

A Shared View of Sharing: The Treaty of orlando

Lynn Andrea Stein, Henry Lieberman, David
Ungar

Two fundamental mechanisms:

Templates: Create new objects in their own image

Empathy: Allows an object to act as if it were some other object

The Treaty of Orlando

- Intent of object oriented programming
- Sharing of data, code, and definition
- Many object oriented languages have implemented sharing through classes
- Class mechanisms impose a rigid type hierarchy
- Solutions proposed by signatories:

Lieberman: Traditional inheritance replaced with delegation

Ungar and Smith: A single type of parent link replaces the class/subclass/instance protocol

Stein: A rapprochement between delegation and inheritance views

- These approaches share a common underlying view
- Two fundamental mechanisms for sharing: Empathy and Template
- Most significant differences between sharing mechanisms along three independent dimensions: Static-Dynamic, Implicit-Explicit, Per Object-Per Group
- Different programming situations call for different combinations
- As systems follow a natural evolution from dynamic and disorganized to static and more highly optimized, the object representation should also have a natural evolutionary path

Two kinds of sharing:

Anticipated:

Foreseen commonalities between different parts of the system.

Desire to share procedures and data between those similar parts.

Accomplished by mechanisms that provide means for the designer to write down the anticipated sharing structure.

Unanticipated:

New behaviour in an object system that does not already provide for it.

New behavior can be accomplished in part by making use of existing components.

Components are used for both anticipated and unanticipated purposes.

Being forced to state the sharing relationships in advance puts a restriction on the kinds of new behaviour.

Supporting unanticipated sharing is important because software evolution often follows unpredictable paths.

Basic Mechanisms

Empathy:

Object *A* *empathizes* with object *B* if *A* doesn't have a its own protocol for responding to *M*, but instead responds to *M* as though it were borrowing *B*'s response protocol.

Template:

A “cookie cutter” for objects.

Strict Template: An object may not gain or lose attributes once it is defined. *Minimal Template:*

The objects can define other attributes as well

Case studies

Actors and Delegation:

Dynamic, Explicit and Per-Object
Message passing

Self

New objects are created by cloning an existing
one. (prototype)

Nonstrict templates

“Standard Inheritance”

A new class for Dumbo

Hybrid

Behaves more like Self, Dumbo is an extended in-
stance of Elephant

Sharing and software evolution

Two kinds of changes in object systems:
Adding new code and Editing existing code.

OO techniques can often extend behavior by adding new code in cases where conventional techniques would require editing existing code.

Occasions arise when we would like to specialize or extend not just a single object, but an entire hierarchy at once.