The Apertos Reflective Operating System: The Concept and Its Implementation

Yasuhiko Yokote
Sony Computer Science Laboratory Inc.

Presentation and Discussions

Irene Huang

May 27th, 2004

ECE750 @ University of Waterloo

Apertos Overview: Introduction

- Open-endedness

  Mobile computing requires freedom of mobility in distributed environment. Hence object needs to be open to evolve and adapt itself to its execution environment

- Dependability

  Object is constrained by its execution environment

- Object Oriented technology

  modularization, reusability, maintainability, single unified perspective of the system, etc.

Apertos Overview: Introduction Cont'd

- Object: *a container of information*
- Metaobjct: *defines the semantics of an object's behavior*
- Metahierarchy: *object and its metaobjects are defined in it*
- MetSpace: *a group of metaobjects*
- Object Migration: *an object changes its group of metaobjects*
- Reflector: *a metaobject representing meta-space*
- MetaCore: *terminal metaobject, no metaspace, provides common primitives for object execution*

Apertos Overview: The Framwork

- Object/MetaObject Separation

  Semantics of execution of an object is provided by metaobjects
  Making exposing internals of an object easy

- Metahierarch

  One object belongs to multiple Metaspace. Metaobject of an object is an object of this metaobject's metaspace.
  Discpline for Metaobject programming.

- Object Migration

  One objects travels from one metaspace to another.
  Solving object heterogeneity problem

Apertos Overview: Implementation

- Reflector
  - A mechanism to represent the metaspace
  - Refelctor class hierarchy
    - mCommon
    - canSpeak() checks the compatibility
    - mZeroObject
    - mBaseObject
- MetaCore
  - The terminal metaobject without knowledge of object ID
  - Context structure representing underlying CPU
  - One object has one Context,
  - MetaCore primitives: access Context structure.

Apertos Overview: Evaluation, Related Work

- Evaluation

  Data presented. Future improvement in implementation discussed

- Reflective Systems

  - 3-Lisp, KRS, and CLOS etc: constructed within language framework, limitation on computing resource management
  - ABCL/R2 and AL-1: management of computing resources in a programming language framework.
  - Silica, CLOS-based window system
  - Apertos: first example applying reflective computing in an object-oriented operating system
  - Reflection in general purpose OS is limited

Apertos Overview: Current Status, Future Work and Conclusions

- Current Status
  - Sony PWS-1550
  - AT&T C++ programming language
- Future work
  - Improving programming interface
  - Improving canSpeak()
- Conclusions
  - Frame work of Object-oriented OS
  - The framework contributes to the open mobile computing world
  - The framework is implemented in a small kernel

Apertos: What I Like

- Very clear structuring of the paper
- Clear explanation of the framework model and concepts
- The write raised the problems to solve and then based on these problem, explained who these problems are solved in the Apertos framework and impelentation

Apertos: What I Don’t Like

- How Metahiearachy benefits meta-programming is not very clear to me. I would like to see more detailed discussion on Metahierarchy

- How does the OS actually works is not very clear to me. I would like to see some discussion on the scheduler by using the proposed framework. Does the reflection brings more benefits to the scheduler? How does the PCB, memory block management works in this OS?

Apertos: Question #1

- How object, metaobject, methierarchy and metaspace are related in Apertos frame work?

Apertos: Answer #1

An *object* is a container of information. A *Metaobject* defines the semantics of an object behavior. A group of metaobjects supports an object and this group of metaobjects is called a *metaspace*. A metaobject also belongs to a metaspace, hence introduces the *metahierarchy* concept. The object and its metaspace are represented within their *metahierarchy*.

Apertos: Question #2

- Why Apertos is designed in such way that object/metaobject are separated? Can you give an example to further illustrate?

Apertos: Answer #2

To inspect the internals of an object is needed. Without metaobject, each object needs to implement methods to expose itself. With metaobject/object separation, the internals of an object can be accessed at meta-level.

For example a scheduler needs to access an object's state information.

Apertos: Question #3

- What mechanism does Apertos use to solve object heterogeneity problem? How does this mechanism work for OS services?

Apertos: Answer #3

Object Migration is the mechanism used in Apertos to solve object heterogeneity problem.

A certain OS service is within a metaspace. The object switches to an other OS service by migrating from the current metaspace to the one that provide the new service.

At the coding level, page 418 gives detailed exmaple

Apertos: Question #4

What is the purpose of introducing metahierarchy in Apertos and how it is used?

Apertos: Answer #4

Metahierarchy provides discipline for metaobject programming where objects/metaobjects are separated.

Apertos: Question #5

How Apertos differentiate itself from other object-oriented Operating Systems such as Amoeba, Chorus and Choices et al. ?

Apertos: Answer #5

In those systems, the basic abstraction is defined by the kernel. The kernel defines the interface to the application objects. Whereas for Apertos, the kernel can be defined as objects and it can provide multiple optimal interface to individual objects.

Apertos: Question #6

What is the reflector for in the implementation of Apertos? Explain how it works

Apertos: Answer #6

- A reflector the metaobject used to represent metaspace. The reflector is defined within a class hierarchy called reflector class hierarchy. The top of the hierarchy is $_{m}$Common which provides object migration, canSpeak(), and descriptors.

Apertos: Question #7

In the reflector hierarchy of Apertos, which reflector provides reflector programming facilities? Describe the reflector migration protocols used in it.

Apertos: Answer #7

mZeroObject provides reflector programming facilities.

The migration protocol is in to steps.

1. receive a descriptor from a reflector that creates a new reflector; check compatibility by canSpeak()

2. transfer the contents of a reflector.

Apertos: Question #8

Which reflector provides concurrent object-oriented programming facilities? Describe the reflector migration protocols used in it.

Apertos: Answer #8

mBaseObject provides concurrent object-oriented programming facilities.

The migration protocol is in to steps

1. transfer a descriptor to the target reflector; check compatibility by canSpeak()

2. transfer contents of an object eagerly or lazily

Apertos: Question #9

What is a MetaCore and what is the use of it in Apertos?

Apertos: Answer #9

A terminal metaobject that has no metspace associated with it. It is similar to a micro-kernel in existing systems. It provides primitives to facilitating object/metaobject separation and object migration, execution. It virtualizes the underlying CPU

Apertos: Question #10

What mechanism is used in MetaCore to represent the underlying CPU? How object is related to this mechanism in the MetaCore?

Apertos: Answer #10

Context structure is used in MetaCore to represent the underlying CPU. Each object is associated with a *Context.* A reflector holds the “meta-of” link, *Context* holds the same link as a cache.

The internals of Context structure can only be accessed by the MetaCore.

Apertos: Question #11

Does MetaCore do scheduling?

Apertos: Answer #11

No, the MetaCore is not concerned with Context Scheduling. It maintain its state to avoid multiple execution of the same Context. So the kernel does not handle scheduling policies. How to make a scheduler? This is an open question for discussion.

Apertos: Question #12

In Apertos, is class hierarchy and metahierarchy related? How does Apertos frame work solve the structural heterogeneity and semantic heterogeneity problems?

Apertos: Answer #12

Class hierarchy and metahierarchy are separately defined.

A class defines the structure of an object, hence solving the structural heterogeneity problems such as programming languages and underlying hardware.

A metaclass defines the computation of an object, hence solving the semantic heterogeneity problems such as difference in communication and difference in lifetime.