QOS FORMAL SPECIFICATION OF ENGINEERING LANGUAGE

[1]BALOUKI ABDESSAMAD ,[2]BALOUKI YOUSSEF , [3]BOUHDADI MOHAMED ,[4]EL HAJJI SAID

Dep. of Mathematics and Computer Science, University Mohammed V Faculty of Sciences, Rabat Morocco

ABSTRACT

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. The Reference Model for Open Distributed Processing (RM-ODP) provides a framework within which support of distribution, inter-working and portability can be integrated. It defines: an object model, architectural concepts and architecture for the development of ODP systems in terms of five viewpoints. Each viewpoint language defines concepts and rules for specifying ODP systems from the corresponding viewpoint. These viewpoints include an Engineering viewpoint, which focuses on the concepts for describing the infrastructure required to support selective distribution transparent interactions between objects. QoS-ODP management is now an important research topic for many universities, institutes and industrial organizations. Several approaches are being used in the field of the formalization of QoS-ODP.

The use of formal methods in the design process of ODP systems is explicitly required. An important point to take into account is the incorporation of the many proofs which have to be performed in order to be sure that the final system will be indeed ”correct by construction”. The Event-B method is being defined as a formal notation. In this paper, we explore the benefits provided by using the proof construction approach to define the protocol of negotiating QoS requirements When engineering objects in different clusters interact. In this context, we investigate the support for the specification of Quality of Service (QoS) in Event-B when modelling open distributed System in the Engineering viewpoint.

Keywords: *RM-ODP, Engineering Language, Object Model, Qos Requirements, Event B.*

1. INTRODUCTION

The rapid growth of distributed processing has led to a need for coordinating framework for the standardization of Open Distributed Processing (ODP). The Reference Model for Open Distributed Processing (RM-ODP) [1-4] provides a framework within which support of distribution, networking and portability can be integrated. It defines a framework comprising five viewpoints, viewpoint language, ODP functions and ODP transparencies. The five viewpoints, called enterprise, information, computational, engineering and technology provide a basis for the specification of ODP systems.
In this paper we treat the need of formal notation of ODP viewpoint languages. The languages Z, SDL, LOTOS, and Esterel are used in RM-ODP architectural semantics part [4] for the specification of ODP concepts. However, no formal method is likely to be suitable for specifying every aspect of an ODP system.
Elsewhere, we used the meta-modelling approach [8] [9] to define syntax of a sub-language for the ODP QoS-aware Engineering viewpoint specifications. We defined a UML [12] [13]/OCL meta-model semantics for structural constraints on ODP Engineering language [10]. We also used the same met-modelling and denotational approaches for behavioral concepts in the foundations part and in the Engineering language [11] [14].
Furthermore, for modelling ODP systems correctly by construction, the current testing techniques [15] [16] are not widely accepted and especially for the Engineering viewpoint specifications. In this paper, we use the event-B formalism as our formal framework for developing distributed systems.
Event B is a method with tool support for applying systems in the B method. Hence we can benefit

from the useful formalism for reasoning about distributed systems given by refinement techniques and from the tool support in B. [17] [18] [19] [20]
In this context, we developed the QoS negotiation process using manager function, with event B. Thus was performed in a stepwise manner from abstract specification to concrete implementation using superposition refinements. The correctness of each step is proved in order to achieve a reliable system. The tools assist the development process by generating the proof obligations needed. These proof obligations can then be proved with the automatic or the interactive proover of the tool. The Rodin Platform for Event-B provides effective support for refinement and mathematical proof. [21] [22]
The paper is organized as follows. Section 2 introduces RM-ODP. Section 3 describes and specifies the process of negotiation QoS in ODP Engineering view point by using trader function. In Section 4, we use event B as refinement support to specify this process of negotiation. Section 5 presents the Rodin platform as tool of proving initial and refinement models. In section 6 we describe related works. A conclusion ends the paper.

2 ENGINEERING VIEWPOINT

2.1 ENGINEERING LANGUAGE

An engineering specification includes the definition of mechanisms and functions required to support distributed interaction between objects in an ODP system. It defines concepts for describing the infrastructure required to support selective distribution transparent interactions between objects, and rules for structuring communication channels between objects and for structuring systems for the purposes of resource management. The engineering viewpoint describes the distribution of processing performed by the system to manage the information and provide the functionality.

