Reference Model of Open Distributed Processing Basic Modelling Concepts in Event-B

Abdessamad JARRAR[1], Nawfal FERFRA[2], Taoufiq GADI[1], Youssef BALOUKI[1]
[1] University Hassan First, Faculty of Sciences and Technologies of Settat, Computing, Imaging and Modeling of Complex Systems Laboratory, Morocco
[2] CMR The Moroccan Pension Fund, Rabat, Morocco

Abstract. We present a set of recommendations to help engineers using the Event-B formal methods to specify the basic modelling concepts of the Reference Model of Open Distributed Processing (RM-ODP). This model is developed by the IUT and ISO in order to standarise the development of Open Distributed Processing (ODP). RM-ODP is criticized for inssuficient definition of the basic modeling concepts which limits the applicability of the model. Therefore, the IUT and ISO provide for users a sematic architecture formalization in several formal methods (LOTOS, ACT ONE, SDL-92, Z, and ESTELLE). However, these formal methods are very basic and suffer from the lack of predefined mathematical operators and also some of them are very poor in term of specification techniques. These issues encourage us to develop our recommendations for specifying and developing the ODP using a more sophisticated method called Event-B. This formal methods is very rich in term of predifined mathematical operator and provides sophisticated techniques that can be used during the specification process. Additionaly, Event-B provide a set of verification proofs that highly guarentee the absence of bugs.

Keywords: Open Distributed Processing, RM-ODP, Formal methods, Event-B, architecture semantic formalization.

1 Introduction

Verifying the absence of bugs in Open Distributed Processing systems is a challenging task in the context of software engineering [2]. Therefore, it is highly recommended to use powerful tools and techniques to study these systems. One of the most successful techniques in this domain is formal methods approach. The use of formal methods highly guarantees strong assurance of bugs' absence by means of mathematical modelization and proofs. However, using formal methods is, in general, not an easy task because it requires a very well understanding of the system processing. Therefore, we propose in this paper a set of recommendations in term of a pattern –similar to design patterns- to help engineering to develop systems using the Event-B formal method [1].

Event-B is a formal method that verifies systems' correctness based on theorem proving which is a method-by-method verification technique [3]. This method have been applied to several ODP systems such as Meteor line 14 driverless metro in Paris in which no bug has been detected [4]. Therefore, we chose the Event-B as a formalization and verification method.

This paper contributes to a stepwise facilitating the modelization and verification of ODP systems. We propose a pattern that illuminates how it is possible to model RM-ODP concepts

ICCWCS 2019, April 24-25, Kenitra, Morocco

DOI 10.4108/eai.24-4-2019.2284096

using Event-B. The approach proposed is this paper covers the most important basic modelling concepts and specification concepts.

The rest of the paper is organized as follows. Section 2 gives some background on the fundamental concepts of ODP and formal methods. Section 3 presents the contribution of the paper. Section 4 illuminate the approach proposed in term of a set of recommendations and guides on how to formally model RM-ODP basic concepts in Event-B. Finally, we conclude in section 5.

2 background and literature review

2.1 RM-ODP parts

The rapid growth of distributed systems encourages scientists to develop a coordinating framework for the standardization of ODP system. Therefore, the International Organization for Standardization (ISO), the International Electro-technical Commission (IEC) and the Telecommunication Standardization Sector (ITU-T) joint effort to develop the Reference Model of Open Distributed Processing [12]. This reference Model provides an architecture that supports distribution, interworking and portability. It is made up of four part, these are:

1. Overview: this first part contains motivation, scoping, justifications and explanation about the use of ODP architecture. It contains also guide lines of how the RM-ODP can be interpreted and a categorization of required areas of standardization [11].
2. Foundations: this brief part with only 18 pages contains the foundations of RM-ODP. It contains concepts of categories, basic interpretation concepts, linguistic concepts, modelization concepts, specification concepts, organization concepts, system and object properties, designation concepts, behaviour and management concepts, and principles of conformance to ODP standards [10].
3. Architecture: this part includes the specification of the characteristics required for a distributed processing system to be open. These are the constraints that the open distributed processing (ODP) must conform [9].

