Event-B Based Formal Modeling of a Controller: a Case Study

Rahul Karmakar, Bidyut Biman Sarkar and Nabendu Chaki

February 7, 2020

Event-B Based Formal Modeling of a Controller: A Case Study

Rahul Karmakar[1*], Bidyut Biman Sarkar[2] , and Nabendu Chaki[3]

[1*]The University of Burdwan, Golapbag, Burdwan, 713104, India
rahulkarmakar6@gmail.com
[2]Techno International, Kolkata – 700156, India
bidyutbiman@gmail.com
[3]University of Calcutta, Kolkata – 700098, India
nchaki@gmail.com

Abstract: Event-B is an event-driven approach for system development. It has the flexibility to develop different discrete control systems. It is a modeling language over a wide range of application domains. It is a refinement based step by step modeling methodology, used to develop the model of the system incrementally. There is a well-tested open-source tool available for model B checking, formalization of mathematical proofs and system validation is RODIN. In this paper, we present a short survey on usage of Event-B based model to locate the research gaps followed by a case study to build a model using 2 stage refinement strategy of event B to stop the precious groundwater wastage and conserve it. We try to model the behavior required for the environment of the system. Our proposed controller then controls the environment. The controller acts accordingly and we achieve the goal of groundwater conservation.

Keywords: Formal Modeling, Event-B, RODIN tool, Industry automation, Eclipse IDE, Water pump controller.

1 Introduction

Event-B [1] is a modeling language and its application range is versatile. Not only a sequential program to distributed systems but it has the privilege to model different control systems. Event-B models the environment which is the necessity to assure the correctness of the proposed systems[3].The Event-B based formal modeling proposed by Jean-Raymond Abrial [1], is a top-down engineering approach consists of step-by-step refinement strategy. The designers, therefore, design the refinement strategies to meet the system requirements and specifications. A model in Event-B consists of two components – context and machine. Context is the static part of a model. It contains all the constants, axioms, sets, and theorems that remain unaffected at any state of a machine. The machine describes a whole model. It sees the context; and also contains the dynamic part consisting of variables, invariants, theorems, and events. The dynamic part is affected when an event has occurred (i.e. when the state of a machine is changed). An event consists of a set of actions, guards, and parameters. The axioms are the predicates made-of constants, and the invariants are the predicates made-of variables (and constants). The refinement can be done by imposing features one-by-one to an abstract model. Thus we can find a set of refined models. At the final refinement, the model has all the desired features. The context of one model can be extended to the context of another model. When a machine (concrete machine) refines another machine (abstract machine), it refines the abstract events of the previous one; it can also extend the concrete events as well [1, 10]. We know that formal modeling allows us to do a more rigorous review of the system and as a result improve the quality of the system. The Rigorous Open Development Environment for Complex Systems (RODIN) [5] tool supports refine based rules and mathematical proofs. It is an eclipse based open-source IDE and can be extended with plugins. The notations of Event-B are either automatically or interactively generate the proof obligation. The main advantage of RODIN is its flexibility for proof obligation and model checking[5]. Starting from the initial model, each of the refinements is proved so that if any error is found at any stage of refinement, the error must not be carry-forwarded to the next refinements. The RODIN platform indicates which rules are successfully proved and which contradict. Event-B has drawn reasonable attention both for industrial and academic communities towards modeling problem specifications and iterative refinements.

In section 2 we try to give a glimpse of the Event-B based formal modeling approaches particularly in industry and safety-critical domain though it has a wide range of domains like Artificial Intelligence, Automatic code generation, E-Commerce, etc. In section 3 we did a case study on a water pump controller. We design the control strategy using Event-B and check the strategies using the RODIN tool.

