

# Interfaces para Serviços de Dados Estruturados

Rodrigo Carneiro Henrique  
[rodrigoh@tecgraf.puc-rio.br](mailto:rodrigoh@tecgraf.puc-rio.br)

# Motivação

- ⦿ Necessidade de integrar aplicações científicas
  - ⦿ Compartilhamento dos dados
- ⦿ Representação dos dados é complexa
  - ⦿ Algoritmos que os processam são complexos
  - ⦿ Estruturas de dados definidas de acordo com os algoritmos
- ⦿ Grandes volumes de dados

# Arquitetura

- ⦿ Arquitetura orientada a componentes de software
  - ⦿ Modelo de componentes distribuídos SCS
  - ⦿ Os servidores de dados são componentes
- ⦿ Navegação hierárquica
  - ⦿ Comum em outras soluções para dados científicos

# Arquitetura (cont.)

- ⦿ Chave unívoca do dado
- ⦿ Navegação sobre descrição dos dados
- ⦿ Múltiplas visões
- ⦿ Obtenção parcial

# Servidor de Dados

- ⦿ Componente SCS
- ⦿ Gerencia as facetas oferecidas
  - ⦿ IComponent
  - ⦿ IMetaInterface
  - ⦿ IReceptacles
  - ⦿ IHierarchicalDataService

# Serviço de Dados

- Gerencia os dados do servidor
- Navegação hierárquica sobre as descrições dos dados
- Responsável por oferecer as visões dos dados
- Criação, atualização e exclusão de um dado
- Transferência de dados entre serviços

# Chave Unívoca

- ⦿ Definida pelo tipo  
`tecgraf::openbus::data_service::DataKey`
- ⦿ Representada como uma sequência de octetos
- ⦿ Pode ser persistida

# Chave Unívoca (cont.)

- Contém informações sobre o servidor de origem do dado
- APIs disponíveis para cada linguagem devem ser utilizadas para extrair estas informações

```
typedef sequence<octet> OctetSeq;  
typedef OctetSeq DataKey;  
typedef sequence<DataKey> DataKeySeq;
```

# Descrição

- ⦿ Definida pelo tipo DataDescription
- ⦿ Contém a chave unívoca do dado
- ⦿ Contém os tipos das visões oferecidas pelo dado
- ⦿ Metadados representados através de pares nome-valor
- ⦿ Pode ser estendida através de herança

# Descrição (cont.)

```
valuetype DataDescription {  
    public DataKey fKey;  
    public string fName;  
    public StringSeq fViews;  
    public MetadataSeq fMetadata;  
};  
typedef sequence<DataDescription> DataDescriptionSeq;  
  
struct Metadata {  
    string fName;  
    any fValue;  
};  
typedef sequence<Metadata> MetadataSeq;
```

# Visão

- Definida pelo tipo  
`tecgraf::openbus::data_service::DataView`
- É uma interface abstrata: pode ser implementada por um objeto remoto (interface) ou por um objeto por valor (valuetype)
- É o dado, do ponto de vista do cliente
- O dado só é transferido através de suas visões

# Visão (cont.)

- Contém a chave unívoca do dado
- Uma visão pode ser uma nova versão de uma outra já existente

```
abstract interface DataView {  
    DataKey getKey();  
    string getInterfaceName();  
};  
typedef sequence<DataView> DataViewSeq;
```

# Obtenção Parcial

- ⦿ Pode ser implementada de duas formas:
  - ⦿ Criação de visões representando as diferentes partes do dado
  - ⦿ Modelagem do dado de forma hierárquica
    - ⦿ Os atributos são representados como descendentes na hierarquia

# Serviço de Navegação Hierárquica

- Definido pelo tipo  
`tecgraf::openbus::data_service::IHierarchicalNavigationDataService`

```
interface IHierarchicalNavigationDataService {  
    DataDescriptionSeq getRoots() raises (ServiceFailure,  
DataAccessDenied);  
  
    DataDescriptionSeq getChildren(in DataKey fKey) raises  
(ServiceFailure, InvalidDataKey, DataNotFound, DataAccessDenied);  
  
    DataDescription getParent(in DataKey fKey) raises  
(ServiceFailure, InvalidDataKey, DataNotFound, DataAccessDenied);
```

