

## CORBA and RM-ODP: parallel or divergent?

This article has been downloaded from IOPscience. Please scroll down to see the full text article.

1999 Distrib. Syst. Engng. 6 82

(<http://iopscience.iop.org/0967-1846/6/2/303>)

View [the table of contents for this issue](#), or go to the [journal homepage](#) for more

Download details:

IP Address: 38.107.179.213

The article was downloaded on 20/02/2012 at 08:02

Please note that [terms and conditions apply](#).

# CORBA and RM-ODP: parallel or divergent?

N Dunlop<sup>†</sup>, J Indulska<sup>†</sup> and K A Raymond<sup>‡</sup>

<sup>†</sup> Department of Computer Science and Electrical Engineering, The University of Queensland, Australia

<sup>‡</sup> CRC for Distributed Systems Technology, Australia

E-mail: dunlop@csee.uq.edu.au, jaga@csee.uq.edu.au and kerry@dstc.edu.au

Received 16 February 1999

**Abstract.** Modern architectures for distributed object environments (or distributed ‘middleware’) are revealing an increasing trend towards standardization. The recent emergence of a standard for open distributed processing, the ISO/IEC Reference Model for Open Distributed Processing (RM-ODP) (ITU-T Recommendation X.901) and the coincidence of the development of the Object Management Group’s Common Object Request Broker Architecture (CORBA), has prompted us to explore the relationship between these architectures. This paper analyses the CORBA architecture as a support environment for open distributed processing by comparing the business requirements for ODP, RM-ODP viewpoints, functions and distribution transparencies as specified in RM-ODP (ITU-T Recommendations X.901–4) with the CORBA architecture. Through this examination it is evident that despite distinctly divergent terminology, there exist significant parallels between CORBA and RM-ODP.

## 1. Introduction

The Reference Model for Open Distributed Processing (RM-ODP) [3–6] is a meta-standard that describes an architectural framework for building open distributed systems. Its goal is to describe an architecture for present and future distributed systems without prescribing an implementation.

The RM-ODP (X.901–4/ISO 10746) consists of four parts:

- Part 1—‘Overview’ [3]. The overview states the motivation and objectives of ODP systems. It includes an overview of the ODP architecture explaining the key components in the architecture, ODP functions and distribution transparency. It introduces the architecture to standards writers and systems designers of ODP systems, explains how it can be applied to future systems, and how to use conformance assessment for existing systems.
- Part 2—‘Foundations’ [4]. The foundation specification contains the definition of the concepts and analytical framework for the description of ODP systems, e.g. object, interface, object state and communication. The foundations specification also introduces the principles of conformance to ODP standards and the way in which they are applied.
- Part 3—‘Architecture’ [5]. The architecture contains a specification of the required characteristics that qualify a distributed processing system as open. ODP systems are defined from five viewpoints, these being the enterprise, information, computational, engineering and

technology. It uses the descriptive techniques from the foundations of RM-ODP [4] to describe the constraints to which ODP systems must conform.

- Part 4—‘Architectural Semantics’ [6]. Architectural semantics is a formalization of the ODP modelling concepts defined in clauses 8 and 9 of the foundations of RM-ODP [4], e.g. object, interface, object state and communication. The formalization is achieved by interpreting each concept in terms of the constructs of the various standardized formal description techniques (e.g. LOTOS, Z and ESTELLE).

The Common Object Request Broker Architecture (CORBA) [11] is a standard for distributed object computing produced by the Object Management Group (OMG). CORBA provides an architectural platform for the support of open, distributed object-oriented applications. The primary motivation of CORBA is to provide heterogeneous, distribution-transparent access to objects through the provision of standardized interfaces. CORBA does not prescribe a technology, it is a specification only, not an implementation. CORBA is also an evolving standard whose concepts and functions have not yet been fully defined.

RM-ODP was developed largely by the research community in response to the need for a standard architecture to support ODP. In many respects the CORBA architecture has been developing in parallel to RM-ODP. It is noted that many of the researchers involved in contributing to the development of the RM-ODP standard have also had significant impact on the development of the CORBA standard.

Despite the distinctly divergent terminology used to express the architecture concepts modelled in CORBA and RM-ODP, the two architectures are significantly comparable. Unfortunately, the use of divergent terminology has led to a number of misconceptions within the IT industry regarding comparison between RM-ODP and CORBA. This paper seeks to address this problem by providing a tutorial style comparison of the current CORBA specification with the RM-ODP standard.

The paper is organized as follows. Section 2 analyses to what extent CORBA compares with the user or business requirements of ODP as specified in the RM-ODP. CORBA's correspondence to the RM-ODP viewpoints and functions is analysed in sections 3 and 4, respectively, followed by the lessons learned during the examination in section 5 to conclude the paper.

## 2. Business requirements for ODP

Clause 6 of the RM-ODP [3] identifies a number of business or organizational requirements of ODP systems, these being: openness, integration, flexibility, modularity, federation, manageability, quality of service, security and transparency.

RM-ODP defines *openness* as the property enabling both *portability* (i.e. the ability to execute different components on different processing nodes without modification) and *interworking* (i.e. the ability to support meaningful interactions between components, possibly residing in different systems) [3]. CORBA provides some support for portability through the definition of a standardized language for interface definition, that is the Object Management Group Interface Definition Language (OMG IDL) [11] and specification of associated language mappings, but there is more to portability (i.e. operating system dependencies). Interworking in CORBA is supported by the Internet Inter-ORB Protocol (IIOP) [12]. CORBA may interwork with another non-CORBA compliant distributed environment (e.g. CORBA/COM [8, 9] and CORBA/DCE bridges [17]), however, there are some limitations with respect to the object model, type model and communication paradigms supported.

