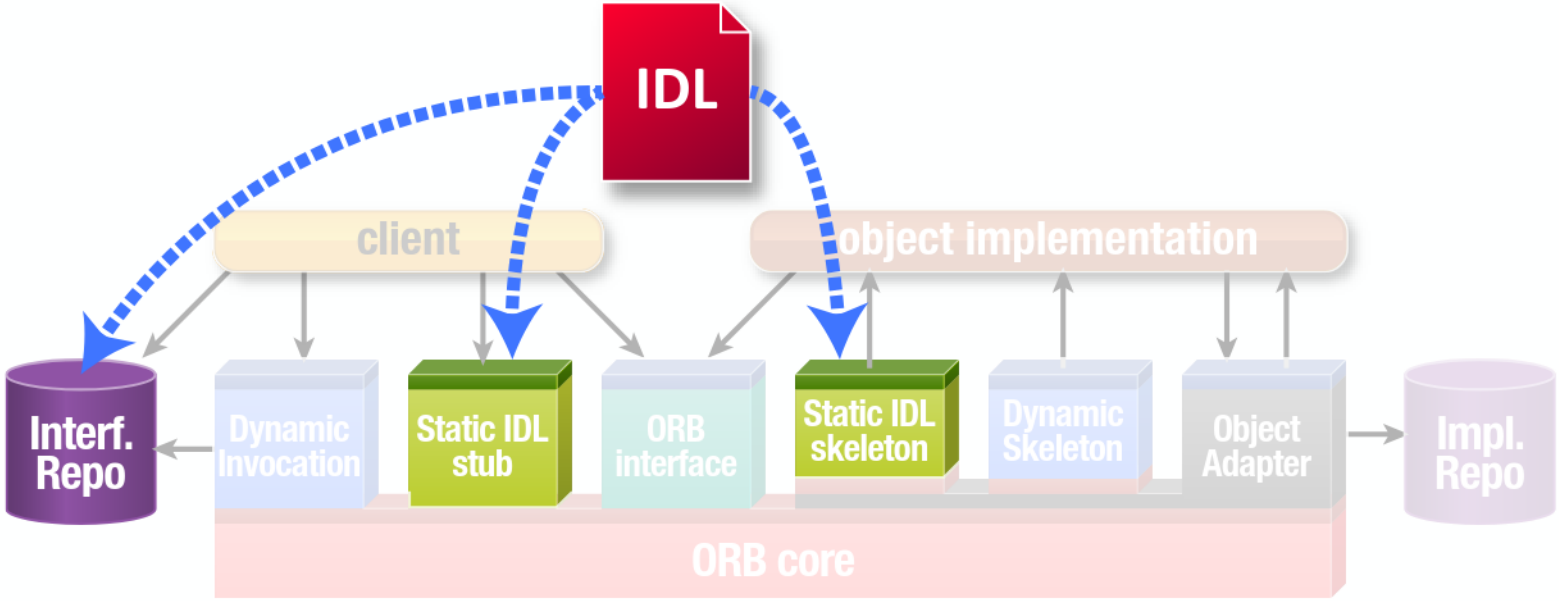
IDL: Interface Definition Language



Principais Componentes da IDL

- Módulos (`module`)
- Interfaces (`interface`)
- Tipos de dados
- Constantes
- Atributos (`attribute`)
- Operações
- Exceções (`exception`)

Tipos de Dados IDL

- Tipos básicos
  - `short` `long` `float` `boolean` ...
- Tipos derivados
  - Usando a palavra-chave `typedef`
- Tipos estruturados
  - `enum` `struct` `union` `array`
- Tipos variáveis:
  - Vetores dinâmicos, `string`, `sequence`
- O tipo `any`

Tipos e Constantes Básicas

- Inteiros: [unsigned] short long
- Reais: float double
- 8 bits: char octet boolean
- Genéricos: any

- Exemplos (constantes):
  - `const double Pi = 3.1415926;`
  - `const string Msg = "Mensagem";`
  - `const unsigned long Mask =(1<<5)|(1<<7);`

Tipos Estruturados

```
enum CartaoCredito {Master, Visa, nenhum};
struct RegistroPessoa {
    string nome;
    short  idade;
}
union Cliente switch (CartaoCredito) {
    case Master:
        string noCartao ;
    ...
}
```

Vetores, sequências e strings

```
// vetores
typedef long longVect[30];
typedef long longArray[2][10];

// sequências
typedef sequence <short> shortSeq;
typedef sequence <short,20> shortSeq20;

// strings de tamanho limitadoa
typedef string<1024> boundedString;
```

Métodos

```
<tipo_retorno> <nome> (<parâmetros>)
   [raises <exceções>]
   [context];
```

- Parâmetros de métodos podem ser:
  - `in`: enviados ao servidor
  - `out`: recebidos do servidor
  - `inout`: ambas as direções

Atributos

```
attribute string nome;
readonly attribute short idade;
```

- Atributos:
  - São declarados como variáveis.
  - Métodos get e set são gerados (readonly: somente get()).
  - TAO (C++):
    - `Float saldo();` → *get*
    - `void saldo(Float valor);` → *set*

Exemplo

```
module Utility {
    typedef long id_type;
    interface Unid {
        id_type GetID();
        void PutID(in id_type id);
    };
};
```

Semânticas de Invocação

- **Síncrona**:
  - Chama método e espera por resultado
  - Comportamento padrão
- **Assíncrona**:
  - Chama método e continua execução.
  - Chamadas de métodos não-confiáveis (sem garantia de sucesso).
  - Só parâmetros `in` e retorno `void` são aceitos .
  - Usa declaração “`oneway`”.