Interaction Signatures and Action Templates in the ODP Computational Viewpoint

OUSSAMA REDA, BOUABID EL OUAHIDI
Mohammed-V University, Faculty of Sciences
Dept of Computer Sciences
Ibn Battouta P.O Box 10 14, Rabat
MOROCCO

DANIEL BOURGET
ENST Bretagne
Dept of Computer Sciences
Technopôle Iroise - CS 83818, 29238 Brest
FRANCE

Abstract:- In this work we raise two issues that we came across when aiming to formalize both interaction signatures and action templates within the ODP computational viewpoint. We discuss these two concepts and present a way to formalize them by introducing a new term to formal descriptions of interaction signatures. In the same spirit as other works, our aim is to address issues concerning how concepts of the ODP computational viewpoint are currently defined as we present some solutions to their formalisation. If required, our work aim to serve as a step to help improve or change the current process of formalizing the ODP computational viewpoint concepts using the UML language.

Key–Words: ODP, Computational Viewpoint, UML, OCL, Meta-modelling, Interaction Signature, Action Template

1 Introduction

The ODP standardization initiative has led to a framework by which distributed systems can be modeled using five viewpoints. For each viewpoint, the Reference Model [1], [2], [3] for ODP provides a viewpoint language that defines concepts and structuring rules for specifying ODP systems from the corresponding viewpoint. These viewpoints include a computational viewpoint, which is concerned with the description of the system as a set of objects that interact at interfaces - enabling system distribution. A computational specification describes the functional decomposition of an ODP system, in distribution transparent terms and is constrained by the rules of the computational language. These comprise amongst others interaction rules.

Recent work within the computational viewpoint such as [4], [5], [6] has mainly addressed the specification of the functional decomposition of an ODP system using UML. Other work [7] has focused on how to consistently formalize concepts of the ODP computational viewpoint and clarify some ambiguities found while aiming to express them formally. The authors discussed the issue concerning whether *Action Templates* belong to the syntactic level or the semantic one. Then, they proposed to introduce the term *Interaction Signature* at the syntactic level, and to reserve *Action Templates* to a semantic level while they interaction signature as syntactic. They also raised a second issue which has to do with the way in which the concept of Causality is used and have proposed some solutions.

From this perspective, we raise the issue of expressing *Operation Signatures* in terms of *Action Templates* and show how to get round the problem of whether *Operation Signatures* are kinds of *Action Templates* or are constituents of *Action Templates*. As we shall see, we propose to solve the problem by formalizing the concept of both *Invocations* and their associated *Terminations* by introducing them as roles played in *Action Templates*. On the other hand, we address another issue concerning how to describe both *Operation Signatures* and *Signal Signatures* on one side and *Flow Signatures* on the other side in terms of *Action Templates*. In fact, *Flow Signatures* differ significantly in their characteristics from both *Operation* and *Signal Signatures*. That is, a *Flow Signatures* has an information type characteristic which is not the case for *Operation* and *Signal Signatures*. Conversely, both *Operation* and *Signal Signatures* have parameters and their numbers as two characteristics which are not significant in *Flow Signatures*. We propose to solve this issue by introducing a new term referred to as **ParametrizedActionTemplate** as we shall see later.

The RM-ODP is not prescriptive about the use of any particular formal description and specification techniques for the specification of ODP systems. Re-

cently there has been a considerable amount of research [8] [9], [10] within the field of applying the UML Language [11], [12] as a formal notation with the ODP viewpoints, and particularly to the ODP computational viewpoint [4], [5], [6].

In this respect, we use the UML language to discuss and present our proposals. Our contribution is based on ideas from the field of defining notations for ODP viewpoints.

The remainder of the paper is organized as follows. In Section 2, we present concepts of *Interaction Signatures* provided by RM-ODP. We discuss in section 3 how to express *Operation Signatures* in terms of *Action Templates*. In section 4 we show how to integrate the *Flow Signatures* concept to the *Operation Signatures* model. A conclusion and perspectives end the paper.