The RM-ODP specifies that *integration* is the property of incorporating various systems and resources into a whole without costly *ad hoc* development [3]. It covers, for example, integration of systems with different architectures and different resources of different performance. Through integration of heterogeneous systems and resources, an object should be able to invoke the service (or use a resource) provided on a different system or node. CORBA allows for the integration of heterogeneous platforms (machine type and operating systems) through abstract IDL and standardized RPC [11].

RM-ODP specifies that *flexibility* is the property of supporting a systems evolution including the existence and continued operation of legacy systems [3]. A very common approach towards incorporating legacy applications and data into the CORBA architecture is to define an OMG IDL specification for the legacy operations and to implement the connection between the new IDL interface (or object wrapper) and the legacy application. Currently, there exist

mechanisms for automating object wrapper development, for example, Java to IDL mapping [20, 21]†.

RM-ODP [3] also specifies that an ODP system should be capable of supporting run-time changes, that is, an ODP system should be capable of being dynamically reconfigured to accommodate changing circumstances. CORBA allows objects to dynamically discover services at run time using the Object Trader Service [11, 26]. Hence, it is possible to add services to the CORBA environment without the need to recompile applications. CORBA also enables clients to rebind to new implementation instances using the object reference and Implementation Repository [11].

*Modularity*, as specified by the RM-ODP [3], is the property of an ODP system that ensures that parts of the system are autonomous but interrelated. The object component architecture of the CORBA specification ensures a high degree of modularity, where autonomous application objects are able to communicate through the use of a standardized interface definition language, OMG IDL [11].

RM-ODP states that *federation* is the property of combining systems from different administrative or technical domains to achieve a single objective [3]. Federation allows completely independent services to cooperate together in order to provide an enhanced service. Federation can be supported through the provision of standardized interfaces that allow a service to share its functionality with other objects in the federation. Through CORBA's provision of standardized interfaces, defined using the OMG IDL, a federation can be built, but there are currently limited mechanisms to support a federation; an exception would be the OMG Object Trader Service [26].

*Manageability* is the property of monitoring, controlling and managing systems resources in order to support configuration, quality of service and accounting policies. RM-ODP specifies management interfaces at object, capsule and cluster levels. The OMG have adopted a System Management architecture [32], which has been based largely on X/Open's‡ System Management architecture [39]. Even though a Systems Management architecture has been adopted, it is recognized that it is not widely used in the management of entities within a production environment.

Provision of *quality of service* is a property of an ODP system related to meeting a set of quality requirements on the systems behaviour. Quality requirements of concern are as follows:

- Provision of timeliness. User requests should be responded to within an appropriate time interval. The CORBA Messaging Service [16] provides for the specification of allowable time windows within which a request must be delivered but does not allow for the specification of the allowable time windows for completion of an operation, which is the subject of the emerging real-time processing RFP [29].
- Availability of system services. Providing availability in an ODP system may mean building redundancy into the system or providing mechanisms for migration where the processing load gets too high. Availability can also

† Note that this is the reverse process of mapping IDL to Java.

‡ Subsequently, X/Open and the OSF have merged to form the Open Group.

be improved by ensuring the architecture design does not require the simultaneous functioning of a large number of critical components. The CORBA architecture does not contain a large number of critical components and allows for the duplication of service implementations across a number of hosts, thereby allowing redundancy to be built into the system. The Object Lifecycle Service [25] provides standardized interfaces for the migration of objects. CORBA does not, however, provide explicit services to support availability.

- Fault tolerance is the level of failure handling provided by the system and the level of autonomy of its objects. The CORBA architecture specifies a number of system exceptions for ORB functions [11], thereby ensuring that failure of one participant in an interaction will not necessarily mandate failure for all participating components. The OMG have recently issued an RFP to address the issue of fault tolerance [18].
- Reliability will be considered to be the level of trust a user has in the systems behaviour. This is influenced by availability, fault tolerance, security and atomicity, which are not fully supported by CORBA. Communication reliability (reliable RPC) is not provided by the CORBA architecture [11]. Consistency of transactions is supported by the Object Transaction Service (OTS).

*Security* is the property of ensuring that system facilities and data are protected against unauthorized access. The adopted CORBA Security Service Specification [31] has defined standardized interfaces for access control, security auditing, authentication, integrity, non-repudiation and key management, see section 4.4.

*Transparency* is the property of masking from applications, the details and the differences in mechanisms used to overcome the problems of distribution. CORBA supports access, failure, migration, location, relocation, persistence and transaction transparency to differing degrees. This issue will be investigated in greater detail in section 4.

### 3. Comparison based on RM-ODP viewpoints

In terms of the RM-ODP viewpoints, this section compares the CORBA architecture with that of the RM-ODP [5]. We identify the relationship between each significant RM-ODP concept and the corresponding concepts in the CORBA architecture [11].

In RM-ODP, the specification of a distributed system is separated into a number of different concerns. This separation of concerns is established by the identification of five viewpoints, each with an associated language. There are five viewpoints in RM-ODP; these are: enterprise, information, computational, engineering and technology.

In order to represent an ODP system from a particular viewpoint, it is necessary to define a structured set of concepts and rules in terms of which that specification can be expressed. This set of concepts and rules provides the *language* for writing the specification of the system from that viewpoint. In this section we discuss each of the viewpoints and associated languages as they relate to the CORBA architecture.

### 3.1. Enterprise viewpoint

The enterprise viewpoint provides the definitions of concepts and rules to enable the specification of an ODP system from a business perspective. An enterprise specification defines the purpose, scope and policies of the ODP system. The viewpoint contains a number of concepts and rules that can be used to model the enterprise, its business direction, and the policies that will be embodied by the system. These include enterprise objects, object roles and policies governing interactions between enterprise objects and their configuration.

