# Formal Modeling and Verification of Context-Aware Systems Using Event-B

Hong Anh Le[1(✉)] and Ninh Thuan Truong[2]

[1] University of Mining and Geology, Dong Ngac, Tu Liem, Hanoi
[2] VNU - University of Engineering and Technology, 144 Xuan Thuy, Cau Giay, Hanoi
{anhlh.di10,thuantn}@vnu.edu.vn

**Abstract.** Context awareness is a computing paradigm that makes applications responsive and adaptive with their environment. Formal modeling and verification of context-aware systems are challenging issues in the development as they are complex and uncertain. In this paper, we propose an approach to use a formal method Event-B to model and verify such systems. First, we specify a context aware system's components such as context data entities, context rules, context relations by Event-B notions. In the next step, we use the Rodin platform to verify the system's desired properties such as safety properties. It aims to benefit from natural representation of context awareness concepts in Event-B and proof obligations generated by refinement mechanism to ensure the correctness of systems. We illustrate the use of our approach on a simple example.

**Keywords:** Context awareness · Modeling · Verification · Event-B

## 1 Introduction

Context-aware systems potentially determine their behaviors and reduces human-computer interaction by providing knowledge context information of their user's environment. Context awareness of an application relates to adaptation, responsiveness, sensitiveness of the application to changes of the context [3]. Since the behaviors of such systems are often complex and uncertain. That could be unacceptable especially when context-aware systems are implemented as safety-critical systems. The results up to date have worked on modeling context awareness with various approaches such as object role modeling, ontology based modeling, logic based modeling [3,14]. They also have proposed several frameworks for context modeling. However, to the best of our knowledge, there does not exist an approach that models context awareness in several aspects such as events of environments, context rules and uncertainty. Furthermore, the resulted model can be formally verified to ensure the correctness of the system.

Formal methods are techniques used for modeling and verifying systems. These techniques prove the correctness of the system mathematically. The B method [1] is a formal software development method, originally created by

P.C. Vinh et al. (Eds.): ICCASA 2013, LNICST 128, pp. 250–259, 2014.
DOI: 10.1007/978-3-319-05939-6_25, 

J.-R. Abrial. The B notations are based on the set theory, generalized substitutions and the first order logic. Event-B [2] is an evolution of the B method that is more suitable for developing large reactive and distributed systems. Software development in Event-B begins by abstractly specifying the requirements of the whole system and then refining them through several steps to reach a description of the system in such a detail that can be translated into code. The consistency of each model and the relationship between an abstract model and its refinements are obtained by formal proofs. Support tools have been provided for Event-B specification and proof in the Rodin platform.

In this paper, we propose to use Event-B as a formal method to model and verify context-aware systems. A context-aware system is somehow considered as a reactive system, i.e. it receives events emitted by context changes and responses to these changes with the providing context knowledge. For this reason, Event-B is a well-suite method for modeling such systems in comparison to others formal methods. The contributions of our proposal are: (1) Natural representation of context-aware systems by Event-B concepts. Context awareness components are then defined formally. The modeling process is also such practical that we can implement a tool which automatically model from the context awareness specification (2) After formalization, significant properties are verified via proof obligations of refinement mechanism automatically (or interactively) without any intermediate transformation.

The rest of the paper is structured as follows: Sect. 2 provides some background of Context awareness and Event-B. In Sect. 3, we introduce an approach to model a context-aware system by formalizing its components using Event-B notations. Section 4 presents a scenario of an Adaptive Cruise Control system in order to demonstrate our approach. Section 5 summarizes some related works. We conclude and provide future works in Sect. 6.

## 2 Backgrounds

As we use Event-B notation to formalize context-aware systems, in this section, we introduce briefly some background of Event-B and context awareness.

### 2.1 Event-B

Event-B is a formal method for system-level modeling and analysis. Key features of Event-B are the use of set theory as a modeling notation, the use of refinement to represent systems at different abstraction levels and the use of mathematical proof to verify consistency between refinement levels [2]. An Event B model encodes a state transition system where the variables represent the state and the events represent the transitions from one state to another. A basic structure of an Event-B model consists of a MACHINE and a CONTEXT.