# Serviço de Navegação Hierárquica (cont.)

```
    DataDescription getDataDescription(in DataKey fKey) raises  
(ServiceFailure, InvalidDataKey, DataNotFound, DataAccessDenied);  
  
    DataView getDataView(in DataKey fKey, in string  
fViewInterface) raises (ServiceFailure, InvalidDataKey,  
DataNotFound, UnknownViewInterface, DataAccessDenied);  
  
    DataViewSeq getDataViewSeq(in DataKeySeq fKeys, in string  
fViewInterface) raises (ServiceFailure, InvalidDataKey,  
DataNotFound, UnknownViewInterface, DataAccessDenied);  
};
```

# Serviço de Gerenciamento

- Definido pelo tipo

tecgraf::openbus::data\_service::IHierarchical  
ManagementDataService

```
interface IHierarchicalManagementDataService {  
    DataKey createData(in DataKey fParentKey, in DataDescription  
fPrototype) raises (ServiceFailure, InvalidDataKey, DataNotFound,  
InvalidPrototype, DataAccessDenied);  
  
    void deleteData(in DataKey fKey) raises (ServiceFailure,  
InvalidDataKey, DataNotFound, DataAccessDenied);
```

# Serviço de Gerenciamento (cont.)

```
    DataKey copyData(in DataKey fSourceKey, in DataKey  
fParentKey) raises (ServiceFailure, UnknownViews, InvalidDataKey,  
DataNotFound, DataAccessDenied);  
  
    void moveData(in DataKey fKey, in DataKey fNewParentKey  
raises (ServiceFailure, UnknownViews, InvalidDataKey,  
DataNotFound, DataAccessDenied);  
  
    void updateData(in DataKey fKey, in DataKey fSourceKey) raises  
(ServiceFailure, UnknownViews, InvalidDataKey, DataNotFound,  
DataAccessDenied);  
};
```

# Serviço de Transferência

- Definido pelo tipo  
`tecgraf::openbus::data_service::IHierarchicalTransferDataService`

```
interface IHierarchicalTransferDataService {  
    DataKey copyDataFrom(in DataKey fSourceKey, in DataKey  
fParentKey) raises (ServiceFailure, UnknownViews, InvalidDataKey,  
DataNotFound, DataAccessDenied);  
  
    void updateDataFrom(in DataKey fKey, in DataKey fSourceKey)  
raises (ServiceFailure, UnknownViews, InvalidDataKey,  
DataNotFound, DataAccessDenied);  
};
```

# Visão não-estruturada

- Definida pelo tipo  
tecgraf::openbus::data\_service::UnstructuredData

```
valuetype UnstructuredData supports
DataView {
    public DataKey fKey;
    public string fHost;
    public unsigned long fPort;
    public OctetSeq fAccessKey;
    public boolean fWritable;
};
```

# File Transfer Channel (FTC)

- Protocolo para transferência de arquivo através de socket
- Define as seguintes operações: open, close, getSize, setSize, getPosition, setPosition, read e write.
- Implementação através de bibliotecas Java, Lua e C++

# CORBA - Objetos por Valor

- ⦿ São definidos pelo tipo `valuetype`
- ⦿ Representam cópias de um dado
  - ⦿ Alterações locais - o objeto original não é alterado no servidor
- ⦿ Criados através de fábricas
  - ⦿ Precisam ser registradas nos ORBs dos clientes
  - ⦿ Uma para cada tipo de objeto

# Exercício 1

- Implementar um cliente para navegar nos dados oferecidos por um serviço.

# Exercício 2

- Transformar o cliente criado no Exercício 1 em um serviço que conte as ocorrências de uma determinada palavra.

Dúvidas ?!?