Thus the computational viewpoint is concerned with when and why objects interact, while the engineering viewpoint is concerned with how they interact. In the engineering language, the main concern is the support of interactions between computational objects. The concepts and rules are sufficient to enable specification of internal interfaces within the infrastructure, enabling the definition of distinct conformance points for different transparencies, and the possibility of standardization of a generic infrastructure into which standardized transparency modules can be placed.

The engineering language is used to define a model for distributed systems infrastructure. The engineering viewpoint is not concerned with the semantics of the ODP application, except to determine its requirements for distribution and distribution transparency.

2.2 ENGINEERING STRUCTURES

The fundamental entities described in the engineering viewpoint are objects and channels.
Objects in the engineering viewpoint can be divided into two categories—basic engineering objects (corresponding to objects in the computational specification) and infrastructure objects (a protocol object). A channel corresponds to a binding or binding object in the computational specification.
The engineering language deals with the basic engineering objects and with various other engineering objects which support them. It relates these objects to the available system resources by identifying a nested series of groupings. The basic units of structure are:

• cluster : A configuration of basic engineering objects forming a single unit for the purposes of deactivation, checkpointing, reactivation, recovery and migration.

• cluster manger : A engineering object that manages the basic engineering objects in a cluster.

• capsule : A configuration of engineering objects forming a single unit for the purpose of encapsulation of processing and storage.

• cluster manager : A engineering object that manages the engineering objects in a capsule.

• nucleus object : an (extended) operating system supporting ODP.

• node : A configuration of engineering objects forming a single unit for the purpose of location in space, and that embodies a set of processing, storage and communication functions (a computer system) .

When engineering objects in different clusters interact, there is a need for a good deal of supporting mechanism. Even if the objects are currently within the same capsule or node, mechanisms are needed to cope with the possibility of one or other of them, terminating, failing or moving elsewhere. The set of mechanisms needed to do this constitute a channel, which is made up of a number of interacting engineering objects. A channel contains the transparency functions required by the basic engineering objects, as specified in the environment contracts in the computational specification. The engineering objects within a channel are divided into three types :

- Stubs are concerned with the information conveyed in an interaction ;

- Binders are used when application semantics are not required; they merely transport the messages. Binders are responsible for managing the binding between the basic engineering objects;

- Protocol objects interact via a communications interface; this models networking.

3 QOS IN THE ENGINEERING SPECIFICATION

A generic framework has been refined in order to be used in two particular areas: communications based on architectures compliant with the OSI reference model and distributed object-based applications compliant with the RM-ODP standards. This last particularization has been adopted as the conceptual base for the scope of this paper.

One of the items identified in the work developed by ISO/ITU-T regarding QoS [5] [6] [7] in ODP is the need for a QoS language capable of representing all the QoS information related to all viewpoints specification ODP system. This is the concrete QoS problem this paper focuses on. It presents a QoS language that is compliant with the QoS concepts of the ISO/ITU-T QoS framework and that can be used for the application-level QoS Engineering specification of a distributed object-based application. This QoS language is expressed by event B in order to take advantage of the support tools such as Rodin platform.[21][22]

The QoS statements in the Engineering specification are those that relate to objectives and responsibilities of the ODP system in its environment. In general, these statements express QoS requirements, which are taken to include requirements on the system from the outside world (its user requirements), as well as the guarantees or claims its designers make in order to meet the user requirements.

QoS requirements are associated with Engineering objectives and responsibilities. These will correspond to requirements expressed on the Engineering objects and their interactions

3.1 The QoS Engineering object model

We illustrate how manager function is used to specify QoS when modelling an open distributed system in the Engineering viewpoint. We investigate end-to-end quality of service (QoS) and highlight that QoS provision has multiple facets and requires complex agreements between Engineering objects, Cluster, node and channel . The Quality of service may be specified in a contract or measured and reported after the event (Fig 1).

Refer to Fig.1. Modelling QoS Activity in Engineering viewpoint.

