An Architectural Framework for Semantic Inter-Operability in Distributed Object Systems

Rainer Kossmann

P.O. Box 3511, Station C
Ottawa, Ontario
Canada K1Y 4H7
Email: kossmann@bnr.ca Fax: (613) 763-7066

ABSTRACT. *This research paper reports results of research on semantic inter-operability in real-time, distributed object-oriented systems. It proposes an architectural framework for distributed heterogeneous object system organization. The major goals of the framework are to provide a basis of inter-operability for different kinds of object models and application domains.*

KEYWORDS: OMG, Corba, Object-Oriented, Architecture, Inter-operability.

1. Introduction

This research paper reports results of research on semantic inter-operability in real-time distributed systems[Ivannikov , 1995]. It proposes an architectural framework for distributed heterogeneous object system organization. The major goals of the proposal are to provide a basis of inter-operability for different kinds of object models and application domains.

The proposed approach is based on the following ideas:

- wide use of reflection for semantic extensions and/or transformations;
- a notion of cover - a reflection-based functional object container with possible nesting;
- three-level architectural schema that consists of views, conceptual and implementation levels.

Section 2 of the paper develops the notion of an ObjectUniverse as the set of all objects residing in a huge address space referred to as the ObjectCosmos. ObjectIdentity is represented as a CosmicAddress in this address space. Section 3 develops the notion of Cover as a hierarchically nested set of containers over the ObjectUniverse. Each cover contains a collection of objects, plus meta-objects and associated services that provide the reflective operation for objects in the cover. Section 4 develops the notion of a three schema model for the ObjectUniverse consisting of a ViewSchema, a CosmicSchema and an ImplementationSchema. Section 5 relates this work to notions of Open Repository Systems.

2. ObjectUniverse

The proposed framework is depicted in Figure 1 as a globally interconnected set of objects known as the ObjectUniverse, positioned in a huge address space referred to as the ObjectCosmos. Each object is identified by an immutable non-reusable ObjectIdentity consisting of a unique CosmicAddress within the ObjectCosmos.

