Journal of Theoretical and Applied Information Technology
15 September 2012. Vol. 43 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

THE SPECIFICATION OF QUALITY OF SERVICE IN OPEN DISTRIBUTED PROCESSING: FORMALISM USED IN MECHATRONIC SYSTEM

YOUSSEF BALOUKI, ABDESSAMED BALOUKI

[1]Dep of Mathematics & Computer Science, University HASSAN 1[ER] Morocco

[2]Dep of Mechanical Engineering, University SULTANE MOULAY SLIMANE Morocco

E-mail: [1] balouki.youssef@gmail.com, [2]balouki@gmail.com

ABSTRACT

The paper discusses two approaches for designing mechatronic systems. the first one is based on using Reference Model for open Distributed Processing (RM-ODP), to specify any kind of mechatronic systems, RM-ODP is a reference model in computer science, which provides a co-ordinating framework for the standardization of open distributed processing (ODP), whereas in the second phase , we introduce Event-B method to formalize and verify mechatronics system. We explore the benefits provided by using the proof construction approach to define the protocol of negotiating QoS requirements When Mechatronic components in different clusters interact. In this context, we investigate the support for the specification of Quality of Service (QoS) in Event-B, when modelling mechatronic systems in ODP Engineering viewpoint.

Keywords: *Mechatronic systems, RM-ODP, QoS Requirements, Event B, Rodin platform.*

1. INTRODUCTION

UML (unified modelling language) is widely used in designing complex and reliable computer science. In mechatronic it provides means for capturing system requirements and for the visual modelling and design of systems on a high level of abstraction [37-40]. However, there is no widely agreed approach to the structuring of such specifications. This adds to the cost of adopting the use of UML for Mechatronic systems specification, hampers communication between system developers and makes it difficult to relate or merge system specifications where there is a need for a structuring framework if it is to be managed successfully. The purpose of the Reference Model for Open Distributed Processing (RM-ODP) is to define such a framework. Used The RM-ODP [1-4] in Mechatronic systems 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.

We used the meta-modelling approach [8] [9] to define syntax of a sub-language for the QoS-aware Engineering viewpoint specifications. We defined a UML/OCL [12][13] meta-model semantics for structural constraints on 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 Mechatronic systems correctly by construction, the current testing techniques [15] [16] are not widely accepted and especially for the Mechatronic 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

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

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].

In this work we are presenting one possibility to integrate RM-ODP, to specify any kind of mechatronic systems, In the second section we will introduce the mapping between engineering viewpoint in ODP and Mechatronic engineering. Then we will describe and specifies the process of negotiation QoS in Mechatronic Engineering by using trader function. In the fourth we use event B as refinement support to specify this process of negotiation. Then we present the Rodin platform as tool of proving initial and refinement models. A conclusion ends the paper.

2. MECHATRONIC ENGINEERING VIEWPOINT

2.1 ODP 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. 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.

2.2 From Engineering Language To Mechatronic Construct

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, cluster manger, capsule, nucleus object, and node.

We used the meta-modelling approach to define mapping between engineering language and Mechatronic domains. One way to do this mapping is to find both the functional structure and the implementation structure and then relate the bounded artifacts to each other. Table 1 shows an overview of the mapping from the engineering language concepts to mechatronic artifacts covering the subset of the structures introduced in this paper.

Table 1– engineering language concepts to mechatronic artifacts mapping overview

Engineering Language Concept	Mechatronic Construct
Control	PID connector/sensor/actuator
Node	Calculateur
Capsule	Address space
Cluster	basic objects forming a simple unit
Engineering Objects	Mechatronic objets / Modules d'un programmes
Engineering Interfaces	input capture card// output capture card
nucleus	Operating System
Stubs	connectors
Chanal	Connexion the mechatronic objects
Protocol objet	communications interface/ system bus

3. QOS IN MECHATRONIC ENGINEERING

A generic framework has been refined in order to be used in two particular areas: communications based on architectures compliant with the OSI

Journal of Theoretical and Applied Information Technology
15 September 2012. Vol. 43 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

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 Mechatronic Engineering specification. 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 mechatronic system 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 Mechatronic Object Model

We illustrate how automated trading function is used to specify QoS when modelling mechatronic systems 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 Mechatronic objects, Cluster, calculateur and channel . The Quality of service may be specified in a contract or measured and reported after the event (Fig 1).

