RM-ODP: The ISO Reference Model for Open Distributed Processing

Antonio Vallecillo
ETSI Informática. Universidad de Málaga
av@lcc.uma.es

1. Introduction

As software technology becomes a core part of business enterprises in all market sectors, customers demand more flexible enterprise systems. This demand coincides with the increasing use of personal computers and today's easy access to local and global communication networks, that together provide an excellent infrastructure for building open distributed systems. However, the specific problems of those large systems are currently challenging the Software Engineering community, whose traditional methods and tools are finding difficulties for coping with the new requirements. Furthermore, current IT departments are also facing serious problems to keep up with the new emerging technologies. A recent analysis by *ComputerWorld* (dated November 10, 1999) shows that 85% of IT departments in the US fail to meet their own companies' strategic business needs. There is no similar study for the Spanish IT departments, but we are sure that the situation is not much better here.

Writing business logic is hard enough. Writing infrastructure code is much more difficult, and developing open distributed applications requires a complex underlying infrastructure. This middleware tier must handle distributed communications, proxy/instance relationships, security, transactional integrity, integration with legacy applications, data transfer to databases, and scalability, to name a few. The more help you get from the infrastructure, the less you'll have to do, leaving more time for your business logic. Which probably is complex enough to keep you amused. In this respect, Desmond D'Souza summarizes very well what businesses demand from technology in today's enterprise [2]:

- First, systems must support the business. Whether they are initially driven by a business need or they arise from a technology opportunity, software systems exist to support the business. Software development must have strong links to business problems.
- Second, systems must adapt and evolve incrementally and quickly. They must be component-based, utilizing an architecture that decouples different business areas, business rules, and technology infrastructure, and that accommodates legacy, heterogeneous and federated system components.
- Third, you must use a clearly defined vocabulary to share business and technology knowledge across different business areas, eliminating historic gaps between corporate objectives, business processes and development.
- Fourth, you must use methods and architectures that scale from small projects to team development of long-lived, mission-critical systems that are robust and high performing, both within and across various enterprise boundaries.
- Finally, systems must be prepared to integrate with other systems, no matter whether they are from the same organization, or external systems from customers, providers, even competitors.

Many enterprises know about of those business needs, although most of them are a long way from effectively meeting them. And the main problem is that they are not in a position to meet them by themselves either. Trying to individually reinvent the wheel is not the solution once again. However, it may seem more sensible to jointly work with other companies towards common solutions that may help each individual enterprise reach their goals, while sharing the problems, the costs, and the efforts. The aim is to reach agreements on how to do things in such way that the global results can be useful to all parties, interoperable, and within a stable framework that remains valid over time, making investments worthwhile. Those are precisely the benefits that International Standards are trying to provide companies with nowadays.

In general, the development of standards is one of the pillars of any engineering discipline, and Software Engineering (SE) is not an exception to this rule. Maybe the youth of SE, and the (reduced?)

size of most commercial applications have not encouraged yet the development and adoption of many International Standards. However, the aforementioned increasing complexity of software systems and the new business needs are making standards indispensable. Those standard are needed not only for reusing the current global experience and know-how in these fields (hence facilitating the reduction of development duration and costs), but also for building systems that can be easily integrated with other systems that conform to the same standards, and to seamlessly interoperate with them.

The ISO (*International Standards Organization*, http://www.iso.ch/) and ITU-T (*International Telecommunication Union*, known as CCITT before, http://www.itu.int/) have been working on a joint standardization effort under the heading of *Open Distributed Processing* (ODP). The goal is to define a reference model to integrate a wide range of future ODP standards for distributed systems and maintain consistency among them. The reference model (known as RM-ODP, *Reference Model – Open Distributed Processing*) provides the coordination framework for ODP standards, creating an infrastructure within which support of distribution, interworking and portability can be integrated.

