

Introducing Open Distributed Processing

Kerry Raymond

kerry@citrus.citr.uq.oz.au

Centre of Expertise in Distributed Information Systems
Centre for Information Technology Research
University of Queensland
QLD 4072 Australia

Abstract

Open Distributed Processing (ODP) is a standardization effort within Layer 7 of the Open Systems Interconnection (OSI) Reference Model. ODP's main goal is to build distributed systems from networked systems, i.e. to achieve a higher level of transparency. The emphasis is on *open* systems through which organizations can interwork world-wide.

1 Introduction

Open Distributed Processing (ODP) is an International Organization for Standardization (ISO) project to develop a Reference Model for distributed processing. The work is undertaken by ISO/IEC SC21 WG7 as a joint effort with CCITT VII/Q19 (Distributed Applications Framework).

2 Goals of ODP

The goals for ODP standardization are:

- Portability of applications in a distributed heterogeneous environment. This includes both static portability (initial process placement) and dynamic portability (process migration).
- Interworking between ODP systems to allow information to be exchanged *meaningfully* and functionality to be used *conveniently*.
- Distribution transparency in an ODP system, removing the distinction between "local" and "remote".

Transparency takes many forms (e.g. transparency of location, heterogeneity, and autonomy). For some forms (e.g. location), ODP aims to achieve total transparency while other forms are expected to achieve only partial transparency (e.g. autonomy).

3 Scope of ODP

Like the Open Systems Interconnection (OSI) Model, the role of ODP's Reference Model is to:

- Define the scope of an ODP system (i.e. what it is and what it is not)
- Determine the requirements of an ODP system
- Develop a framework to identify:
 - What components are required for an ODP system
 - How the components interrelate
 - The interfaces and operations of the components
 - The need for standardization of the components

- Select appropriate modelling constructs and specification tools to describe ODP systems and their components

The Reference Model will describe the interfaces and operations of an ODP component generically and might identify a number of alternatives that could be used to implement that component. The Reference Model will need to include a compatibility matrix to indicate which alternatives for one component are compatible with alternatives for another component.

The ODP project itself will not further refine the component descriptions as the ODP project is *not* intended to produce a single solution or specific protocols. ISO and CCITT will “spin-off” separate projects to perform the detailed analysis and standardization of these components and their alternative implementations. Like the OSI Reference Model, the ODP Reference Model is intended to provide the “big picture”.

4 Viewpoints and Aspects

The work of ODP in analysis and description is structured around the notion of **viewpoints** and **aspects**.

4.1 Viewpoints

Any description of a system depends on the perspective of the writer. For example, a university is seen by an accountant as a balance sheet of income and expenses, by a researcher as laboratories and libraries, and by a student as classes and examinations. The ODP viewpoints capture five different perspectives of a computer system:

Enterprise for social and (human) organizational requirements and policies

Information for data and its interrelationships

Computational for the operations performed by the system and their interfaces

Engineering for the high-level design of the system; its components and their interworking

Technology for low-level requirements for hardware and software

4.2 Aspects

The ODP aspects are a method to group issues by functionality. There are three aspects which *use* distribution:

- Processing, storage, and user access

and four aspects which *provide* distribution:

- Communication, identification (including naming and addressing), management, and security

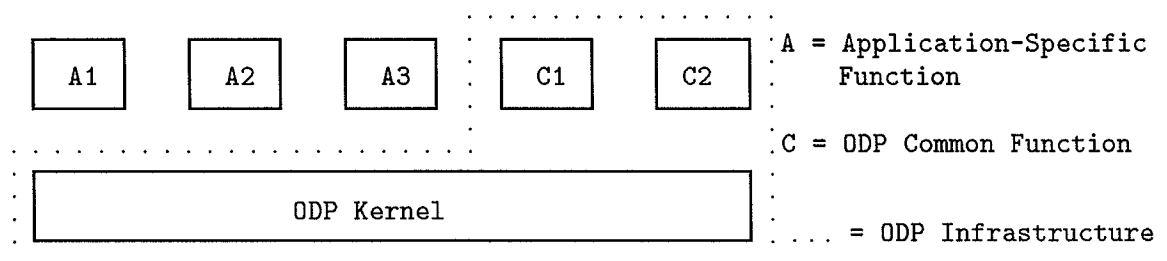


Figure 1: ODP Reference Model in the Computational Viewpoint

5 ODP Reference Model

To date, the emphasis has been on the development of the ODP Reference Model in the Computational and Engineering viewpoints.

