International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Towards a QoS - Aware ODP Computational Viewpoint

Oussama Mohamed REDA

Algorithms, Networks, Intelligent Systems and Software Engineering (ANISSE) Research team Department of Computer Sciences, Faculty of Sciences of Rabat, Mohammed V University in Rabat, Morocco
o.reda[at]um5r.ac.ma

Abstract: ***QoS characteristics are fundamental to the expression of QoS requirements in ODP system. A QoS characteristic represents an identifiable ad quantifiable aspect of a system, a service, or a resource. Examples of usual temporal QoS characteristics relative to interactions between two objects are: transit delay, jitter and throughput. Specifications of such temporal characteristics require two interaction points. The ODP computational model of interfaces is complex and contains inconsistencies which make specification of QoS on computational entities a difficult task. End-to-end QoS Characteristics can only be defined on primitive (atomic) computational entities. This work proposes a simple model of primitive computational interfaces and interactions which intrinsically support the definition (specification) of end-to-end QoS Characteristics in a computational specification.***

Keywords: ODP, QoS, Computational Viewpoint, Interface, Interaction, Computational specification

1. Introduction

The Reference Model (RM) of Open Distributed Processing (ODP) [1], [2], [3], [4], defines a set of concepts and architecture for the construction of ODP systems in terms of five viewpoints. The computational viewpoint supports three models of interactions, each of which has an associated kind of computational interface: signals and signal interfaces, flows and stream interfaces, operations and operation interfaces.

Qos (Quality of service) specification on interfaces and interactions in the computational viewpoint is difficult. Indeed, RM-ODP states QoS characteristics can be specified only on primitive (signal) computational entities. In order to specify QoS on operations and flows they have to be refined into primitives. The computational viewpoint prescribes rules for those refinements. However, the semantics of conceptual relationships between operations flows and signals are not well defined in the RM-ODP computational view-point.

Works [5], [6], [7], [11] have raised those semantic relationships issues, then proposed in reliable solutions. Based on these works [8] [12] [13] have partially proposed a new conceptual model of interfaces (QoS-capable interfaces) for the specification of QoS in the computational viewpoint. The aim of the current work is the completion of the conceptual model of QoS-capable ODP computational interfaces.

The reminder of the paper is as follows. Section 2, presents the computational model interfaces and interactions as well as how QoS is treated in the computational viewpoint. Section 3 defines concepts related to QoS specification on interactions and interfaces necessary to understand the following sections. Section 4 is the core of this work. It proposes a new conceptual view on flows which is flows quantification. Section 5 shows how flows quantification allows support of Qos specification simply in the computational viewpoint. A conclusion and perspectives end the paper.

2. QoS in the Computational Viewpoint

The computational viewpoint is directly concerned with the distribution of processing but not with the interaction mechanisms that enable distribution to occur. The computational specification decomposes the system into objects performing individual functions and interacting at well defined interfaces.

Interactions between computational objects are essentially asynchronous and can take three forms:

- Operations, that are similar to procedures, and are invoked on designated interfaces;
- Flows, that are abstractions of continuous sequences of data between interfaces;
- Signals, which are elementary atomic interactions.

Operations reflect the client/server paradigm. An operation is an interaction between a client object and a server object which requests (an invocation) the performance of some function by the server. There are two types of operations:

- An interrogation, in which the server returns a response (a termination) to the client request (See Figure 1);
- An announcement, in which there is no response to the client request.

Signals are the lowest level of description of interactions between computational objects. A signal is a pairwise, atomic shared action resulting in one-way communication from an initiating computational object to a responding computational object (in this context responding means accepting the communication). This means:

- That the signal occurs at a defined point in time and, hence, is a point of reference for measurement purposes (e. g. in QOS observations);
- That a failure is identical for, and visible to, all participants.