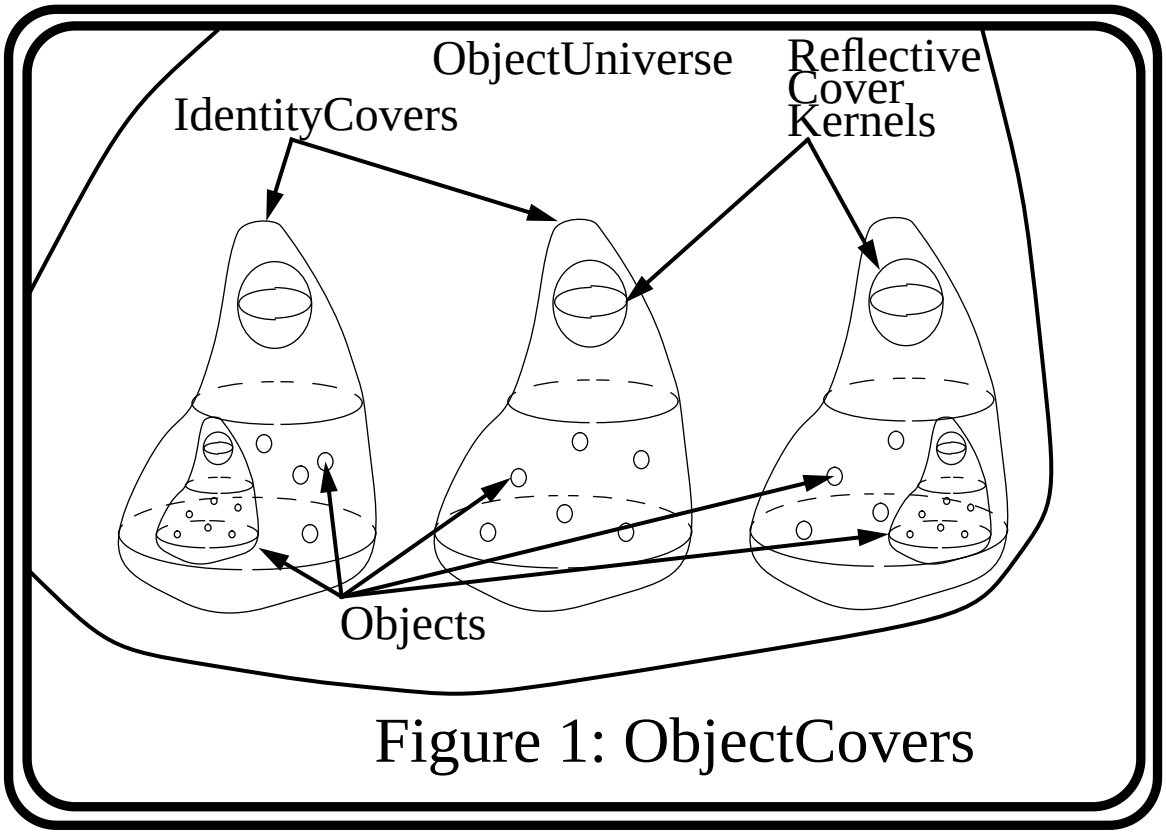

Figure 1: ObjectCovers

Objects are of two major types: as normally understood at present, and as activity causal agents referred to as sparcs. A sparc is an abstraction of the primary causal agent of any activity such as process, task, thread, co-routine, agent, etc. Sparcs are also objects having an ObjectIdentity / CosmicAddress. When a sparc invokes an operation on an object, it drives a thread through that object. Objects which have no thread operating through them are called passive objects, whereas objects which have threads being driven through them by sparcs are called active objects.

3. Covers

3.1 Introduction - Identity Cover

Objects of the ObjectUniverse are covered by semantic covers. A semantic cover can be visualized as a container for a collection of objects / meta-objects that fully and completely describes the contained objects. A key point to note is that every cover is effectively a meta-system, because it describes and controls the objects it covers. Covers are also objects.

An IdentityCover is a specific type of compositional cover that partitions the ObjectUniverse into collections of objects. In its simplest form, the IdentityCover simply lists the CosmicAddress of the objects it contains. IdentityCover are self-descriptive, i.e. they identify themselves and their content. They may contain a cover kernel which is responsible for managing the operation on objects within the IdentityCover.

Covers may be hierarchically nested as shown in Figure 1, which depicts an IdentityCover over the ObjectCosmos. The outermost cover is referred to as the ObjectUniverseCover. We utilize the notation [Cover[*]] to represent a cover in a textual description. The [*] notation is suggestive of the hierarchical nesting of covers and is capable of unambiguously identifying a specific cover

instance, as in for example [Identity[BNR[Ottawa]]], which is the IdentityCover listing the identities of BNR-Ottawa objects, including the Identity of [Identity[BNR[Ottawa]]] itself.

Each IdentityCover may contain within it one or more additional orthogonal semantic covers describing further semantics properties of objects contained within the IdentityCover. In this paper we limit our examination to the additional covers [Class[*]], [Activity[*]], [Consistency[*]] and [Security[*]] described next. Other covers may also exist, for example, we note that OMG's IDL Module construct[OMG, 1992] is capable of specifying an [Interface[*]] cover over the ObjectCosmos. Another example would be a [Version[*]] cover specifying object versioning for the ObjectUniverse.

The cover kernel depicted in Figure 1 is a flexible mechanism which permits the insertion of further semantic covers into the IdentityCover. The cover kernel in an IdentityCover has an IdentityCover-independent component, as well as an IdentityCover-dependent component which varies for each IdentityCover. The IdentityCover-independent component provides the mechanism to slot in additional covers into the IdentityCover kernel.

3.2 [Class[*]] / [Being[*]]

[Class[*]] is a semantic cover which defines the complete set of classes covering the ObjectCosmos in terms of their behaviour and attributes. Since [Class[*]] is a cover, it is of course also a meta-system.

We refer to [Identity[*]] and [Class[*]] together as [Being[*]]. [Being[*]] completely specifies the existential semantics for the ObjectUniverse. These existential semantics are in a sense analogous to the notion of real objects in the ObjectUniverse and their spatial properties: where they are, what they are made of and what can be done on or with them.

3.3 Activity[*]] / [Doing[*]]

Each sparc operative as a causal agent in the ObjectCosmos is coupled to it's own [Activity[*]] meta-object that describes the sparc's current state and sphere of influence in the ObjectCosmos. As the sparc's activity proceeds, its associated [Activity[*]] meta-object expands and contracts to encompass the objects on which it is operating. [Activity[*]] therefore completely specifies the operational semantics for the ObjectUniverse. These are in a sense analogous to causal change agents and their effects and progress in time in the real cosmos. We also refer to [Activity[*]] by the synonym [Doing[*]].

