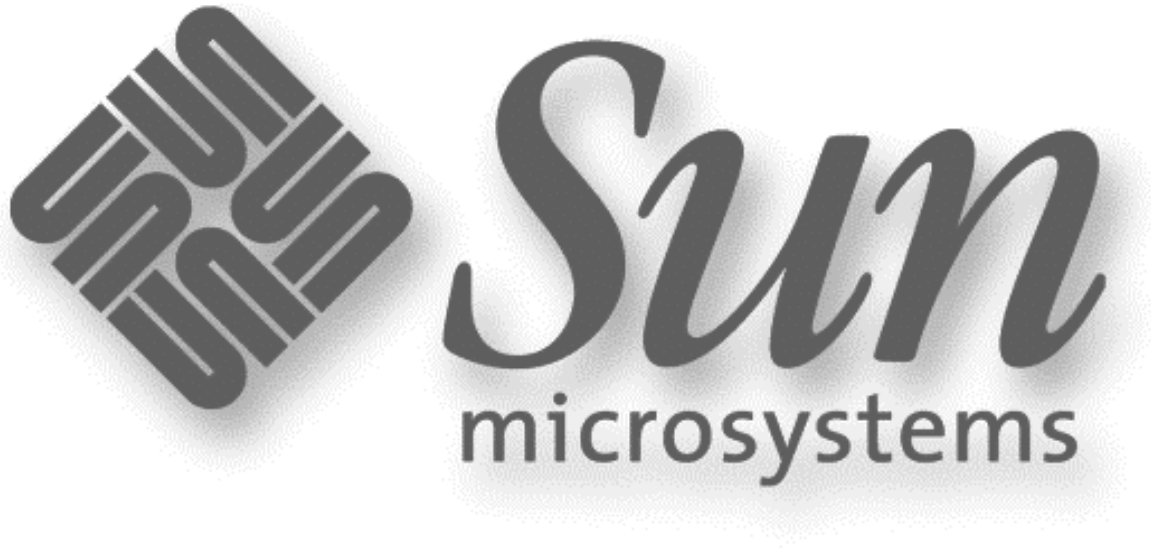
CORBA, JavaIDL, NEO, Joe and more...

Daniel J. Berg
Chief Technology Officer
U.S. Reseller Channel
daniel.berg@Sun.COM



Agenda...

- **CORBA Overview**
  - Architecture
  - IDL
- **Solaris NEO**
  - CORBA Implementation
  - Features
- **Joe**
- **JavaIDL**

CORBA Overview

Heterogeneous Distributed Computing

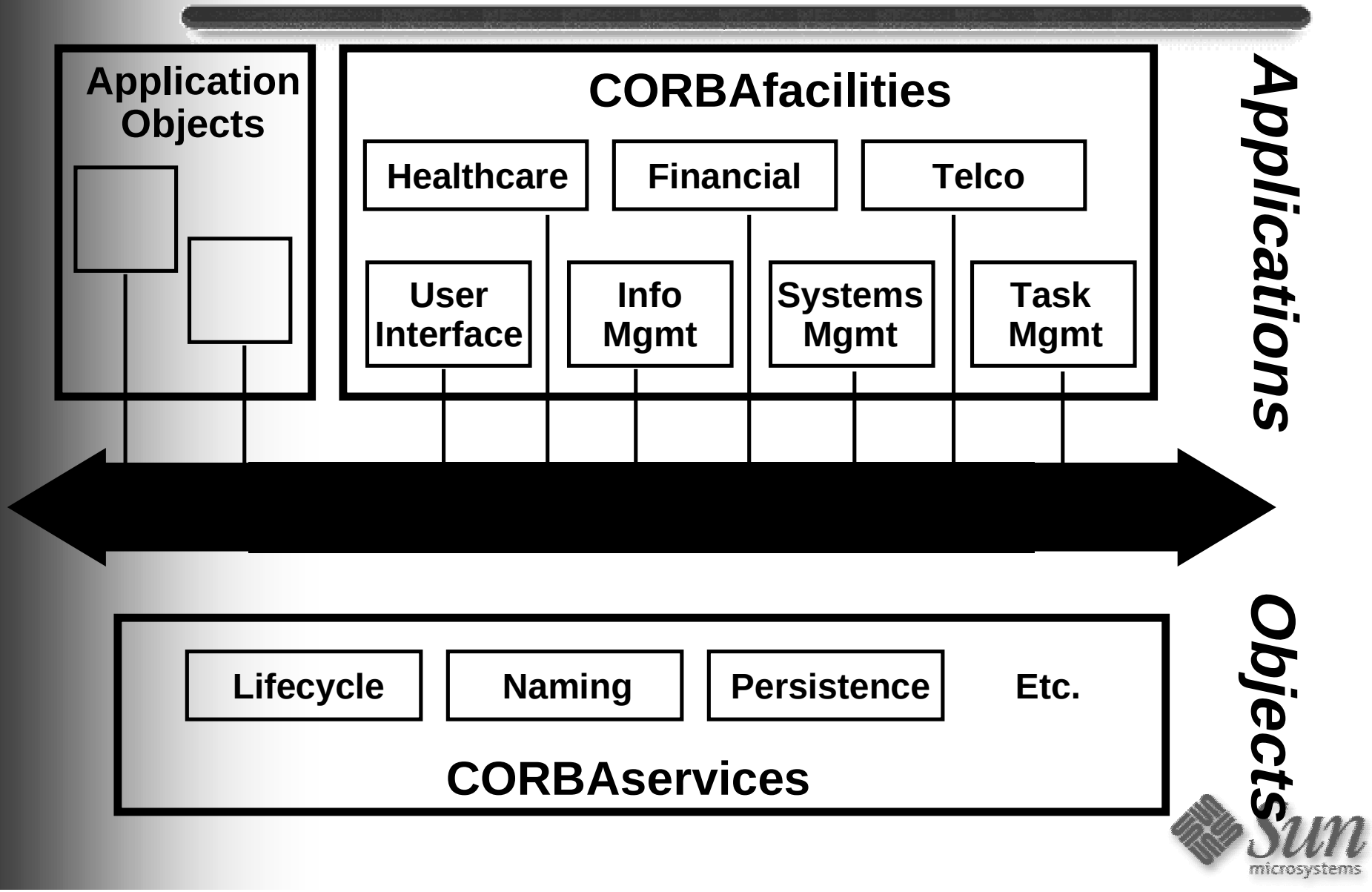
- Diversity in HW and SW
- Need for Interoperability, from pagers to supercomputers
  - Software is the problem today
    - Development time and cost
    - Software Systems Integration
- Component Software is the Solution
  - interoperability and reuse

Sun microsystems

Networked Objects: The Benefits

- Easier to build enterprise applications
- Better way to deploy enterprise applications
- Easier to modify enterprise applications
- Easier to administer networked systems