In this QoS Engineering object model, QoS management activities are driven by a manager object that is responsible to obey the system constraints that are in force on objects interactions while filling roles for responding user requirements QoS management would be used as the following stages of an activity:

• A priori the QoS requirements may be built into the system configuration by system design;
• Before initiation the QoS requirements can be conveyed to a manager before an activity is initiated
• At the initiating the activity the QoS requirements can be negotiated between a serverchannel and the manager
• During the activity the QoS requirements may change during the period of the activity due to changed requirements, detected performance loss, explicit indications from the server objects or explicit indication from one or more object clients;
• After the activity, possibly to carry out trend analysis, contract analysis, performance monitoring, etc.

In an Engineering specification, an interaction is defined when it's necessary to define the objectives of the Engineering objects interacting. Such QoS definition include definition of: The Engineering objects interacting, The purpose of the interaction and ODP system functions (migration, recovery, cloning, reactivation).

3.2 Negotiation QoS in client-to-server communication to reach QoS agreement

The concept of QoS negotiation between Engineering objects includes the trader object between them. In general, the mechanisms of three-party negotiation are used, including client, server and a trader.

We define two negotiation mechanisms between three parties:

• The first mechanism uses a single parameter and allows the negotiation from a proposed maximum. In this paper we focus on this mechanism of negotiation.

• The second mechanism allows the parties to specify the ranges in which they are able to operate and they can agree on a limit, a value or a threshold within a range. (Bounded negotiation).

3.3 Bounded negotiation

1) The client user specifies a desired operating range, providing a lower limit L and an upper limit U, where L<= U.

2) The server could refuse the request if it knows that cannot satisfy the user. If the server does not refuse the request but cannot operate over the full range proposed by client, it could determine a new value U ' for the upper limit, which is worse than the proposed value U, L<= U' <= U (the server could also choose to work internally to a higher quality but does not report this fact to the trader).

The server does not alter the value of the lower limit L. The new upper limit U ' and lower limit are provided to traders.

3) The trader could refuse the request, if accepted, it could select a value V belonging to range defined by L e U ' , L<= V<= U ' . The value V is returned to the Manager.

4) The server leaves the V value unchanged.

5) The V value is selected and returned to the ClientChannel, it is the value of agreement.

This mechanism is illustrated in **Figure 2.**

The system is negotiating QoS between Engineering objects	FUN-1

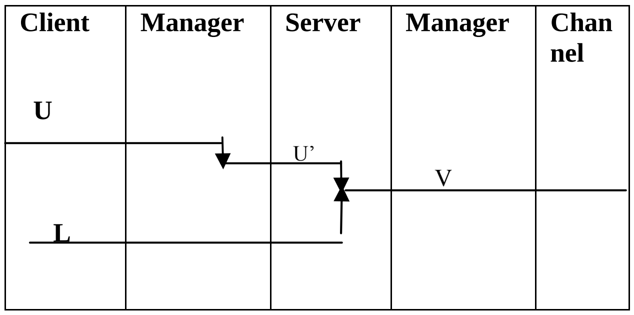

Fig.2. QoS bounded parameter negotiation.

4 SPECIFYING QOS NEGOTIATION WITH EVENT B

4.1 Informal presentation of the QoS single parameter negotiation protocol

The QoS single parameter negotiation can be represented in the diagram of **figure 3** where the are supposed to represent the various phases we have just described as indicated by the arrows:

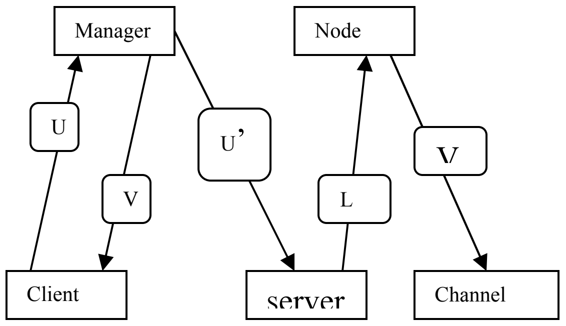

Fig. 3. Schematic view of the QoS negotiation protocol

We have described the normal behaviour of the protocol, where channel and manager Engineering object negotiate successfully a QoS value. We shall also describe below a degraded behaviour, where the two objects fail to achieve this QoS value.