2 Interaction Signatures concepts

In this section, we present the *Interaction Signatures* concepts as they are defined in the computational viewpoint. These definitions will serve us to discuss the ideas of the rest of the paper. the definitions are given as follows:

A computational interface template is an interface template for either a signal interface, a stream interface or an operation interface. Each interface has a signature:

- A signal interface signature comprises a finite set of action templates, one for each signal type in the interface. Each action template comprises the name for the signal, the number, names and types of its parameters and an indication of causality (initiating or responding, but not both) with respect to the object which instantiates the template.

- An operation interface signature comprises a set of announcement and interrogation signatures as appropriate, one for each operation type in the interface, together with an indication of causality (client or server, but not both) for the interface as a whole, with respect to the object which instantiates the template. Each announcement signature is an action template containing both the name of the invocation and the number, names and types of its parameters. Each interrogation signature comprises an action template with the following elements : the name of the invocation; the number, names and types of its parameters, a finite, non-empty set of action templates, one for each possible termination type of the invocation, each containing both the name of the termination and the number, names and types of its parameters.

- A stream interface comprises a finite set of action templates, one for each flow type in the stream interface. Each action template for a flow contains the name of the flow, the information type of the flow, and an indication of causality for the flow (i.e., producer or consumer but not both) with respect to the object which instantiates the template.

3 Operation Signatures and Action Templates

When trying to formalize these concepts we have met with an issue concerning *Action Templates* and how they are currently used and defined currently. In other work such as [7] discussions have focused on whether an Action Template concept lays on a syntactic level or a semantic one. Here, we do not confront this issue as we attempt to solve the problem on a syntactic level. We think that the difficulty of formalizing *Action Templates* stems from the fact that sometimes, *Interaction Signatures* seem to be kind of*Action Templates*, while other times they comprise a set of *Action Templates*. In fact, *Announcement Signatures* are kind of *Action Templates*. In contrast, *Interrogation Signatures* consist of two kinds of interactions which are Invocations and their associated *Terminations*. Thus, it is not evident whether *Operation Signatures* are kind of *Action Templates* or comprise *Action Templates* and it is difficult to merge these two faces of *Operation Signatures* in order to formalize them in one blow.

Furthermore, *Invocations* and *Terminations* seem to be kinds of Action Templates. However, the definition of *Interrogation Signatures* above is a little bit ambiguous. Indeed, *Interrogation Signatures* are defined as comprising Actions Templates (the Invocations) which themselves (the Invocations) comprise a finite non empty set of *Action Templates* (the terminations). This definition is a bit confusing when trying to formalize *Interrogation Signatures*(Invocations and *Terminations*). To eliminate this ambiguity, we can see this definition from another perspective. In fact, we can look at *Interrogation Signatures* as ones comprising both Invocations and their corresponding *Terminations* which are now linked with an association.

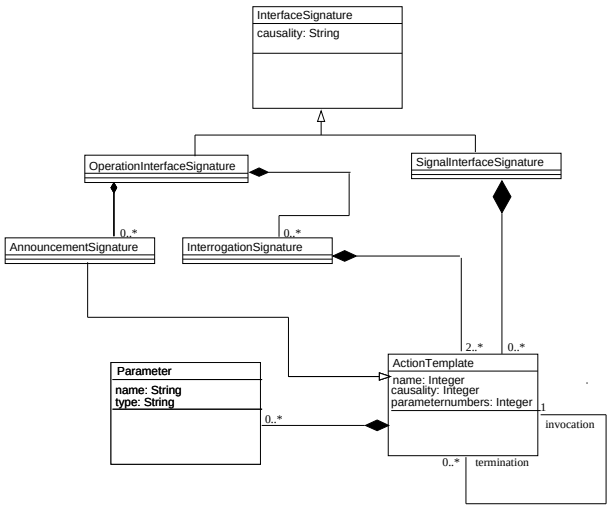

Figure 1: Operation Signatures in terms of Action Templates

So, our proposals to solve this issue is to introduce roles (invocation role, termination role) to *Action Templates* (see figure 1). Having said that, *Announcement Signatures* are now kind of *Action Templates*, while *Interrogation Signatures* comprise *Action Templates*, and that roles introduced to *Action Templates* are there for distinguish between Invocations and their associated *Terminations*.