OMG's Object Management Architecture



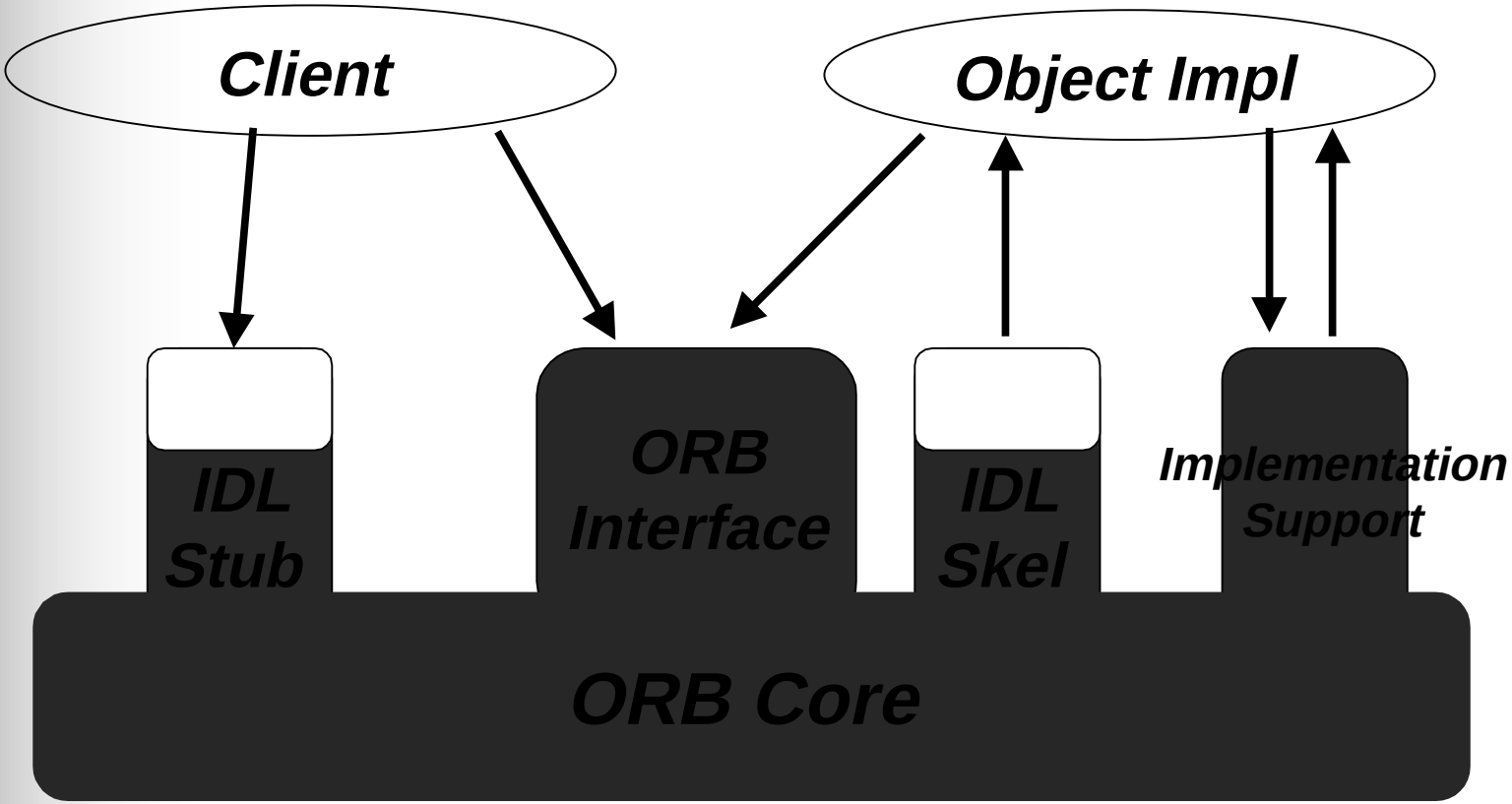
Sun microsystems

- **CORBA objects can be located anywhere on a network**
- **CORBA objects can run on any platform**
- **CORBA objects can be written in any of several languages**
- **Location, Platform and Language are invisable**

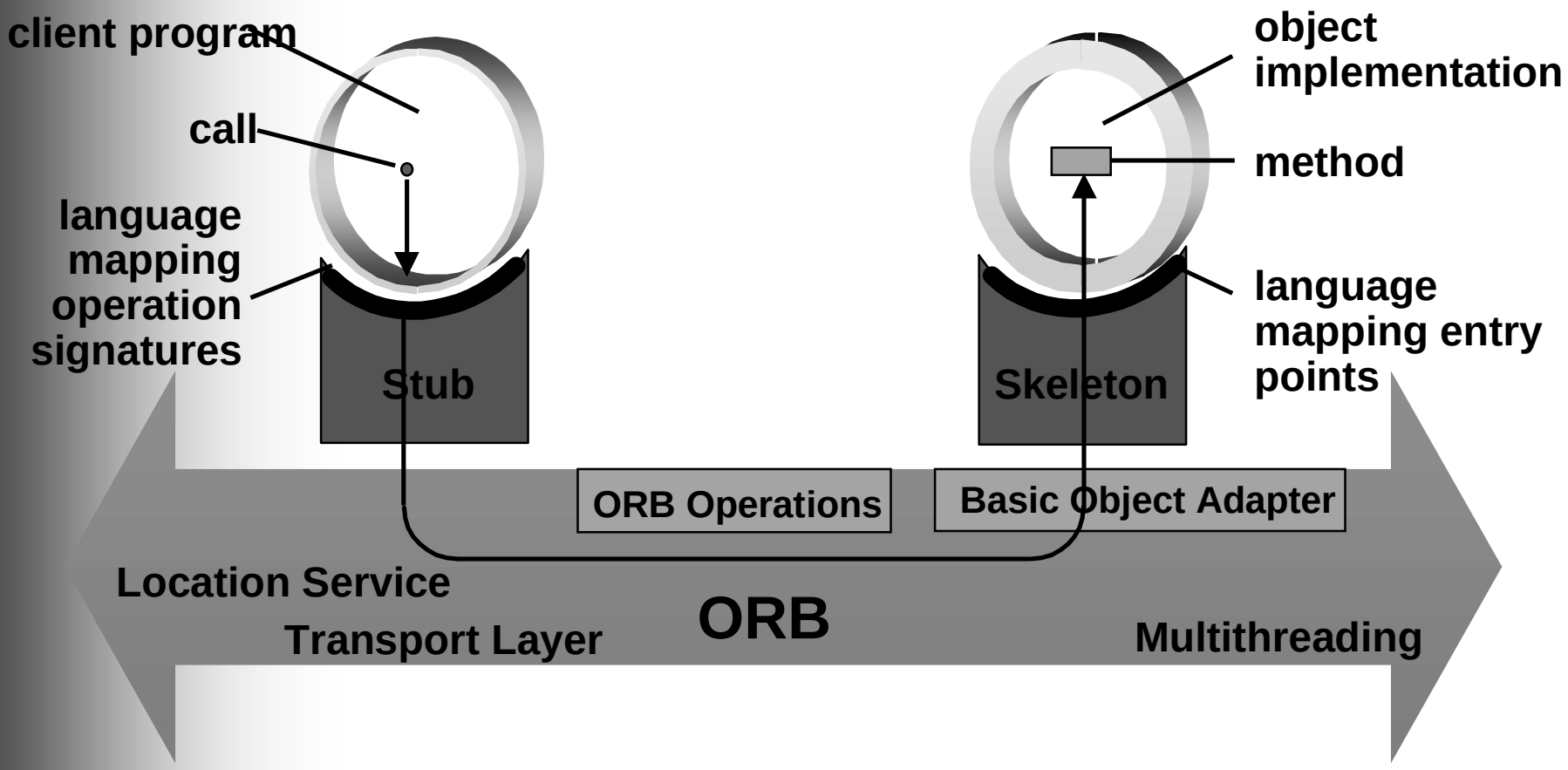
The ORB

- **Responsible for the following mechanisms:**
  - Finding object implementation
  - Preparing the remote object to receive a request
  - Communicating data making up the request
- **All of this is independent of where the object is locatated**