A growing number of organizations are using business object modelling to describe, evaluate and re-engineer their business processes. It was envisaged that a means to model the enterprise through the CORBA architecture was to be provided by the Common Business Objects architecture [10]. Common business objects are a set of predefined enterprise-level objects representing those business semantics that can be shown to be common across most businesses. Similar to the RM-ODP's notion of an enterprise object, CORBA's common business objects were to capture information about a real world or business concept, the operations on that concept, the constraints on those operations and the relationships between business concepts.

Remarkably, the determination as to what will *actually* constitute common business objects in the CORBA environment still remains to be resolved after the Business Object Component architecture (BOCA) [7] was rejected at its ballot for adoption, with the exception of the Task and Session CBOs [33] and Workflow Management [38]. It should be noted, however, that four new RFPs have now been drafted in its place; these being the UML Profile for Enterprise Distributed Computing [36], UML Profile for CORBA [35], Human-Usable Textural Notation for UML Profiles [19] and the CORBA Mapping for Business Object initiative UML Profiles [15].

### 3.2. Information viewpoint

The information specification of an ODP system is a model of the information that it retains and of the information processing that it carries out. The information viewpoint defines what information should be stored and its structure, where the data will be stored and relationships between data items and associated constraints. In RM-ODP, data, relationships and constraints are expressed through the *static schemata* (data that will remain constant), *dynamic schemata* (data that can be modified) and *invariant schemata* (upon which the specializations of static and dynamic are based). Within the CORBA architecture, the Unified Modelling Language (UML) [37] specifies constructs suitable for describing static dynamic and invariant schemata. Static schemata can be modelled using the constructs described in the UML *static structure diagrams* (e.g. objects, attributes and associations). Dynamic schemata can be described using the constructs defined in the UML *statechart diagrams* that provide a means to specify object states and the allowable transitions between object states. Invariant schemata can be described using the constraint mechanisms of the UML class model.

### 3.3. Computational viewpoint

The computational viewpoint is intended to depict the ODP system from a distribution-transparent perspective. This abstraction of an ODP system is intended to support the interests of application designers and developers, whose primary concern is developing application programs to solve business problems. The complexity of distribution and details of the infrastructure used to support the distribution should be abstracted. CORBA provides varying degrees of support for the concepts expressed in RM-ODP's computational viewpoint; they are characterized below.

**3.3.1. Object model.** Though CORBA supports RM-ODP's notion of a computational object type (expressed in OMG IDL), an important characteristic of the concept of an object in RM-ODP is that an object can support *multiple* interface types and provide multiple instances of the same interface type [11]. The benefit of being able to model *multiple* interfaces on a *single* object is that each interface can represent a different view or a projection of the set of services offered by the object. That is, an object may provide an interface for 'normal' operations and provide a separate interface for the management operations. In this way, clients can be offered functionally distinct views of the object implementation.

In contrast, it is currently the case that CORBA objects outwardly expose a *single* interface type, which may be derived from multiple interface types through interface inheritance. Note that this is *not* the same as the RM-ODP notion of offering multiple interfaces. In addition, interface inheritance is the only form of subtyping in CORBA, which is more limited than that supported by the RM-ODP.

Interestingly, the OMG, now recognizing the need for support of multiple interfaces, have issued an RFP CORBA Components [14] in relation to the redefinition of the object model to support the notion of a 'component'. A component is a collection of objects and can be considered to be an extension and specialization of the object type. The component will be a completely new concept in the CORBA architecture. Component types will be specified in IDL and represented in the Interface Repository. Components will be able to provide multiple object references which are capable of supporting distinct (i.e. unrelated by inheritance) CORBA interfaces. The component will also have a single distinguished reference whose interface will act as a container for the other provided interfaces. This distinguished reference and interface will be referred to as the *component reference* and *component interface*, respectively. The other interfaces provided by the component will be referred to as *provided interfaces*.

**3.3.2. Binding model.** Interactions between computational interfaces are only possible if a binding has been established between them. A binding couples two or more object interfaces for the purpose of communication. It results from a *binding action* or set of *binding actions* of which RM-ODP describes two models. These are implicit binding *operations* and explicit binding *actions*.

Implicit binding operations occur when a client requests a server operation interface to which it is not currently bound. Binding in this case is done silently on the initial use of the interface reference or upon being passed the interface reference.

An explicit binding is accomplished through a sequence of binding actions executed prior to the first use of that binding. There are two different kinds of explicit binding action; these are *primitive* and *compound*. Primitive binding actions enable the binding of *two* computational interfaces, provided they are of the same class (i.e. signal, stream or operational) and are of complementary causality (i.e. producer and consumer, client and server) and their signature types are complementary. Compound binding actions enable a *set* of interfaces to be bound together, using a binding object to support the binding. Through the process of *compound* binding, a series of *primitive* binding actions will be used to establish the connection between communicating interfaces. Compound bindings can be used to support multi-party interactions.

CORBA supports an *implicit binding* model and does not provide support for *explicit binding* actions (primitive or compound).

**3.3.3. Invocation semantics.** In RM-ODP, it is possible to specify three different types of interface, each supporting a different style of object invocation. These are *operational*, *stream* and *signal* interfaces.

Operational interfaces support the established client/server model of interaction where client objects invoke operations at the interface to server objects, being the standard remote procedure call model. RM-ODP supports two styles of operational interface, these being *interrogations* and *announcements*. An interrogation is an operation where the caller blocks awaiting the return of the results. An announcement is an operation that has no result and the caller will not block awaiting the return of results.

