CORBA Einführung

CORBA dient zum Aufruf verschiedener definierte Methoden und Attribute in einem Server aus Clients verschiedener Programmiersprachen.

Bestandteile von CORBA:

- Client → Dynamic invocation interface, Static IDL stubs
- Beide → ORB interface
- Object implementation/Server → Static IDL skeleton, Dynamic skeleton interface → Portable object adapter
- Die ORBs haben ein Interface repository und ein implementation repository beide sind Teile des ORB
    - Interface repository wird benötigt damit die ORB weiß wie er die Daten weitergeben muss in was er sie marshaln muss und ob er diesen Dateityp benutzen kann
        - Es wird überprüft:
            - Typen in der Methodensignatur überprüfen
            - Vererbungshierarchie checken
            - Kommunikation zwischen den verschiedenen ORB´s
    - Implementation repository nur damit man zwischen verschiedenen ORBS kommunizieren kann

IDL/-Compiler

Interface Definition Language (IDL)

Die IDL Files beinhalten die zu Grunde liegenden Methoden-und Attributdefinitionen, jedoch nicht deren Implementationen, außerdem dient als Schnittstelle zwischen mehreren Programmiersprachen. CORBA greift dann darauf zu und gibt an wie bei einem Aufruf deren Typen und Methoden zu benutzen sind.

Die IDL definiert nur die public Elemente in einer Server Applikation.

Nun ein Beispiel dazu:

```
module Finance {
  typedef sequence<string> StringSeq;
  struct AccountDetails {
    string     name;
    StringSeq  address;
    long       account_number;
    double     current_balance;
  };
  exception insufficientFunds { };
  interface Account {
    void deposit(in double amount);
    void withdraw(in double amount) raises(insufficientFunds);
    readonly attribute AccountDetails details;
  };
};
```

In IDL kann man einen Module-Namen angeben. Dieser fungiert wie ein Java-Package und hällt so die entsprechenden Klassen zusammen. Mit dem Scopeoperator kann man aus einem anderen Module auf eine andere Methode/Struct/Interface/usw... zugreifen.

Es kann read only Attribute und read and write Attribute geben.

Methoden haben eine fix definierte Richtung (entweder: in, out, inout), dies wird je nachdem entschieden ob der Server einen Parameter bekommt (in) oder der Client einen bekommt (out). Außerdem kann eine Methode eine Exception „raisen“, dies entspricht einem „throws“ in Java. Es gibt 30 verschiedene vordefinierte Exceptiontypen die Systemexceptions heißen.

Man kann nicht jeden Datentyp benutzen, zum Beispiel kann man string, boolean oder long benutzen oder einen eigen-definierten Typen (user-defined) verwenden. Diese können: struct, sequence, Union, Enum, Array und typedef sein.

Der CORBA Standard definiert das Verwenden der IDL zwischen folgenden Programmiersprachen:

C, C++, Java, Ada, Smalltalk, COBOL, PL/I, LISP, Python und IDLScript

Für einige andere Programmiersprachen gibt es auch inoffizielle Schnittstellen.

Zum generieren des Stubs und Skeletons(serverseitiger Stub) benötigt sprachenspezifische Übersetzer für die IDL.

IDL Compiler

Der IDL Compiler übersetzt die Definitionen der IDL in die jeweiligen Definitionen der spezifischen Sprache.

Dieser Compiler erzeugt also den Stub für Client und Server für jedes IDL File, je nachdem was dann benutzt wird muss der Stub unterschiedlich implementiert werden (Server tatsächliches „berechnen“ und im Client aufrufen). Der Stub wird auch oft Proxy genannte.

Namingservice

Der Namingservice oder Tradingservice ermöglicht es dem Client den Server und seine Methoden einfacher zu lokalisieren und dem Server sich an einer fixen Stelle im Netzwerk/WAN/LAN anzumelden und erreichbar zu sein. Dieser Namingservice kann bei jedem Verzeichnisdienst (LDAP …) angemeldet werden.