4.2 Requirement document

The requirement document which we propose now is far less precise than the previous informal explanations we have given. It does not propose an implementation. It essentially consists in explaining what kind of believe each Engineering object may have at the end of the protocol:

the QoS is negotiated by the manager of the channel.:

The final QoS value is published by channel Manager	FUN-2

We explain what a required QoS:

The first QoS value is required by client object.	FUN-3
The final QoS value is approved by binder object.	FUN-4

The both client and server may believe at the end of the negotiation protocol:

Client and server might either believe if the QoS value is conformably negotiated or failed	FUN-5

The binder relate of both the client and the server:

When the QoS value is published by manager, the binder and server believe that the QoS value has been negotiated successively. Otherwise, they believe that negotiation failed.	FUN-6

The case where the server refuses the requested QoS.:

If the binder refuse QoS value, the negotiation is aborted	FUN-7

4.3 Refinement strategy

Before engaging in development of such a system, it is profitable to clear identify what our design strategy will be. This is done by listing the order in which we are going to take account of the various requirements we proposed in the requirement document of the previous section. Here is our strategy for constructing the QoS bounded negotiation protocol:

- We started with very simple model allowing us to taking account of requirements (FUN-2, FUN-3) concerned with the maximum and minimum value of QoS .
- In the first refinement, we take account of requirements FUN-4 to FUN-6 telling us that the QoS value is negotiated by the client and the server and approved by the manager.
- In the last refinements, we introduce the channel between client and server. A channel is a configuration of stubs, binders, protocol objects and interceptors providing a binding between a set of interfaces basic engineering objects, through which interaction can occur.

4.3.1 Initial model

The first model contains a partial specification of the QoS bounded parameter negotiation protocol. It deals with requirements FUN-1, FUN-2 and FUN-3. The protocol is executed in one shot.

4.3.1.1 Formalizing the state

The state of our model is made of to parts : the static part and the dynamic part. The static part contains the definition and axioms associated with some constants, the dynamic part contains the variables which are modified as the system evolves.

Constants: Uchannel_max Lchannel_min	**Axm0_1**:Uchannel_max€ IR **Axm0_2** :Lchannel_min € IR

The negotiated QoS value is variable typed in invariants inv0_1 , inv0_2 and inv0_3.

variables**:** **Uneg**	**Inv0_1** : Vnegoc € IR Inv0_2 **: Uneg <= Uchannel_max** Inv0_3 **: Uneg > Lchannel_min**

4.3.1.2 Formalizing the events

At this stage, we can observe two transitions, which we shall call events in the sequel. They correspond to QoS value proposed and even accepted or refused.

The initial value of **Uneg** is **U**. the final value of the protocol must be less than **U** and greater than **L** .

Init : Uneg := U Uchannel_max > 0 Uchannel_min > 0	**brp** Ufinal <= U Ufinal > L

4.3.2 First refinement

We are now going to proceed with a refinement of initial model. A refinement is more precise model than initial one. It is more precise but is should not contradict the initial model. In this refinement, we introduce the channel. Thus the value of QoS required by the channel to ensure Liaison reliably between engineering objects.

4.3.2.1 Refining the state

In this first refinement, we introduce the concept of status. It is made of three distinct elements: working, success and failure as shown below.

Constants: working success failure	Axm1_1 : working ≠ success Axm1_2 : success ≠ failure Axm1_3 : working ≠ failure

We replace the abstract variable V by a concrete one V_channel indicated in invariant inv1_1. We introduce the status v_channel and v_sever of the channel and server one respectively. Such variables are member of the STATUS as indicated implicitly in invariants inv1_2, inv1_3 and inv1_4.

Variables: V_server v_channel v_client	**Inv1_1** : 0 < Vnegoc <= Vchannel_max **Inv1_2** : v_server = success ⬄ v_channel >= V_server **Inv1_3** : v_client = success ==> v_client <= v_channel and v_server >= v_client **Inv1_4**: v_server = failure ⬄ v_channel < v_client or v_server < v_client

Requirement FUN-6 is formalized in inv1_4 where it said that the client accept when the server does.

4.3.2.2 The events

In this refinements, we introduce many new events : Client_snd, Server_failure, Server_accept and Channel_refuse. The four of them clearly refine skip (since their action are concerned by new variables), and also maintain invariants inv1_2 and inv1_3 regarding the status of the channel and server.