In CORBA, the interrogation operation is supported by a synchronous remote procedure call where the client program, or thread, blocks when a remote invocation is made, waiting until the results arrive. The synchronous remote procedure call is the dominant form of communication in CORBA applications. An announcement is supported in CORBA by the *oneway* variant of the synchronous remote procedure call. In addition, CORBA supports what is termed '*deferred synchronous*' invocations which are non-blocking, remote procedure calls, but require the client thread to subsequently poll to see if results are available. The OMG is currently working on a standard for asynchronous messaging [16] that will potentially provide two asynchronous invocation models being callback and polling.

The streams interface in RM-ODP is intended to support continuous flows of data upon which quality-of-service characteristics apply. Typically, a consumer object will connect to the stream interface of a producer object or vice versa, several streams can be grouped in a single interface (e.g. an audio stream and a video stream). CORBA does not provide direct support for stream interfaces, though the OMG have now adopted a specification for the management

**Table 1.** Computational activities.

ODP activity	Explanation	Supported in CORBA
Initiate signals	Initiate or respond to signals	No
Produce flows	Produce or consume flows	No
Initiate operations	Initiate or respond to announcements and interrogations	Yes
Bind interfaces	Explicit binding operation	Implicit binding only
State access	Access and modify object state	Only through the functions of the implementing language
Instantiation	Instantiate object and interface templates	Enables both through the ORB services
Delete interfaces	Delete one or more interfaces	Provides an interface for deletion—does not provide its implementation
Spawn, fork, join	Spawn, fork and join activities	CORBA applications have to use services of the underlying operating system
Trading	Trading for an interface	Yes, (using <i>resolve_initial_references()</i> ), CORBA trader [26] conforms to RM-ODP
Is_a_subtype	Test if a computational signature is a subtype of another	The ORB provides this operation—CORBA subtyping is based on inheritance
Failure detection	Infrastructure failure (e.g. communication or security failure)	Provides system exceptions to ensure failure is not critical for all participants

and control of logically continuous flows of data between objects [13].

Underpinning both the operational and stream interface types is the isochronous signal interface. The signal interface is a low-level one-way communication action. It is intended to provide the basis upon which to build more complex communication primitives (i.e. the operational and stream interface types). It is not expected that an application itself would invoke a signal interface directly. Correspondingly, CORBA applications have access to isochronous low-level one-way communication actions through the sockets interface of the supporting platform.

**3.3.4. Computational activity.** In RM-ODP, computational objects can perform the following activities. Table 1 displays those computational activities that the CORBA architecture can support.

### 3.4. Engineering viewpoint

The engineering viewpoint is primarily concerned with providing the mechanisms and functions for supporting distribution-transparent interaction between objects in an ODP system. Similar to the computational viewpoint, the engineering viewpoint is an abstraction of an ODP system, deemed useful for systems designers and programmers. The engineering viewpoint is concerned with providing the infrastructure to support distribution transparencies to the application designers and programmers. CORBA has varying degrees of support for the concepts expressed in RM-ODP's engineering viewpoint.

**3.4.1. Engineering concepts.** RM-ODP defines a number of architecture concepts in the engineering viewpoint, which exist to support distribution transparency for application objects. These engineering concepts are described below and how they relate to the CORBA architecture.

- Basic Engineering Object (BEO). BEOs exist in the computational viewpoint as 'computational objects' that

interact through computational bindings. CORBA supports the notion of the BEO through executables that are comprised of a set of application and supporting infrastructure objects.

- Cluster. A cluster is the smallest grouping of objects within a capsule. It can be considered to correspond to the notion of linking modules to form an executable program image. Interaction between objects in a cluster will occur using either a local binding (for objects communicating in the same cluster) or using a distributed binding supported by a channel (for objects communicating in different clusters). In the future, CORBA components [14] will closely correspond to the concept of a cluster.
- Cluster Manager. When a system is unable to provide processing, storage or communication functions continuously, there is a need to support functions that enable objects (or groups of objects) to be checkpointed, recovered, migrated and deactivated. The Cluster Manager exists to provide these functions. It is expected that the CORBA component interface will be the logical place to implement Cluster Manager services.
- Capsule. A capsule owns storage and a share of the node's processing resources. A capsule is a unit of protection and is generally the smallest unit of independent failure supported by the operating system. The grouping of clusters into capsules is done to reduce the cost of object interaction. In CORBA an equivalent notion to the capsule would be a container [14]. A component implementation will execute in a container. Containers will provide Portable Object Adaptors (POAs) that the component implementations use to create object references and activate server instances.
- Node. At the outer level, objects are physically located and associated with processing resources by grouping them into nodes. Nodes can be thought of as representing independently managed computing systems. A node can be anything that has a strongly integrated view of resources, as long as the systems designer can consider

it as a whole. In relation to the CORBA architecture, an equivalent concept would be the host.

- **Nucleus.** The nucleus provides basic processing, storage and communication functions. The nodes are under control of the nucleus, which is responsible for the initialization, and supporting communication and storage facilities. In the CORBA architecture, both the ORB and the local operating system provide the functions of the nucleus.
- **Channel.** A channel is the infrastructure that allows communication between BEOs. Channels are composed of stubs (for argument marshalling), binders (which maintain the association between BEOs and keep track of endpoints when they move or fall) and protocol objects (which manage the actual communication by providing directory services to translate logical addresses into physical ones, and provide transmission of data with sufficient quality and reliability requested by users). Though CORBA has no direct equivalence of a channel, it does accommodate the notion of stubs for parameter marshalling. Binders in CORBA could be considered to be the ORB itself. The ORB and underlying network provide the management of actual physical communication between cooperating objects.

