2.3.4 LINGUAGEM DE DEFINIÇÃO DE INTERFACES – IDL

A linguagem IDL (*Interface Definition Language*) é usada para descrever as interfaces dos objetos CORBA, sendo puramente declarativa. Uma definição IDL especifica cada atributo e parâmetro das operações definida em uma interface. Uma interface IDL oferece as informações necessárias para desenvolver clientes que usam operações disponíveis remotamente. Tanto o cliente quanto à implementação de objetos não são escritos em IDL, mas em linguagens de programação para os quais foram definidos os mapeamentos dos conceitos em IDL. O mapeamento de um conceito em IDL para uma construção de uma linguagem destino dependerá das facilidades existentes.

A gramática de IDL é um subconjunto de C++ com construções adicionais para os mecanismos de requisições de operações. Ela permite a definição de tipos, de constantes, de exceções e de módulos, além de prover suporte aos recursos de pré-processamento para inclusão de arquivos e substituição de macros.

Toda declaração de interface contém um cabeçalho e um corpo. O cabeçalho é composto por um identificador que o nomeia e por uma especificação opcional de herança. O identificador define um nome do tipo interface, sendo uma referência para o objeto que suporte tal interface. O corpo de uma interface pode conter declarações: de constantes, de tipos, desde que não sejam novas interfaces, de exceções, de atributos, e de operações, incluindo o nome da operação, seu modo de execução, os tipos de seus parâmetros e do dado de retorno. A assinatura de uma interface é caracterizada por suas operações e atributos. Entretanto, uma definição de atributo é logicamente equivalente à declaração de um par de operações de acesso: um para a leitura e outra para escrita.

As especificações em IDL são fortemente tipadas, pois definem o tipo de retorno das operações, o tipo dos parâmetros e dos atributos. A figura 1 exibe os elementos que constituem a estrutura da interface IDL.

- **Módulos**

  Identificado pela palavra chave *module* e serve para agrupar um conjunto de interfaces. Uma declaração de módulo pode incluir: declarações de tipos; declarações de constantes, declarações de exceções, declarações de interfaces e outros módulos.

- **Atributos**

  Descrevem propriedades de um objeto e por *default* podem ser consultados e modificados. Atributos que não podem ser modificados devem ser declarados como *readonly*.

- **Interfaces**

  Uma interface define uma classe de objetos CORBA. Ela especifica as instâncias de serviços que uma classe possui, seus atributos e as exceções que podem gerar para indicar quando uma operação não teve sucesso. IDL permite herança múltipla, o que significa que uma interface pode ser derivada de uma ou mais interfaces.

- **Operações**

  O conjunto de declarações de operações em uma interface define os serviços que um objeto é capaz de executar. Uma operação define os parâmetros de entrada e saída necessários para um certo serviço. O tipo de uma operação é o tipo do resultado de sua execução.

- **Tipos de Dados**

  Os tipos de dados são usados para descrever os valores aceitáveis para parâmetros, atributos, exceções e valores de retorno. CORBA classifica os tipos de dados em duas formas: básico e construídos. Os tipos de dados básicos do modelo CORBA são mostrados na tabela 3, com a sua sintaxe em IDL e descrição. Da mesma forma, são apresentados os tipos construídos ou estruturados na tabela 4.

```
module <identificador>
{
<declaracoes_de_tipo>;
<declaracoes_de_constantes>;
<declaracoes_de_excecoes>;

     inteface <identificador> [:<lista_interfaces_base>]
     {
          <declaracoes_de_tipo>;
          <declaracoes_de_constantes>;
          <declaracoes_de_atributos>;
          <declaracoes_de_excecoes>;

          [<tipo_operacao>]<identificador>(<parametros>)
          [raises excecao] [contexto];
               .
               .
          [<tipo_operacao>]<identificador>(<parametros>)
          [raises excecao] [contexto];
               .
               .
     }
}
```

Figura 1: Estrutura da Interface IDL.

Tipo	Sintaxe	Descrição
Inteiro	Short	Número inteiro de 32 bits com sinal
Inteiro	Long	Número inteiro de 64 bits com sinal
Inteiro	unsigned short	Número inteiro de 32 bits sem sinal
Inteiro	unsigned long	Número inteiro de 64 bits sem sinal
Ponto flutuante	Float	Número de ponto flutuante de 32 bits (IEEE)
Ponto flutuante	Doublé	Número de ponto flutuante de 64 bits (IEEE)
Caracteres	Char	
Booleano	Boolean	Tipo booleano com valores “TRUE”e “FALSE”
Octeto	Octet	Tipo de dado opaco de 8 bits que garante que nenhuma conversão será feita durante a sua transmissão entre os sistemas
Genérico	Any	Tipo que pode representar qualquer tipo básico ou estruturado.

Tabela 3: Tipos básicos IDL.

Tipo	Sintaxe	Descrição
Enumerado	enum <identifier>	Lista ordenada de identificadores
Estrutura	struct <identifier>	Estrutura ordenada de pares
União	union <identifier>	Um discriminante seguido de uma instância de um tipo apropriado para o discriminante.
Vetor	Array	Vetor de tamanho fixo de valores de um único tipo
seqüência	sequence <<type>, <const>>	Vetor de tamanho variável de valores de um único tipo, sendo que o tamanho da seqüência é determinado em tempo de execução.
Vetor de caracteres	String	Vetor de caracteres de tamanho variável.

Tabela 4: Tipos Estruturados IDL.