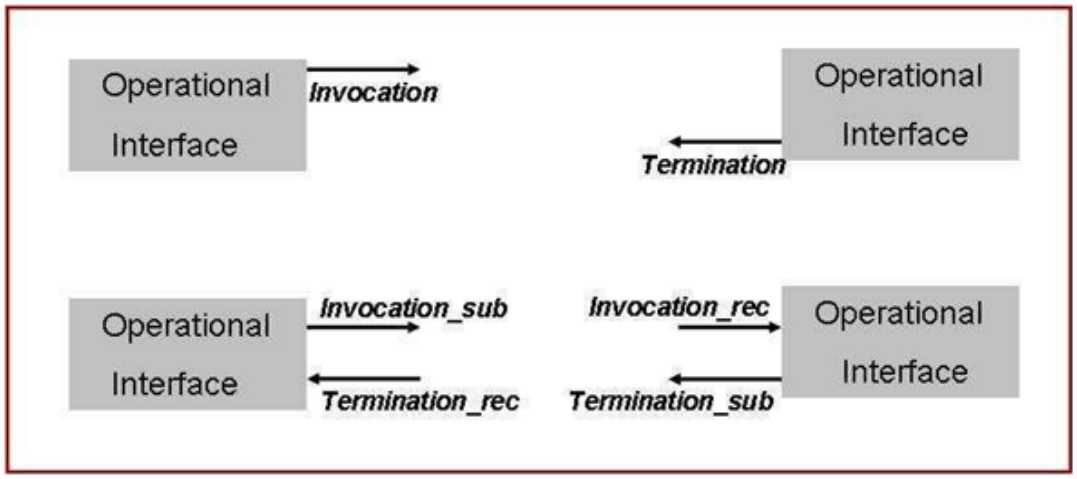

Figure 1: Interactions refinements in the Computational viewpoint of RM-ODP

An operation or a flow can be explained in terms of a combination of several signals. An interrogation, for instance, can be understood as a sequence of signals [3] [8]: invocation emission (by the client object), invocation receipt (by the server object), termination emission (by the server), termination receipt (by the client) (See Figure 1). In contrast, since the exact semantics of flows is not given in the computational model, their map-ping on signals is not defined. Modeling operations or flows in terms of signals becomes necessary in order to define end-to-end QOS characteristics [3] [8], and the operation of multiparty binding and bindings between different kinds of interface (e. g. stream to operation interface bindings).

3. From Parameterized interactions to QoS-capable interfaces

This section provides definitions of some concepts as well as propositions which are necessary to understand the reminder of the paper. From now on, we intently drop the term signature there where it is found it will lighten the definitions of the introduced concepts. Wherever the term interaction or interface is present, it is implicitly followed by the term signature without explicitly mentioning it. We also drop the terms interaction and interface whenever there is no surrounding ambiguity in the context they are dropped in. We let *by* be the contracture of *by and only by*.

Definition 1:

An *Action Template* is defined *by* the name of the action (interaction) and its causality.

Proposition 1:

All Interaction Signatures are Action Templates.

Proof:

See [9], [10].

Proposition 2:

Interaction Signatures but flows are parameterized (i. e contain finite set of parameters as well as their name and numbers).

Proof:

See [9], [10].

Definition 2:

A *Parameterized* interaction signature is an *Action Template* with a finite set of parameters as well as their numbers.

Corollary 1: See [9], [10].

1) Interaction signatures are of two kinds: ***Parameterized* interactions signatures** and flow interactions signatures.
2) Operation Interfaces signatures and Signal Interfaces signatures are composed *by* ***Parameterized* interactions signatures**.
3) A stream interface signature is composed *by* a set of flowing interactions signatures.

Definition 3:

See [9], [10].

A ***Functional*** Interface Signature is an interface signature composed *by* ***Parameterized*** **Interaction signatures**. **Corollary 2:** See [9], [10].

Interface Signatures are of two kinds, namely; ***Functional*** Interface Signature and Stream Interface Signature.

Definition 4:

A ***QoS-labeled interaction*** is a *Primitive Parameterized* interaction [12], [13], [15].

Definition 5:

A ***QoS-capable interface*** is a *Functional* interface signature composed *by* ***QoS-labeled*** interactions [12], [13], [15].

