Action Templates and Causalities in the ODP Computational Viewpoint

Raúl Romero and Antonio Vallecillo
Dpto. de Lenguajes y Ciencias de la Computación
University of Málaga, Spain
{jrromero,av}@lcc.uma.es

Abstract

The RM-ODP is a reference model that provides a coordinating framework for Open Distributed Processing standards, and offers a well-defined and comprehensive set of concepts and functions for the specification of ODP systems. Some years after its release as International Standard, an ISO Study Group will evaluate the need for a revision of RM-ODP, a customary process for ISO standards. The goal is to make use of the experiences gained from the use of the RM-ODP framework during that period, in order to propose improvements or changes if required. In order to serve as an input to that Group, this paper raises two small issues that we have discovered when trying to formalize the ODP Computational Viewpoint: the need of an independent term for referring to the signature of an Action Template, and the way in which Causalities are currently defined and handled. A proposal for addressing these issues is presented for discussion.

1. Introduction

The ISO and the ITU-T jointly developed a Reference Model for Open Distributed Processing (RM-ODP) [8, 11-14], which provides the coordination framework for ODP standards, and creates an infrastructure within which support of distribution, interworking and portability can be integrated. The goal of this joint standardization effort is to define a reference model to integrate a wide range of future ODP standards for distributed systems and maintain consistency among them.

RM-ODP provides five generic and complementary viewpoints of the system and its environment: *enterprise*, *information*, *computational*, *engineering* and *technology*. Each of them has its own specific viewpoint language, defining concepts and rules for specifying ODP systems from the corresponding viewpoint.

The *enterprise viewpoint* focuses on the purpose, scope and policies of an ODP system. The *information viewpoint* describes the semantics of information and of information processing. The ODP *computational viewpoint* describes the functionality of a system and its environment, in terms of a configuration of objects that interact at interfaces. The *engineering viewpoint* focuses on the mechanisms and functions required to support distributed interactions between objects in the system. Finally, the *technology viewpoint* focuses on the choice of technology for that ODP system.

After formalizing the enterprise and the information viewpoints concepts [3, 4] using the Maude language and system [5, 6, 7], we recently started working on the formalization of the computational viewpoint specifications [2], for which other formalization efforts also exist [1, 9, 10, 15]. Our work has allowed us to explore the basic concepts defined in ODP, in addition to those specific to the computational viewpoint. Furthermore, some case studies have been developed, and a metamodel for the Computational Viewpoint has been proposed in [2]. The metamodel describes the concepts used in a computational viewpoint specification and the relationships between them.

In general, we find that Parts 2 [12] and Part 3 [13] of the ODP Reference Model are two excellent standards, fully consistent, and solidly conceived and architected. However, the inherent complexity of some of the concepts and functions defined in these two standards, their (sometimes) cryptic definition, and the lack of examples and real applications for most of the concepts, may hinder their understandability for readers which are not familiar with such terms. Having said that, we also discovered that, once understood, the concepts provided by RM-ODP are really valuable for the specification of open and distributed systems, and that everything fits in the conceptual framework with a clock-maker precision.

However, we also discovered that the use of these standards might help uncovering some small details that cannot be easily detected otherwise. Thus, based on our experiences with the case studies and the definition of the metamodel, we observed two issues in the ODP computational viewpoint. First, the term *Action Template* seems to cover both the syntactic (i.e., signature) and semantic (i.e., behavioral) aspects of an action template. The problem is that there is no specific concept for

referring just to the signature of an action template, which may seem to be required in some situations, as we shall later see. To solve this issue we propose, roughly, to include the concept *Interaction Signature*, which will specify just the syntactic part of an Action Template.

The second issue has to do with the way in which the concept of *Causality* is used. The Standard allows specifying causalities at different granularity levels (object, interface signature, and action template), but in an asymmetric manner. We propose a homogeneous treatment of causalities for both interface signatures and action templates.

In this paper we will discuss these two issues in more detail, together with the corresponding proposals for addressing them. Our proposals try to serve as an input for the current ODP revision work, and for discussion purposes.

The structure of this document is as follows. First, Section 2 describes inconsistencies found when dealing with the concept of *Action Template*. Section 3 deals with the distinction between *causalities* contained at the object, interface, and action template levels. Finally, Section 5 draws some conclusions.

2. Action Templates

The problem we found with action templates is about the way in which this concept is used for defining operation, signal, and stream signatures. In particular, the problem appeared in the metamodel when trying to model the existing relation between interface signatures, interaction signatures, and action templates.