POA/-Manager

Der POA(Portable Object Adapter) ist eine Art Container, in den die Objekte, die der Server per Remote anbietet gelegt werden. Verschiedene POAs können unterschiedliche Niveaus an Qualität haben. Ein POA kann single-threaded, multi-threaded, usw… sein. Diese unterschiedlichen Niveaus kommen von den Objekten, die hinein gelegt werden, daher können diese eben dieselben unterschiedlichen Niveaus an Qualität haben. Also können durch mehrere POAs in einem Server-Prozess, mehrere Objekte unterschiedlicher Qualitäten angeboten werden. Außerdem gibt es noch die Eigenschaften transient(temporär) und persistent, die beschreiben ob die Objekte bei einem

Crash des Server-Prozesses bestehen bleiben oder nicht. CORBA erstellt standardmäßig einen „rootPOA“, der multi-threaded und transient ist. Deswegen muss man bei den meisten Implementierungen kein POA extra erstellt werden, wenn die Eigenschaften der Implementation keine extra Eigenschaften des POA benötigen.

Ein POA-Manager kontrolliert nun den Fluss der eingehenden Anfragen. Der POA manager, der mit dem root POA verbunden ist, ist beim Start standardmäßig auf holding gesetzt, das heißt das Anfragen die eingehen automatisch in eine Warteschlange gelangen. Dadurch kann sich der Server-Prozess fertig initialisieren, ohne sich um eingehende Anfragen kümmern zu müssen. Wenn der Server fertig initialisiert ist, setzt dieser den POA-Manager auf active, sodass die eingehenden Anfragen bearbeitet werden.

ORB – Object Request Broker

Auf beiden Seiten also Client und Server muss die ORB Library vorhanden sein damit die Interaktion zwischen beiden funktioniert.

Beim Aufrufen einer Methode wird der Befehl des Clients gemarshalt und dann dem ORB übergeben, dieser hat dann die Aufgabe dem gegenüberliegenden ORB (serverseitig) diesen Code übers Netzwerk zu schicken, danach wartet der clientseitige ORB, nun wird der ORB in dem Server den Code über den Stub unmarshaln, danach wird die benötigte Methode aufgerufen und wieder zurück an den Server ORB gegeben. Dieser gibt dann das Ergebnis wieder dem Client ORB.

Im Grunde genommen ist also ein ORB der Übermittler aller Daten zwischen Server und Client.

Schreiben einer CORBA Applikation (simpel)

Man kann eine Corba Applikation so schreiben indem man eine IDL schreibt (oben beschrieben)

Dann lässt man sich die Stubs generieren(Orbacus, Jacorb usw)

Danach kann man den Client und den Server erstellen je nach dem in was für einer Sprache man sich den Stub generieren hat lassen.

Nun meldet man den Server an einem Nameservice wie LDAP an und kann über den ORB den Client und den Server kommunizieren lassen.

Dynamic Invocation Interface DII

- Client-seitiger Aufruf einer Methode die zur Compile-Zeit noch unbekannt ist
- Um einen Aufruf durchzuführen:
  - Objekt Referenz der Adresse des Servers
  - Operation die ausgeführt werden soll
  - Parameter dieser Operation
- Methoden:
  - Get_interface()
  - Describe_interface()
  - Create_request()
  - Invoke oder send ()

Dynamic skeleton Interface DSI

- Das gleiche auf der Serverseite
- Das Objekt weiß zur Compilezeit noch nicht welches Interface es implementieren muss
- Der Server muss die Methode invoke(ServerRequest) implementieren
- Das Ergebnis kann dann via set_result() oder set_exception() auf dem ServerRequest gesetzt werden

Weitere CORBA Services

Notification	Transaction	Persistent state
Life cycle	Security	Event

Quellen

[1] A Simple C++ Client/Server in CORBA; Carlos Jiménez de Parga; Codeproject; September 2009; Online:http://www.codeproject.com/Articles/24863/A-Simple-C-Client-Server-in-CORBA; abgerufen 1.3.2012

[2] ”Distributed systems –
principles and paradigms” Andrew S. Tanenbaum, Maarten van Steen; 2007; Pearson Prentice Hall

[3] CORBA Frequently Asked Questions,http://www.omg.org/gettingstarted/corbafaq.htm

[4] CORBA Specifications, http://www.omg.org/spec/CORBA/3.1/

[5] Distributed Systems Technologies, Lorenz Froihofe