Client_snd
When
V_client = working
v_channel = accept
v_server = accept
Then
V_client := accept
End
	Client_ failure
	When
	v_channel = propose
	v_server = failure
	Then
	V_client := failure
	End

Server_accept **When** v_channel = accept **Then** v_server := accept **End**	Server_failure **When** V_client = working v_channel = failure **Then** v_server := failure **End**

Event brp defined below, is also a new event refining *SKIP*, This is clearly a convenient abstraction but not a final implementation. In fact, this direct access will be removed in the next refinement.

Init :
Vnegoc := P
Vserver_max > 0
Vchannel_max > 0
V_Client:=required
v_channel := prescribed
v_server := proposed
	brp
	When
	V_Client ≠ working
	v_server ≠ working
	v_channel ≠ working
	Then
	skip
	End

The refinement of event brp must refine its abstraction, which is a non-deterministic event.

4.3.3 Second refinement

In this refinement, the binder will enter into the scene by cooperating with client and server objects in order to negotiate the QoS value. In fact, the client will not access any more directly the server value as was the case of the previous refinement, this will be done by the binder. We then introduce this binder which is situated between client and server.

4.3.3.1 The state

The state is first enlarged with two variables to be used by binder Vbinder. A Boolean variable publish indicated implicitly by inv2_1, and a real variable Vbinder as indicated in inv2_2, inv2_3 and inv2_4.

variables: required Vbinder	**Inv2_1:** required=true ==> Vbinder <= Vchannel_max **Inv2_2** : Vbinder > 0 **Inv2_3** : Vbinder <= Vserver **Inv2_3** : Vbinder <= Vclient

4.3.3.2 The events

The initialization event is extended in a straightforward fashion as indicated below. The Boolean value publish is set to False at the beginning so that the only two events which can be fired are the ones described next.

Init :
Vnegoc := P
Vserver_max > 0
V_server := proposed
V_channel:= prescribed
required := FALSE
	Client_accept
	When
	V_client = required
	Publish = TRUE
	Then
	V_server := accept
	End
		Server_accept
		When
		V_server := proposed
		Publish = TRUE
		Then
		V_client := accept
		End

Client_refuse **When** V_client = required V_client>> v_channel Publish = FALSE **Then** V_client := refuse **End**	Server_refuse **When** V_server = proposed V_server>>v_channel Publish = FALSE **Then** V_server := refuse **End**	Binder_accept **When** V_server~ = v_channel V_client ~ = V_server Publish = FALSE **Then** V_binder := accept **End**

5 CONCLUSION

The Reference Model of Open Distributed Processing (ODP RM) is intended to create an international standard for the design and realization of open distributed systems. The use of formal methods in the design process of ODP systems is explicitly required. The intent of our work is to give some insights and formal reasoning. Theses activities are supposed to be performed before undertaking the effective coding of computer system, so that the system in question will be correct by construction.

We presented our approach for developing open distributed system in Event B. we constructed an ordered sequence of embedded models, where each of them is supposed to be a refinement of the one preceding it in that sequence. This means that a refined, more concrete, model will have more variables than its abstraction. Indeed, these approach are suitable to define and constrain ODP information viewpoint specifications. In parallel, we are applying the same formal method to specify the dynamic behavior of open distributed systems.

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: Prescriptive Model, ‘’ ISO/IEC DIS 10746-3, 1994.

[4]. ISO/IEC, ‘’RM-ODP-Part4: Architectural Semantics, ‘’ ISO/IEC DIS 10746-4, July 1994.

[5]. ISO/IEC TR 13243 – Information technology – Quality of service –Guide to methods and

[6]. Mechanisms (November 1999)

[7]. 6. ITU-T Recommendation G.1000 - Communications quality of service: a framework and Definitions (November 2001)

[8]. ITU-T E.860 Framework of a service level agreement (June 2002)

[9]. M. Bouhdadi et al., ‘’A UML-Based Meta-language for the QoS-aware Enterprise Specification of Open Distributed Systems’’ IFIP Series, Vol 85, Springer, (2002) 255-264.

[10]. Mohamed Bouhdadi and Youssef Balouki. Semantics of Behavioral Concepts for Open Virtual Enterprises’. Series: Lecture Notes in Electrical Engineering, , Vol. 27 .Springer, 2009. p.275-286.

