

Object Planes

David Chisnall

March 25, 2010

The Road to Object Orientation

Evolving Abstractions for Decomposition

Introducing Object Orientation

The Problem with Object Orientation

Operating Systems

The R[Pleaseinsertintopreamble]le of Operating Systems

Operating System Abstractions

Object Planes

Introducing Object Planes

Design Analogies

Other Use Cases

Implementations

Programming in the '50s

- ▶ Programming is hard.
- ▶ 150 instruction limit per program on STANTEC ZEBRA not a problem—no one can write maintainable programs that big.
- ▶ Primitive flow control.
- ▶ No encapsulation.

Decomposition into Subprograms

- ▶ David Wheeler et al invent *closed subroutine* in 1951.
- ▶ Decompose programs into reusable algorithms.
- ▶ No encapsulation of data

Abstraction of Data

- ▶ Simula 67 introduces *classes* representing *abstract data types*.
- ▶ Decompose data and operations into separate modules.
- ▶ Not a very clean way of representing processes.

Object Orientation

- ▶ Alan Kay asks 'what is the simplest unit of encapsulation simple enough to represent part of a program?'
- ▶ Answer: a computer.
- ▶ Object orientation is defined as *simple computers that communicate by exchanging messages.*

Objects as Simple Computers

- ▶ Objects can do anything that computers can.
- ▶ Store data.
- ▶ Perform operations on data.
- ▶ Run simple models of itself that communicate via message passing?
- ▶ ... oops.

Object Orientation is Not a Recursive Abstraction

- ▶ Computers contain objects.
- ▶ Objects can't contain objects.
- ▶ Should have been obvious with Smalltalk-76: Why is the VM not an object capable of containing other VM objects?
Because the OOP model is not expressive enough.

What is an Operating System?

An operating system is a collection of things that don't fit into a language. There shouldn't be one.

Dan Ingalls, 1981

- ▶ Operating systems hide details of the computer from programmers.
- ▶ Operating systems hide programs from each other.

Processes

- ▶ Simple models of computers.
- ▶ Contain data, perform operations on them.
- ▶ Communicate with other processes via some mediated mechanism (e.g. message passing)
- ▶ Sounds familiar...

Recursive Processes

- ▶ Processes can contain threads.
- ▶ Threads can communicate without mediation.
- ▶ Processes can spawn other processes.
- ▶ Child processes can communicate via the OS, not via the parent.
- ▶ *Exactly the same limitations as objects.*

Side Note: Virtual Machines

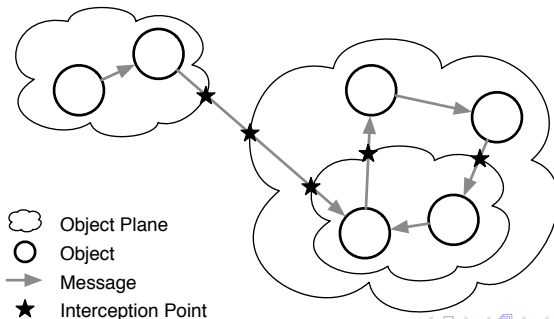
- ▶ Popek and Goldberg described the requirements for recursive virtualization in 1974.
- ▶ Most systems still do not support recursive virtualisation.

Who Cares?

- ▶ Lots of programs are now pretend operating systems.
- ▶ Web browsers run scripts.
- ▶ Office apps run macros.
- ▶ These must be isolated from each other and from the real system.
- ▶ The OS does this for processes, why not subprocesses?

Introducing Object Planes

- ▶ Every object lives in a plane.
- ▶ Messages sent outside the plane are mediated at the plane boundary.
- ▶ Objects are a special case of planes: An object is a plane with no children.



Object Planes and Namespaces

Namespaces:

- ▶ Compile-time attribute of a class.
- ▶ Direct communication for code in the same namespace.
- ▶ Compiler mediates communication outside the namespace.

Object Planes:

- ▶ Run-time attribute of an object.
- ▶ Direct communication for code in the same plane.
- ▶ Runtime library, virtual machine, or operating system mediates communication outside the plane.

Processes are Object Planes

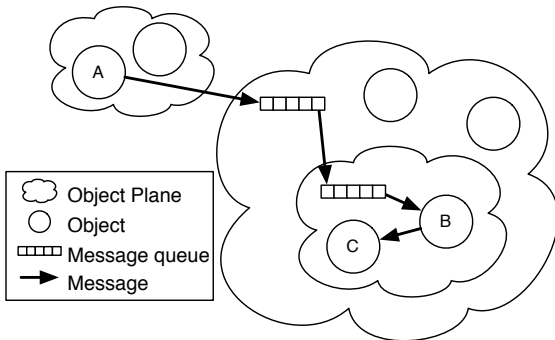
- ▶ Messages sent out of the plane are mediated by the OS.
- ▶ Messages inside the plane are sent directly.

Planes for Persistence

- ▶ Make each document a plane.
- ▶ Record every message sent to the document.
- ▶ Persistent undo history.

Planes for concurrency

- ▶ Make each thread a plane.
- ▶ Enqueue every message sent between planes (optionally return a future / promise).
- ▶ Actor model at arbitrary granularity.



Planes for Security

- ▶ Run untrusted code in a sub-plane.
- ▶ Block all messages out of the plane except those on a whitelist.
- ▶ Simple security is good security!

Implementations

- ▶ Prototype Objective-C implementation in Étoilé (no longer maintained)
- ▶ Objective-C implementation in clang / GNUstep libobjc2 (compiler and runtime support).
- ▶ Smalltalk implementation by INRIA Lille.
- ▶ Java implementation in progress by INRIA Lille.

Acknowledgements

Thanks to Damien Pollet for his help on refining the concept and to his team at INRIA Lille for work on the Smalltalk and Java implementations.