Definition 6:

An ***outgoing*** interaction is a *QoS-labeled* interaction going from an *client QoS-capable interface* out to a *server QoS-capable interface* or vice-versa [12], [13], [15].

Definition 7:

An ***incoming*** interaction is a *QoS-labeled interaction* coming from a *server QoS-capable interface* into an *client QoS-capable interface* or vice-versa [12], [13], [15].

4. Flow quantification for computational QoS specification

Quantification of flowing interactions

In order to complete the conceptual model of QoS-Capable computational interfaces, we need to model flows as primitives. Flows can be used to model, for example, the flow of audio or video information in a multimedia application or in voice-based telecommunication services, or the continuous flow of periodic sensor readings in a process control application. Modeling operations or flows in terms of primitives becomes necessary in order to define end-to-end QOS characteristics and the operation of multiparty binding and bindings between different kinds of interface (e. g. stream to functional interface bindings) [3].

In the previous section, we have defined primitives as incoming and outgoing inter-actions. Thus modeling flows in terms of primitives comes down to modeling flows in terms of both incoming and outgoing interactions. However, we have seen in the previous sections that incoming and outgoing interactions are parameterized interactions (discrete interactions). In contrast, flows are defined in the computational model as continuous sequences of interactions (continuous flow of data). Indeed, the computational language defines a flow by two characteristics, namely its name and its type, which specifies the nature and format of data exchanged. Thus, an ODP flow defines no parameters in the computational language.

The exact semantics of flows is not given in the computational model of RM-ODP and their mapping on primitives is not defined. In order to model flows in terms of primitive parameterized interactions (incoming and outgoing interactions) we need to "parameterize" flows. «Parmeterizing» flows come down to transform it to a parameterized interaction (an interaction with parameters).

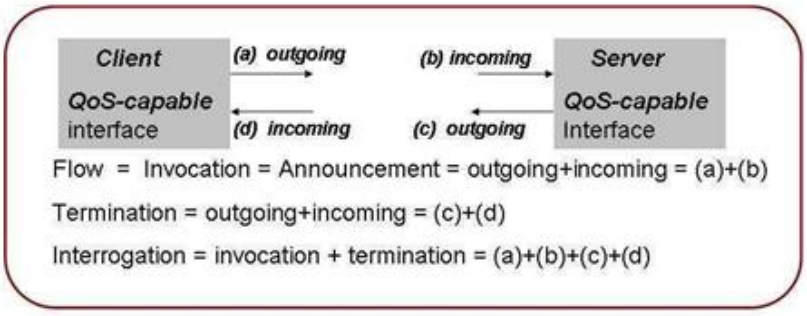

Figure 2: Conceptual model of QoS-capable interfaces

In order to parameterize flows we refine flows into a set of subflows (fragments). Each subflow constitutes a parameter on its own. An analogue example of this concept is the fragmentation processed by the transport layer in a data network. For instance, TCP/IP networks do fragment data received from higher layers (application layers) in order to inject (data) into the IP layer. Thus, even the data is a continuum from the higher layers point of view, it is no more viewed such as such in the IP layer. An example of this is are IP telephony networks which consider voice no more as continuous flows of bits but rather as a set of data fragments transmitted from a router to another.

This is the same concept we introduce in the computational language so as that flows do no more be conceived such as continuous sequences of interactions but rather as a set of huge data quantities. Since in the current work we are only considering flows in the computational viewpoint, the way to parameterize (fragment) flows is not our concern since this must be treated in the engineering viewpoint.

Now that we have parameterized flows (flows parameters are the fragments pieces of flows) they can be considered as parameterized interactions. Since they are parameterized interactions they can be defined as functional interfaces (definition 3 in previous section). It is shown in [] that a functional interface composed by Parameterized interactions can be redefined by primitive parameterized interactions (PIS). Thus, from definition 4 in previous section we can consider a flow as PIS. and consequently as QoS-labeled inter-action. From definition 5 in previous section we can consider stream interfaces as QoS-capable interfaces. Consequently, computational objects can interact now only through QoS-capable interfaces and thus, QoS characteristics can be specified on every kind of computational interactions including "flows" (See Figure 1).