Sparcs are [Activity[*]] abstraction primitives necessary for specifying [Consistency[*]] and [Security[*]] semantics as further described below.

3.3 [Consistency[*]]

[Consistency[*]] deals with consistency semantics in the ObjectCosmos under conditions of sparcs with intersecting / colliding [Activity[*]] covers. Two [Activity[*]] meta-objects collide if they both expand to include one or more identical objects at the same time. The function of [Consistency[*]] is to ensure that the ObjectUniverse remains in a consistent state once [Activity[*]] collision ceases. The notion of transaction is an example of [Consistency[*]].

3.4 [Security[*]]

[Security[*]] semantics specify the security semantics operative in the ObjectCosmos, by specifying the ‘keys’ held by sparcs that permit them to open the appropriate ‘locks’ covering the objects they wish to access.

4. ObjectSchema

The framework deals with multiple schema analogous to the ANSI / SPARC 3-schema model for data[ANSI, 1975]. These schema are the CosmicSchema, the ImplementationSchema and the ViewSchema. These correspond approximately to the ANSI / SPARC conceptual schema, internal schema and external sub-schema.

4.1 CosmicSchema

The CosmicSchema is a description of the ObjectCosmos in terms of at least its orthogonal covers of [Being[*]], [Doing[*]], [Consistency[*]] and [Security[*]]. Figure 1 is a CosmicSchema showing only the [Identity[*]] sub-cover of [Being[*]].

4.2 ImplementationSchema

Figure 2 depicts the mapping of a CosmicSchema into an ImplementationSchema. At the ImplementationSchema, there are a number of activity engines (AE) which can be loaded with sparcs in order to perform the sparc’s activity. Activity engines are generally limited to being able to perform only a few types of activity covers from the total set of activity cover types in the ObjectCosmos (e.g. UNIX SUN/OS process, active Microsoft Window, agent...).