ORB Struture



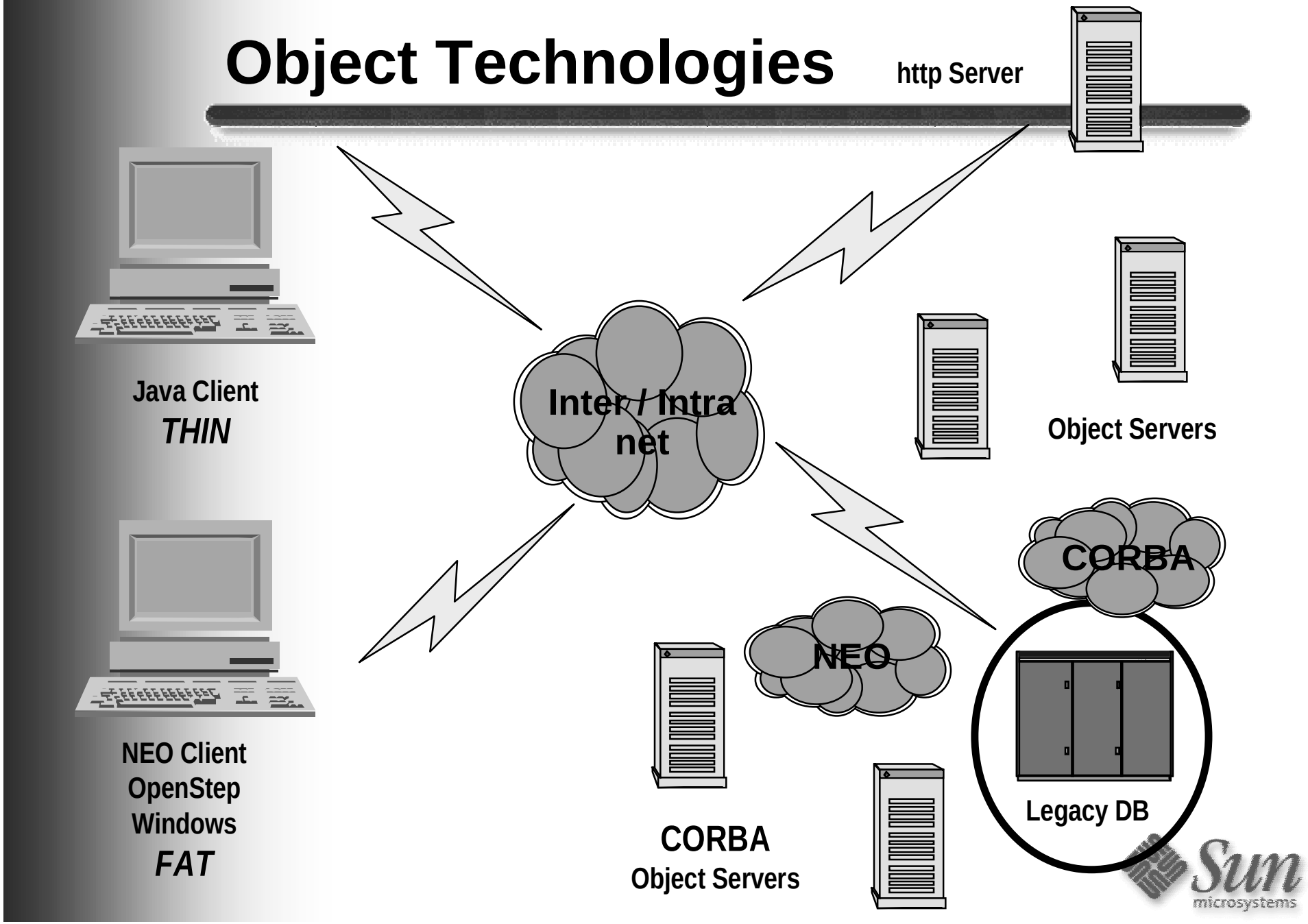
Invocation


client program
call
language mapping operation signatures
Stub
object implementation
method
language mapping entry points
Skeleton
ORB Operations
Basic Object Adapter
Location Service
Transport Layer
ORB
Multithreading

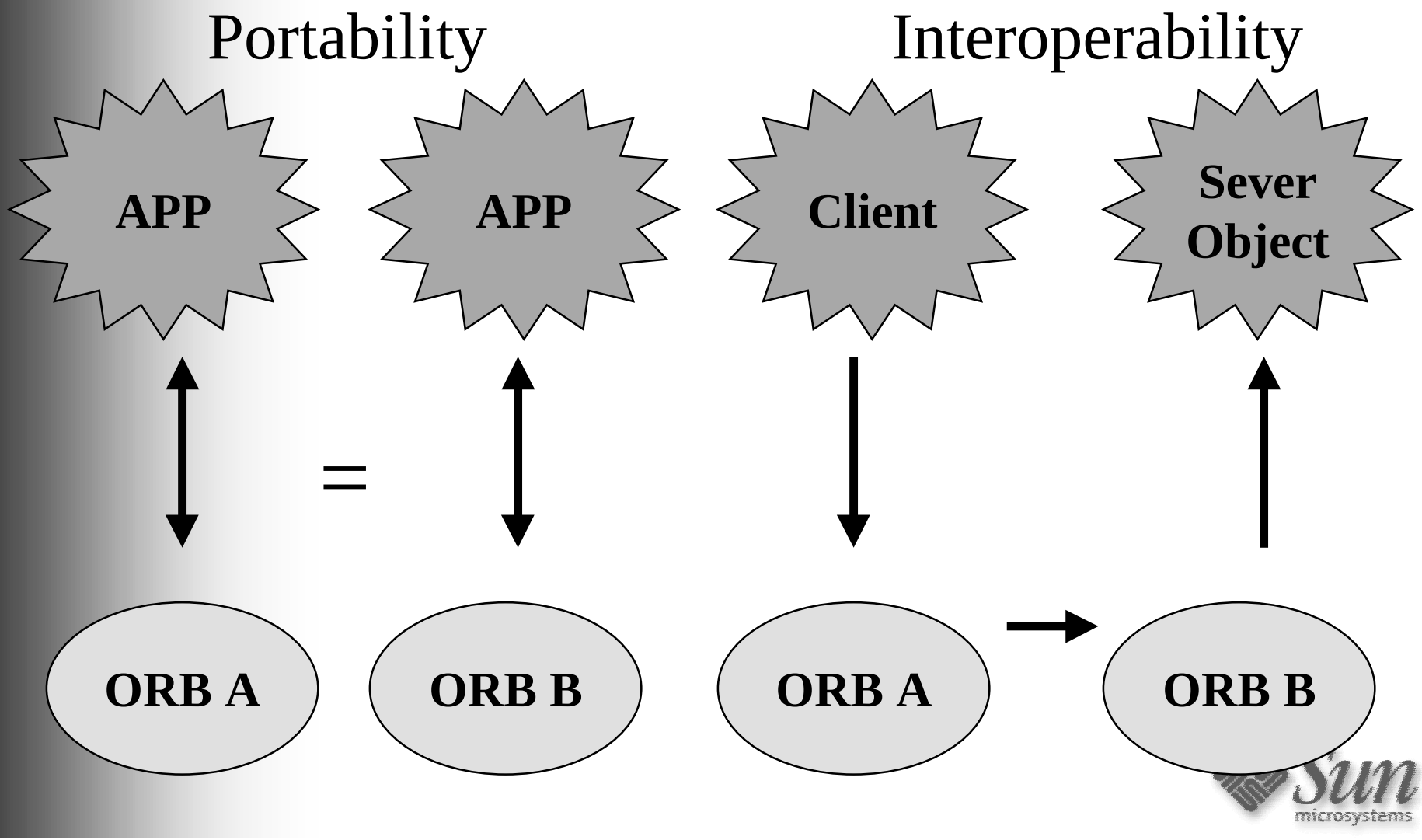
CORBA Benefits

- **SW development tools don't change**
- **"best practice" for Software lifecycle**
  - OO analysis and design
  - OO language implementation
    - Languages, DataBases, User Interfaces
  - Distributed Object Environment for Deployment
- **Legacy apps on equal basis via IDL and wrapper code**
- **Maximize Programmer Productivity**
  - off-the-shelf tools
  - standardized CORBAservices, CORBAfacilities
  - platform independence
- **Code reuse**
  - components, as-is, in new or dynamically reconfigured apps
  - new services via stepwise refinement
- **Mix and match tools in a project**
  - Java, C/C++, Objective-C