Definability of QoS characteristics on computational entities

Interactions between objects can be modeled implicitly or explicitly through a binding object to which are attributed a set of QoS characteristics. An interaction not occurring through a binding object can be considered as occurring at single location (interface). This limits the set of QoS requirements on an interaction (for example, a request response service) to declaration referring to a single point. Thus, throughput, integrity and turn-around trip delay statements can be defined. On the other hand, QoS statements relative to relatives to transit delay or jitter require the knowledge of two interaction points (two interfaces). Whenever a QoS statement on QoS characteristic related to two interaction points (interfaces) must be specified, the interfaces involved in the interaction must be primitives [3]. Thus, computational QoS characteristics can only be defined for atomic primitive computational entities.

A computational operational interface has a partial of the interaction it is involved in. Indeed, a client interface invoking an operation from a server interface never knows the instant of arrival of the request of the server [3] [14]. Similarly, a server interface cannot know the instant of arrival of the response of the server (termination) to the request of the client. Consequently, the interaction model of an operational interface can specify QoS statements related to turn-around delay of interaction but cannot specify QoS requirements on transit delay or jitter since it needs the knowledge of two interaction points.

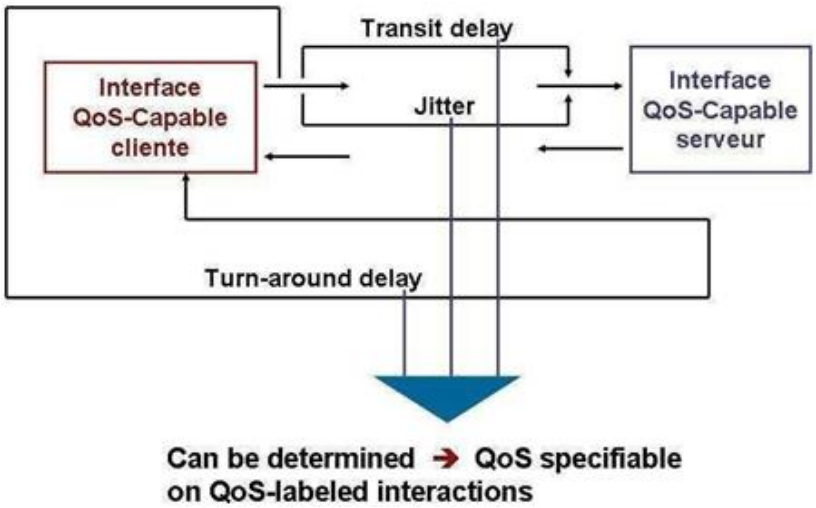

Figure 3: Temporal QoS characteristics in the computational viewpoint

In contrast, QoS statements (transit delay, throughput, jitter, etc.) relative to two interaction points can be specified in the computational QoS-capable interface model. For example transit delay and jitter of an interaction can be measured by the two given inter-action points ((a) and (b)) or ((c) and (d)) (See Figure 3). Similarly, specification of QoS requirements on internal time processing of a QoS capable interface is possible by the two given interaction points (b) and (c). Turn-around trip delay can be measured between (a) and (d).

5. Conclusions & Perspectives

This work proposes a simple model of primitive computational interfaces (QoS-capable interfaces) and interactions (QoS-labeled interactions) which intrinsically supports the definition (specification) of end-to-end QoS Characteristics in a computational specification. The work provides a conceptual framework for a QoS-aware computational viewpoint. In order to provide interoperability in ODP systems, a computational specification is by three structuring rules: Typing rules, interaction rules and binding rules [1] [3].

No particular formal description and specification techniques for the specification of ODP systems have been prescribed by RM-ODP to be used. Recently, UML4ODP *FDIS (Use of UML for ODP systems specification; Final Draft International Standard) [*18] became the framework of choice in industry for ODP systems specification using UML 2.0 [16]. In [11] [12] [15] [10] [14] we enhanced the computational metamodel of UML4ODP with QoS concepts and specified them using UML/OCL 2.0 [16] [17] to specify typing and interaction rules. However, since the QoS-capable model proposed in [11] [12] [15] [10] is only a partial model (does nit take into account QoS characteristics specification on flows), the QoS-capable model we propose in the current work provides a basis on which we can fully specify a QoS-aware computational metamodel in UML4ODP.