A machine is defined by a set of clauses which is able to refine another machine. We briefly introduce main concepts in Event-B machine as follows:

VARIABLES represent the state variables of the model of the specification.
INVARIANTS described by first order logic expressions, the properties of the attributes defined in the VARIABLES clause. Typing information, functional and safety properties are described in this clause. These properties are true in the whole model. Invariants need to be preserved by events clauses.
EVENTS define all the events that occur in a given model. Each event is characterized by its guard (i.e. a first order logic expression involving variables). An event is fired when its guard evaluates to true. If several guards evaluate to true, only one is fired with a non deterministic choice. The events occurring in an Event B model affect the state described in VARIABLES clause.

An Event B model may refer to a CONTEXT describing a static part where all the relevant properties and hypotheses are defined. A CONTEXT consists of the following items:

SETS describe a set of abstract and enumerated types.
CONSTANTS represent the constants used by the model.
AXIOMS describe with first order logic expressions, the properties of the attributes defined in the CONSTANTS clause. Types and constraints are described in this clause.

### 2.2 Context-Aware Systems

The term "context-aware" was first introduced by Bill Schilit, he defined contexts as location, identities of objects and changes of those objects to applications that then adapt themselves to the context [12]. Many works have been focused on defining terms of context awareness. Context-aware systems can be constructed in various methods which depend on requirements and conditions of sensors, the amount of users, the resource available on the devices. A context model defines and stores context data in a form that machines can process. Baldauf *et al.* [3] summarized several most relevant context modeling approaches such as key-value, markup scheme, graphical object oriented, logic based and ontology based models.

In this paper, we consider the environment in which a system is operating as contexts. Therefore, there are many kinds of contexts such as position, acceleration of the vehicle and/or temperature, weather, humidity, etc. The system uses sensors to capture the contexts data. Processing of the system is context-dependent, i.e. it react to the context changes (for example: if the temperature is decreased, then the system starts heating). The system's behaviors must comply with the context constraints properties (for instance: the system does not start heating, even though the operator executes heating function when the temperature is vey high).

## 3 Formalizing Context Awareness

In this section, we consider a simplified context-aware system and represent its components in set theory. Base upon these definitions, we then use Event-B notations to formalize a context-aware system.

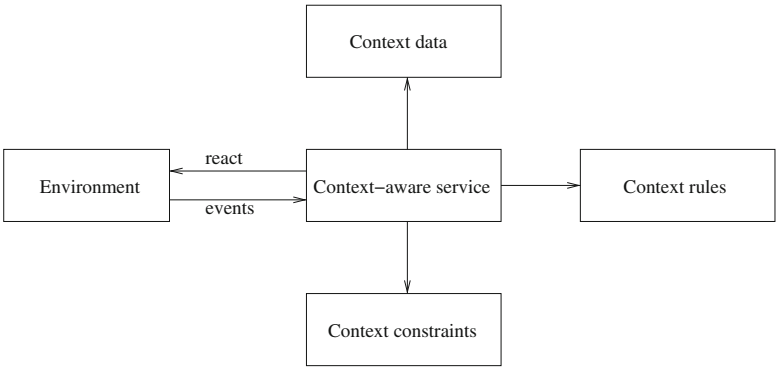


**Fig. 1.** A simple context-aware system

### 3.1 Set Representation of Context Awareness

Firstly, we introduce a simple structure of context-aware systems consisting of five components depicted in Fig. 1. A basic operation of the system is that if there is any change from the environment, it sends events to core context-aware service. This component then uses both context data entities and context rules to reason about the situation. Finally, it reacts to environment via its behaviors. During that process, the system still has to fulfill the constraints.

**Definition 1 (Context-aware system).** *A context-aware system is denoted by a 4-tuple,* $CaS = \langle E, R, CD, CC \rangle$ *where* $E$ *and* $R$ *represent for the environment events and context rules respectively,* $CD$ *denotes context data entities and its relations and* $CC$ *states the system's constraints.*

We go further for definitions of context rules and context entities. Let us assume that rules of context-awareness are in the form of ECA (event-condition-action), i.e. if an an event $e$ occurs in condition $C$ then do action $A$. Hence, we present definitions for each element $r, r \in R$ as follows:

**Definition 2 (Context rules).** *A rule is defined by 3-tuple* $r = \langle e, a, c \rangle$*, where* $e, c$ *are event and condition of the rule respectively, while* $a$ *states the action of the rule.*

Context data consists of context entities and their relations. This component takes a role as a data storage of the system.

