Interoperability in Distributed Systems

CS 451 Lecture 2003

Prof. R.K. Joshi
Dept. of CSE
IIT Bombay
RPC not enough!

Why?

Need to handle multiple programming languages for implementations (decouple communication from implementation)

RPC is only low level comm. Paradigms, but not a full fledged infrastructure for distributed applications development
Interoperability

cobol

C

C++

java
Consequences of supporting Interoperability

Interoperability vs. portability vs. Compatibility?
Ideas for obtaining a basic architecture supporting interoperability?

- Type system
- Mappings

CTS $\rightarrow$ LTS
- C client stub
- Java client stub
- Cobol server stub
- C server stub
- C++ server stub ..

LTS $\rightarrow$ CTS
The solution

use a standard language for describing interface
The solution: Export

Interface Of Program A

CTS to JAVA

Skeleton Specification

Skeleton Realization

Export Logic
The solution: Import

- Interface of Program A
- Proxy Specification
- Client Implementation
- Import Logic

CTS to C
What constitutes an Interface Specification Language

- A Type system
- Basic and aggregate types
- Call semantics
- Parameter directions
- Reuse of types

control constructs excluded!
IDL: Types

Basic types: Int, bool, char, float,
Aggregate type specifiers [], Struct
Interface type: interface foo{ .. }
Function type: int f(int a, char c);
Direction specifiers:

int f((in int a, in char c, out
char c, inout int b));

Exceptions e,e2;
Throwables: foo throws e;

Call semantics:
Interface I {
  One way f(int a);
}

Reuse through inheritance
interface J extends interface I { ...}
A Distributed System Infrastructure

- Basic Remoting Architecture with interoperability
- Added Services for Applications building
  - Location services
  - Activation services
  - Repositories
  - Event services
  - Persistence
  - Timing services
  - Transactions servers
  ...

Example Distributed Systems Infrastructure

- CORBA
- COM/DCOM → .NET
- J2EE