Object Technologies



Interoperability Vs. Portability



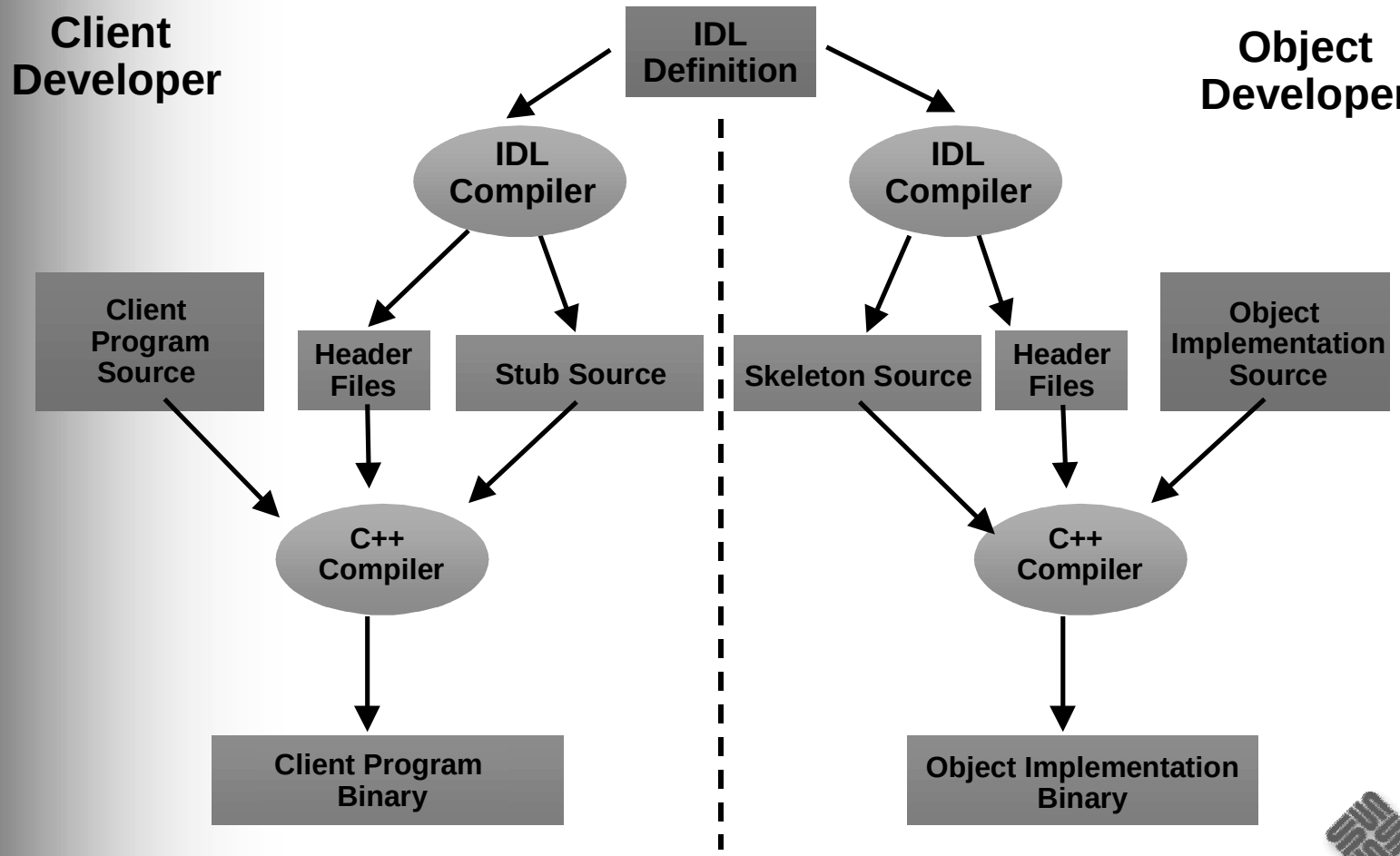
Interface Definition Language

- **IDL - Describes only the interface**
- **Language independent description**
- **IDL is mapped to into languages**
- **IDL mappings Sun has today**
  - C, C++, Java

```
module Bank {
  interface Teller {
    float getBalance();
    void deposit(in float amt);
    void withdraw(in float amt);
    };
  };
```

Sun microsystems

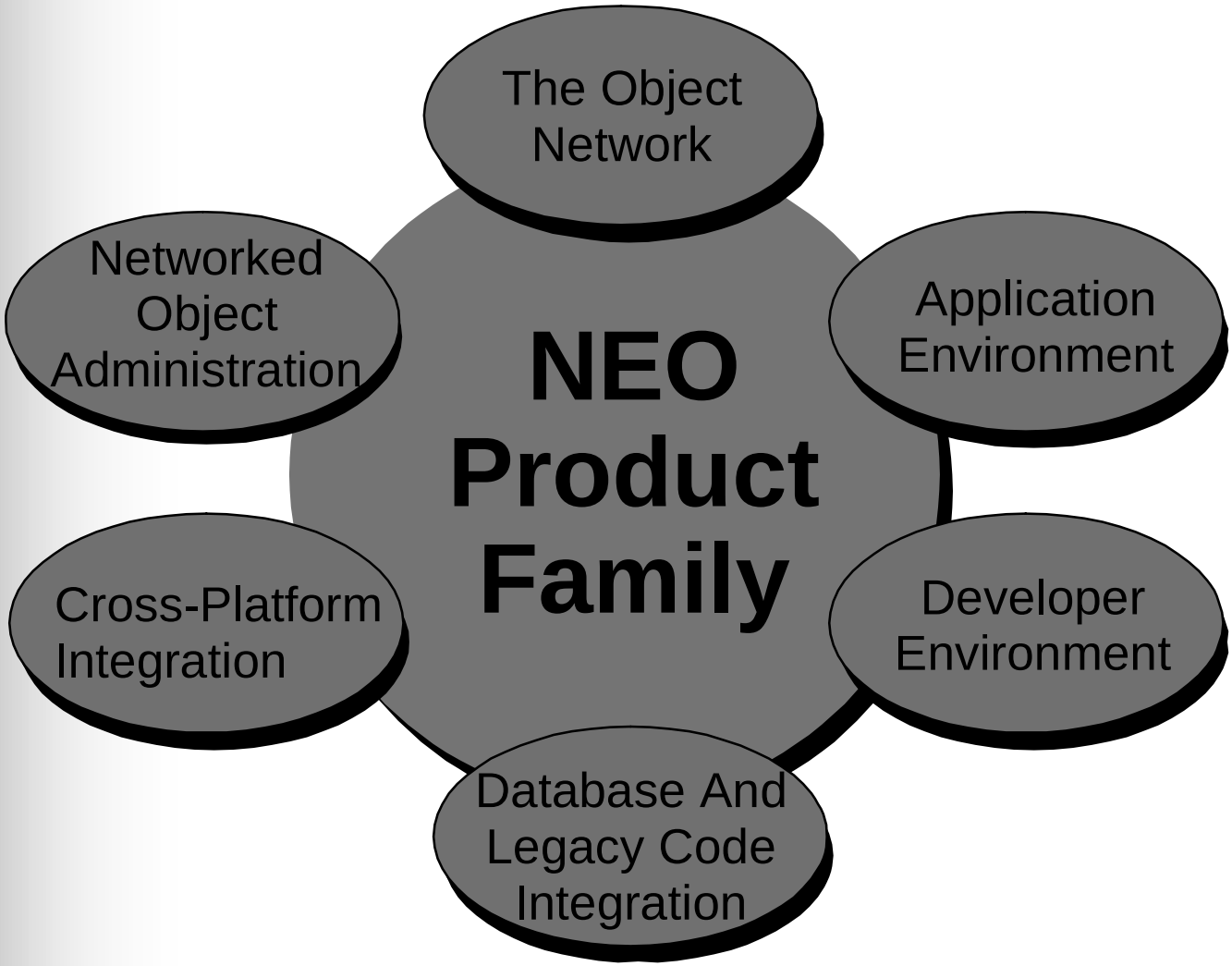
Client / Object Development Process



Sun microsystems

Solaris NEO

NEO: The Complete Environment



Sun microsystems

Things to think about

- Installation support
- Centralized server management
- Finding common services
- Debugging: tracing/logging
- Exception handling
- Safe simultaneous requests
- “Servant” code
- State persistence
- Object life-cycle, activation, deactivation
- Server startup and shutdown

NEO Server

- **Builds on and simplifies NEO Network ORB server process activation**
- **Provides transparent management of the availability of object implementations grouped in a server program**
  - Automates server process startup on arrival of request for any object in server program
  - Automates server process shutdown after period of inactivity
- **With Persistent Object Availability, minimizes use of system resources**