2 System Review

In this section, we are trying to give a brief review of different control strategies developed using Event-B. We consider mainly industrial automation and safety-critical systems. A few works are considered for the review though there a handsome number of works are there. We start with an article by Michel Butler [2] where he proposed a control strategy of a water pump. He tries to achieve the control goal and control strategy using Event-B modeling language. The system has monitored variable for checking the water level of the tank, Controlled variable controls the environment and switches on and off the pump. The Input and Output variables tell the water level value stored in sensors and registers. The event UpdateWaterLevel is used to update the current water level of the tank. DecreseWaterLevel and IncreseWaterLevel events refine the previous event UpdateWaterLevel. Guard variables are used to enforce time-bound in the environment events. The control event ControlLow is used to switch on and off the pump by checking the threshold value of the tank. The maximum threshold value signifies that the tank is full and it will stop the pump automatically and vice versa [2].

Jean-Raymond Abrial [1] in his book Modeling in Event-B system and software engineering discussed different applications developed using Event-B modeling. In chapter 2 he discussed controlling cars on a bridge. Here he develops a control system for scheduling the cars from the mainland to Iceland via a bridge. The system development starts with the initial model where only the mainland and Iceland is considered. In the first refinement, the bridge is introduced and assumes that no car is on the bridge. Then traffic lights are introduced. The Greenlights allows a car to go and Red light stops a car. Then sensors are introduced in the system for identifying a car whether it is entering the bridge or leaving the bridge. The one way and both way nature of the bridge are also considered. The controller has different events and functions to refine the environment. In chapter 16 we find the design of the location access controller where people can find their locations. The design controls the sensors and other devices to get accurate location information. Finally, in chapter 17 the design of a train system controller is proposed. It will act as a software agent to control trains. The goal of the controller is to provide the safety and reliability of a train network. All the examples are well formalized with Event-B notation and they are deadlock-free [1].

In the paper[3], the authors show how the platforms and trains are managed by the controller. They identified and designed the network topology using the event-driven approach. They consider sensors, switches, actuators for the purpose. Different events and actions are developed. Safety rules are also imposed on those actions. The model is developed in the RODIN tool and all the consistencies are checked and it is available online [3].

In reference [4] authors discussed a Programmable Logic Controller (PLC). The PLC is meant for a huge factory setup where radiopharmaceuticals are produced. The initial model says that the Automatic production line only starts and there is no production. Then the progress event of work is done step by step with the help of the refinements. Safety is the main concern. The starting refinement deals with the safety rule where the action of the cylinders considered. The action of the cylinder tells the progress event to the next step. The next refinements deal with the safety rule for the people. A safety alarm is used for safety. The position of the delivery is very important and to get the correct information about the delivery system and this rule is used in the succeeding refinements. The final rule tells the successful accomplishment of delivery and for that, a confirmation message from the device is introduced as an end rule. If all the steps of the system complete and safety rules are followed, then the risk must be minimized that we discussed above.We find an article about an aircraft landing system [8]. The failure of a critical system like aircraft causes a fatal end. Different landing gear arrangement is used in aviation and it has sequential operations like open the doors of gearboxes, extending and closing the doors. The landing gear system has three main components like the mechanical system, the digital system and the pilot interface. Event B is used not only to model the system behavior of the landing gear system but to prove the safety requirements. The first refinement is about the press up and presses down events of the system. The next refinement is about opening and closing the doors of the aircraft. Observation of the gear is done in the next refinement. Next sensors and actuators are introduced in the system. Failure detection and adding lights are also done in the system. These refinements are modeled by the event B notations. The refinements are verified by the RODIN tool [8]. The integration of domain-based features using event B modeling is done in an article. A case study of noise gear velocity is shown in this paper [9]. Designers find event B based modeling and RODIN based proof obligation of the requirements of the system in safety and embedded systems [6].