First, according to Part 2 [12–9.11], a *Template* is "*the specification of the common features of a collection of <X>s in sufficient detail that an <X> can be instantiated using it*". From this definition, we directly obtain that an *Action Template* can be defined as "*the specification of the common features of a collection of actions in sufficient detail that an action can be instantiated using it*."

Then, we looked at how Action Templates are used in Part 3 in the Computational Viewpoint, in which they play a very relevant role.

First, we see that Part 3 indicates [13 – 7.1.12] that "*an announcement signature is an action template*" (the underlined text is ours). Another reference appears when referring to interrogation signatures. Although we expected a similar definition, what we find in Part 3 is that "*an interrogation signature comprises an action template*" [13 – 7.1.12].

The concept *action template* appears again when defining stream interface signatures, which comprise a finite set of action templates.

So the first issue is whether signatures "are" action templates, or "comprise" action templates.

Furthermore, there is the issue of the way in which action templates are used in Part 3 for defining signatures. Commonly, signatures (of both interactions and interfaces) are considered to remain at the syntactic level, i.e., they are supposed to describe just the names and types of the actions and their parameters. Semantic information (e.g., behavior) is not usually covered by signatures. However, this does not seem to be consistent with the use of action templates for defining signatures, since action templates might also include behavioral specifications (cf. Part 2).

Certainly, this is also corroborated by Part 4 [14 – 4.4.2.12], which states that: "*It should be noted that the text in ITU-T Rec. X.902 | ISO/IEC 10746-2 treats an interface signature as a set of action templates associated with the interactions of an interface. Given that an action template is likely to include semantic information as well as syntactic. Common interpretations of interface signature deal primarily at the syntactic level, however.* […]".

We propose to solve these two issues by introducing a term that refers to the syntactic information specified by an action template, and that we have called *Interaction Signature*. This term can be used to define the signatures of announcements, interrogations, terminations, signals and flows (see Figure 1), that now are Interaction Signatures. This does not contradict the current standard text and, in fact, allows the separation of the syntactic and the semantic information specified by an action template.

Moreover, interface signatures (an abstract class that simply generalizes operation, signal and stream interface signatures) now comprise sets of interaction signatures, which seems to be more in line with the intent of the RM-ODP standard.

Finally, and as shown in the figure, the parameters of the action template are now associated to the syntactic part of such action template, that is, to its *Interaction Signature*, which also seems to be more natural than attaching them directly to the *Action Template*.

3. Causalities

Clause 13.3 of Part 2 states that "*the identification of causality allows the categorization of roles of interacting objects*". Furthermore, that clause provides "*a basic set of roles*" and specifies that a "*causality implies a constraint on each behaviour of the participating objects while they are interacting*".

Meanwhile, Clause 7.1 of Part 3 defines where the indication of the causality must be defined in each case, and for each element. For signal interfaces, their interface signatures "*comprise a set of finite action templates, one*

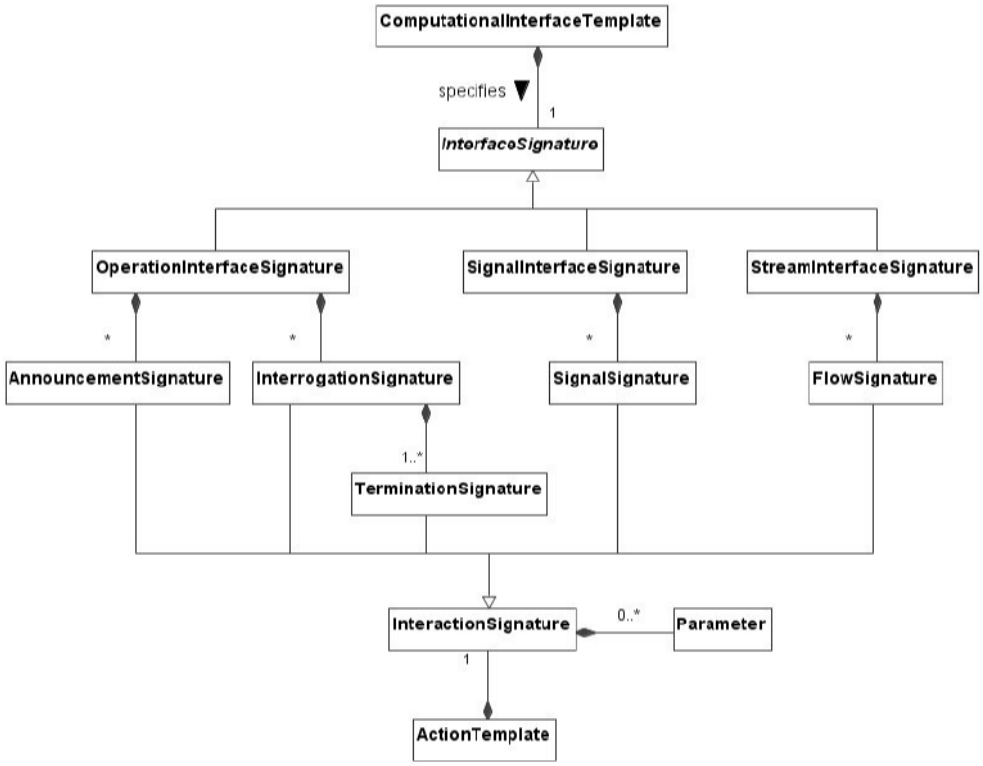

Figure 1. Interaction signatures specify the syntactic information of Action Templates