Among the contributions that RM-ODP provides for the development of open distributed applications are the following:

- RM-ODP offers a conceptual framework and an architecture that integrates aspects related to the distribution, interoperation and portability of software systems, in such way that hardware heterogeneity, operating systems, networks, programming languages, databases and management systems are transparent to the user. In this sense, RM-ODP manages complexity through a "separation of concerns", addressing specific problems from different points of view.
- RM-ODP offers a coordinating framework for the standardization of ODP, able to integrate current and future standards, and maintain consistency among them.
- RM-ODP provides a short, clear and explicit specification of concepts and constructs that define semantics, independently of the representation, methodologies, tools and processes used for the development of open distributed applications. RM-ODP offers a vocabulary and a common semantic framework to all the applications' participants (from managers to users, from designers to developers), and encourages the use of formal notations (such as ESTELLE, LOTOS, SDL, or Z) for the definition of those concepts and the specification of the architecture.

The following sections will explore with more detail the structure of RM-ODP, its concepts, constituent standards, and the benefits it may offer to the developer of open distributed applications.

2. The reference model RM-ODP

Distributed systems can be very large and complex, and the many different considerations which influence their design can result in a substantial body of specification, which needs a structuring framework if it is to be managed successfully. The purpose of the RM-ODP is to define such a framework.

2.1 Basic Standards

RM-ODP consists of four basic International Standards:

- **Overview** (ISO/IEC 10746-1; ITU-T X.901): Contains a motivational overview of ODP, giving scoping, justification and explanation of key concepts, and an outline of the ODP architecture. It contains explanatory material on how the RM-ODP is to be interpreted and applied by its users, who may include standard writers and architects of ODP systems.
- **Foundations** (ISO/IEC 10746-2; ITU-T X.902): Contains the definition of the concepts and analytical framework for normalized description of (arbitrary) distributed processing systems. It introduces the principles of conformance to ODP standards and the way in which they are applied. In only 18 pages, this standard sets the basics of the whole model in a clear, precise and concise way.
- **Architecture** (ISO/IEC 10746-3; ITU-T X.903): Contains the specification of the required characteristics that qualify distributed processing as open. These are the constraints to which ODP standards must conform. This recommendation also defines RM-ODP *viewpoints*, subdivisions of the

specification of a whole system, established to bring together those particular pieces of information relevant to some particular area of concern.

- **Architectural Semantics** (ISO/IEC 10746-4; ITU-T X.904): Contains a formalization of the ODP modeling concepts by interpreting each concept in terms of the constructs of the different standardized formal description techniques.

2.2 Fundamental Concepts

A good framework should allow different parts of the design to be worked on separately if they are independent, but should clearly identify those places where different aspects of the design constrain one another. In order to achieve this, RM-ODP uses several structuring approaches:

- The specification of a complete system in terms of *viewpoints*.
- The use of a *common object model* for the specification of the system from every viewpoint.
- The definition of an infrastructure that provides *distribution transparencies* for system applications, hiding the complexity and problems of specific concerns.
- The definition of a set of *common functions* that provide general services needed during the design and development of open distributed systems.
- A framework for the evaluation of conformance based on *conformance points*.

2.2.1 Viewpoints

Most complex system specifications are so extensive that no single individual can fully comprehend all aspects of the specifications. Furthermore, we all have different interests in a given system and different reasons for examining the system's specifications. A business executive will ask different questions of a system make-up than would a system implementor. The concept of RM-ODP viewpoints framework, therefore, is to provide separate viewpoints into the specification of a given complex system. These viewpoints each satisfy an audience with interest in a particular set of aspects of the system. Associated with each viewpoint is a *viewpoint language* that optimizes the vocabulary and presentation for the audience of that viewpoint.

The RM-ODP framework provides five generic and complementary viewpoints on the system and its environment:

- The **enterprise** viewpoint, which focuses on the purpose, scope and policies for the system. It describes the business requirements and how to meet them.
- The **information** viewpoint, which focuses on the semantics of the information and the information processing performed. It describes the information managed by the system and the structure and content type of the supporting data.
- The **computational** viewpoint, which enables distribution through functional decomposition on the system into objects which interact at interfaces. It describes the functionality provided by the system and its functional decomposition.
- The **engineering** viewpoint, which focuses on the mechanisms and functions required to support distributed interactions between objects in the system. It describes the distribution of processing performed by the system to manage the information and provide the functionality.
- The **technology** viewpoint, which focuses on the choice of technology of the system. It describes the technologies chosen to provide the processing, functionality and presentation of information.

