Dynamic Management of Any Values

Jeff Parsons
jp4@cs.wustl.edu

October 13, 1999

Distributed Object Computing Group

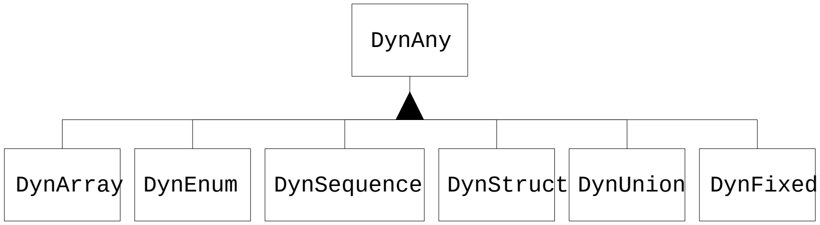
Motivation for Dynamic Any

- **Context:**
  - All data over the wire is in the form of the datatype Any
  - Insertion/extraction of data by overloaded operators $<<=$ and $>>=$
  - Operators for application-defined types generated by IDL compiler
- **Problem:**
  - Requires compile-time knowledge of type - a drawback for:
    - bridges
    - event channels that support filtering
    - browsers
    - debuggers
    - generic user interface tools

How Dynamic Any Addresses the Problem

- Out
  - Composes a value at runtime whose type is unknown
  - Prepares it for sending inside an Any
- In
  - Receives an Any from an invocation - type unknown at runtime
  - Interprets its type
  - Decomposes it into its constituent values

The Dynamic Any Classes



DynAny

DynArray DynEnum DynSequence DynStruct DynUnion DynFixed

Interface DynAny added to CORBA spec (version 2.2) February 1998

Dynamic Any Characteristics

- **Portable**
  - Encapsulates ORB-specific code
  - Developer deals only with functions defined in Interface `DynAny`
- **Composite**
  - Contained elements (if any) must also be DynAnys
  - Decomposition can be recursive

Dynamic Any Characteristics - continued

- **Locality-constrained**
  - Local to address space of creation
  - Cannot be sent over the wire
  - Object reference cannot be stringified
- **Type-static**
  - Type cannot change once created
  - ”Empty” `DynAny` can be created for incremental demarshalling

Examples of Use

- **Decomposition**
  - Get typecode with `DynAny` `type` function
  - Get `TCKind` from typecode
  - Switch statement - select `get` function or iterate/recurse
- **Composition** - get type info from:
  - Interface Repository
  - Translation tables
  - Notification Service