Figure 1 shows the ODP Reference Model in the Computational Viewpoint. In this viewpoint, there are a number of functions, some of which are specific to the particular application (e.g. booking a hotel room) while others are common to a number of applications (e.g. international currency conversions). Declaring a function to be an **ODP common function** is a subjective decision but typically the function is used by a wide range of applications. It is anticipated that naming, directory, access control, and object management services and operations will be included in the initial set of ODP common functions and will probably be the target of standardization efforts in the coming years.

The ODP infrastructure consists of the ODP common functions and the ODP kernel. The ODP kernel provides distributed, transparent interaction between application-specific and common functions.

Figure 2 shows the relationship between the ODP Reference Model in the Computational Viewpoint and the Engineering Viewpoint.

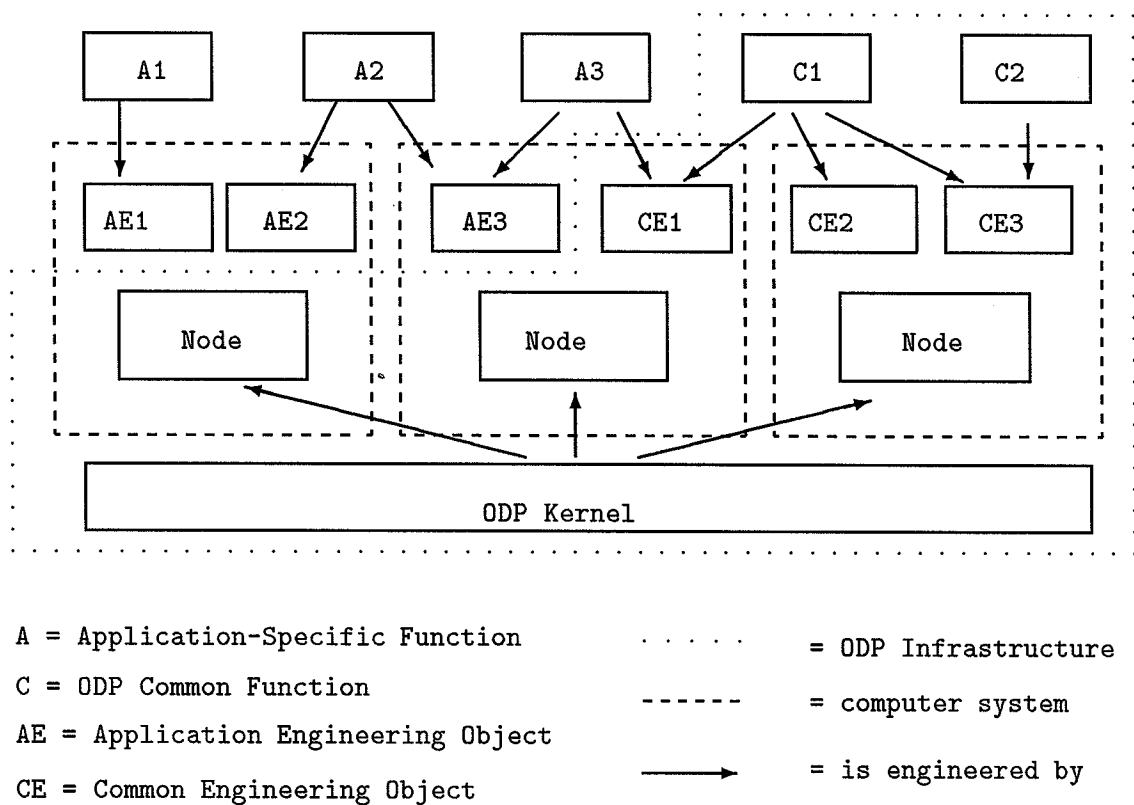


Figure 2: ODP Reference Model in the Engineering Viewpoint

In the Engineering Viewpoint, an ODP system shows functions distributed across a number of computer systems (which might be either a single computer, a virtual machine, or a cluster architecture). Within each computer system, there are the "local" resources (processing, storage, and communications) called the **node**. The ODP kernel is the sum of the resources of the nodes in the ODP system and the communications network which joins them.

Engineering objects (processes) implement the functions in the computational viewpoint. Each application-specific function could be realized by one or more engineering objects (either application-specific or common). Common functions are realized by one or more common engineering objects. An engineering object could be shared between a number of functions. For example, the same database system could be used to implement logically disjoint functions, e.g. hotel bookings and staff payments.

An engineering object executes in only one computer system although it might be replicated at other computer systems.

