

# What can network management gain from ODP and its type management?

Jadwiga Indulska  
jaga@cs.uq.oz.au  
Dept of Computer Science  
University of Queensland

Kerry Raymond  
kerry@citr.uq.oz.au  
Centre for Information Technology Research  
University of Queensland

CRC for Distributed System Technology

An extended abstract

Today, computer networking tends to world-wide connectivity, through a variety of networks and networking protocols. The hardware platforms are manufactured by a large number of vendors; the software platforms are supplied by a mixture of vendors and third-party and vendor-independent software houses. The computer networks are owned and operated by many organizations and subject to a variety of regulatory and legislative authorities.

Network management, which focuses on means for effective management of the network resources and network behaviours, will have to operate in such an open, heterogenous and autonomous environment. Consolidation of many management systems and their further automation will be necessary. However, this will increase the complexity of network management and a support, which would simplify the problem, will be needed. Such a support may be found in Open Distributed Processing (ODP) which is to provide a means for modularization ( by object orientation) and integration of distributed, heterogenous systems. It can be seen, that the management of world-wide connected networks and Open Distributed Processing have some common requirements. Both require "openness", which could be briefly characterized as an ability in both the adaptability of existing heterogenous services and run-time introduction of new services. In fact, if future network management is to deal successfully with heterogeneity, it will have to evolve to an open distributed network management system. Therefore, it will have to adopt concepts and methods which are created by the Open Distributed

Processing Reference Model in order to overcome problems of distribution and heterogeneity.

In particular, the following ODP concepts are important for network management:

- object orientation,
- service trading,
- global type management.

The reason is that in an open system little commonality can be assumed, as some network management systems will be developed independently. In order to use management services globally, there must be a common understanding of the nature of those services and information, independent of their representation or provision by a particular computer network. Many network management systems will be too large to use pair-wise agreements on the semantics of data and operations. If they are built as open distributed applications, they could be supported by an ODP type management system to describe and relate the types of information, services and entities within the open network management system. Type management is a vital component which underpins many parts of any distributed environment.

An open network management system, as with any open distributed system, will continue to evolve. New types of information, services and network management techniques will come into existence. Some will be completely novel; others will update, extend, or be very similar to existing types. Although some applications will be able to operate within the confines of the types known at their compile-time, many applications will need to interwork with types discovered at run-time. As the population of information and services in an open distributed system varies with the addition and deletion of sites and services, it is unrealistic to provide only static, compile-time or manual binding between objects. Run-time binding will be needed which, in turn, requires a means for the dynamic selection of appropriate information and network management services.

Therefore, the following requirements which are set for open distributed processing are also desirable for a network management system:

1. **Dynamic selection of services.** Although instances of types enjoy certain similarities, not every instance of a type is suitable for a particular application. For example, two network management servers might offer services with the same interface type, but one server might be cheaper, closer, or have some functional or non-functional attribute which causes it to be completely unacceptable or, at least, less desirable. Therefore, the selection of suitable type instances in an open distributed system might involve selection constraints and criteria based on attributes. The ODP trader supports attribute-based selection. The trader uses support from the ODP type management to supply information about both the types of services advertised in the trader and the types of attributes associated with those services.

2. **Dynamic binding of operations.** Dynamic binding of operations is also a very crucial feature of an open network management system, as not all parts of distributed system can be compiled and linked together. Also, it is vital that an open network management system be dynamically extensible. It must be possible to introduce new objects or to exchange some objects with new compatible ones (e.g. new versions of services ) without re-compilation or re-linking of the system.
3. **Run-time type checking.** Run time binding implies a need for run-time type checking. During the binding, the types of services must be checked to ensure that they are compatible (which implies more flexible matching than just checking for identical types). During subsequent interactions using that binding, the operations, parameters, and results must be type-checked to ensure that they conform to their interface type description.

Another area where network management can gain using an ODP approach is integration. Integrating applications is a complex issue, and the solutions are often largely application specific. Open network management systems can also require integration. The merging of two independent systems will require the federation of the infrastructure components (such as traders and directories or data bases used by network management systems). Type management facilitates both integration and federation by providing a common understanding of the types themselves and the relationships between types (e.g. various forms of equivalence, subtype, and conversion relationships).

An open distributed management system can be built using different languages (specification and programming languages) which might use different type models. In order to provide solutions to the problems outlined above, an open distributed system needs both a common notion of type and knowledge about relationships between types. Such a type model should be able to express any type which can be defined in any specification or programming language. It will need to reflect some features (e.g. inheritance) of the type systems used in specific languages when such features might affect type checking during run-time binding of modules. In particular, the type system has to support polymorphism in both forms:

- inclusion polymorphism (inheritance) to allow services to be replaced by compatible ones in run-time,
- parametric polymorphism to support the creation of generic functions.

The paper will present all the requirements for the type model which could support openness of any distributed system, including a network management system. It will characterise relationships between types which facilitate dynamic selection of services, dynamic binding of services and integration of service domains. The paper also proposes a type model which includes the following first-class objects:

- data types, which can be divided into two classes:

- primary (basic) data types (e.g. char, integer, a type to express parametric polymorphism)
- constructed data types
  - \* derived types (composition of types)
  - \* type constructors (e.g. sets, records)
- operation types, consisting of:
  - operation names
  - names and data types of input parameters
  - names and data types of terminations and results
- object interface types (service types), consisting of:
  - a set of operations and behaviour of the interface
- object types, consisting of:
  - a set of interfaces

The model also includes a predefined set of relationships between types. In addition, the ODP type manager is able to learn about new relationships in run-time (new relationships can be defined and introduced to the type manager) and to start recording pairs of this relation. The relationships can cross boundaries between interface types, operation types and data types.

Such a type model can be built from scratch, as a new type model, but the paper will also show that in order to provide required functionality, an extension to well known standards (e.g. ASN.1, ACT.ONE) is feasible.