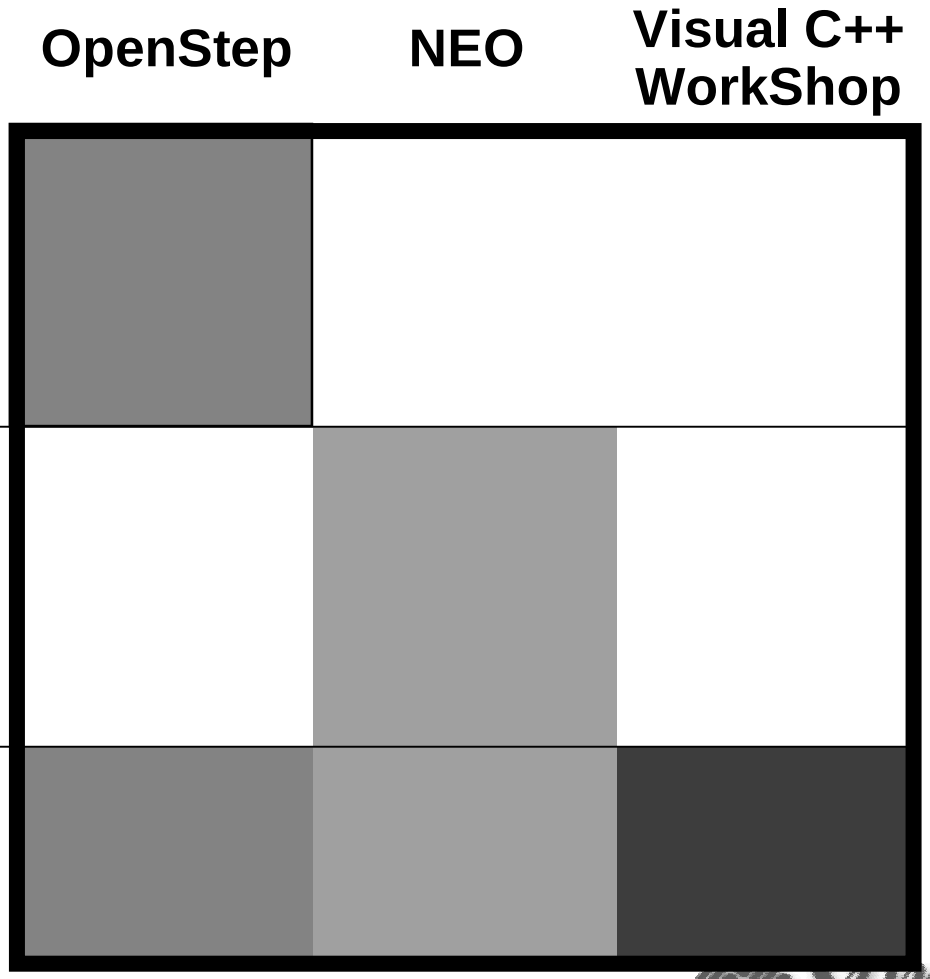
NEOshare - Basic Shared Services

- **Workgroup Support and Shared Service Finder**
  publish/subscribe by Workgroup
- **Concurrency Requests**
  creation and management of threads and deadlock avoidance
- **Server and Persistent Object Availability**
  automatic start, re-start and management of object context
- **Server Management**
  balance computer loads, collect logging and tracing information, track errors
- **Application Installation**
  software installation, registration and upgrade
- **Data Store Manager**
  support for fine-grained objects, type safety, and caching of attribute values
- **Implementation**
  servant creation and management. smart object references, exception handling, object tracing, and message logging

Sun microsystems

NEO Development Environment

- Interface Builder
- Project Builder
- Icon Builder
- Header Viewer
- IDL Compiler
- NEOshare Dev Framework
- Network Object Debugger
- Networked Object Constructor
- SPARCcompiler C/C++
- SPARCworks Teamware
- SPARCworks iMPACT



Sun microsystems

Connectivity for MS-Windows

- Leverage Windows development tools and desktops
- No knowledge of CORBA required
- Win95, WinNT 3.5.1, Win NT 4.0
- CORBA 2.0 compliant

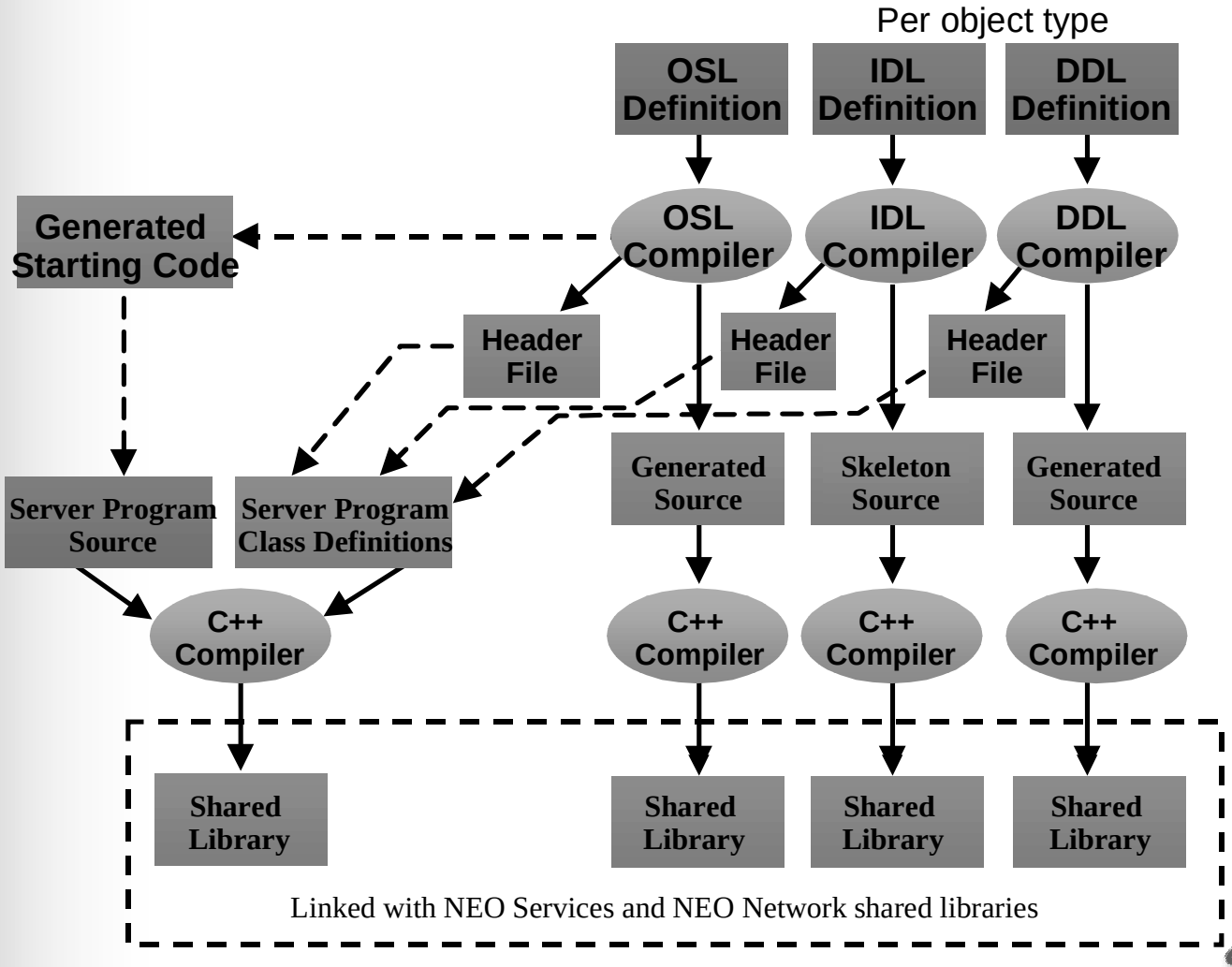
Sun microsystems

Features

- **GUI Tool for browsing and installing NEO shared services**
- **Real-time object conversion between OLE/COM and CORBA**
- **Bi-directional interoperability among OLE, COM and NEO**
- **No changes to client software required**
- **Supports OLE Automation, ActiveX and COM interfaces**
- **Small Footprint -> less than 2 MB**
- **Standards**
  - OMG IIOP for network communications
  - OMG COM/CORBA interoperability