3 A Case Study on a Controller Design

We use the Event-B Modeling Language to address the aspect of consumable groundwater conservation. The consumable freshwater is essential for the sustainability of mankind and day by day the water demand exceeds its availability. The careless wastage of water is one of the major reasons behind it. Countries like India the main source of drinking water is the groundwater. Most households have a pump to withdraw groundwater. The main goal of the system is to control the pump connected to a water tank automatically to save the wastage of water. We start with an abstract model of the system. Then in each refinement, we consider the requirements and refine the model. The controller communicates with the environment with the help of sensors and the communication is bidirectional. The controller receives input from the environment via the sensor and the controller produces outputs accordingly to change the environment [2]. The three major steps are given below: Firstly, we model the environment to behave accordingly: It is an abstract model and we omit the controller completely. Here we have the pump and the water tank. Secondly, we model the capacitive liquid level sensor and relay switch to change the environment: Here we introduce these components and ensure that they interact with each other correctly and change the environment accordingly. Finally, we model the controller to get statistical analysis: Here we introduce a strategy which will enable us to perform some statistical analysis generated by the controller. The main advantage of this development strategy is that we start from a basic model and using step by step refinements we reach the final model. We have the flexibility to introduce different components until required at any stage.

3.1 Requirement Document

We start our system modeling with the initial model where we have the minimum functionalities like start filling the tank or stop filling. Then we add refinements and we consider detail requirements of the system. We introduce a capacitive liquid level sensor, relay switch in the system for checking the water level and start and stop the pump accordingly.

Table 1. The environments and functions of the system.

ID	*Description*
ENV-1	The system consists of a Pump, a Water Tank, a Relay Switch, and a Capacitive Liquid Level Sensor.
ENV-2	When the relay is on, the pump is started to withdraw groundwater; when the relay is off, the pump stops.
ENV-3	The capacitance level liquid sensor gives the minimum and maximum water level of the water tank.
ENV-4	There is a microcontroller that controls the whole system.
FUNC-1	The Controller will control the groundwater withdrawal by the pump.
FUNC-2	When the Capacitive Liquid Level Sensor gives minimum value, the controller switch on the relay; hence the pump is started.
FUNC-3	When the sensor gives maximum value, the system switches off the relay; hence the pump stopped.

3.2 Initial Model

The initial model gives the highest level of abstraction of the system. Here we can only see the water tank. The tank is started to fill or stop filling based on the water level in the water tank. We have a microcontroller in our system which will sense the current water level of the tank. The block diagram of the system is shown in figure 1.

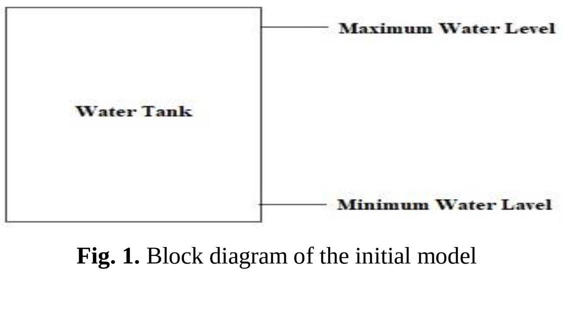


Fig. 1. Block diagram of the initial model

If it finds the water level lesser than the minimum threshold then it will initiate the pump to start filling the tank. When the water level crosses the maximum threshold then it immediately stops the pump. The water level in the tank can be expressed in formal notation as following constants: WATER_LEVEL_MAXIMUM and WATER_LEVEL_MINIMUM. The type of these constants can be expressed using the following axioms.
Axiom1 is WATER_LEVEL_MAXIMUM which belongs to a natural number N and axiom2 WATER_LEVEL belongs to N. Therefore, the Water Tank has two states FULL or EMPTY. These values belong to the set TANK_STATUS. The relation can be expressed using axioms 4 and 5.
Axiom4 represents the TANK_STATUS which is {EMPTY, FULL} and axiom5 say EMPTY is not equal to FULL.
The filling of water into the tank has two states STARTED or STOPPED. These values belong to the set FILLING_STATUS.Axiom6represents FILLING_STATUS which is equal to the set {STARTED, STOPPED}.
Axiom7 is defined as STARTED not equal to STOPPED.
All the sets, constants and axioms are placed within the context of the initial model. We assume that the following variables will keep track of the dynamic states of a machine:
PresentWaterLevel The present water level in the tank status denotes whether the Water Tank is full or not.
filing status denotes whether the tank is started filling. The type of the variables and their relationships with each other and with the sets and constants are represented by the invariants: We use four invariants.Invariant1 represents the present water level of the tank which is a natural number. Invariant2 is for tankStatus which belongs to TANK_STATUS and Invariant3 is filing status which belongs toFILLING_STATUS.Invariant4 represents the WaterLevel which is less than or equal to the WATER_LEVEL_MINIMUM where tank status equals EMPTY.
An event is a set of actions may occur or to be performed during the execution of the system. The occurrences of an event change the state of the machine. Therefore, one or more variables are modified during the execution of an event. We introduce guard for the events. The basic events of the initial model START_FILL and STOP_FILL are formalized using Event-B notations and are bellowed [1, 2, 5].