Finally, to complete our proposal, we must add a constraint which asserts that whenever an Action Template plays the role of an Invocation the set of its corresponding *Terminations* is not empty. we leave this to later in the work.

4 Flow Signatures and Action Templates

Now that we know how to express *Operation Signatures* in terms of *Action Templates*, we turn our attention to Flow signatures, and see how to formalize them in terms of *Action Templates*. We shall see how to integrate *Flow Signatures* with the *Interaction Signatures* model and clarify some inconsistencies by introducing a new term that we call **ParametrizedActionTemplate**.

When we look at how *Flow Signatures* are defined, we can see they are described as kind of *Action Templates*. However, when taking a close look to this, we realize that it is not convenient to derive Flow Signatures directly from already formalized *Action Templates*. In fact, *Flow Signatures* do not involve parameters and their numbers as characteristics which describe them statically. Moreover, *Operation Signatures* are not characterized by the Flow Information Type which is strictly belonging to *Flow Signatures*. Thus, we cannot express both *Flow Signatures* and *Interrogation Signatures* directly in terms of *Action Templates* in one go.

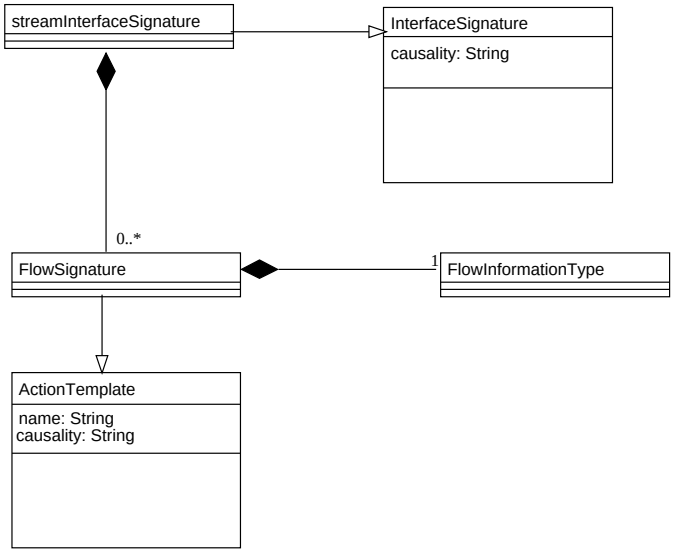

Figure 2: Flow Signatures and Action Templates

We resolve this problem by introducing the term **ParametrizedActionTemplate** as an intermediate level between interrogation Signatures and *Action Templates*(see figure 2). Now, *Operation Signatures* will be derived indirectly from *Action Templates* via **Parameterized Action Template** while Flow Signatures derive its description directly from Action Templates. In doing so, the description of *Action Templates* will change. Indeed, since *Action Templates* are the common descriptive elements between *Operation Signatures* and *Flow Signatures*, an Action Template will have neither parameters, nor their numbers in its description. In fact, these two attributes belong now to the term **ParametrizedActionTemplate** and Action Templates are now expressed in terms of the minimal description consisting of the name and causality of *Action Templates* which is conceptually more convenient.

Having done this, we can join the two models elaborated above into one model that describes all the *Interaction Signatures* in one blow (see figure 3).

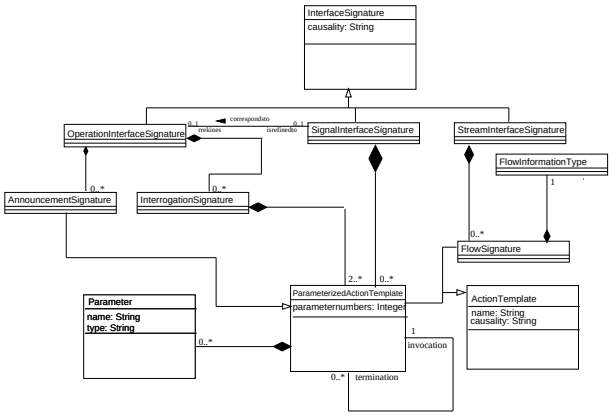

Figure 3: An UML model for Interaction Signatures

As we mentioned above, we have to add a constraint which ensures that the set of *Terminations* associated to an *Invocation* is never empty. But, as *Interrogation Signatures* are related now to **Parametrized Action Templates**, the constraint will belong to the **ParametrizedActionTemplate** term. The constraint written in OCL is as follows:

InterrogationSignature
self.ParameterizedActionTemplate.termination $\rightarrow$ size > 0

This constraint occurs in the context of Interrogation Signature. Now that we have joined all the pieces of the puzzle together, the final model can be seen as a consistent description of Interaction Signatures within the ODP computational viewpoint.

5 Conclusion and perspectives

In our past work [20], we have proposed a UML-Based language for the QoS-aware enterprise specification of ODP systems in which we focused mainly on the specification of QoS from an enterprise viewpoint. When trying to deal with the QoS concepts within the computational viewpoint we have met with the issues as discussed here. So, we decided to clarify some ambiguities relevant to the computational viewpoint. Now we have done that, our work serves as a contribution within the field of formalizing the ODP computational viewpoint, at the same time that it helps us to move forward safely and confidently. We are now dealing with the issue of formalizing QoS concepts from the computational viewpoint.

References:

[1] ISO/IEC, *Basic Reference Model of Open Distributed Processing-Part1: Overview and Guide to Use* ISO/IEC CD 10746-1, 1994.

[2] ISO/IEC, *RM-ODP-Part2: Descriptive Model* ISO/IEC DIS 10746-2, 1994.

[3] ISO/IEC, *RM-ODP-Part3: Perspective Model* ISO/IEC DIS 10746-3, 1994.

[4] R. Romeo et al., *Modelling the ODP Computational Viewpoint with UML 2.0* IEEE International Enterprise Distributed Object Computing Conference, 2005.

[5] D.H.Akehurst et al., *Addressing Computational Viewpoint Design*, Seventh IEEE International EDOC, IEEE Computer Society, 2003

[6] Behzad Bordbar et al, *Using UML to specify QoS constraints in ODP*, Computer Networks Journal pp. 279-304, 2002

[7] R. Romero et al., *Action templates and causalities in the ODP computational viewpoint* WODPEC'04 pp. 23-27. 2004

[8] M.W.A. Steen and al., *Applying the UML to the ODP Enterprise Viewpoint*, http://www.cs.ukc.ac.uk/pubs/1999/819, 1999.

[9] P.F. Linington et al., *The specification and testing of conformance in ODP systems*, http://citeseer.nj.nec.com/170353.html, 1999.

[10] M. W. A. Steen et al.,*Formalising ODP Enterprise Policies*, IEEE Com. Soc. Press, EDOC'99, 1999.

[11] G. Booch et al., *The Unified Modelling Language Guide* Addison Wesly, 1998.

[12] J. Rumbaugh and al., *The Unified Modelling Language Reference Manual*, Addison Wesly, 1999.

[13] OMG, *UML2.0 OCL Final Specification*, OMG Document ptc/03-10-14, 2003.

[14] pUML group,*The Precise UML* http://www.cs.york.ac.uk/puml

[15] M. Gogolla et al., *State Diagrams in UML- A Formal Semantics Using Graph Transformation*, proceedings of ICSE'98,1998.

[16] K. Lano et al., *Formalising the UML on Structured Temporal Theories*, Conference ECOOP'98, 1998.

[17] R. Breu et al., *Systems Views and Models of UML*, Physical Verlag, 1998.

[18] A Evans et al., *Core Meta-Modelling Semantics of UML: The pUML Approach*, Proceedings of UML'99, pp 140-155, 1999

[19] J-M. Bruel et al.,*Transforming UML Models to Formal Specifications*, UML'98-Beyond The Notation, 1998.

[20] B. El Ouahidi et al., *An UML-based Metalanguage for the QoS-aware Enterprise Specification of Open Distributed System*, PROVE'02, Kluwer Academic Publishers IFIP series, pp. 255-266, 2002.