Architectural semantics: the recommendation presented in this part is an integral part of the RM-ODP. It contains a formalization of ODP modelling concepts obtained by the interpretation of each concept according to the constructions of the different standardized formal description techniques [6].

2.2 Formal methods

Modeling can be defined as the representation of a real-world system either graphically representation or mathematically. It is a stage before construction any system that illuminate its expected structure. Modeling helps also to study and test some system properties to reduce the risk of failures. For example, by modeling correctly an auto-driving cars system, we are able to prove theoretically that it is impossible for two cars to collide. Modeling can be done by means of graphical representation or mathematical equations. The first one is easy to read and can visualize the system structure better, which means it can be shown to clients. However, it is not very accurate in term of representation and its verification techniques are not powerful. On the other hand, mathematical representations need mathematical knowledge to be read and cannot be shown to most clients. It is also not easy to use due to the required accuracy and the huge number of proofs. Despite all that, it is very powerful for detecting

errors and ensures a strong assurance of bugs' absence. Therefore, we think that the use of formal methods based on mathematical representation is highly required.

In event-B, a system is developed as sequence of models. These models are refined in successive steps by making each model richer in term of details. In other words, we start with a very abstract model called initial model, and then we refine it to get a more concrete models. These models are made up of contexts and machines [7]. Contexts are the static parts of models; they are presented in term of sets, constants and axioms, whereas, machines are the dynamic part of models. In a machine, the dynamic status of the system is described by means of variables; the statuses of the system are constrained by invariants. Invariants are mathematical properties describing the necessary condition that must be preserved during the system life-time. Statuses transitions are described by events, which are a set of actions. Each action changes the value of certain variable. Events may have some necessary conditions to be triggered; these conditions are called guards. The figure below illustrates the development process when using event-B:

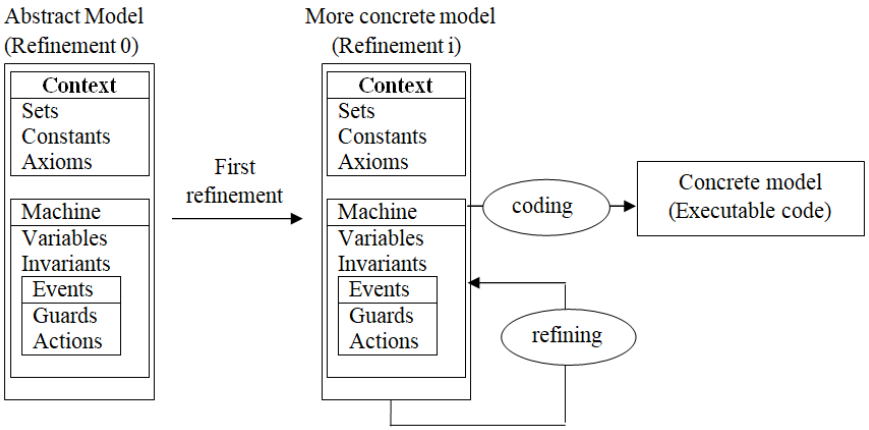

Fig. 2. Process of development in event-B [16]

To ensure that the system will not contain bugs, some mathematical proofs called proof obligations must be performed. One of the most important proofs among proof obligation is invariants preservation that ensures the preservation of all invariant before and after an event trigger. They include also dead-lock freedom to ensure that the system will never be in a status where no guard is verified, which means no event can be trigger. Almost all these proofs are established automatically by means of an eclipse platform called Rodin [8]. It provides the fundamental functionality for syntactic analysis and proof-based verification of Event-B models [8]. In some cases, we may be forced to interfere manually by guiding Rodin in order to perform some proofs. To do so, it may be necessary to specify the hypotheses that Rodin should consider in order to perform proofs. The reason why this should be done is that Rodin ignore some properties if he that they are not necessary for the proof. In other cases, we propose some additional hypothesis that will be proved later separately and will help proving the current desired proof.