A viewpoint is a subdivision of the specification of a complete system, established to bring together those particular pieces of information relevant to some particular area of concern during the design of the system. Although separately specified, the viewpoints are not completely independent; key items in each are identified as related to items in the other viewpoints. However, the viewpoints are sufficiently independent to simplify reasoning about the complete specification. The mutual consistency among the viewpoints is ensured by the architecture defined by RM-ODP, and the use of a common object model provides the glue that binds them all together.

2.2.2 Objects

ODP system specifications are expressed in terms of objects. An object is a representation of an entity in the real world. It contains *information* and offers *services*. A system is composed of interacting objects.

The use of the object paradigm provides abstraction and encapsulation, two important properties for the specification and design of complex systems. Abstraction allows highlighting those aspects of the system relevant from a given perspective, while hiding those of no relevance. Encapsulation is the property by which the information contained in an object is accessible only through interactions at the interfaces supported by the object. Because objects are encapsulated, there are no hidden side effects of interactions. It also implies that the internal details of an object are hidden from other objects, which is crucial for dealing with heterogeneity, multiple implementations, interoperability and portability.

2.2.3 Distribution transparencies

Transparencies arise from the fact that, when contemplating a distributed system, a number of concerns become apparent which are a direct result of the distribution: the system components are heterogeneous, they can fail independently, they are at different and, possibly, varying locations, and so on. These concerns can either be solved directly as part of the application design, or standard solutions can be selected, based on best practice. If standard mechanisms are chosen, the application designer works in a world which is transparent to that particular concern; the standard mechanism is said to provide a *distribution transparency*. RM-ODP defines several distribution transparencies:

- The *access* transparency masks differences in data representation and invocation mechanisms to enable interworking between heterogeneous objects.
- The *failure* transparency masks from an object the failure and possible recovery of other objects (or itself) to enable fault tolerance.
- The *location*, *migration* and *relocation* transparencies allow the search and invocation of services independently from their location, and from any relocation or migration of the calling or called objects.
- The *replication* transparency masks the use of a group of compatible server objects to support an interface, while the *persistence* transparency permits the state of objects to be saved and restored.
- Finally, the *transaction* transparency masks coordination of activities amongst a configuration of objects to achieve consistency.

RM-ODP defines a set of functions and structures to achieve those transparencies. The system designer can choose which ones to use, since each one may or may not be relevant to a particular application, and each one conveys a cost (in time and resources). RM-ODP does not force the designer to select them all, but in case one is incorporated, it should conform to the model.

2.2.4 Common functions

In addition to these structuring approaches, RM-ODP gives outline definitions of a number of common functions. Those functions provide a set of common services that are either fundamental or widely applicable to the construction of ODP systems. Detailed specifications for those functions are the subject of separate and specific standards.

The functions are organized into four groups –management, coordination, repository, and security–, and most of them are either introduced by the engineering language to provide the support needed for its structures, of from convenient building blocks for the provision of transparencies. Functions are provided by objects, although it is generally left for more detailed standards or individual implementors to decide whether each function is provided by a single object, several functions are provided by one object, or a function is provided by a set of interacting objects. One of the most important common function is *trading*, that will be discussed later.

2.2.5 Conformance assessment

As previously mentioned, one of the benefits that International Standards offer is the possibility of integrating systems that conform to the same recommendation, ensuring their interoperability. In open distributed systems, the evolution and heterogeneity of their constituent components force the need of mechanisms to allow their independent instantiation and deployment by different parties that do not know each other [5], and the coexistence of multiple developers and vendors of components that implement the same specification. Therefore, conformance statements and mechanisms should be put in place to check that a component fulfills a given specification, or to ensure that a given component can replace an old one. Analogously, conformance statements are also used as invariants in the *refinement* process that brings the system specifications to a final working system that fulfills those requirements. In each step of this refinement process, conformance points and conformance statements can be defined to ensure that the result will conform to the specifications.