### 3.5. Technology viewpoint

A technology specification in RM-ODP is intended to delineate the choices of technology (hardware and software) for an ODP system. The technology specification should express how the specifications of the ODP system are implemented and identify the technology relevant to the construction of the distributed systems environment. With CORBA, the technology specification would include such considerations as: the choice of ORB product, operating system platform, and hardware platform.

## 4. Comparison with RM-ODP functions

The ODP functions defined by the architecture specification of [5] are those that are either fundamental or widely applicable to the construction of ODP systems. The CORBA architecture supports the functions of RM-ODP to varying extents and this will be investigated in the following section.

### 4.1. Management functions

RM-ODP management functions are categorized by node management, thread management, object management, cluster management and capsule management. Each of these categories will be examined below.

**4.1.1. Node management.** Node management controls the processing, storage and communication functions within a node. The node management function is provided by each nucleus and offers the following node management interfaces.

- **Thread management.** RM-ODP prescribes the ability to spawn (dependent) threads and to fork (independent) threads. Operations to join, delay and synchronize

threads are also prescribed. Although CORBA does not specify standardized interfaces for the use of threads, this does not preclude developers from accessing the standard thread libraries provided by the supporting operating system and programming language.

- **Clock access and timer management.** RM-ODP specifies functions to determine current system time and the ability for applications to start, monitor and cancel timers. The CORBA architecture defines the Time Service which provides functions to enable users to obtain current time, ascertain the order in which events occurred, generate events based on timers and alarms and compute the interval between two events. The Time Operations and Internationalization Service (TOIS) [34] is also an adopted technology which provides a means for 'localizing' the presentation of data according to the cultural preferences of the users. An Enhanced Time Service is currently the subject of an RFP which is a service aiming to extend the capabilities of the existing OMG Time Service [11] to satisfy the demands of real-time, fault-tolerant applications regarding the use and management of clocks, time intervals and timers.
- **Channel creation and interface location management.** RM-ODP specifies functions to enable the binding between an engineering object and an instance of the trading function. In the CORBA environment, all objects can potentially access the functions/interfaces of the OMG Object Trader Service [26] through standard RPC. RM-ODP also requires that object interfaces be available for binding to objects in other capsules. The CORBA object reference is a relocation transparent reference. An object reference is valid for the lifetime of the object. If an object implementation migrates at any point during its lifetime, the client holding the object reference will not need to be aware of this relocation. However, if the client was already bound to a migrated server, that client will now need to update their object reference.
- **Capsule template instantiation and capsule deletion management.** RM-ODP references functions to instantiate capsule templates and deletion of capsules. We anticipate that the CORBA Components architecture [14], will provide operations for both the removal and deregistration of containers.

**4.1.2. Object management.** Within the RM-ODP architecture, the management of a distributed environment exists at a number of levels of abstraction; these are *object*, *cluster* and *capsule*. Object management functions underpin those provided in cluster management, similarly, cluster management functions underpin those offered by capsule management. In RM-ODP, object management specifies the functions to both checkpoint and delete objects from an environment. It is possible to checkpoint the state of an object within the CORBA architecture using the functions of the Persistent Object Service (POS 1.0) [27], due to be succeeded by the Persistent State Service (PSS 2.0) [28], currently the subject of an RFP. CORBA also provides *interfaces* for deleting objects in the Object Lifecycle Service but does not provide an *implementation* of these services; this is the responsibility of the systems programmer. Migration of

**Table 2.** Coordination functions.

ODP function	Explanation	Supported in CORBA
Event notification	Enables the recording of event histories.	Provided by the Event Service [11], which has mechanisms for decoupled communication between objects supporting both a push and pull model of communication. The Event Service has been extended by the Notification Service which enables consumers to subscribe to particular classes of events [24].
Checkpointing and recovery coordination	Policies that govern when a cluster should be checkpointed, recovered where a cluster should be recovered, where checkpoints should be stored and which checkpoint is recovered.	Mechanisms to support the existence of policies regarding the coordination of checkpointing and recovery exist in the Externalization Service. Development of these policies is the responsibility of the application designer.
Deactivation, reactivation and migration	Deactivation and reactivation policies for clusters.	Policies to co-ordinate deactivation, reactivation and migration activities are the responsibility of the application designer, though there exist mechanisms within the Externalization Service for use in the implementation of these policies.
Group (multiparty bindings)	Mechanisms to coordinate the interactions of groups of objects in a multiparty binding.	No
Replication	A specialization of the group function, where each member (replica) must participate in <i>all</i> interactions in the same order.	It is possible to replicate objects within the CORBA environment using the <i>externalize()</i> and <i>internalize()</i> functions of the Externalization Service. There is no guarantee that all members in a replicated group will participate in <i>all</i> interactions, and that <i>all</i> interactions will occur in the <i>same</i> order.
Transaction functions	Consistency of concurrent transactions (ACID) properties should be provided by the architecture.	The OTS supports consistency of transactions. Underpinning this service is the Concurrency Control Service (CCS). Applications are still required to mark the start and end of the transaction.
Engineering Interface Reference Tracking Function	Maintains information on the possession of engineering interface references, enabling the garbage collection of unused references.	No

single objects within the CORBA environment is supported through the provision of standardized *interfaces* (move, copy and delete) within the Object Lifecycle Service. Provision of an *implementation* of the Object Lifecycle Services will need to be provided by the systems programmers and applications will need to invoke them directly. Consequently, CORBA cannot be considered to be migration transparent.