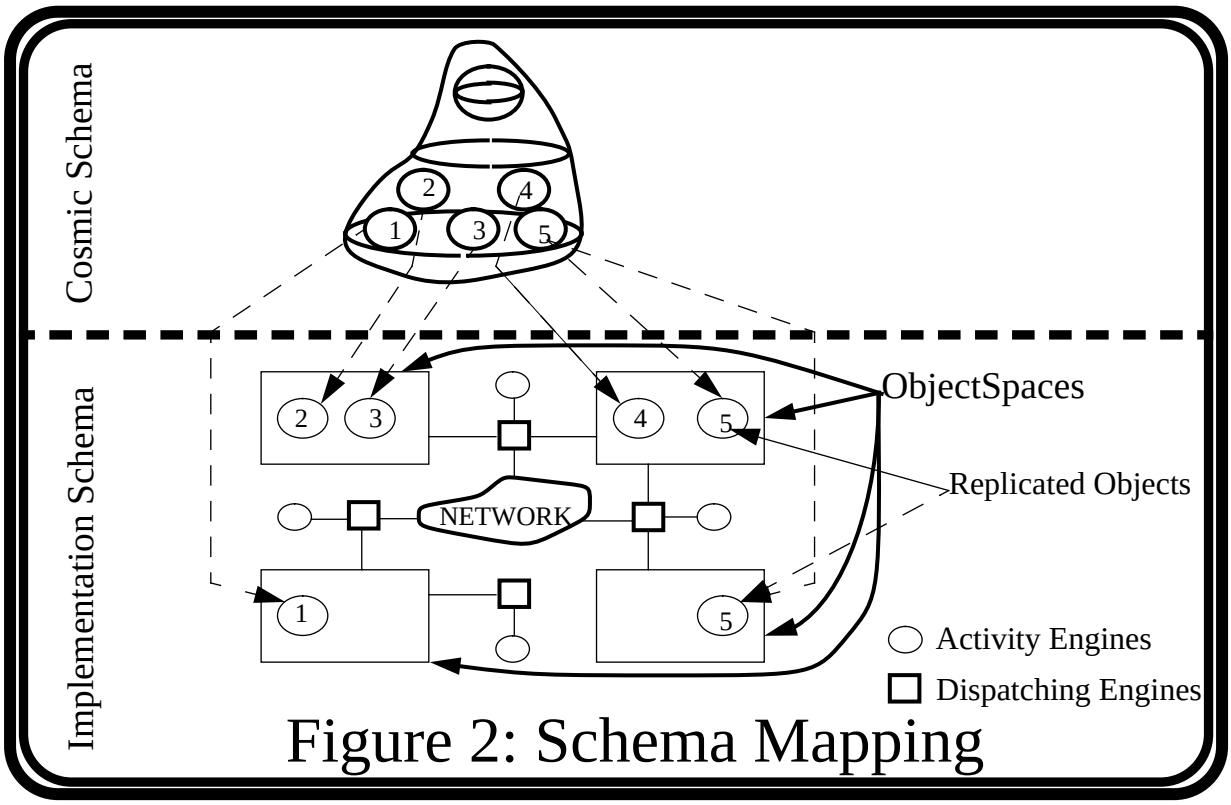

Figure 2: Schema Mapping

Objects of the ImplementationSchema are contained in memory abstractions referred to as ObjectSpaces in Figure 2. Objects have a unique ImplementationAddress within an ObjectSpace.

DispatchingEngines perform dispatch of operations on objects originating from an activity running on an ActivityEngine. DispatchEngines map the CosmicAddress of a target object into an ImplementationAddress for that object the local ObjectSpace and performs the dispatching operation. DispatchEngines therefore very similar to an OMG ORB. If the object is not local to a DispatchEngine (i.e. it is not contained in a local ObjectSpace), then the dispatch will interact with other DispatchEngines supporting the ObjectCosmos to properly dispatch the operation.

DispatchEngines have knowledge of the semantic covers over each of its ObjectSpaces and perform the appropriate semantic transformations required to support semantic inter-operability between two objects existing under their respective covers, where such semantic inter-operability is indeed possible.

Objects may also be replicated, relocated or moved within the above ImplementationSchema Platform. Such mechanisms are managed by DispatchEngines. Failures and restarts are performed by DispatchEngines or with their assistance, and generally involve relocating objects to avoid failed components ActivityEngine , DispatchEngine and ObjectSpace.

In summary, the ImplementationSchema identifies the distribution of the CosmicSchema over the ImplementationSchema Platform. I.e. it identifies the distribution of the objects of the CosmicSchema over the ActivityEngines, DispatchEngines and ObjectSpaces of the ImplementationSchema platform.

4.3 ViewSchema

A ViewSchema resides above the CosmicSchema and provides different perspectives and views of the CosmicSchema in accordance with the requirements and privileges of a viewer.

5. Enterprise Repository

The notion of hierarchically nested IdentityCovers and their distribution over ObjectSpaces is technology. At some level of [Identity[*]] below the ObjectUniverseCover of Figure 1, we apply IdentityCover technology to a specific form of human endeavour by defining the notion of EnterpriseCover. EnterpriseCovers are IdentityCovers with the additional anthropic role of being related to society and its needs[Kossmann, 1994]. We also refer to EnterpriseCover as an EnterpriseRepository. EnterpriseRepositories correspond to enterprises recognized as legal independent entities in international law such as persons, organizations and nations. The notion of a legally recognized enterprise is important because it is associated with security and authentication mechanisms that are required to be able to fully specify security semantics for the ObjectCosmos. Such mechanisms rely on SecurityCovers which map signing authorities (people) of an enterprise to sparcs (corresponding locus of action representing people) that perform activities on people's behalf. Such mappings are of course part of [Security[*]].

REFERENCES

[ANSI, 1975] "ANSI/X3/SPARC Study Group on Data Base Management Systems, 'Interim Report,'", FDT (ACM SIGMOD bulletin), Vol. 7, No. 2, 1975.

[Ivannikov et. al, 1995] "An Architectural Framework for Semantic Inter-Operability in Real-Time Distributed Object Systems", BNR Internal Report, V. Ivannikov, R. Kossmann, V. Kamensky, E. Kusikov, N. Kuzjurin, T. Kouzminov, S. Kuznetsov, B.Novikov, Y.Pogudin, L. Shabanov, I.Shevlyakov, V. Shnitman, S.Chernonozhkin, September 1995.

[Kossmann, 1994] “Repositories”, ANSI X3H4 Open Repository Systems, document 94-148, R. Kossmann, July 1994.

[OMG 92.11.1, 1992] “OMG Object Management Architecture Guide (OMA Guide), Revision 2.0”, Object Managment Group, September 1, 1992, OMG TC Document 92.11.1.