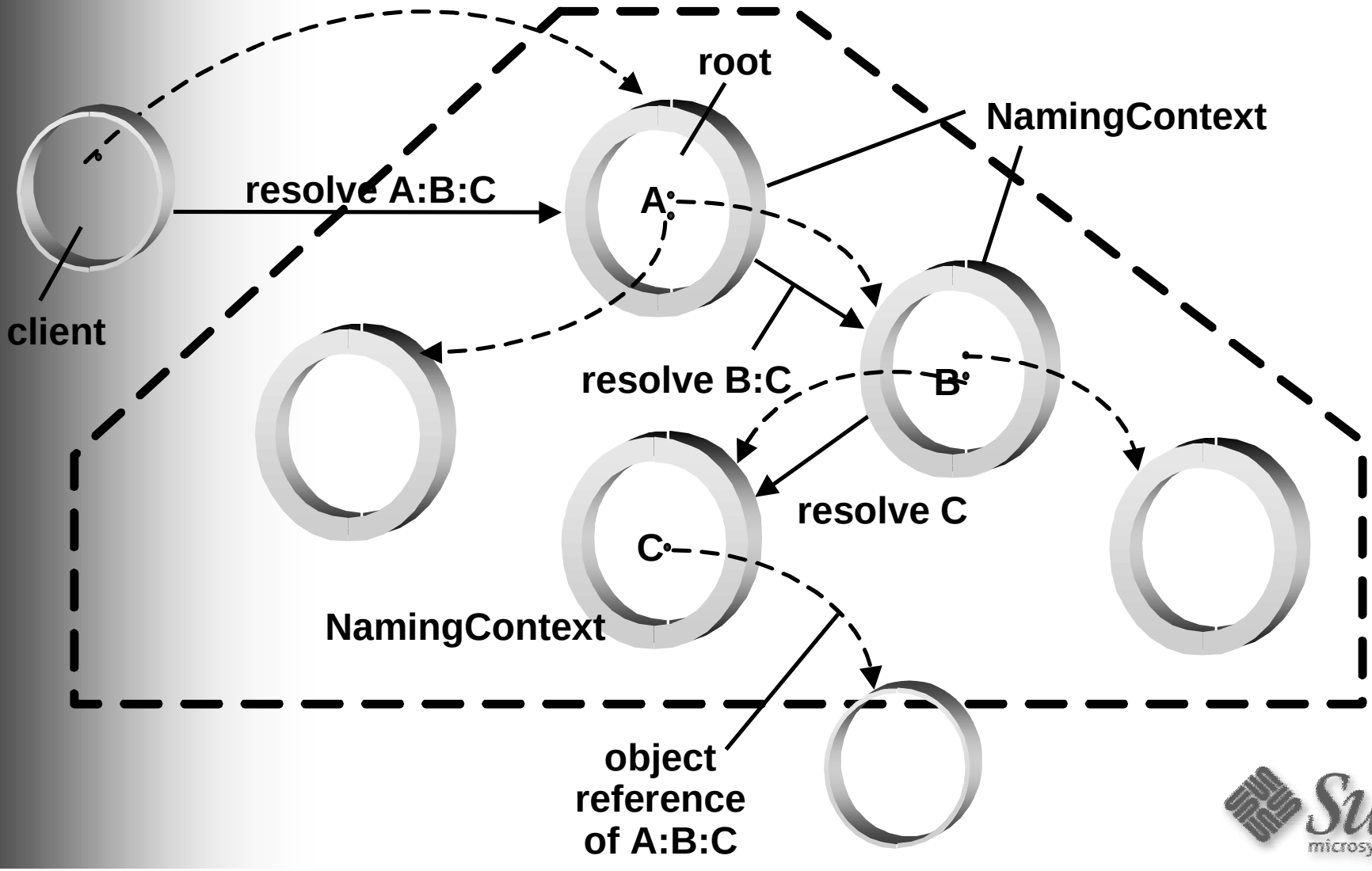
Sun microsystems

Server Development Process



Sun microsystems

NEO Naming Service

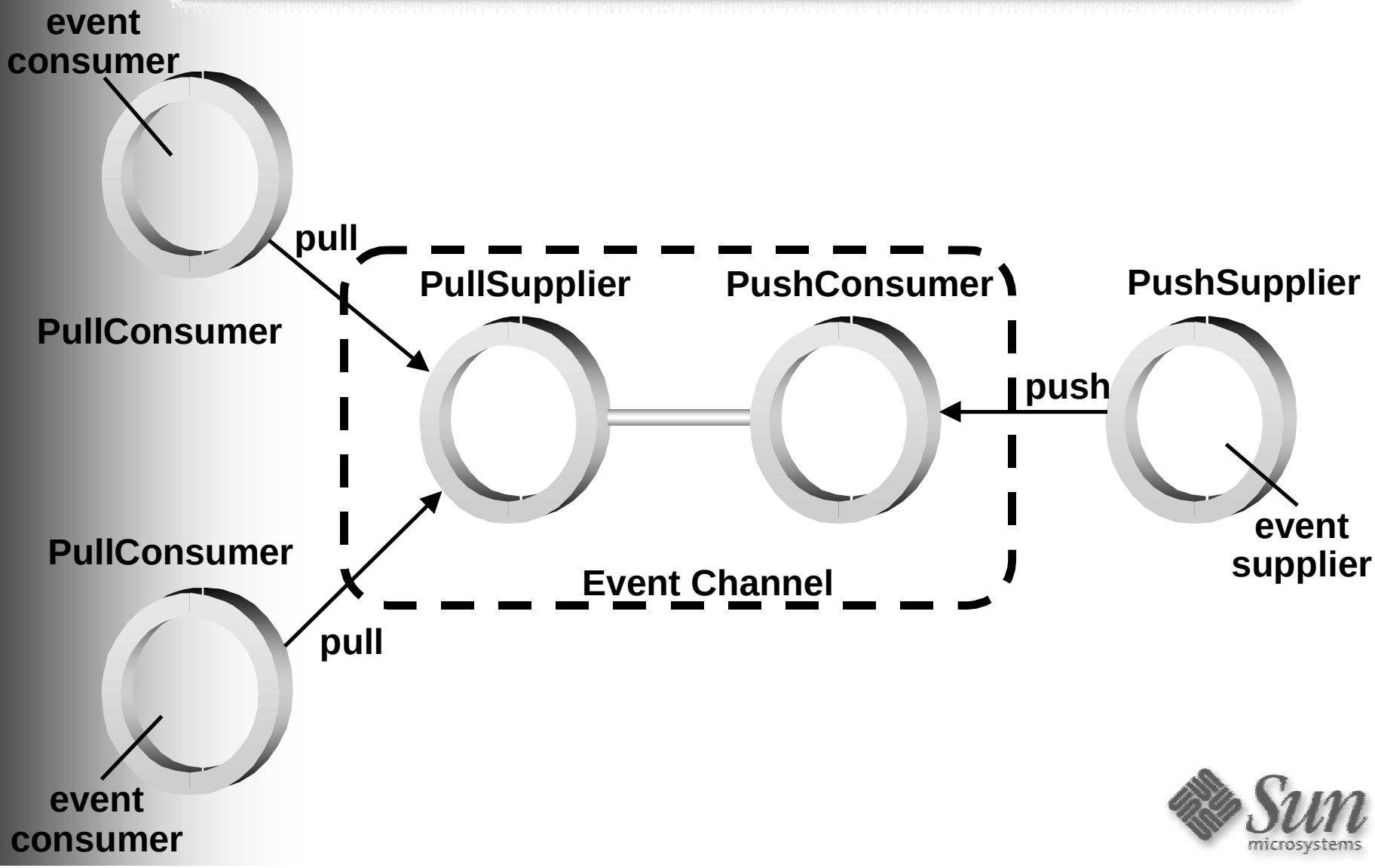


Sun microsystems

NEO Naming Service

- Standardized way of storing and retrieving object references by name
- Provides a hierarchical naming scheme for objects and naming contexts
- Uses canonical representation for compound names (i.e. does not mandate any particular syntax)