[11]. 10.Y. Balouki, H. Belhaj and al. Event B for ODP Enterprise Behavioral Concepts Specification, Proceedings of the World Congress on Engineering 2009 Vol I, WCE '09, July 1 - 3, 2009, London, U.K., Lecture Notes in Engineering and Computer Science, pp. 784-788, Newswood Limited, 2009

[12]. 11. Youssef Balouki and Mohamed Bouhdadi. ‘Using BPEL for Behavioral Concepts in ODP Enterprise Language’, Virtual Enterprises and Collaborative Networks, IFIP, Vol. 283, pp. 221-232, Springer, 2008

[13]. 12. J. Rumbaugh et al., the Unified Modelling Language, Addison Wesley, 1999.

[14]. 13. B. Rumpe, ‘’A Note on Semantics with an Emphasis on UML, ‘’ Second ECOOP Workshop on Precise Behavioral Semantics, LNCS 1543, Springer, (1998) 167-188.

[15]. 14. Mohamed Bouhdadi and Youssef Balouki and El maati Chabbar, ‘Meta-modelling Syntax and Semantics of Structural Concepts for Open Networked Enterprises’, Lecture Notes in

Computer Science, Vol. 4707, pp. 45-54, Springer, 2007

[16]. 15. Myers, G. The art of Software Testing, John Wiley &Sons, New York, 1979

[17]. 16. Binder, R. Testing Object Oriented Systems. Models, Patterns, and Tools, Addison-Wesley, 1999

[18]. 17. http://www.event-b.org/

[19]. 18. Joochim, T., Snook, C., Poppleton, M. and Gravell, A. (2010) TIMING DIAGRAMS REQUIREMENTS MODELLING USING EVENT-B FORMAL METHODS. In: IASTED International Conference on Software Engineering (SE2010), February 16 – 18, 2010, Innsbruck, Austria

[20]. 19. J.-R. Abrial & S. Hallerstede, Refinement Decomposition and Instantiation of Discrete Models: Application to Event-B, Fundamenta Informaticae, 77(1-2), 2006, 1-28.

[21]. 20. C. Snook & M. Butler, UML-B and Event-B: an integration of languages and tools. Proc. IASTED International Conf. on Software Engineering (SE2008), Innsbruck, Austria, 2008.

[22]. 21. RODIN. Development Environment for Complex Systems (Rodin). 2009. http://rodin.cs.ncl.ac.uk/.

[23]. 22. Jean-Raymond Abrial: A System Development Process with Event-B and the Rodin Platform. ICFEM (2007) 1-3

[24]. 23. ISO/IEC, “ODP Type Repository Function”, ISO/IEC JTC1/SC7 N2057, 1999.

[25]. 24. ISO/IEC, “The ODP Trading Function”, ISO/IEC JTC1/SC21 1995.

[26]. 25. J.-R. Abrial. Tools for Constructing Large Systems (a proposal). In Rigorous Development of Complex Fault-Tolerant Systems. M. Butler, etc. (Eds). LNCS 4157 Springer, 2006

[27]. 26. J.-R. Abrial, M. Butler, S. Hallerstede, L. Voisin. An Open Extensible Tool Environment for Event-B. ICFEM 2006

[28]. 27. M.J. Butler and S. Hallerstede The Rodin Formal Modelling tool. BCS-FACS Christmas 2007 Meeting Formal methods in Industry London, 2007

[29]. 28. J.-R. Abrial, Tutorial - Case study of a complete reactive system in Event-B: A mechanical press controller. Proc. 5th International Symposium on Formal Methods (FM’2008), Turku, Finland, 2008.

[30]. 29. D. Cansell, D. Méry & J. Rehm, Time Constraint Patterns for Event B Development. Proc. Formal Specification and Development in B, 7th International Conf. of B (B 2007), Besancon, France, 2007. 140-154.

[31]. 30. J. Bicarregui, et al, Towards Modelling Obligations in Event-B. Proc. International Conf. of ASM, B and Z Users, London, UK, 2008, 181-194.

[32]. 31. E. Letier & A.V. Lamsweerde, Agent-Based Tactics for Goal-Oriented Requirements Elaboration. Proc. 24th International Conf. on Software Engineering (ICSE’02), Orlando, Florida, USA, 2002, 83-93.