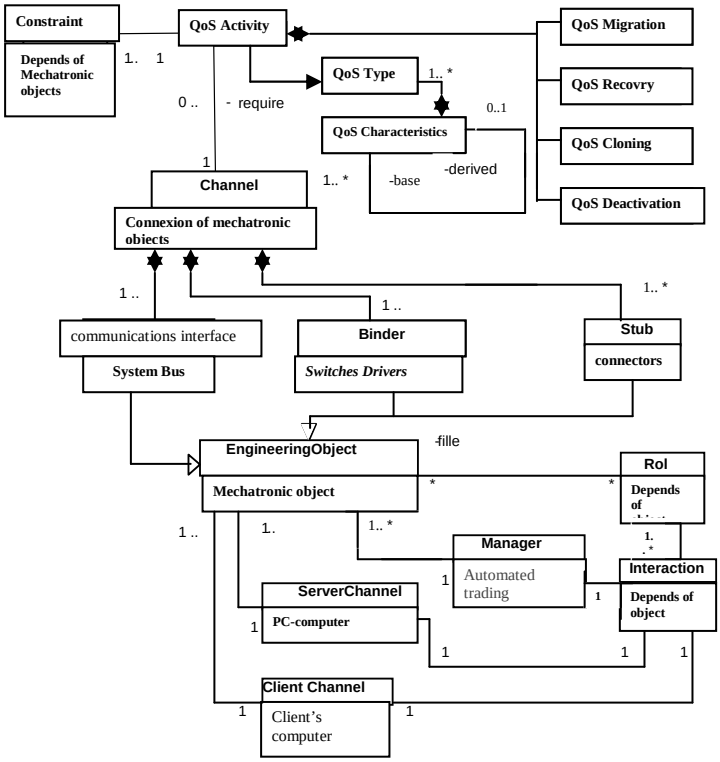

Fig.1: Modelling QoS Activity in Mechatronic Engineering.

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 an automated trading (manager). before an activity is initiated

• At the initiating the activity the QoS requirements can be negotiated between a PC-computer (server channel) and the automated trading (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

Journal of Theoretical and Applied Information Technology
15 September 2012. Vol. 43 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

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 automated trading (Trader) object between them. In general, the mechanisms of three-party negotiation are used, including Client's computer (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's computer (client) user specifies a desired operating range, providing a lower limit L and an upper limit U, where L≤ U.

2) The executive PC-computer (server) could refuse the request if it knows that cannot satisfy the user. If the PC-computer (server) does not refuse the request but cannot operate over the full range proposed by Client's computer (client), it could determine a new value U ' for the upper limit, which is worse than the proposed value U, L≤ U' ≤ U (the Pc-reciever (server) could also choose to work internally to a higher quality but does not report this fact to the trader). The PC-computer (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 and U, 'L≤ V≤ U '. The value V is returned to the Manager.

4) The Pc-reciever (server) leaves the V value unchanged.

5) The V value is selected and returned to the Client channel, it is the value of agreement.This mechanism is illustrated in Figure 2.

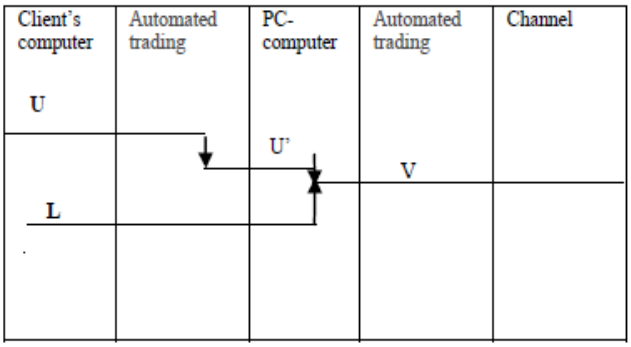

Fig.2. QoS bounded parameter negotiation

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 events (Client_snd, Server_snd, Trader_snd, Client_rcv, Server_rcv, Trader_rcv) are supposed to represent the various phases we have just described as indicated by the arrows:

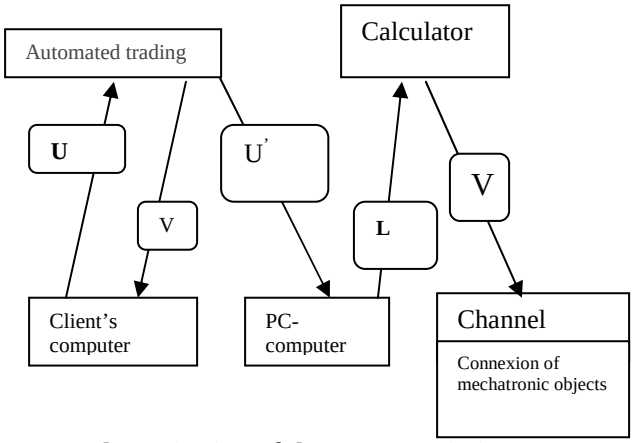

Fig. 3. Schematic view of the QoS negotiation protocol in Mechatronics systems

Protocol we have described the normal behaviour of the protocol, where channel and Automated trading (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.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

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:Listen

1	The system is negotiating QoS between Engineering objects
2	The final QoS value is published by Automated trading (Manager)
3	The first QoS value is required by Client's computer
4	The final QoS value is approved by Switches Drivers binder object.
5	Client's computer and server might either believe if the QoS value is conformably negotiated or failed
6	When the QoS value is published by Automated trading (manager), the Switches Drivers (Binder) and PC-computer (server) believe that the QoS value has been negotiated successively. Otherwise, they believe that negotiation failed.
7	If the binder refuse QoS value, the negotiation is aborted

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 (2,3) concerned with the maximum and minimum value of QoS .

* In the first refinement, we take account of requirements 4 to 6 telling us that the QoS value is negotiated by the Client's computer (client) and the PC-computer (server) and approved by the Automated trading

* In the last refinements, we introduce the channel between Client's computer (client) and PC-computer (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 1, 2 and 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. The negotiated QoS value is variable typed in invariants inv0_1 , inv0_2 and inv0_3.

Constants: Uchannel_max Lchannel_min	**Axm0_1**:Uchannel_max∈ IR **Axm0_2** :Lchannel_min ∈ IR
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.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

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 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.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

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**

6 CONCLUSION

Used The Reference Model for Open Distributed Processing (RM-ODP) in Mechatronic systems provides a framework within which support of distribution, networking and portability can be integrated. We used the meta-modelling approach to define mapping between engineering language and Mechatronic domains. 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 Rodin Platform for Event-B provides effective support for refinement and mathematical proof.

REFRENCES:

[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 Mechanisms (November 1999)

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

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

[8].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.

[9].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.

[10]. Youssef 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

[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

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

[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.

[14]. Mohamed Bouhdadi and Youssef Balouki and El maati Chabbar, ‘Meta-modelling Syntax

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

and Semantics of Structural Concepts for Open Networked Enterprises', Lecture Notes in Computer Science, Vol. 4707, pp. 45-54, Springer, 2007

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

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

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

[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

[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.

[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.

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

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

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

[24]. ISO/IEC, "The ODP Trading Function", ISO/IEC JTC1/SC21 1995.

[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

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

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

[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.

[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.

[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.

[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.

[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.

[33]. Abdessamad Balouki, Balouki Youssef, M.Bouhdadi , S.El Haji- QOS Formal Specification of Engineering JATIT 7vol 22 No2.

[34]. Mrozek Z. (2002a): Computer-aided design of mechatronic systems.— Sci. Fasc., Cracow Univ. of Technol., Series: Electrical and Computer Eng., No. 1, (in Polish).

[35]. Mrozek Z. (2002b): Design of the mechatronic system with help of UML diagrams. — Proc. 3-rd Workshop Robot Motion and Control, Bukowy Dworek, Poland, pp. 243– 248.

[36]. Mrozek Z. (2002c): Methodology of using UML in mechatronic design. — Pomiary Automatyka Kontrola, No. 1, pp. 25– 28, (in Polish).

[37]. ZBIGNIEW MROZEK Computer aided design of mechatronic systems Int. J. Appl. Math. Comput. Sci., 2003, Vol. 13, No. 2, 255–267