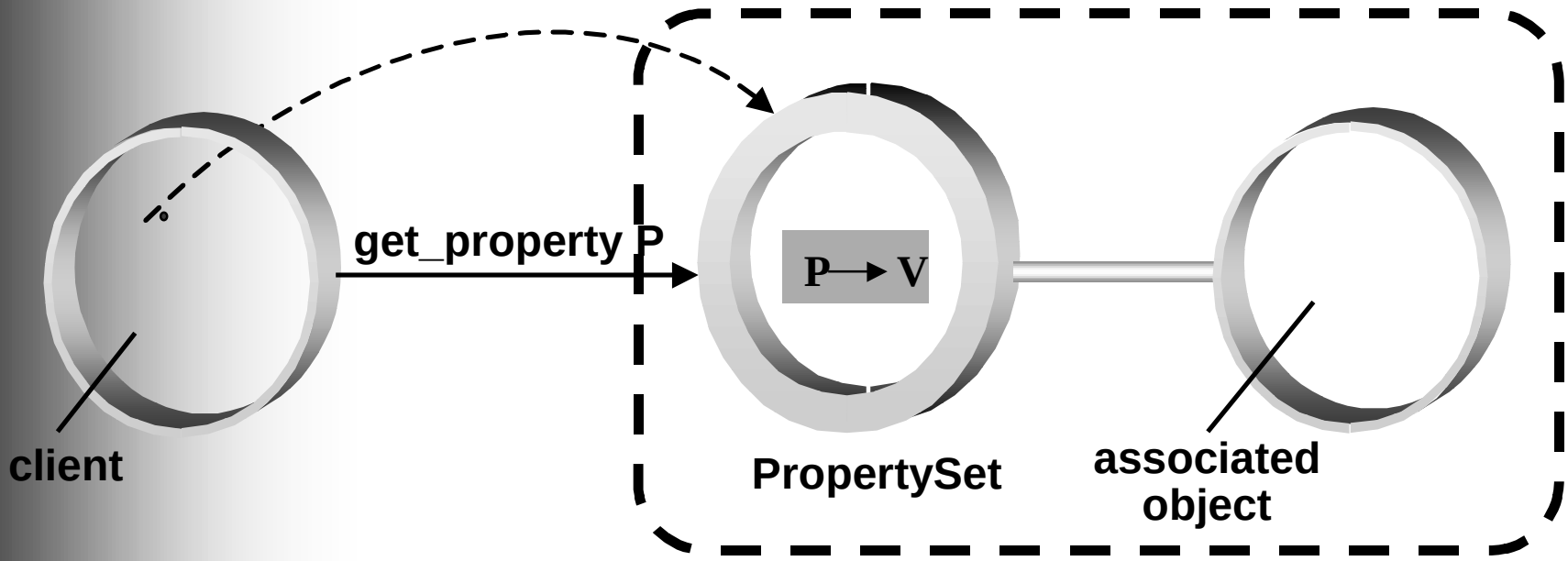
NEO Event Service



NEO Event Service

- Supports asynchronous event notification between event producers and consumers
- Event channels decouple suppliers and consumers and support multiple suppliers and multiple consumers
- Supports push-style and pull-style delivery models and event “fan-in” and “fan-out” (multicast)
- Two implementations provided with different qualities-of-service: (1) fully persistent (2) transient events, persistent connections

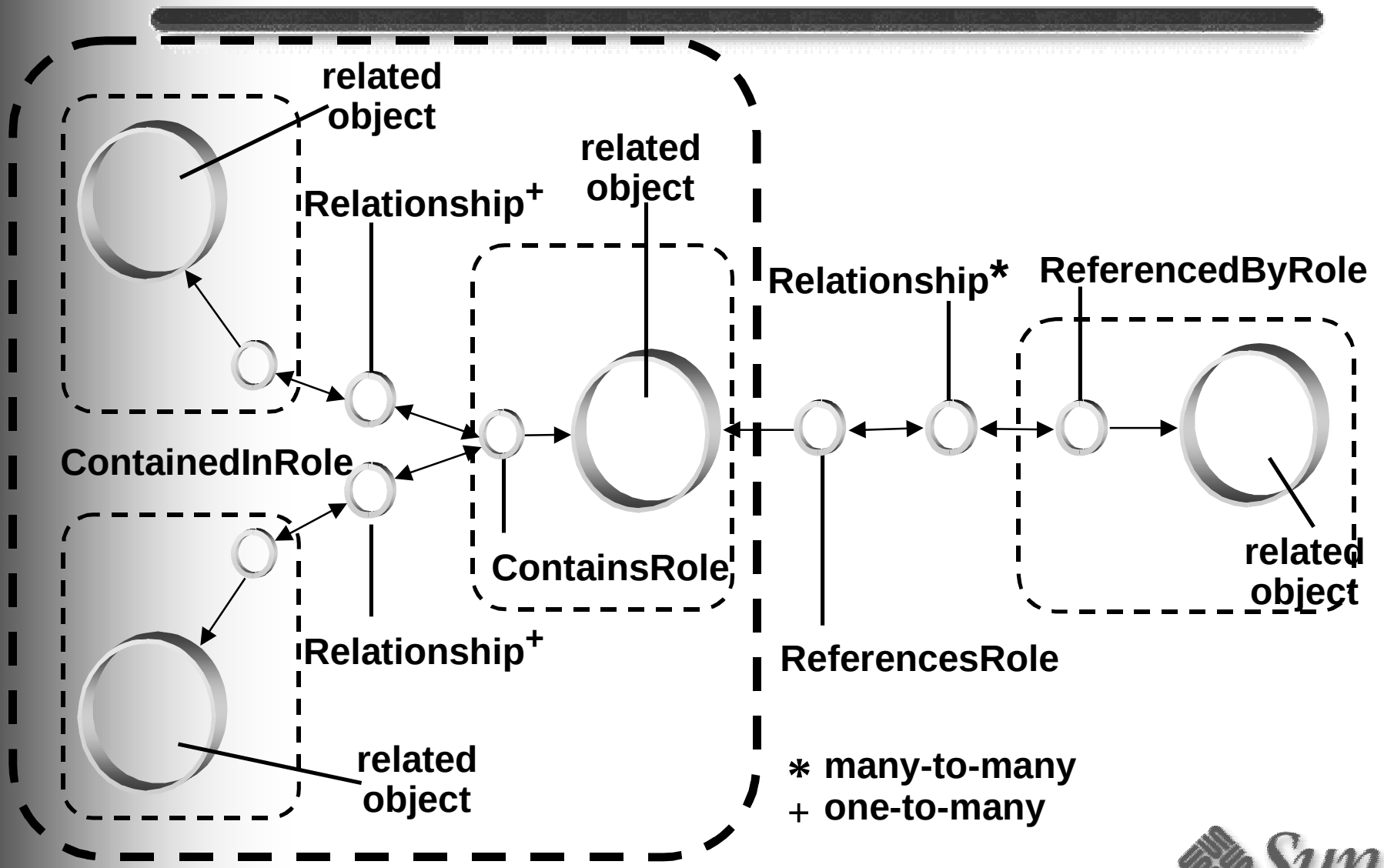
NEO Property Service



NEO Property Service

- Simple, extensible service
- Enables properties to be dynamically associated with any object independent of its static IDL interface attributes
- Does not require the involvement of the associated object
- PropertySet is first-class networked object that maintains a set of key-value pairs