**4.1.3. Cluster management.** RM-ODP specifies the following functions in reference to cluster management.

- Cluster checkpoint. Cluster checkpointing provides a function to checkpoint the state of each object within a cluster. It maintains the configuration of objects within the cluster and sufficient information to re-establish the distributed binding that involved the objects within the cluster. It is possible to checkpoint the state of a specified group of objects within the CORBA architecture using operations of the Externalization Service. To externalize an object is to record the object's state in a stream of data. If a client object wishes to store the state of multiple objects (or related sets of objects) the application will issue a *begin\_context()* operation before the first *externalized()* request and then issues an *end\_context()* following the last *externalize()* request. The operations of the Externalization Service build on the interfaces provided by the Relationship Service. Maintenance of configuration information in order to re-establish distributed bindings is not provided in the CORBA architecture.
- Cluster deletion, deactivation and failure. RM-ODP specifies the ability to delete all objects within a cluster, the cluster manager and any infrastructure objects solely supporting the cluster (e.g. stubs and binders and protocol objects). CORBA provides interfaces to delete single objects within the Object Lifecycle Service, and will also support deletion of related groups of objects and their associated supporting infrastructure objects through the CORBA Components architecture. Cluster deactivation involves checkpointing of each object within the cluster and then deleting the cluster and its supporting infrastructure. This functionality will be provided within the CORBA architecture through the CORBA Components architecture.

**Table 3.** Repository functions.

ODP function	Explanation	Supported in CORBA
Storage	Stores data in a repository. Can be used in checkpointing and reactivating clusters, capsules and objects.	The CORBA architecture does not provide a generic storage function for application use as there exist many stable database technologies to serve this purpose.
Information organization	Manages a repository of information described by an information schema. Functions include modifying and updating the information schema, querying the repository using the query language, modifying and updating the repository.	RM-ODP defines a generic repository and functions for storing information, but CORBA does not. CORBA defines specific information repositories, for example, the Implementation Repository.
Relocation	A repository of 'locations for interfaces'.	The Implementation Repository supports management of a repository for the locations of object interface implementations.
Type repository	Manages a repository of type specifications and relationships. It provides functions to query the type specification, assert relationships between types and query relationships between types.	The CORBA architecture supports a type of repository known as the Meta-Object Facility (MOF) [22].
Trading	Mediates advertisement and discovery of interfaces.	The CORBA Object Trader Service [26] conforms precisely to the RM-ODP notion of a trader.

- Cluster reactivation and recovery. Cluster reactivation is sometimes a capsule management function because the cluster manager may have been deleted during the process of deactivation. In any case, a cluster can be recovered from one of the checkpoints. If the associated cluster manager has not been deleted, a cluster can also initialize its own recovery, otherwise recovery is a capsule management function. Reactivation and recovery functions for groups of objects are available within the CORBA architecture through the *internalization()* operations of the Externalization Service.
- Cluster migration. Cluster migration involves cloning the source cluster into a target capsule, followed by the deletion of the source cluster. It is possible to achieve the migration of groups of objects within CORBA, using the Externalization Service to *externalize()* or checkpoint a group of objects, then *internalize()* or copy these objects to a new location. Deletion of the source objects could be provided using the interfaces of the Object Lifecycle Service [25], provided that an application developer writes an implementation of these interfaces.

**4.1.4. Capsule management.** The RM-ODP specifies the following functions in reference to capsule management:

- Cluster template instantiation. This function is intended to instantiate clusters (including recovery and reactivation), checkpointing of all clusters in a capsule, (using cluster management functions) and the deactivation of all clusters within a capsule.
- Capsule deletion. Deletion of all clusters within a capsule and supporting infrastructure objects (e.g. stubs, binders, and protocol objects).

We anticipate that CORBA Components [14] will result in the provision of container-level operations that will provide support for the RM-ODP capsule management functions (possibly with the assistance of the POA).

#### 4.2. Coordination functions

The co-ordination functions of RM-ODP are compared with the CORBA functions in table 2.

#### 4.3. Repository functions

The repository functions of RM-ODP are compared with the CORBA functions in table 3.

#### 4.4. Security functions

The CORBA architecture provides standardized interfaces for all of the security-related considerations noted in the RM-ODP specification (access control, security audit, authentication, integrity, confidentiality, non-repudiation and key management), but not implementations thereby ensuring security technology independence. This can potentially lead to islands of interoperability due to incompatible security technology (e.g. the interworking of heterogeneous access control schemes). The OMG is currently addressing this issue through the Secure ORB Interworking RFP [30].

### 5. Lessons learned

This paper has examined the level of convergence between two prominent architectures for open distributed processing (CORBA and RM-ODP). The examination has revealed that although CORBA does not currently support all of the concepts and functionality specified in the RM-ODP [3–6],

these architectures are parallel in their interpretation and support of an ODP system in a number of respects.

CORBA, unfortunately, provides little or no support for the concepts expressed in the RM-ODP enterprise viewpoint despite the staggering amounts of effort invested by the OMG into this area. The issue of defining an enterprise modelling architecture is again the subject of a new suite of RFPs.

CORBA does, however, provide strong support for the concepts expressed in the RM-ODP information viewpoint. An investigation of CORBA's computational model reveals some significant deviations from RM-ODP. The binding model in RM-ODP provides a richer set of operations than that of CORBA. CORBA still provides no support for multicast and plug-and-play binding (although there exist RFPs to document these issues, they remain, however, largely unresolved).

CORBA and RM-ODP have similar support for the major concepts of the engineering and technology viewpoints, for example, the RM-ODP cluster and the soon-to-be-released CORBA Components [14]. Nonetheless, there exist a myriad of minor divergences between CORBA and RM-ODP in the detailed level of the engineering viewpoint. However, this may be reasonably argued to be due to an unbalanced, overspecification of this particular viewpoint in the RM-ODP.

CORBA is not an attempt to embody all of the principles and concepts as specified in the RM-ODP. CORBA is an evolving standard whose complete set of concepts and functions are yet to be realized. RM-ODP was developed largely by the research community in response to the need to provide a standardized framework for the development of open distributed architectures. CORBA, however, has been developed in response to natural market pressures by a consortium of users, computer vendors and industry alliances.