3 Contribution

The Reference Model of Open Distributed Processing is mainly based on distributed processing development and the use of formal specification techniques. Whereas, RM-ODP documents are criticized of being very abstract and ambiguous, formal methods provide a very specific models by mean of mathematical representation of distributed systems. In the ITU-T X.904 – Computational formalization [5], the semantic architecture presented in several formal languages (LOTOS, ACT ONE, SDL-92, Z, and ESTELLE).The main contribution of this paper is presenting the architecture semantic using Event-B alongside recommendation on how to develop ODP systems using Event-B. this method provide several advantages compared to other methods such as the independence of temporal ordering, the richness in sense of mathematical predefined operators in addition to the use of techniques to model such as refinement.

Furthermore, Event-B is a formal method based on proof obligations that provide a strong assurance of bugs' absence. Therefore, Event-B is a very suitable formal method to develop ODP systems.

4 Computational formalization

Object modelling is a formalization approach based on abstraction and encapsulation. By means of abstraction we describe the system main functionality separately without considering details, while encapsulation hides all kinds of heterogeneity, security, mechanism, localization, etc [17].

The object modelling concepts cover basic modelling concepts and specification concepts. Basic modelling concepts define the most fundamental concepts of ODP system such as action, objects, interface, and activity [19]. Whereas, specification concepts stands for the reasoning approach used for ODP systems design such as refinement, invariant and compatibility.

In this section, we describe how to formalize the main basic modelling in event-B. We provide also an example of self driving cars along the description to facilitate the understanding of approach.

4.1 Object

An object may be any system component, for example an object may be router or a computer in a network and may be a car in an auto driving cars system. To describe an object in Event-B, a set OBJECTS is defined in the context. After that, we can introduce an object as variable and presented it as an object by mean of a typing invariant (inv 1). In this paper, an abstract recommended model for development of any ODP system is presented and it is up to ready to integrate the concept in the system [20].

```
CONTEXT
        RM-ODP
SETS
        OBJECTS
END
```

MACHINE
RM-ODP
VARIABLES
object
INVARIANTS
inv1 : object ∈ OBJECTS

The rest of concepts are presented along an example of self driving car modelling, therefore the OBJECTS are substituted by a set CARS to facilitate models understanding.

4.2 Environment of an object

The environment of an object is described in term of its relation with it. Any object input or output is considered as a part of the environment. In general, the environment is interacting with object within an event as an event parameter. For example, in an auto driving cars system, the environment which is weather is affecting the car (which is the object) in term of deciding either it is allowed to move or not [21]. The event parameter in this case is wind speed which should be less than about 35 miles per hour to allow car moving. This may be formalized below by an event that prevents car from moving and associate to it the state stopping.

moving_preventing
ANY
wind_speed
car
WHERE
grd1 : wind_speed ∈ ℕ
grd2 : wind_speed ≥ 35
grd3 : car ∈ CARS
THEN
act1 : state(car)=stopping
END

4.3 State of an object

The state of an object is described as a total function from the set OBJECTS to a set of STATES. The set of states is presented in term of enumeration [15]. For example, states moving/stopping can be an enumeration of car states in the auto driving cars system.

SETS
OBJECTS
STATES
CONSTANTS
moving
stopping
AXIOMS
axm1 : partition(STATES,{moving},{stopping})
END

The partition predicate is an easy way to enumerate sets. Mathematically, the partition predicate is defined as follows:

$$partition(S, x, y) \Leftrightarrow x \cup y = S \ \wedge \ x \cap y = \emptyset$$

Where x and y are two subsets of a set S.

The state function that associates to a car a certain state is defined by means of an invariant in the machine as follows:

$$\text{inv2} \ : \ \text{state} \in \text{CARS} \rightarrow \text{STATES}$$

4.4 Action

We must be careful here to avoid confusion between RM-ODP action and Event-B action. The first is an operation that changes the object states, whereas, the event-B action changes the value of a variable. Therefore, we will refer to them in this section as ODP_action and B_action.