2.3 Some other RM-ODP Standards

In addition to the four basic standards that provide the foundations of RM-ODP, ISO and ITU-T continuously work on the definition of more standards that complement the basic model. In this section we will discuss four of those standards, which cover some of the basic concepts that RM-ODP uses: the trading function, the naming framework, the interface definition language, and how objects reference and bind to other object interfaces.

2.3.1 The trading function

The concept of trading is fundamental to RM-ODP. A trader is used to store potential service offers, so servers must register their interfaces with it; a simple form of advertisement. Potential clients send queries to a trader to look for the services they need. Once a match is found, the client is sent the location reference for the interface, i.e. the service, it requires. The client can then interact with the service provider directly. In this respect, the trader supplies a locating service.

Standard ISO/IEC 13235-1 (ITU-T X.950) specifies the trading function independently from any implementation. The specification covers the case of federated trading among traders from different vendors, and defines conformance assessments to this norm. An important characteristic of this standard is the use of a formal notation (Z) for specifying the behavior of traders.

A second standard (ISO/IEC 13235-3; ITU-T X.951) supplements the previous recommendation, specifying how the trading function can make use of OSI's X.500 directory services.

2.3.2 Naming Framework

The definition of a global naming scheme for large open and distributed systems is a difficult issue, even more when names have to be simultaneously managed by different independent parties. Therefore, standard ISO/IEC 14771 (ITU-T X.910) specifies a context-dependent global naming framework valid under the scope of RM-ODP.

2.3.3 ODP Interface Definition Language

In order to specify the services that ODP objects offer, standard ISO/IEC 14750 (ITU-T X.920) defines a textual language for the description of object interfaces, known as ODP Interface Definition Language (ODP IDL). It allows the definition of the data structures that an object manages, and the signature of the methods it implements. One of the benefits of this language is that it is independent from any possible representation of data and implementation of the object methods, hence providing the required degree of object encapsulation. In addition, the ODP IDL is perfectly consistent with the CORBA IDL, developed by the international consortium OMG (*Object Management Group*, http://www.omg.org/).

2.3.4 Interface references and bindings

In an object model, services are described by interfaces, and therefore interface references become crucial for object interactions and federation. In RM-ODP, a reference to an interface provides the information needed to establish bindings among objects, including those that support several communication protocols, or those located across different management domains. In addition, a reference to an interface should also contain enough information to account for the previously mentioned relocation transparency. The way interface references and bindings are defined in RM-ODP is described in the standard ISO/IEC 14753 (ITU-T X.930).

2.4 Standards currently under development

In addition to the standards that we have just mentioned, the ISO and ITU-T technical committees are currently working on further standards that complement the reference model RM-ODP. Those new standards cover some of the aspects of the model that were originally mentioned in the four basic standards that define the model, but that needed more in-depth specifications. Some of the drafts currently under development and that will soon be published as International Standards are the following:

- ODP-Protocol Support for Computational Interactions (ISO/IEC 14752; ITU-T X.931)
- ODP-Type Repository Function (ISO/IEC 14769; ITU-T X.960)
- ODP-Reference Model: Enterprise Viewpoint (ISO/IEC 15414; ITU-T X.911)
- ODP-Reference Model: Quality of Service (ISO/IEC 15935; ITU-T X.905)

3. Is RM-ODP really practical?

Once the basic structure and composition of RM-ODP, its motivation, and its potential benefits have been presented, we will discuss here about its practical utility. In fact, its complexity and high level of abstraction has discouraged many people from effectively using it for specifying and building open distributed applications. However, we will show here that this is not necessarily true, and that RM-ODP can provide real and practical benefits when using it for specifying, designing and constructing real ODP applications.