**Definition 3 (Context data).** *Context data is denoted by a 2-tuple* $CD = \langle E, R \rangle$*, where* $E$ *is a set of context entities and* $R$ *is a set of functions mapping between sets of context entities.*

**Definition 4 (Environments).** *Environment is a set of events stated by a set:* $E = \{e\}$*, where* $e$ *is an event that is sent to context aware core service.*

### 3.2 Modeling Context-Aware System

Event-B is based on classical set theory, we thus use it to model context-aware systems according to definitions given in Subsect. 3.1. We present transformation rules between a context-aware system and an Event-B model as follows:

- Rule 1: Recall that, we represent context data by either sets or set's elements. Hence, we formalize it as sets or constants of an Event-B Context.
- Rule 2: Each rule $r = \langle e, a, c \rangle$ is mapping to an Event-B event, since its structure is similar to ECA format. More specifically, conjunction of e and c are guards of Event-B event while a is modeled in the body of the event (see Table 1). All these events are included in either Event-B abstract machines or a refined ones.
- Rule 3: Each event that is triggered by environment is represented by an Event-B event. Example: A context-aware system includes a sensor for detecting Wind speed is high or low, it is then formalized by two events: *detectStrongWind* and *detectWeakWind*. Two events of this sensor are included in one Event-B machine.
- Rule 4: A constraint of the context-aware system is a desired property that the system should maintain. That standpoint matches to the meaning of Event-B invariants, we thus model Context constraints by a set of invariants.

We summarize transformation rules used for modeling in Table 2.

### 3.3 Incremental Modeling Using Refinement

In fact, the development of context-aware systems often starts from the scratch requirements, then it is built gradually when we have new requirements about context entities and reasoning. Therefore, it requires to have a suitable modeling method for incremental development. As we have described in Subsect. 3.2,

**Table 1.** Modeling a context rule by an Event-B event

| IF ($e$) | |
|---|---|
| ON ($c$) | WHEN ($e \wedge c$) |
| ACTION (a) | THEN (a) END |

**Table 2.** Transformation between context-aware systems and Event-B notations.

| | Context-aware concepts | Event-B notations |
|---|---|---|
| Rule 1 | Context data $CD$ | Sets, constants |
| Rule 2 | Context rules $r = \langle e, c, a \rangle$ | Events |
| Rule 3 | Environments triggers $E$ | Events |
| Rule 4 | Context constraints $CC$ | Invariants |

a context-aware system is transformed to abstract Event-B. It is apparently suitable for modeling initial context-aware systems. In this subsection, we answer the question how our approach fits to incremental development of such systems.

The refinement mechanism of Event-B makes it possible to model context-aware systems incrementally. We begin with abstract machines to model the very beginning system, after that we refine these machines by concrete ones to represent new requirements of the systems. It is proved through Event-B proof obligations that concrete machines are checked with invariants of abstract ones. According to Rule 4 in Subsect. 3.2, all constraints are represented by Invariants, therefore a new constructed context-aware system at any refined step preserves all constraints of the initial step.

## 4 An Example: Adaptive Cruise Control System

We demonstrate our approach by modeling a scenario of an Adaptive Cruise Control (ACC) system. First, we introduce the scenario, then we apply modeling method presented in Subsect. 3.2 and finally we check the significant properties.

### 4.1 Scenario Description

ACC controls car's speed is based on the driving conditions which are enhanced with context-aware features such as weather conditions, close target conditions, road conditions. The constructed ACC system has three sensors for detecting weather conditions, road conditions and close target. When a car travels in a raining condition or sharp bend, ACC reduces car's speed. If there is no rain or the road is not shape, then ACC resumes the preset speed. When a car detects a target close, ACC reduces car's speed to target's speed. If no target is detected, then it resumes the initial speed.

