

Distributed Processing: CORBA, DCE, and Java

Andy Bond
Keith Duddy
Kerry Raymond

{bond, dud, kerry}@dstc.edu.au
CRC for Distributed Systems Technology
University of Queensland
Brisbane 4072 Australia

Abstract

1. Introduction

[need to devise a common simple example expressed in CORBA IDL, DCE IDL, and Java]

[IN an ancient paper, we once compared DCE and ANSAware under a number of categories. Might find the categories useful to structure this work too.]

- software components (what pieces you get when you open up the box)
 - * e.g. compare CDS against Naming, Trading in CORBA
- applications development
 - * Paradigms
 - * IDL syntax
 - * Supported IDL Data Types
 - * Methods of exchanging information during an RPC
 - * RPC parameter direction
 - * Types of RPCs
 - * Inheritance from IDL files
 - * Interface type identifier
 - * programming languages
 - * Offering an interface
 - * Support for multiple instances of the same interface type by an object
 - * Forms of Binding
 - * Advanced programming Concepts
 - * Concurrency
 - * Security
-
- infrastructure
 - * Platforms
 - * Interoperability between heterogeneous platforms
 - * Transport Protocols
 - * Process interaction transparencies

- * Data transparencies
- * Support for Robustness
- * Support for scalability and incremental growth
-
- systems administration -- what administration do you do at different levels
 - * Global Domain
 - * Local Domain
 - * Node
 - * Object
 - *

See /home/ashley/doc/papers/aw_vs_dce/final.doc for more details (Frame).

2. CORBA

Consists of ORB, services and facilities (horizontal and vertical). list them.

aimed to standardise O-O specifications, not implementations

many language mappings

strong C++ influences, but O_O in CORBA is a simpler model

multiple inheritance of interfaces

object-oriented in theory, not so O-O in practice due to the cost of fine-grained objects

no thread support in CORBA2 spec (only in impls)??

infrastructure is presented as objects

3. DCE

Consists of RPC/IDL, CDS, Security, Time, Threads, DFS

aimed to distribute C programs, no general support for other programming languages

aimed to build a common implementation, spec is defined by implementation

priority was interoperability, sex came second

object-based rather than object-oriented

infrastructure is presented as libraries

4. Java

aimed to be used in embedded systems, e.g. toasters and washing machines

need to be safe. Type safety (no pointers). Memory safety (garbage collection)

Java is O-O and simpler than C++, e.g. single-inheritance only for implementations (but multiple for interfaces)

Success of "java" often misattributed to the language being simple. On the contrary, the success of "Java" is based on:

- its interpreted and hence portable (remember how portable C compilers made C portable and hence popular)

- browsers contain “sandpit” interpreter for Java Virtual Machine (no perceived risk)
- can download applets via WWW (easy distribution)

But other languages could be interpreted using JVM and downloaded as applets.

Java as a language does not cause distributed processing. Downloading an applet can be a means of distribution of an application, but this isn't distributed processing and it can be done with any applet regardless of language of development. Due to sandpit security, applets tend to do compute-and-display tasks. Greater functionality requires call-back to host site.

Java programs can call socket library. In applets, sockets are restricted back to “host site”. Some firewalls interfere with socket use (even to host site). Hence “tunnelling” is used.

Distributed applications can be built from java programs using sockets for communication, but this is hard work. Nicer to use some kind of RPC rather than sockets.

Java RMI allows Java programs to remote-call other objects. Enables distributed processing, not so scalable without the kind of services offered by DCE and CORBA. Still, sufficient for many purposes, and easy to do -- just inherit from RMI class.

Hence Java ORBs (ref to famous book by Vogel and Duddy). Hence Java DCE.

threads??

infrastructure is implemented as objects/libraries????

5. Comparison between CORBA, DCE, and Java

CORBA is believed to be light-weight compared with DCE, but usually does not compare equivalent functionality

Java considered simple and light-weight but is it too early in its life to judge on equal basis. Most O-O languages start out simple but get complicated to be more industrial-strength.

Java RMI likely to be popular for simple distributed processing. More constraints imply move to CORBA (e.g. if multiple languages required), or DCE (if “industrial strength” security required).

References