START_FILL function	**STOP_FILL function**
START_FILL WHERE grd1:presentwa-terlevel=WATER_LEVEL_MINIMUM THEN act1: tankstatus=EMPTY act2: fillingstatus=STARTED END	STOP_FILL WHERE grd1:presentwa-terlevel=WATER_LEVEL_MAXIMUM THEN act1:tankstatus=FULL act2: fillingstatus=STOPPED END

To begin execution, the first event which is occurred in every system is the INITIALISATION. In this event, the initial values are assigned to the variables. Therefore, every machine must have an INITIALISATION event. It must not have guard and parameter. It is assumed that the water tank is initially empty. There are two main events: START_FILL occurs when the water level in the tank reaches its minimum level.STOP_FILL occurs when the water level in the tank reaches its maximum level. When an event makes changes to the variables it must preserve the hypothesis associated with those variables and the system must be deadlock-free. The proof-obligation rules are provided to confirm these preservations. The RODIN tool automatically generates proof obligations and tries to prove the model. It shows errors when there is any contradiction.

3.3 Refinement 1:

Introducing the Capacitive Liquid Level Sensor, Relay Switch and Pump. In this refinement, the abstraction is removed and the capacitive liquid level sensor, relay switch and the pump can be seen. Since we have a sensor, we don't have to bother about the water level. The whole setup is shown in figure 2.

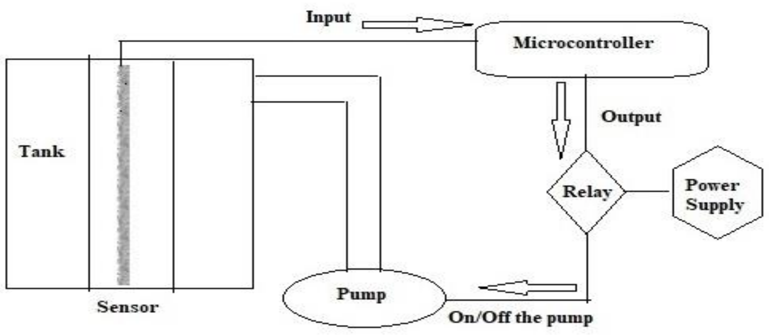


Fig. 2. Overall Block diagram of the refined system

The sensor's output is taken to decide whether to on or off the relay to start or stop the pump respectively. In this refinement, the pump will start and stop automatically by the capacitance sensor and relay switch of the pump. When the water tank becomes empty, the capacitance value becomes lowest i.e. MINIMUM; then the microcontroller ON the relay switch to start the pump. On the other hand when the water level reaches its maximum level i.e. the water tank becomes full, the capacitance value becomes highest i.e. MAXIMUM; then the microcontroller OFF the relay switch to stop the pump from further water withdrawal. Therefore, in this refinement, we need two sets of statuses RELAY = {ON, OFF} and CAPACITANCE = {MINIMUM, MAXIMUM}. In the machine refinement, the two variables named capacitance and relay are used to track the current statuses of the sensor and the relay respectively. There is a need to draw a relation between the present water level of the initial model and the capacitance value of the refined model. Capacitance equals MINIMUM and presentWaterLevel is equal WATER_LEVEL_MINIMUM. When capacitance equals to MAXIMUM and presentWaterLevel is equal to WATER_LEVEL_MAXIMUM. The refinement also has three events. INITIALIZATION event sets the initial value for the variables. Initially, capacitance equals MINIMUM and relay is ON. START_PUMP refines START_FILL to ON the relay when capacitance value is MINIMUM and STOP_PUMP refines STOP_FILL to OFF the relay when capacitance value is MAXIMUM. The events in this model are starting and stopping the pump are formalized using Event-B notations and given bellow [1, 2, 5].