[33]. 32. C. Ponsard & E. Dieul, From Requirements Models to Formal Specifications in B. Proc. International Workshop on Regulations Modelling and their Validation and Verification (REMO2V’06), Universitaires de Namur, Luxemburg , 2006, 249-260.

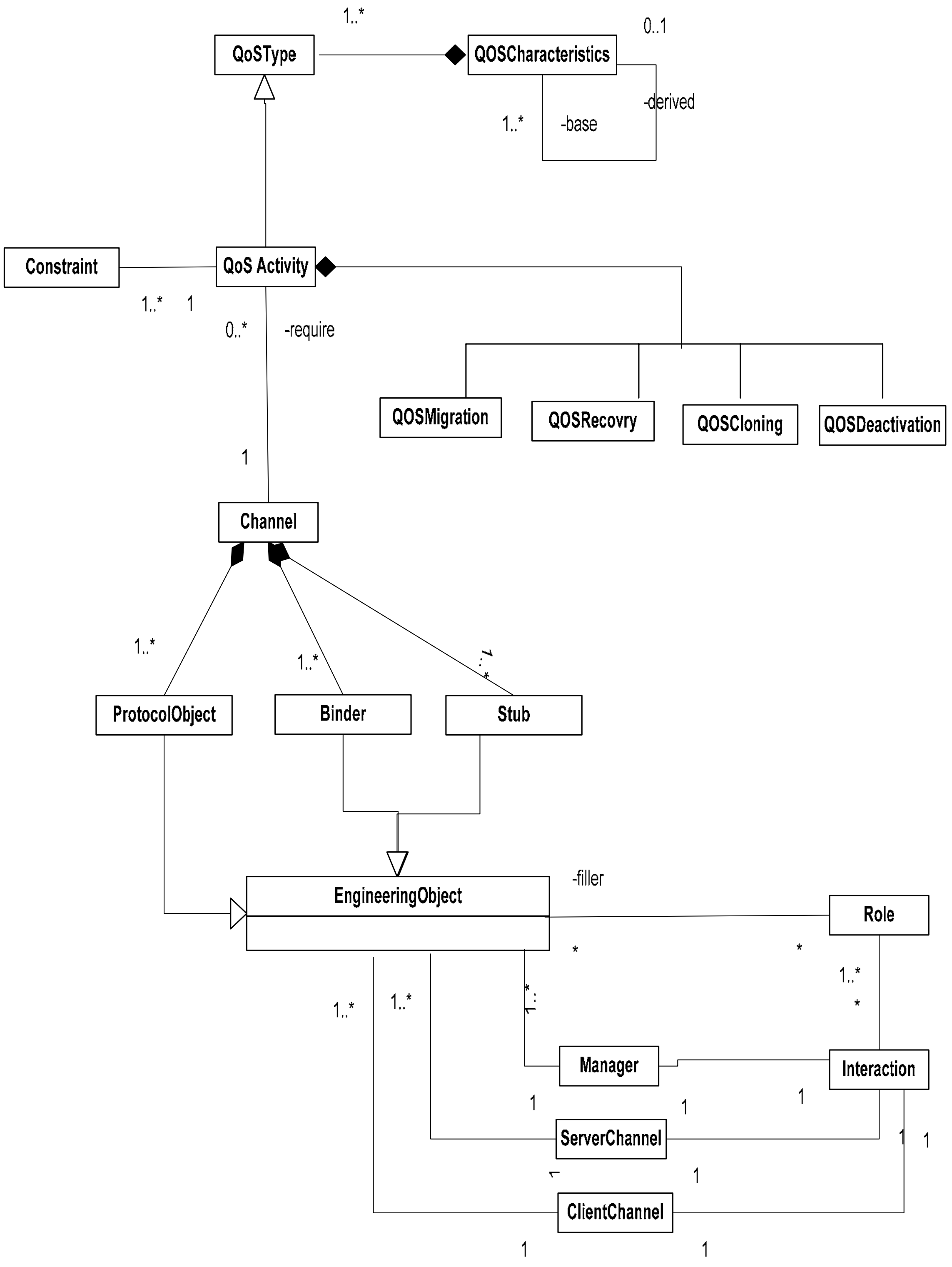

Fig.1. Modelling QoS Activity in Engineering viewpoint