6 Modelling and Specification

The use of natural language in existing standards permits numerous interpretations leading to incompatible implementations. Therefore the ODP group is committed to the use of formal methods for modelling and specifying ODP systems and ODP-related standards. To achieve this aim, the ODP group is defining fundamental modelling concepts (e.g. object, behaviour, and interaction) and establishing if and how these fundamental concepts can be represented in specification languages. From these fundamental concepts, high-level concepts (e.g. client, service, replication) can be specified. Using formal specification, standards for ODP components and systems will be more precise and implementations can potentially be verified.

The ODP group has not yet selected a specification language but has determined criteria for the language. The chosen specification language:

- Must represent the fundamental modelling concepts
- Must be able to specify the higher-level concepts from the fundamental concepts (i.e. the language must be able to have sufficiently powerful constructors)
- Should be able to describe properties and constraints (whether static, temporal, or real-time)
- Should support specifications at the different levels of abstraction used throughout the design process
- Should be modular and object-oriented
- Should support verification and reasoning (i.e. it must have a formal semantics)
- Should have sufficient syntactic sugar, tutorial material, and support tools to assist its users

The ODP group is evaluating well-known formal description techniques including LOTOS [1], Estelle [2], and Z [3,4].

7 Trader: The First Spin-off Standard

The ODP requirements analysis has identified **trading** as an ODP common function in the identification (naming) aspect. The trader has been described as a “dating service for objects”; it is a mixture of classified advertising and match-making. Servers **export** (advertise) their services in the appropriate classifications to the trader. Each service type has associated attributes (or properties), e.g. cost and features available. For a printer, the attributes might be price per page, whether the printer understands Postscript, and so on.

The export operation must provide values for the attributes while a client presents the trader with a list of its requirements, expressed in terms of the attributes. Depending on the client’s needs, the trader can:

- Supply details of all suitable servers and their services. For example, “List the Postscript printers”.
- Select the most suitable service. For example, “Choose the Postscript printer which is cheapest”.
- Reserve the most suitable service. For example, “Reserve the cheapest Postscript printer for a 1000 page document”.
- Invoke the most suitable service on behalf of the client. For example, “Print this document on the cheapest Postscript printer”.

The services exported through the trader can be either software services or real-world products (e.g. houses, cars, plane tickets). The clients of the trader can be software applications or human users browsing via user-friendly interfaces.

While the trader will be used directly by applications as a broker (e.g. trading stocks, buying airline tickets, and selling spare parts), it can also be used to construct other components of the ODP infrastructure. For example, the trader could be used to advertise under-utilized processor capacity, network bandwidth, or disk space which would provide the information base for process or file migration.

Although the more obvious uses of the trader are the export of services “for sale”, the role could be reversed by clients exporting their need-for-service and waiting for the server to contact them. This is appropriate for clients with an interest in services not routinely available (e.g. a house in Toorak costing less than \$100,000).

Other interesting uses of the trader include the development of entrepreneurial servers which advertise high-level services through the trader. These entrepreneurs will not provide these services directly but will construct these services from lower-level services found through the trader. For example, package holiday services could be constructed from travel and accommodation services. Expert systems and knowledge-based techniques could be used to construct intelligent services operating within a limited universe of discourse (e.g. holidays, financial planning).

Since it is unrealistic to have a single world-wide trader, it is necessary to distribute the trading function. Loosely-coupled trader federations have been proposed as the most suitable structure for trading across many organizations, each with its own trading policies.

Figure 3 shows three federated (interworking) traders. Requests made by a client might be satisfied by services exported at any of the federated traders (subject to security and other policies, e.g. cost).