START_FILL function	**STOP_FILL function**
START_PUMP REFINES START_FILL WHERE grd1: capacitance=MINIMUM THEN act1:relay=ON END	STOP_PUMP REFINES STOP_FILL WHERE grd1: capacitance=MAXIMUM THEN act1:relay=OFF END

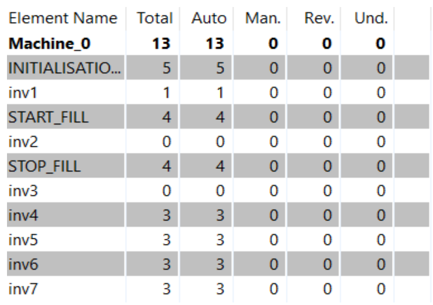
3.4 Refinement 2:

Here we propose some additional events and actions for statistical analysis of water usage like EndOfDayEndOfMonth and EndOfYear. Based on the statistical data of the previous month, the maximum water consumption limit per day is set for the next month. The model generates a statistical report on a daily, monthly and yearly basis for usage, monitoring, and validation.
EndOfDay: The water consumption per day is added to a set to generate a statistical analysis for a month. **EndOfMonth:** The water consumption per month is added to a set to generate a statistical analysis for 12 months. A set will store all the data.
EndOfYear: A set will store the yearly report.
We limit our daily use of water from statistical analysis. We get the limit from the last few years' daily usages of water. We will have the actions which will withdraw water according to the limit received by the statistical analysis. We can start and stop the pump according to the water consumption limit also. One household can get a previous month or year data so they can get the average use of water and use the pump accordingly.

4 Model Analysis and Validation

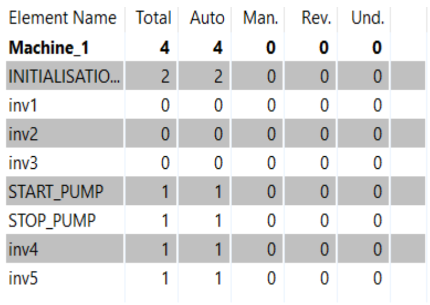
This analysis helps to maintain water consumption by a household in a great way.

We are still working on this refinement and add more features like getting statistical data when draining and filling are performed at the same time. This model may be applied to a municipality pumping station in the future for an optimized water supply. The Rodin Platform is an Eclipse-based open-source integrated development environment (IDE) for Event-B. It provides operative provision for refinement and mathematical proof. We prove all our assumptions and model consistency in the RODIN platform. We generate all the proof obligations in the RODIN tool and validate the water conservation model using the context, sets, constraints, events, and axioms of START_FILL function, STOP_FILL function in refinement 1, and additional events like EndOfDayEndOfMonth, and EndOfYear. The proposed controller not only controls the whole system but manages the redundant withdrawal of water. The controller addresses the excessive overflow of water from the water tank and helps to control the state. We are capable to successfully distinguish all the environments and functionalities. We can formalize the environments and functionalities in the final model using the RODIN tool. We also prove all the proposed functions for the controller. We have 14 automatic proof for the initial model and 5 automatic proof for 1st refinements. The proof statistics from RODIN toolset are shown in figure 3(a) and 3(b). Here all the events of Machine 0 and Machine 1of the proposed Event-B model are automatically proved. The proposed invariant rules are also been discharged using RODIN auto-prover.