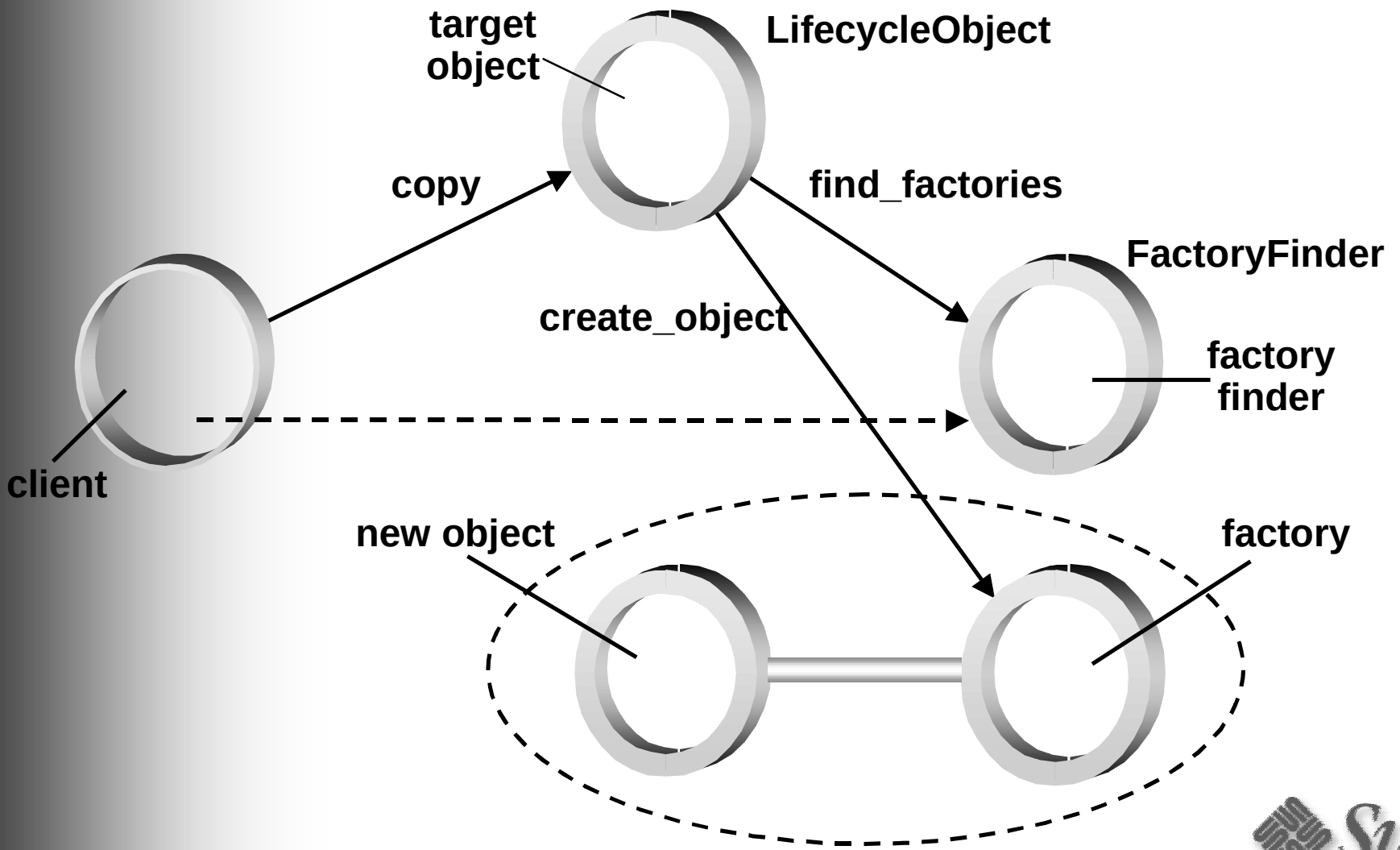
NEO Relationship Service



NEO Relationship Service

- Provides standardized way of linking networked objects in a way that does not require involvement of objects
- Supports one-to-one, one-to-many and many-to-many binary relationships
- Relationships are first class objects themselves
- Two common kinds of relationships are predefined: containment and reference
- Navigation of relationships comparable to use of object references

NEO Lifecycle Service



NEO Lifecycle Service

- **Conventions for creating, deleting, copying and moving objects**
- **Client's model of creation is defined in terms of factory objects**
- **Compound life-cycle operations address copying, moving and deleting objects that are related to other objects**
  - Builds on Relationship Service traversals of graphs of related objects

Joe

Joe

- **CORBA 2.0**
- **IIOP**
- **CORBA compliant ORB**
- **Remote call-back**
  - asynchronous event notification
  - eliminates polling
- **Firewall support**
- **Requires NEO**
- **Older IDL mapping**

Sun microsystems

JavaIDL

Java IDL 1.1

- **100% Pure Java ORB**
- **Full IIOP implementation**
- **CORBA 2.0 IDL to Java mapping**
- **CORBA 2.0 standard COS Naming**
- **Does not use a Interface Respository**
  - Has access to other IR info from other CORBA implementations

JavaIDL or RMI?

- **RMI**
  - Java to Java
  - Private protocol
  - Pass by value
  - Distributed garbage collection
- **JavaIDL**
  - Java client -> CORBA server
  - CORBA client -> Java server
  - IIOP protocol
  - CORBA services

idltojava

- Compiles IDL to Java source
- idltojava -fclient -fserver test.idl

JavaIDL Example

- IDL code that defines a simple interface

```
module Bank {
  interface Teller {
    float getBalance();
    void deposit(in float amount);
    void withdraw(in float amount);
    };
  };
```

- Run idltojava -fserver -fclient bank.idl

JavaIDL Example - The Servant

- tellerServant **is the implementation of the** Teller **IDL interface**
- **The servant is a subclass of** _TellerImplBase
- tellerServant **contains one method for each IDL operation**

Servant Code...

```
class tellerServant extends _TellerImplBase {
  float balance = 0;

  public float getBalance() {
    return(balance);    }

  public void deposit(float amount) {
    balance += amount;    }

  public void withdraw(float amount) {
    balance -= amount;     }
  }
```

JavaIDL Example - The Server

- **Servers** main() **method**
- **Creates ORB instance**
- **Creates servant instance and tells ORB about it**
- **Gets naming context and registers the new object**
- **Waits for invocation of the new object**

Server Code...

```
public class BankServer {
  public static void main(String args[]) {
    try {
      // create and initialize the ORB
      ORB orb = ORB.init(args, null);

      // create servant and register it with the ORB
      tellerServant bankRef = new tellerServant();
      orb.connect(bankRef);

      // get the root naming context
      org.omg.CORBA.Object objRef =
        orb.resolve_initial_references("NameService");
```

Sun microsystems

Server code (cont.)

```
NamingContext ncRef =
  NamingContextHelper.narrow(objRef);

// bind the Object Reference in Naming
NameComponent nc = new
  NameComponent("TheBank", "");
NameComponent path[] = {nc};
ncRef.rebind(path, bankRef);

// wait for invocations from clients
java.lang.Object sync = new java.lang.Object();
synchronized (sync) {
  sync.wait();
  }
}
```

JavaIDL Example - The Client

- **Client** main()
- **Get naming context**
- **Get a tellerRef**
- **Call methods on the object**

Client Code...

```
public class Client {
  public static void main(String args[]) {
    try {
      // create and initialize the ORB
      ORB orb = ORB.init(args, null);

      // get the root naming context
      org.omg.CORBA.Object objRef =
        orb.resolve_initial_references("NameService");
      NamingContext ncRef =
        NamingContextHelper.narrow(objRef);
```

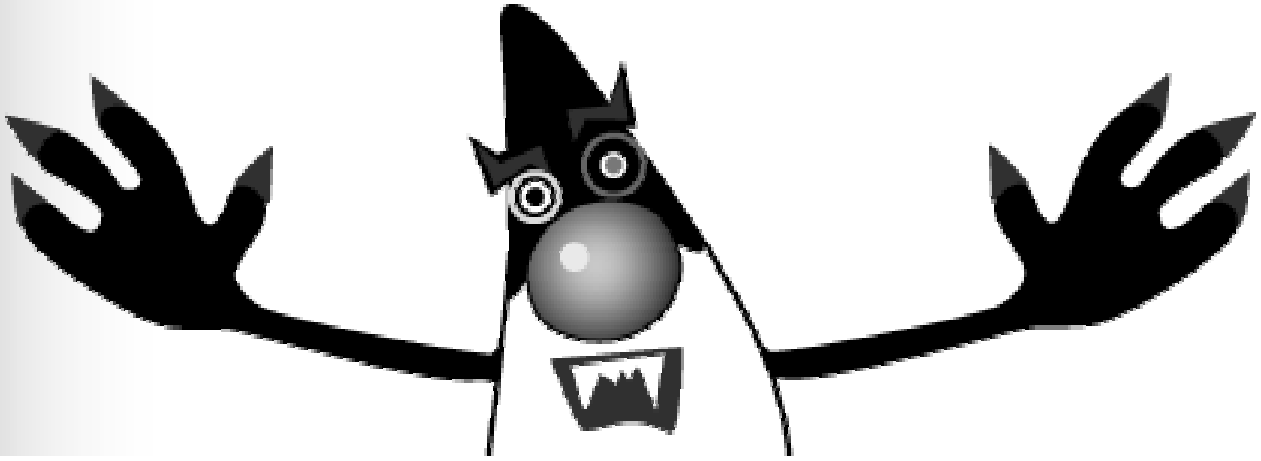
Client Code (cont.)

```
// resolve the Object Reference in Naming
NameComponent nc = new
  NameComponent("TheBank", "");
NameComponent path[] = {nc};
Teller tellerRef =
  TellerHelper.narrow(ncRef.resolve(path));

// call the Bank server object and print results
System.out.println("Balance: " +
  tellerRef.getBalance());
tellerRef.deposit(345.89F);
System.out.println("Balance: " +
  tellerRef.getBalance());
}
```

The Demo

MAY FRIGHTEN YOUNG CHILDREN . . .



Sun microsystems

Thank You!

Questions?

Sun microsystems


Sun
microsystems