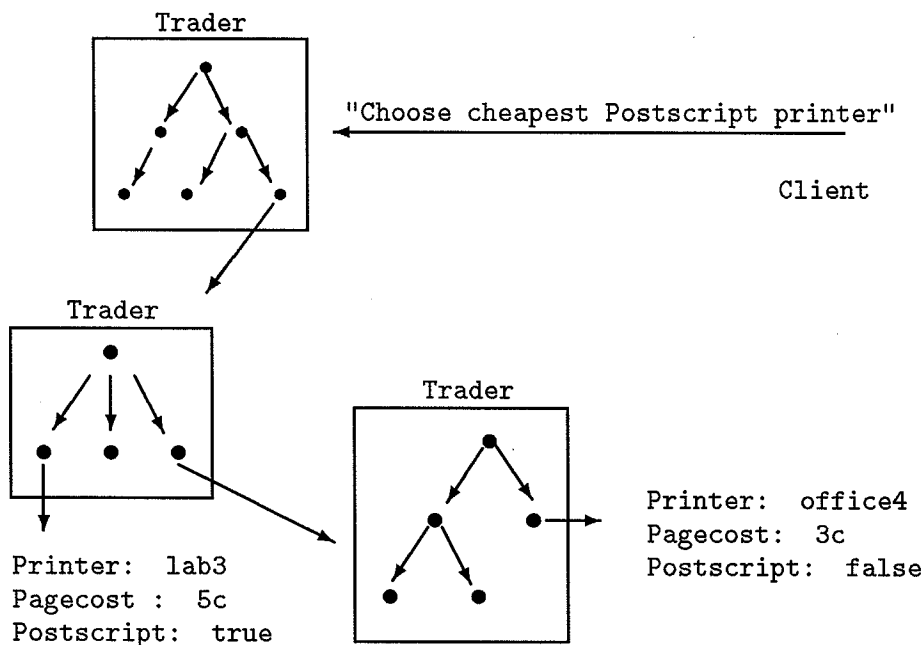


Figure 3: Traders Federate to Match-make Clients and Servers over a Wide Area

There are a number of similarities between a trader federation and the Electronic Directory Standard X.500 [5]. The extent to which the trader can build on X.500 is a subject for research. However, X.500's main purpose is to map logical names to physical addresses while the trader's main purpose is to map attribute expressions onto logical names.

Although the ODP standard is not expected to be completed until 1996, it is hoped that the Trader standard will be completed in 1995. However, the work on the Trader standard has not yet formally commenced as the Trader New Work Item is currently being balloted; it is expected to commence in May 1991.

8 Type Management

Managing data types throughout an ODP system is a problem not yet seriously addressed by the ODP standardization effort.

While some data types, e.g. monetary units, might require international standardization, others could be agreed upon within a single country or organization. Finally, there might be types which are only for local use. Where data types are subject to any degree of standardization, it is inevitable that applications developers will be satisfied with the standardized definitions. The definitions might be considered:

- Too general. These applications will need to create subtypes which are specializations of the original type. A specialized subtype might assign undefined or default values to certain components or restrict the range of values which can be permitted in certain fields. For example, a Woman is a specialized subtype of Person for which Sex = Female.

- Not sufficiently general. These applications will need to create subtypes which are extensions of the original type. An extended subtype adds additional fields or components to the original type.

Type management functions must understand the type hierarchies constructed by specialization and extension and be able to determine when instances of subtypes can be treated as instances of a supertype or another subtype of a common ancestor.

Rather than include type management within each common function (e.g. the current description of the trader includes service type management), it will be more effective to make type management an ODP common function in its own right. This will avoid the need to ensure that all interworking common functions agree on the type definitions.

9 Conclusions

This paper can give only an overview of ODP Reference Model and the Trader standard. The state of the art is contained in the output documents from the October 1990 ODP/DAF meeting: [6-8] (for ODP architecture), [9,10] (for modelling and specification), and [8,11] (for Trader). Contact Standards Australia to obtain these documents, join the mailing list, or participate in the standardization effort.

Acknowledgements

The author's research and participation in the ODP standardization effort is supported by Telecom (Australia) Research Laboratories.

References

- [1] International Organization for Standardization, "Information Processing Systems - Open Systems Interconnection - LOTOS, A Formal Description Technique based on the Temporal Ordering of Observation Behaviour," ISO IS 8807, 1988.
- [2] International Organization for Standardization, "Information Processing Systems - Open Systems Interconnection - Estelle - A Formal Description Technique based on an Extended State Transition Model," ISO IS 9074.
- [3] J. M. Spivey, *The Z Notation: A Reference Manual*, International Series in Computer Science, Prentice-Hall International, 1989.
- [4] J.M. Spivey, "Understanding Z: A Specification Language and its Formal Semantics," in *Cambridge Tracts in Theoretical Computer Science*, Cambridge University Press, 1988.
- [5] CCITT Recommendation of the X.500 Series, "X.500 The Directory : Overview of Concepts, Models and Services," CCITT X.500 / ISO 9594-1, 1988.
- [6] "Report on Topic 4.1 - Requirements for Structures and Functions," ISO/IEC JTC1/SC21/WG7 N308, October 1990.
- [7] "Report on Topic 4.2 - Structuring of Functions," ISO/IEC JTC1/SC21/WG7 N309, October 1990.
- [8] "Report on Topic 4.3 - Functions and Interface Definitions," ISO/IEC JTC1/SC21/WG7 N310, October 1990.
- [9] "Working Document - Architectural Semantics, Specification Techniques, and Formalisms," ISO/IEC JTC1/SC21/WG7 N314, October 1990.
- [10] "Recommendation X.9yy: Basic Reference Model of Open Distributed Processing - Part 2: Descriptive Model," ISO/IEC JTC1/SC21/WG7 N315, October 1990.
- [11] "Working Document on the Trader," ISO/IEC JTC1/SC21/WG7 N312, October 1990.