for each type of signal in the interface. Each action template comprises the name of that signal, the number, name and types of its parameters and an indication of causality with respect to the object which instantiates the template". Same for stream interfaces. However, for operation interfaces we noticed that causalities are not treated in the same way. In that case, the operation interface signature comprises, apart from a set of announcement and interrogation signatures, as appropriate, the indication of causality for the interface as a whole with respect to the object that instantiates the template.

Clause 7.2.2 of Part 3 (Interaction Rules) clearly refers to the causality "*in the interface's signature*", that seems to support the specification of causalities at the interface signature level. More precisely, sub-clauses 7.2.2.1 and 7.2.2.2 are clear and explicit when referring to this issue.

According to signal interaction rules [13 – 7.2.2.1], "*a computational object offering a signal interface of a given signal interface type*

- *initiates signals that have initiating causality in the interface's signature;*
- *responds to signal that have responding causality in the interface's signature.*"

Similarly, according to stream interaction rules [13 – 7.2.2.2], "*a computational object offering a stream interface*

- *generates flows that have producer causality in the interface's signature;*
- *receives flows that have consumer causality in the interface's signature."*

Thus, we find that, whereas the indication of causality for signal and stream interfaces is defined at the action template level, for operations it is defined at the object's interface signature level.

One of the reasons behind these decisions seems to be the fact that, for operations, the causality indication provided by the interface signature determines the causality for each action template in the interface, depending on what we are really using: invocations or terminations. However, when dealing with signal or stream interface signatures, which comprise signals or flows that may go in different directions (i.e., incoming and outgoing actions), there is no clear relationship between the causality of the interface signature, and the causalities of the individual interactions that comprise the interface signature. Thus, causalities should be defined

both for the interface with respect to the object that interacts and for each individual action template.

For example, let us consider a simple stream. Its signature could have both incoming and outgoing action templates defined for it. However, as mentioned in the interaction rules, we need to consider an indication of causality with respect to the role that the computational object plays in the communication process. This requires indicating that causality in the interface signature, which would indicate the object that produces the flow and the object that consumes it.

To address this issue, we propose to include causality definitions at both levels. However, calling it "causality" at both levels might be confusing too. Actually, the definition of causality in Part 2 refers to objects only, i.e., the granularity of causality is defined at the object level – more precisely at the object's interface signature level. But in Part 3, causality indications seem to be used at two different levels: object interface signature and action template. There is a clear need to align the granularities for these different definitions.

Thus, we propose defining causalities in every level at which this term is involved. This means incorporating causalities in individual action templates and in interface signatures. In operations, in which the causality defined at the interface signature level determines the causality of the individual interactions, a constraint should enforce such a relationship.

Figure 2 shows our proposal, where the indication of causality appears not only at the interface signature level—to specify the roles played by computational objects in the communication process as a whole—but also at the interaction signature level.

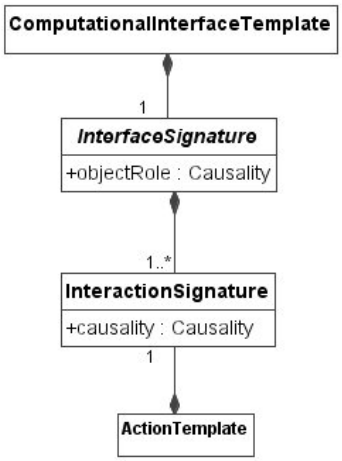

Figure 2. Indication of causality at two levels

4. Conclusions

RM-ODP was created at the beginning of last decade, but it is becoming now probably the best framework for specifying and developing large open and distributed applications. In the first place, the complexity of the applications is reaching the level where many traditional software engineering methods do not seem to be able to cope with. However, RM-ODP was specifically conceived to specify those large and complex open systems, and therefore is perfectly fit to address their specification and design. Furthermore, the level of maturity reached by the RM-ODP seems to be the adequate to fulfill the requirements of current businesses and organizations.