Nevertheless, the similarities between CORBA and RM-ODP are extensive and their areas of difference are current or proposed work in progress with the OMG. Despite divergent motivations CORBA and RM-ODP are producing parallel solutions. It is unfortunate that differences in terminology are obscuring the correspondence between these two architectures. We urge future standards developers to aim for greater cohesion between existing standards and evolving standards.

## Acknowledgments

The work reported in this paper has been funded, in part, by the Co-operative Research Centre Program through the Department of Industry, Science and Tourism of the Commonwealth Government of Australia.

## References

- [1] Blair G and Stefani J 1998 *Open Distributed Processing and Multimedia* (Longman, Addison Wesley)
- [2] Iida K and Kikuchi J 1997 *Evaluation of a Method for Reliable Message Transfer Communication in CORBA (EDOC '97: Proc. 1st Int. Enterprise Distributed Object Computing Workshop) (October, 1997, Australia)*
- [3] ITU-T Recommendation X.901 (ISO/IEC 10746-1) ODP Reference Model Part 1 *Overview* June 1995, webpage [http://www.dstc.edu.au/AU/research\\_news/odp/ref\\_model/standards.html](http://www.dstc.edu.au/AU/research_news/odp/ref_model/standards.html)
- [4] ITU-T Recommendation X.902 (ISO/IEC 10746-2) ODP Reference Model Part 2 *Foundations* January 1995, webpage [http://www.dstc.edu.au/AU/research\\_news/odp/ref\\_model/standards.html](http://www.dstc.edu.au/AU/research_news/odp/ref_model/standards.html)
- [5] ITU-T Recommendation X.903 (ISO/IEC 10746-3) ODP Reference Model Part 3 *Architecture* January 1995, webpage [http://www.dstc.edu.au/AU/research\\_news/odp/ref\\_model/standards.html](http://www.dstc.edu.au/AU/research_news/odp/ref_model/standards.html)
- [6] ITU-T Recommendation X.904 (ISO/IEC 10746-4) ODP Reference Model Part 4 *Architectural Semantics* June 1995, webpage [http://www.dstc.edu.au/AU/research\\_news/odp/ref\\_model/standards.html](http://www.dstc.edu.au/AU/research_news/odp/ref_model/standards.html)
- [7] The Object Management Group 1998 *Business Object Component Architecture (BOCA)* (Framingham, MA: Object Management Group) bom/98-01-14, <ftp://ftp.omg.org/pub/docs/bom/98-01-14.pdf>
- [8] The Object Management Group 1997 *COM/CORBA Part A Specification* (Framingham, MA: Object Management Group) orbos/97-09-07, <ftp://ftp.omg.org/pub/docs/orbos/97-09-07.pdf>
- [9] The Object Management Group 1997 *COM/CORBA Part B Specification* (Framingham, MA: Object Management Group) orbos/97-09-19, <ftp://ftp.omg.org/pub/docs/orbos/97-09-06.pdf>
- [10] The Object Management Group 1997 *Common Business Objects RFI* (Framingham, MA: Object Management Group) bom/97-06-02, webpage [http://www.omg.org/library/schedule/Common\\_Business\\_Objects.RFI.htm](http://www.omg.org/library/schedule/Common_Business_Objects.RFI.htm)
- [11] The Object Management Group 1998 *The Common Object Request Broker Architecture and Specification, CORBA/IIOP 2.2 Specification* (Framingham, MA: Object Management Group) formal/98-07-01, webpage <http://www.omg.org/corba/corbaiiop.htm>
- [12] The Object Management Group 1997 *Common Secure IIOP (CSI) Specification* (Framingham, MA: Object Management Group) orbos/96-06-20, webpage [http://www.omg.org/techprocess/meetings/schedule/Technology\\_Adoptions.htm](http://www.omg.org/techprocess/meetings/schedule/Technology_Adoptions.htm)
- [13] The Object Management Group 1997 *Control and Management of A/V Streams* (Framingham, MA: Object Management Group) telecom/(97-05-07) <ftp://ftp.omg.org/pub/docs/formal/98-06-05.pdf>
- [14] The Object Management Group 1997 *CORBA Component Model* (Framingham, MA: Object Management Group) webpage [http://www.omg.org/techprocess/meetings/schedule/CORBA\\_Component\\_Model.RFP.html](http://www.omg.org/techprocess/meetings/schedule/CORBA_Component_Model.RFP.html)
- [15] The Object Management Group 1998 *CORBA Mapping for Business Object Initiative UML Profiles* (Framingham, MA: Object Management Group) bom/98-10-05, ad/98-10-13, <ftp://ftp.omg.org/pub/doc/bom/98-10-04.pdf>
- [16] The Object Management Group 1998 *CORBA Messaging* (Framingham, MA: Object Management Group) webpage [http://www.omg.org/library/schedule/Technology\\_Adoptions.htm](http://www.omg.org/library/schedule/Technology_Adoptions.htm)
- [17] The Object Management Group *DCE/CORBA Interworking RFP* (Framingham, MA: Object Management Group) webpage [http://www.omg.org/library/schedule/DCE\\_CORBA\\_InterworkingRFP.htm](http://www.omg.org/library/schedule/DCE_CORBA_InterworkingRFP.htm)
- [18] The Object Management Group *Fault Tolerant CORBA RFP* (Framingham, MA: Object Management Group) <ftp://ftp.omg.org/pub/docs/orbos/98-04-01.pdf>
- [19] The Object Management Group 1998 *Human-Usable Textual Notation for UML Profiles* (Framingham, MA: Object Management Group) bom/98-10-04, ad/98-10-12, <ftp://ftp.omg.org/pub/doc/bom/98-10-03.pdf>
- [20] The Object Management Group 1996 *IDL to Java RFP* (Framingham, MA: Object Management Group)

