SystemException) = 0;
        virtual CORBA::Long value(const char*)
            throw(Stock::UnknownStock,
                  CORBA::SystemException) = 0;
    };
};
```


Classe Servente para a Interface Quoter

```
class QuoterImpl : public POA_Stock::Quoter {
public:
    QuoterImpl(const char* my_name);
    char* name()
        throw(CORBA::SystemException);
    CORBA::Long value(const char* name)
        throw(Stock::UnknownStock,
              CORBA::SystemException);
private:
    string m_name; // classe string ANSI/ISO
    map<string,CORBA::Long> m_map; // map da STL
};
```


QuoterImpl::name()

```
char*
QuoterImpl::name()
    throw(CORBA::SystemException)
{
    char* nm = CORBA::string_dup(m_name.c_str());
    if (nm == 0)
        throw CORBA::NO_MEMORY();
    return nm;
}
```

QuoterImpl::value()

```
CORBA::Long
QuoterImpl::value(const char* stock)
    throw(Stock::UnknownStock,
          CORBA::SystemException)
{
    map<string,CORBA::Long>::iterator iter =
        m_map.find(stock);
    if (iter == m_map.end())
        throw Stock::UnknownStock(stock);
    return (*iter).second;
}
```


QuoterImpl::QuoterImpl()

```
QuoterImpl::QuoterImpl(const char* my_name)
    : m_name(my_name)
{
    // inicializa m_map com informações lidas
    // de um arquivo ou banco de dados
    // ...

    //      (num quoter real, m_map deveria
    //       ser atualizado periodicamente)
}
```

Mainline do Servidor de Cotações

```
int main(int argc, char* argv[]) {
   using namespace CORBA; using namespace Stock;
   ORB_ptr orb = ORB_init(argc, argv);
   Object_ptr obj=
      orb->resolve_initial_references("RootPOA");
   POA_ptr poa = POA::_narrow(obj);
   QuoterImpl quoter_impl("DowJones");
   Quoter_ptr quoter = quoter_impl->_this();
   poa->the_POAManager()->activate();
   orb->run();
   release(quoter); release(poa);
   release(obj); release(orb);
}
```


Obtenção de Object References

- Clientes precisam obter object references
- Uma object reference é obtida como resultado de alguma operação chamada pelo cliente
  - Clientes não podem "construir" ou "inventar" object references
- Uma object reference pode ser convertida em string e depois reconvertida em object reference
  - Você pode guardar essa string num arquivo
  - Pode divulgá-la como bem entender

Exemplo de Cliente C++

```
int main(int argc, char** argv)
{
    using namespace CORBA;
    using namespace Stock;
    string objstr; // classe string ANSI/ISO
    ifstream is("quoter_string_objref");
    is >> objstr;
    ORB_ptr orbref = ORB_init(argc, argv);
    Object_ptr objref =
        orbref->string_to_object(objstr.c_str());
```


Exemplo de Cliente C++ (cont.)

```
    Quoter_ptr quoter = Quoter::_narrow(objref);
    char* name = quoter->name();
    Long value = quoter->value(argv[1]);
    cout << "Cotação de " << argv[1]
         << ", fornecida por " << name << ": "
         << value << endl;
    string_free(name);
    release(quoter);
    release(objref);
    release(orbref);
    return 0;
}
```