The ACC must conform to a safety property that is if driver applies the brake, then ACC stops the car in whatever condition. Immediately when the driver releases the brake, ACC resumes the initial speed.

### 4.2 Modeling ACC Scenario

In this scenario, there are three sensors, following the approach presented in Sect. 3, we specify the initial system with one abstract machine and three concrete machines corresponding to the car and three sensors respectively (depicted in Fig. 2).

Two events **DetectBreak** and **DetectNoBreak** are modeled by two events of the abstract machine. In Fig. 3, we formalize the context input sensor detecting weather by a concrete machine which has two events **DetectRain** and **DetectNoRain**. These two events represent two context rules of raining conditions. Modeling context reasoning with road and target sensors are similar.

We finally specify the desired properties of the system such as safety, liveness by Event-B invariants clauses. ACC system should comply the context constraints, for example: if the weather is rain, then car's speed is slower than the

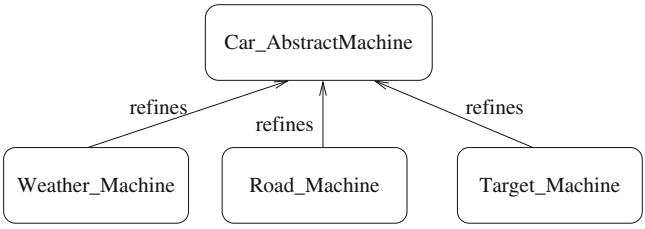


**Fig. 2.** ACC Event-B model overview

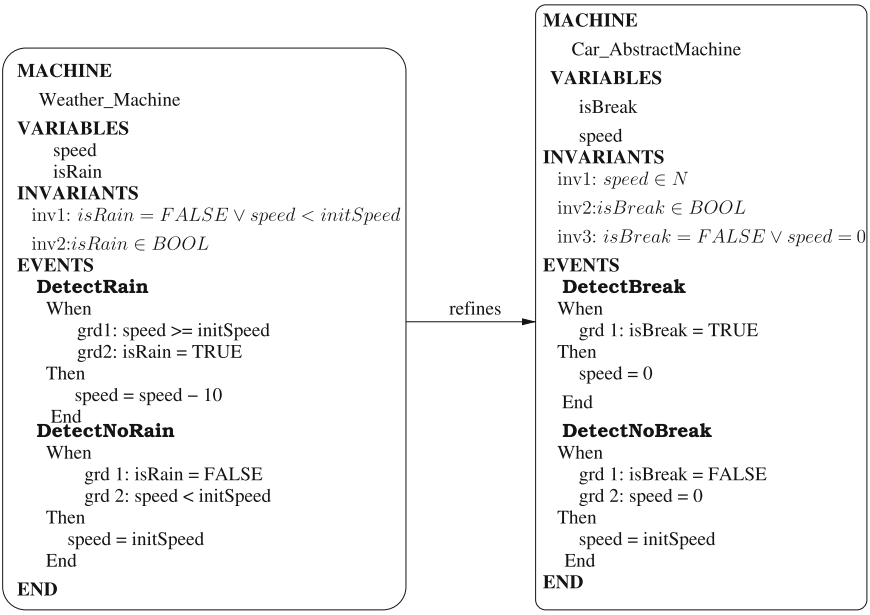


**Fig. 3.** Part of ACC system modeled by Event-B

initial one (stated in Fig. 3 by *inv1* of *Weather_Machine*: $isRain = FALSE \vee speed < initSpeed$). Road and Target context constraints are modeled similarly. Moreover, the system also fulfills a safety property that is if the break is applied, then the speed is zero. This property is specified by the invariant clause *inv3*: $isBreak = FALSE \vee speed = 0$.