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 CD 10746-2*, 1994.
[3] ISO/IEC, RM-ODP-Part3: Perspective Model, *ISO/IEC DIS 10746-3*, 1994.
[4] ISO/IEC, RM-ODP-Part4: Architectural semantics, *ISO/IEC DIS 10746-4*, 1994.

[5] OUSSAMA REDA, BOUABID EL OUAHIDI, DANIEL BOURGET, Interaction Signatures and Action Templates in the ODP Computational Viewpoint, *EDOC Workshops 2006: 38, 10th IEEE International Enterprise Distributed Object Computing Conference EDOC*, 2006.
[6] OUSSAMA REDA, BOUABID EL OUAHIDI, DANIEL BOURGET, Resolving the ODP Computational Viewpoint Interaction Signatures in Terms of Action Templates using UML, *IC-TIS'O7: Information and Communication Technologies International Symposium, April 3-5*, 2007.
[7] OUSSAMA REDA, BOUABID EL OUAHIDI, DANIEL BOURGET, Interaction signatures and Action Templates in The ODP Computational Viewpoint, *Proceedings of the 6th WSEAS International SEPADS'07, Corfu Greece, Feb 16-19*, 2007.
[8] OUSSAMA REDA, BOUABID EL OUAHIDI, DANIEL BOURGET, Towards a Refinement of the Open Distributed Systems Interactions Signatures, *WSEAS transactions on communications, vol.6, pp.601-607*, Apr 2007.
[9] OUSSAMA REDA, BOUABID EL OUAHIDI, DANIEL BOURGET, On UML Modeling of Computational Interfaces & Interactions in the UML4ODP Computational Language, *In Proceedings of the 12th WSEAS International multiconference, Advances in computers, CSCC'08, Crete Island*, July 23-25, Greece, 2008.
[10] OUSSAMA REDA, BOUABID EL OUAHIDI, DANIEL BOURGET, UML4ODP: OCL 2.0 Constraints Specification & UML Modeling of Interfaces in the Computational Metamodel, *WSEAS Transactions on Computers international Journal*, February 20, 2009.
[11] OUSSAMA REDA, BOUABID EL OUAHIDI, DANIEL BOURGET, OCL 2.0 Constraints Specification On Computational Interfaces of ODP Applications, *Proceedings of CARI'08 (Africain Conference on Research in Computer Science and Applied Mathematics*, CARI'08, 2008.
[12] OUSSAMA REDA, BOUABID EL OUAHIDI, DANIEL BOURGET, Embedded QoS aspects in the UML4ODP Computational Metamodel, *International Conference on Multimedia Computing and Systems*, ICMCS'09, 2009.
[13] OUSSAMA REDA, BOUABID EL OUAHIDI, DANIEL BOURGET, Towards QoS-aware ODP Computational interfaces, *The 7th ACS/IEEE International Conference on Computer Systems and Applications*, AICCSA 2009.
[14] OUSSAMA REDA, Méta Modélisation UML des traitements distribuées à support de la QoS dans UML4ODP, *Thése de doctorat, Faculté des sciences de Rabat*, 11 July 2009.
[15] OUSSAMA REDA, BOUABID EL OUAHIDI, DANIEL BOURGET, Typing rules specification on ODP QoS-capablec Computational interfaces, *ARIMA Journal, CARI'08 special issue*, ARIMA 2009.
[16] OMG, UML 2.0 Superstructure Specification, *OMG document formal/05-07-04*, 2005.
[17] OMG, UML 2.0 OCL Final Specification, *OMG Document ptc/03-10-14*, 2003.
[18] ISO/IEC, ITU-T Recommendation X.906 | ISO/IEC 19793, Use of UML for ODP system specifications, *SC 7/WG19 and ITU-T*, 2007