An ODP_action is modelled in event-B by the performance of an event. The effect is the instantaneous change in state (or the null change) of the objects with which that ODP_action is associated. An object is defined by a set of variables that describe the current state of the object; the values of these variables may change by means of B_actions in an event. In the auto driving car example, we model the ODP_action that change the state of the car from moving to stopping using an event Stop_car as below:

Stop_car
ANY
car
WHERE
grd1 : car ∈ CARS
THEN
act1 : state(car)≔stopping
act2 : speed(car)≔0
END

4.5 Interface

An abstraction of the behaviour of an object obtained by identifying the operations associated with that object that is to form the substance of the interface. In the auto driving cars example, we may define an interface vehicle in the initial model that express all the possible operation that a vehicle perform such as moving, stopping, accelerating, etc. this can be expressed as an iUML-B representation [14].

4.6 Activity

While Event-B is based on graphical representation, the presentation of an activity as a single headed directed acyclic graph of action does not exist directly in Event-B. However, we can express an activity by forcing a certain order of ODP_actions. This can be done by defining an enumeration denoting the various steps of activity, then we use guard to ensure that the correct event is the only one that may occurs. The steps' transition is done by means of an action the associate to the current step variable the next wanted action.

For example, let A, B, C, and D four ODP_actions that should occur in this order. We define an enumeration of set that include all possible steps beside two additional ones that are: the start and the end. The steps enumeration set is defined as follows:

Partition (STEPS,{start},{step1},{step2},{step3},{step4},{end})

After that, a variable called current_step is defined indicating the current step. The current_step is initialized by start, then a guard in the A event ensure that the current step is the start. A B_action is also added that indicate the next step which is step1. The B event will have a guard that current_step is step1, so on and so forth.

4.7 Behaviour of an object

The behaviour of an object is the set of all possible activities that may occur. The actual activity that will occur depends on the environment of the object and the current state [13]. A very practical way to present the behaviour o an object is by using iUML-B which is a graphical representation of the different states of the object.

4.8 Communication

Communication may be modelled in event-B through the interaction between object. This communication is presented in several forms in the event-B model. In some cases, it is presented as an invariant (for example: two cars should keep a minimum distance between them) or as an event (for example: a car may follow another), etc. all interactions between two objects are guided by a communication between them. In general, the communication between objects is performed by exchanging outputs and inputs of associated variables.

5 Conclusion

Modelling an ODP system that follows the RM-ODP with a significant number of components is, in general, a challenging task that should be performed carefully. Moreover, the use of a sophisticated formal method such as Event-B rise also the complexity of the task, therefore, we propose a number of recommendations and guides that helps modelling such systems. These recommendations and guides illuminate how to model the main concepts of object modelling.

The recommendations proposed in this paper were used in several works [16, 18, 22-25] and all proof obligations were performed correctly using Rodin platform. Therefore, the use of these recommendations highly guarantees the correctness of the model thus the discharging of proofs.

References

[1] Qi Zhang, Zhiqiu HuangJian Xie: Distributed System Model Using SysML and Event-B, International Conference on Machine Learning and Intelligent Communications, MLICOM 2017: Machine Learning and Intelligent Communications pp 326-336(2018).

