Architecture of the CORBA Component Model

IR, DII and DSI

Overview

"Dynamic" clients and servers are programs which use interfaces that are not known at development time

- Desired interfaces must be found at runtime
  - through user interaction
  - through control files (e.g. scripting language interpreter)
- IR/IFR: Interface Repository
- DII: Dynamic Invocation Interface
- DSI: Dynamic Skeleton Interface

Interface Repository

- Object Service, defined in module CORBA
- Defines dynamic type definitions
  - nearly complete representation of IDL (except for ordering)
  - maintenance of repository up to sysadmin
- hierarchical and flat access operations
- read and write operations
  - MICO: idl --feed-ir
- initial reference (“InterfaceRepository”)
- introspection (_get_interface())
- one interface per metatype (InterfaceDef, ModuleDef, StructDef, …)

Repository Identifier

- Each IDL type has its own repository identifier
- used in IOR, and as a key in the IFR
- different formats:
  - IDL:<prefix><scope><version>
    - IDL:omg.org/CosNaming/NamingContext:1.0
    - IDL:Uebung1/Server:1.0
    - #pragmas: prefix, ID, version
    - #pragma prefix “hpi.uni-potsdam.de”
  - RMI:<class name>:<hash code>
  - DCE:<uuid>
  - LOCAL:<string>

IR Base Types: IRObject

```
enum DefinitionKind {
    dk_none, dk_all,
    dk_Attribute, dk_Constant, dk_Exception, dk_Interface,
    dk_Module, dk_Operation, dk_Typedef,
    dk_Alias, dk_Struct, dk_Union, dk_Enum,
    dk_Primitive, dk_String, dk_Sequence, dk_Array,
    dk_Repository, dk_Wstring, dk_Fixed,
    dk_Value, dk_ValueBox, dk_ValueMember,
    dk_Native, dk_AbstractInterface, dk_LocalInterface
};
interface IRObject {
    readonly attribute DefinitionKind def_kind;
    void destroy ();
};
```

IR Base Types: Contained

```
interface Contained : IRObject {
    attribute RepositoryId id;
    attribute Identifier name;
    attribute VersionSpec version;
    readonly attribute Container defined_in;
    readonly attribute ScopedName absolute_name;
    readonly attribute Repository containing_repository;
    struct Description {
        DefinitionKind kind;
        any value;
    };
    Description describe ();
    void move ( in Container new_container, in Identifier new_name,
                in VersionSpec new_version);
};
```

IR Base Types: Container

```
interface Container : IRObject {
    Contained lookup (in ScopedName search_name);

    ContainedSeq contents (in DefinitionKind limit_type,
                                        in boolean exclude_inherited);

    ContainedSeq lookup_name (in Identifier search_name,
        in long levels_to_search,      in DefinitionKind limit_type,
        in boolean exclude_inherited);
    struct Description {
        Contained contained_object;
        DefinitionKind kind;
        any value;
    };
    DescriptionSeq describe_contents (in DefinitionKind limit_type,
        in boolean exclude_inherited, in long max_returned_objs);
```

IR Base Types: Container (2)

```
// write interface
ModuleDef create_module (in RepositoryId id,
        in Identifier name, in VersionSpec version);

ConstantDef create_constant (in RepositoryId id,
        in Identifier name, in VersionSpec version,
        in IDLType type, in any value );

StructDef create_struct (in RepositoryId id,
        in Identifier name, in VersionSpec version,
        in StructMemberSeq members);

//...
```

IR Base Types: IDLType

```
interface IDLType : IRObject {
    readonly attribute TypeCode type;
};

interface TypedefDef : Contained, IDLType {
};
```

Primitive Types in the IR

```
enum PrimitiveKind {
    pk_null, pk_void, pk_short, pk_long, pk_ushort, pk_ulong,
    pk_float, pk_double, pk_boolean, pk_char, pk_octet,
    pk_any, pk_TypeCode, pk_Principal, pk_string, pk_objref,
    pk_longlong, pk_ulonglong, pk_longdouble,
    pk_wchar, pk_wstring, pk_value_base
};

interface PrimitiveDef: IDLType {
    readonly attribute PrimitiveKind kind;
};
```

Repository

```
interface Repository : Container {
    Contained lookup_id (in RepositoryId search_id);
    PrimitiveDef get_primitive (in PrimitiveKind kind);

    StringDef create_string (in unsigned long bound);
    SequenceDef create_sequence (
            in unsigned long bound,
            in IDLType element_type
    );
    //...
```