Some years after its release as International Standard, an ISO Study Group will evaluate the need for a revision of RM-ODP, a customary process for ISO standards. The goal is to make use of the experiences gained from the use of the RM-ODP during that period, in order to "tune" it according to the findings, and to propose improvements or changes if required.

In order to serve as an input to that Group, this paper has raised two issues that we discovered when trying to formalize the ODP Computational Viewpoint: the need of an independent term for referring to the signature of an Action Template (without taking into consideration the semantic information that an action template also contains), and the way in which Causalities are currently defined and handled. Proposals for addressing these issues have been presented. First, the term Interaction Signature has been proposed for capturing the syntactic aspects of an action template. This allows a consistent definition of all interaction signatures (announcements, interrogations, terminations, signals and flows), as shown in Figure 1. Second, we propose the definition of causalities at two levels: interface signature and interaction signature. This seems to resolve the apparent mismatch in Part 3 of the RM-ODP standard.

Finally, just to mention the need for more examples, case studies and documents describing experiences in the use of RM-ODP, in order to help software engineers fully understand the concepts in the Reference Model, whose complexity (and sometimes cryptic definitions) make them difficult to learn, understand, and properly use to specify and design large open distributed applications.

Acknowledgements The authors would like to acknowledge the work of many ODP experts who have been involved in investigating and addressing the problems of the computational specification of ODP systems. Although the views in this paper are the authors' solely responsibility, they could not have been formulated without the detailed discussions with ISO experts on ODP, in particular with Akira Tanaka and Dave Akehurst.

This work has been partially supported by Spanish Project TIC2002-04309-C02-02.

5. References

[1] D. H. Akehurst, J. Derrick and A.G. Waters. "Addressing Computational Viewpoint Design." In *Proceedings of the 7th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2003)*, pages 147-159, Brisbane, Australia, Sept. 2003. IEEE CS Press.

[2] R. Romero and A. Vallecillo. "Formalizing ODP Computational Specifications in Maude". In *Proceedings of the 8th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2004)*, Monterey, California, September 2004. Copyright IEEE CS Press.

[3] F. Durán and A. Vallecillo. Specifying the ODP information viewpoint using Maude. In H. Kilov and K. Baclawski, editors, *Proceedings of Tenth OOPSLA Workshop on Behavioural Semantics*, pages 44-57, Florida, Oct. 2001. Northeastern University.

[4] F. Durán and A. Vallecillo. Formalizing ODP Enterprise specifications in Maude. *Computer Standards & Interfaces*, 25(2):83-102, June 2003.

[5] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseger and J. Quesada. Maude: specification and programming in rewriting logic. *Theoretical Computer Science*, 285:187-243. Aug. 2002.

[6] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer and C. Talcott, Maude 2.0 manual. Available in http://maude.cs.uiuc.edu, June 2003.

[7] S. Eker, J. Meseguer and A.Sridharanarayanan. The Maude LTL model checker. In F. Gaducci and U. Montanari, editors, *Proc. Of the 4th International Workshop on Rewriting Logic and its Applications (WRLA 2002)*, volume 71 of *Electronic Notes in Theoretical Computer Science*, pages 115-142, Pisa, Italy, Sept. 2002. Elsevier.

[8] P. Linington. *RM-ODP: The architecture*. In K. Milosevic and L. Armstrong, editors, Open Distributed Processing II, pages 15-33. Chapman & Hall, Feb. 1995.

[9] E. Najm and J.B. Stefani. A formal operational semantics for the ODP computational model. *Computer Networks and ISDN System*, 27:1305-1329, 1995.

[10] E. Najm and J.B.Stefani. Computational models for open distributed systems. In H. Bowman and J. Derrick, editors, *Proceedings of FMOODS'97*, pages 157-176, Canterbury, 1997, Chapman & Hall.

[11] ITU-T Recommendation X.901 | ISO/IEC 10746-1: *ODP Reference Model Part 1. Overview*. Geneva, Switzerland, 1998.

[12] ITU-T Recommendation X.902 | ISO/IEC 10746-2: *ODP Reference Model Part 2. Foundations*. Geneva, Switzerland, 1996.

[13] ITU-T Recommendation X.903 | ISO/IEC 10746-3: *ODP Reference Model Part 3. Architecture*. Geneva, Switzerland, 1996.

[14] ITU-T Recommendation X.904 | ISO/IEC 10746-4: *ODP Reference Model Part 4. Architectural semantics*. Geneva, Switzerland, 1998.

[15] R. Sinnot and K.J. Turner. Specifying ODP computational objects in Z. In E. Najm and J.B. Stefani, editors, *Proceedings of FMOODS'96*, pages 375-390, Canterbury, 1997. Chapman & Hall.