[2] Ciancarini, P., Poggi, F., Rossi, D., & Sillitti, A. Analyzing and predicting concurrency bugs in open source systems. In Neural Networks (IJCNN), 2017 International Joint Conference on (pp. 721-728). IEEE (2017).
[3] Fayollas, C., Palanque, P., Fabre, J. C., Martinie, C., & Déléris, Y. Dealing with Faults During Operations: Beyond Classical Use of Formal Methods. In The Handbook of Formal Methods in Human-Computer Interaction (pp. 549-575). Springer, Cham (2017).
[4] Lecomte, T., Deharbe, D., Prun, E., & Mottin, E. (2017, November). Applying a formal method in industry: a 25-year trajectory. In Brazilian Symposium on Formal Methods (pp. 70-87). Springer, Cham.
[5] ISO/IEC 10746-4:1998/Amd.1 Information technology — Open Distributed Processing — Reference Model: Architectural semantics — Part 4 AMENDMENT 1: Computational formalization (2001).
[6] ITU Recommendation X.904 I ISO/.IEC CD 10746-4, Open Distributed Processing - Reference Model- Part 4: Overview (1994).
[7] J.-R. Abrial: Modeling in Event-B: System and Software Engineering, Cambridge University Press New York (2010).
[8] C. Rodin, M. Jastram, and M. Butler: User's Handbook.(2011).
[9] ITU Recommendation X.903 I ISO/.IEC 10746-3: 1995, Open Distributed Processing - Reference Model- Part 3: Overview.
[10] ITU Recommendation X.902 I ISO/.IEC 10746-2: 1995, Open Distributed Processing - Reference Model - Part 2: Overview.
[11] ITU Recommendation X.901 I ISO/.IEC CD 10746-1, Open Distributed Processing - Reference Model- Part 1: Overview (1994).
[12] The international standard ISO/IEC 10746, information technology – open distributed processing – reference model: Architectural semantics, first edition (1998).
[13] Asprino, L., Nuzzolese, A. G., Russo, A., Gangemi, A., Presutti, V., & Nolfi, S. (2017). An Ontology Design Pattern for supporting behaviour arbitration in cognitive agents. Advances in Ontology Design and Patterns, 32, 85.
[14] Snook, C., Hoang, T. S., & Butler, M. Analysing security protocols using refinement in iUML-B. In NASA Formal Methods Symposium (pp. 84-98). Springer, Cham (2017).
[15] DONG, Q., JI, M. Q., ZHU, Y. F., & YANG, F. (2018). The CORAS Approach for OPM Based Risk Management. DEStech Transactions on Engineering and Technology Research, (pmsms) (2018).
[16] Jarrar, A., & Balouki, Y. Towards Sophisticated Air Traffic Control System Using Formal Methods. Modelling and Simulation in Engineering, (2018).
[17] Genilloud, G., & Wegmann, A. A foundation for the concept of role in object modelling. In Enterprise Distributed Object Computing Conference, 2000. EDOC 2000. Proceedings. Fourth International (pp. 76-85). IEEE (2000).
[18] Jarrar, A., & Balouki, Y. (2018). Formal modeling of a complex adaptive air traffic control system. Complex Adaptive Systems Modeling, 6(1), 6 (2018).
[19] Cheatham, M., Vardeman II, C. F., Karima, N., & Hitzler, P. (2017, October). Computational Environment: An ODP to Support Finding and Recreating Computational Analyses. In WOP@ ISWC (2017).
[20] Oliveira, I. L., Câmara, J. H., Torres, R. M., & Lisboa-Filho, J. Design of a Corporate SDI in Power Sector Using a Formal Model. Infrastructures, 2(4), 18 (2017).
[21] Laassiri, J. Data Security and risks for IoT in intercommunicating objects. In Proceedings of the 2nd international Conference on Big Data, Cloud and Applications(p. 3). ACM (2017).

[22] Jarrar, A., Balouki, Y., & Gadi, T. Formal Specification of QoS Negotiation in ODP System. International Journal of Electrical and Computer Engineering (IJECE), 7(4), 2045-2053 (2017).
[23] Jarrar, A., Gadi, T., & Balouki, Y. Modeling the Internet of Things System Using Complex Adaptive System Concepts. In Proceedings of the 2nd International Conference on Computing and Wireless Communication Systems (p. 22). ACM (2017).
[24] Jarrar, A., Bellasri, O., Chougdali, S., & Balouki, Y. Formal Specification and Verification of Transmission Control Protocol. In Proceedings of the 2nd International Conference on Computing and Wireless Communication Systems (p. 30). ACM (2017).
[25] Jarrar, A., Balouki, Y., Gadi, T., & Chougdali, S. Modeling Aircraft Landing Scheduling in Event B. In International Conference on Information Technology and Communication Systems (pp. 127-142). Springer, Cham (2017).