ModuleDef

```
interface ModuleDef : Container, Contained {
};

struct ModuleDescription {
    Identifier name;
    RepositoryId id;
    RepositoryId defined_in;
    VersionSpec version;
};
```

ConstantDef

```
interface ConstantDef : Contained {
    readonly attribute TypeCode type;
    attribute IDLType type_def;
    attribute any value;
};
struct ConstantDescription {
    Identifier name;
    RepositoryId id;
    RepositoryId defined_in;
    VersionSpec version;
    TypeCode type;
    any value;
};
```

StructDef

```
struct StructMember {
    Identifier name;
    TypeCode type;
    IDLType type_def;
};

typedef sequence <StructMember> StructMemberSeq;

interface StructDef : TypedefDef, Container {
    attribute StructMemberSeq members;
};
```

InterfaceDef

```
interface InterfaceDef : Container, Contained, IDLType {
    attribute InterfaceDefSeq base_interfaces;
    boolean is_a (in RepositoryId interface_id);

    struct FullInterfaceDescription {...};
    FullInterfaceDescription describe_interface();

    OperationDef create_operation (in RepositoryId id,
            in Identifier name, in VersionSpec version,
            in IDLType result, in OperationMode mode,
            in ParDescriptionSeq params,
            in ExceptionDefSeq exceptions, in ContextIdSeq
    contexts);

    //...
```

TypeCodes

- predefined IDL type, in module CORBA
- “by-value” description of an IDL type
- difference to IDLType:
  - “stand-alone” (no relationship to container)
  - describes types only as far as relevant for data transmission
    - e.g. description of all fields of a struct
    - does not contain all operations of an interface
- Created through IFR, ORB, or language mapping
  - C++: _tc_<typename> (MICO: only with --typecode)

TypeCode IDL

```
enum TCKind {
    tk_null, tk_void,
    tk_short, tk_long, tk_ushort, tk_ulong,
    tk_float, tk_double, tk_boolean, tk_char,
    tk_octet, tk_any, tk_TypeCode, tk_Principal, tk_objref,
    tk_struct, tk_union, tk_enum, tk_string,
    tk_sequence, tk_array, tk_alias, tk_except,
    tk_longlong, tk_ulonglong, tk_longdouble,
    tk_wchar, tk_wstring, tk_fixed,
    tk_value, tk_value_box,
    tk_native,
    tk_abstract_interface,
    tk_local_interface
};
```

TypeCode IDL (2)

```
interface TypeCode {
    boolean equal (in TypeCode tc);
    TCKind kind ();

    // for tk_objref, tk_struct, tk_union, tk_enum, tk_alias,
    // tk_value, tk_value_box, tk_native, tk_abstract_interface
    // tk_local_interface and tk_except
    RepositoryId id () raises (BadKind);

    // for tk_objref, tk_struct, tk_union, tk_enum, tk_alias,
    // tk_value, tk_value_box, tk_native, tk_abstract_interface
    // tk_local_interface and tk_except
    Identifier name () raises (BadKind);
```

TypeCode IDL (3)

```
// for tk_struct, tk_union, tk_enum, tk_value,
// and tk_except
unsigned long member_count () raises (BadKind);

// ...
```

ORB Operations on TypeCodes

```
module CORBA{
    interface ORB{
        // ...
        TypeCode create_struct_tc (
                in RepositoryId id;
                in Identifier name,
                in StructMemberSeq members
        );
        TypeCode create_interface_tc (
                in RepositoryId id,
                in Identifier name
        );
        // ...
```

DII

- Dynamic Invocation Interface
- Request Objects
  - created using object reference (C++: _request)
  - contains operation name, parameters, result
  - provides methods for sending request, receiving result
- ORB typically requires all type information from the application

Request Initialization

```
// IDL
interface Server{
    string reverse(in string message);
};
//C++
    Object_var obj = orb->string_to_object(ior);
    Request_var req = obj->_request("reverse");
    req->add_in_arg() <<= "Hello, world";
    req->result()->value()->set_type(CORBA::_tc_string);
```

Request Invocation

```
req->invoke();
// check req->env()->exception()
CORBA::String_var result;
req->result()->value() >>= result;
```

- alternatively: send_deferred(), send_oneway()
- deferred invocation:
  - request->poll_response()
  - orb->get_next_response()
  - orb->poll_next_response()

Dynamic Skeleton Interface

- servants don’t “know” at implementation time which interface they implement
- derive from PortableServer::DynamicImplementation
  - char* _primary_interface(const ObjectId&, POA_ptr) = 0;
  - void invoke(CORBA::ServerRequest_ptr) = 0;