In the first place, there are some real applications that have been successfully built using RM-ODP, and some companies that have already adopted this reference model in their IT departments for specifying, designing and developing their systems (from Swiss banks to US Telecommunication companies). Industry is starting to effectively make use of RM-ODP.

Second, we can already count with solid commercial technology to support the model. For instance, CORBA (*Common Object Request Broker Architecture*) is a distributed object platform defined by the OGM. CORBA is consistent with RM-ODP, and provides widely accepted technologic solutions to support some of its viewpoints. CORBA also provides a set of common services (similar to the RM-ODP common functions) to support the development of large open distributed applications in heterogeneous environments. In the telecommunications industry, TINA (*Telecommunications Information Networking Architecture*), defined by TINA-C (TINA *Consortium*), describes an architecture for the development of telecommunication applications based on the concepts defined by RM-ODP. Currently TINA provides the most widespread and accepted architecture in this field. With the support of those technologies, building systems using the RM-ODP concepts is no longer a visionary and risky business.

Coming down to our particular business, the use of RM-ODP may offer us some interesting advantages:

- First, it may help us thinking from different perspectives (or *viewpoints*), greatly improving the requirement collection and analysis phases of the development of applications.
- Second, RM-ODP offers an infrastructure and a common reference model within which different requirements expressed in separate languages (those from the viewpoints) can be consistently integrated.

- Third, RM-ODP provides a set of already established *reasoning patterns* to help us specify and design our system. Those patterns assist us identify the fundamental entities of the system and the relations among them. In this sense, RM-ODP encourages us to ask the right questions to the right people, and with the appropriate degrees of abstraction and precision for building useful system specifications.
- Finally, RM-ODP provides system designers and developers with a set of mechanisms and common services to alleviate their jobs, together with a technological infrastructure that supports the model. This infrastructure is currently mature enough for building robust, efficient and competitive applications, interoperable with other systems that also conform to the same standards, and backed by industrial products with enough acceptance.

To know more about RM-ODP, in addition to reading the International Standards, some meetings provide discussion forums for some of those topics. For instance, the workshops on Behavioral Semantics that Haim Kilov usually organizes in conjunction with the main conferences on Object-Orientation (such as ECOOP, OOPSLA or TOOLS). They are mainly dedicated to discuss about how to incorporate precise (even formal) semantics into system analysis and design, and present many real experiences on the use of RM-ODP [1,3,4]. Additionally, ISO's subcommittee SC7 plenary meetings (like the one sponsoring this journal Special Issue) gather every year all ISO technical working groups on these matters. In them ISO experts not only discuss about the ongoing and future work related to ISO Standards, but also try to disseminate their knowledge and experience, present the benefits that International Standards may bring to the industry, and encourage people to use them when building open distributed systems.

4. References

1. K. Baclawski, H. Kilov, A.E. Thalassinidis, K. Tyson (Eds). *Proceedings of the 8th OOPSLA Workshop on Behavioral Semantics*. Denver (CO). November 1999.
2. D. D'Souza. Enterprise integration. Enterprise components with Catalysis/UML. In [1] pp. 49-63.
3. H. Kilov, W. Harvey (Eds.). *Object Oriented Behavioral Specifications*. Kluwer Academic Publishers, 1996.
4. H. Kilov, B. Rumpe, I. Simmonds (Eds.). *Behavioral Specifications of Business and Systems*. Kluwer Academic Publishers, 1999.
5. C. Szyperski. *Component Software. Beyond Object-Oriented Programming*. Addison-Wesley Longman, 1998.

Biography

Antonio Vallecillo holds a BsC in Mathematics, a MsC in Pure Maths, and a PhD in Computer Science. He currently works at Málaga University as an Associate Professor, and is also the Head of Málaga University IT Services. Most of his professional experience comes from the computer industry, where he has worked as a software developer and Marketing manager in several IT international companies, both in Spain and in UK. His current research interest include component models for open systems, open distributed processing, and the industrial use of formal methods.