- orbos/96-08-01, webpage [http://www.omg.org/library/schedule/IDL\\_to\\_Java\\_RFP.htm](http://www.omg.org/library/schedule/IDL_to_Java_RFP.htm)
- [21] The Object Management Group 1997 *Java to IDL RFP* (Framingham, MA: Object Management Group) orbos/97-03-08, webpage [http://www.omg.org/library/schedule/Java\\_to\\_IDL\\_RFP.htm](http://www.omg.org/library/schedule/Java_to_IDL_RFP.htm)
- [22] The Object Management Group 1997 *Meta Object Facility Specification (MOF)* (Framingham, MA: Object Management Group) ad/97-08-14, <ftp://ftp.omg.org/pub/docs/ad/97-08-14.pdf>
- [23] The Object Management Group 1996 *Multiple Interfaces and Composition RFP* (Framingham, MA: Object Management Group) orb/96-01-04, webpage [http://www.omg.org/library/schedule/Multiple\\_Interfaces\\_and\\_Composition.htm](http://www.omg.org/library/schedule/Multiple_Interfaces_and_Composition.htm)
- [24] The Object Management Group 1997 *Notification Service* (Framingham, MA: Object Management Group) telecom/97-01-03, webpage [http://www.omg.org/library/schedule/Notification\\_Service\\_RFP.htm](http://www.omg.org/library/schedule/Notification_Service_RFP.htm)
- [25] The Object Management Group 1996 *Object Lifecycle Service* (Framingham, MA: Object Management Group) webpage <http://www.omg.org/corba/sectrans.htm#life>
- [26] The Object Management Group 1996 *Object Trader Service* (Framingham, MA: Object Management Group) webpage [http://www.omg.org/library/schedule/Technology\\_Adoptions.htm](http://www.omg.org/library/schedule/Technology_Adoptions.htm)
- [27] The Object Management Group 1994 *Persistent Object Service 1.0* (Framingham, MA: Object Management Group) webpage <http://www.omg.org/corba/csindx.htm>
- [28] The Object Management Group 1997 *Persistent State Service 2.0 RFP* (Framingham, MA: Object Management Group) orbos/97-06-07, webpage [http://www.omg.org/library/schedule/Persistent\\_State\\_Service\\_2.0\\_RFP.htm](http://www.omg.org/library/schedule/Persistent_State_Service_2.0_RFP.htm)
- [29] The Object Management Group 1997 *Realtime CORBA 1.0 RFP* (Framingham, MA: Object Management Group) orbos/97-09-31, webpage [http://www.omg.org/techprocess/meetings/schedule/Realtime\\_CORBA\\_1.0\\_RFP.htm](http://www.omg.org/techprocess/meetings/schedule/Realtime_CORBA_1.0_RFP.htm)
- [30] The Object Management Group 1997 *Secure ORB Interworking RFP* (Framingham, MA: Object Management Group) orbos/97-02-04, webpage [http://www.omg.org/library/schedule/Technology\\_Adoptions.htm](http://www.omg.org/library/schedule/Technology_Adoptions.htm)
- [31] The Object Management Group 1998 *Security 1.2 RTF* (Framingham, MA: Object Management Group) webpage [http://www.omg.org/library/schedule/Security\\_1.2\\_RTF.htm](http://www.omg.org/library/schedule/Security_1.2_RTF.htm)
- [32] The Object Management Group 1996 *System Management Facility* (Framingham, MA: Object Management Group) 1995/(95-12-02)-1995/(95-12-06), webpage [http://www.omg.org/library/schedule/Technology\\_Adoptions.htm](http://www.omg.org/library/schedule/Technology_Adoptions.htm)
- [33] The Object Management Group 1998 *Task and Session CBOs* (Framingham, MA: Object Management Group) bom/98-07-05, webpage [http://www.omg.org/library/schedule/Technology\\_Adoptions.htm](http://www.omg.org/library/schedule/Technology_Adoptions.htm)
- [34] The Object Management Group 1997 *Time and Internationalization Service* (Framingham, MA: Object Management Group) cf/97-06-16, webpage [http://www.omg.org/library/schedule/Technology\\_Adoptions.htm](http://www.omg.org/library/schedule/Technology_Adoptions.htm)
- [35] The Object Management Group 1998 *A UML Profile for CORBA* (Framingham, MA: Object Management Group) bom/98-10-03, ad/98-10-11, webpage <ftp://ftp.omg.org/pub/doc/bom/98-10-02.pdf>
- [36] The Object Management Group 1998 *A UML Profile for Enterprise Distributed Computing* (Framingham, MA: Object Management Group) bom/98-10-02, ad/98-10-10, <ftp://ftp.omg.org/pub/doc/bom/98-10-01.pdf>
- [37] The Object Management Group 1997 *UML—Unified Modelling Language Specification* (Framingham, MA: Object Management Group) ad/(97-08-02)-ad/(97-08-09), webpage [http://www.omg.org/library/schedule/Technology\\_Adoptions.htm](http://www.omg.org/library/schedule/Technology_Adoptions.htm)
- [38] The Object Management Group 1998 *Workflow Management Facility* (Framingham, MA: Object Management Group) bom/98-06-07, webpage [http://www.omg.org/library/schedule/Technology\\_Adoptions.htm](http://www.omg.org/library/schedule/Technology_Adoptions.htm)
- [39] Siegel J 1996 *CORBA Fundamentals and Programming* (New York: Wiley)