Element Name	Total	Auto	Man.	Rev.	Und.
Machine_0	**13**	**13**	**0**	**0**	**0**
INITIALISATIO...	5	5	0	0	0
inv1	1	1	0	0	0
START_FILL	4	4	0	0	0
inv2	0	0	0	0	0
STOP_FILL	4	4	0	0	0
inv3	0	0	0	0	0
inv4	3	3	0	0	0
inv5	3	3	0	0	0
inv6	3	3	0	0	0
inv7	3	3	0	0	0

Fig. 3(a). proof stat of Machine 0



Element Name	Total	Auto	Man.	Rev.	Und.
Machine_1	**4**	**4**	**0**	**0**	**0**
INITIALISATIO...	2	2	0	0	0
inv1	0	0	0	0	0
inv2	0	0	0	0	0
inv3	0	0	0	0	0
START_PUMP	1	1	0	0	0
STOP_PUMP	1	1	0	0	0
inv4	1	1	0	0	0
inv5	1	1	0	0	0

Fig. 3(b). proof stat of Machine 1

These are the simple rules so the proofs are less and we did not need any iterative proofs so far. The full proof design makes the system implementation and testing smoother and we can firmly deal with the real-time system design using Event-B.

5 Conclusion

We start from an abstract model and design the model incrementally to meet the required goals. A system is modeled into event B notations. The proposed system has a machine and context that deals with the environment. For detailing the system different events are introduced. Event-B based modeling has the facility to consider the environment with the software part. All the variables, invariants and guards help to meet all constraints of the system. RODIN platform plays a great role to prove the models. The application domain could be anything but all we need is the ability to formalize it. All the applications we discussed above are safety-critical and failures can be instantly traced back. Researchers can design plug-in for the RODIN framework for automatic code generation in many languages like Java and C. Our attempt will be on exploring security issues in E-commerce and distributed transaction domains. In this paper, we tried to summarize the work of event B, especially for controller design. We find the survey very useful in different perspectives. First of all event, driven modeling is very useful in almost every application domain from industry to communication. As the modeling approach in Software Engineering minimizes the design ambiguities and meets the requirements beforehand in a simulated environment. All interactive proofs of a model could not be checked by the RODIN. But we have the facility to design our model checker on the Eclipse platform as a plugin of RODIN.

References

1. J-R Abrial, Modeling in Event-B system and software engineering [C] Cambridge University Press, UK, 2010.
2. Michael Butler, “Using Event-B Refinement to Verify a Control Strategy”, Unpublished.
3. Simon Hudon, Thai Son, and Hong, "Development of Control systems Guided by Models of Their Environment", Electronic Notes in Theoretical Computer Science 280, Elsevier, 2011,pp.57–68.
4. Kaiming-FU, Bin Fang, Yafen-LI and Huijie-LI., Research on Event-B based formal modeling and verification of automatic production line,IEEE,pp.3690-369,2016[28th Chinese Control and Design Conference(CCDC), China,2016].
5. Michael Jastram and Prof. Michael Butler, RODIN User’s Handbook. DEPLOY project,2010.
6. Event-B Homepage: http:// http://www.Event-B.org, last accessed 5/9/2017.
7. Abdolbaghi Rezazadeh, Neil Evans and Michael Butler, “Redevelopment of an Industrial Case Study Using Event-B and RODIN”,2007[BCS-FACS Christmas 2007, Meeting Formal Methods In Industry The British Computer Societ,2007].
8. Dominique Méry and Neeraj Kumar Singh, "Modelling an Aircraft Landing System in Event-B", Communication and Computer Science, Springer Vol. 433,pp.154-159,2014.
9. Dominique Méry, Rushikesh Sawant and Anton Tarasyuk, Integrating “Domain-Based Features into Event-B: Noise Gear Velocity A Case Study”, Tiziana Margaria, Bernhard Steffen (Eds.) [MEDI 2015, LNCS, vol. 9344, pp 89-102 ,Springer, Heidelberg,(2015)].
10. Dominique Cansell, DominiqueM´ery, "The Event-B Modelling Method: Concepts and Case Studies”, W. Brauer J. Hromkovic G. Rozenberg A. Salomaa(eds) [EATCS Series, Monographs, Theoretical Computer Science(Logics of Specification Languages), vol. 57, pp 47-152, Springer ,Heidelberg, 2008].