### 4.3 Verifying the System's Properties

After modeling, we are able to verify the system safety properties with the Rodin tool. All desired properties are described as four invariants clauses of abstract and refined machines. The Rodin tool generates proof obligations (PO) for these invariants that are proved to be preserved through events for both refined and

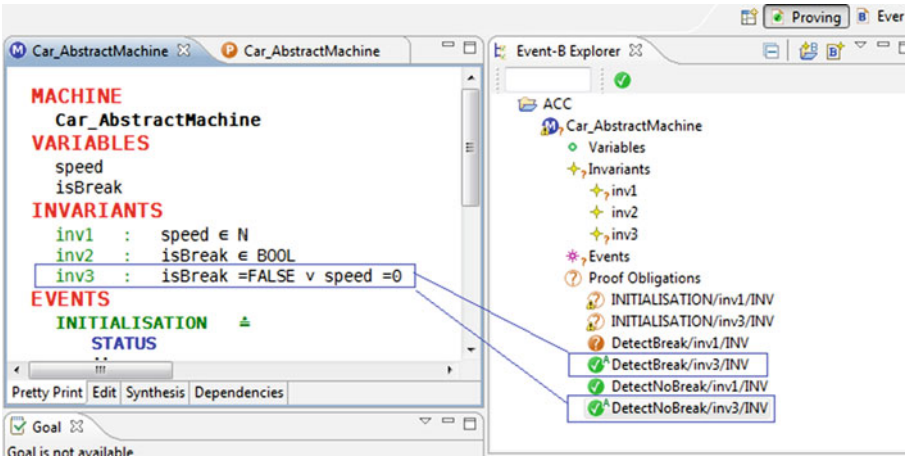


**Fig. 4.** Verification of safety property in Rodin

abstract machines. For instance: the generated proof obligations of the safety property (formalized by *inv3* of the abstract machine *Car_AbstractMachine*) are "*DetectBreak/inv3/INV*" and "*DetectNoBreak/inv3/INV*". The former is the invariant preservation PO of event *DetectBreak*, while the later is the one of event *DetectNoBreak*. The POs are presented in detail as follows:

"*DetectNoBreak/inv3/INV*": $isBreak = FALSE \wedge speed = 50 \vdash isBreak = FALSE \vee speed = 0$.

"*DetectBreak/inv3/INV*": $isBreak = TRUE \wedge speed = 0 \vdash isBreak = FALSE \vee speed = 0$.

These POs are proved trivially and done automatically with the Rodin tool as illustrated in Fig. 4.

## 5 Related Work

Many papers have been proposed for modeling and verifying context-aware systems with various approaches. Most research efforts that are based on markup scheme model have defined and extended markup languages. Henricksen *et al.* [7] proposed to represent contextual data by *Comprehensive Structure Context Profiles* (CSCP). Indulska *et al.* [8] extended CC/CP model to define a set of CC/PP components and attributes to express a various types of context information and context relationships.

Some researchers followed the graphical model approach to model contextual data. Mostefaoui [11] presented a three-layered data model for context. Benselim and Hassina [4] recently presented an UML extension for representing and modeling context by creating some stereotypes that are described by several tagged values and some constraints.

Almost all ontology-based approaches have used high-level ontologies to formalize context information and models. Shehzad *et al.* [13] introduced a formal

modeling method in context aware systems using OWL. Ejigu *et al.* [6] also proposed ontology based reusable context model that providing structure for contexts, rules and their semantics. The problem with these two pieces of work is that there was no verification mechanism presented.

More recently, Tran *et al.* [15] introduced a ROAD4Context framework which is based on Role-Oriented Adaptive Design (ROAD) [5] to model context-aware systems. However, in order to verify the system, it takes more intermediate steps to translate a ROAD4Context model to a Petri net model and then use SPIN to check the system's behaviors. Furthermore, the transformation rules are not presented generally.

## 6 Conclusions and Future Work

The use of context-awareness plays an important role in reactive and interactive systems. Context aware computing is applied in many fields such as mobile, embedded systems, etc.. Modeling and verifying context-aware systems are difficult tasks due to their complex behaviors. In this paper, we introduce a proof-based approach to model and verify such systems. The advantages of our approach are natural representation of context-aware concepts to model and the use of invariant preservation proof obligations generated by refinement mechanism in Event-B to verify the correctness of the system. However, in this paper, we just consider a simple case of context awareness. Limitation of data types in Event-B method is also a weak point when modeling complex context data.

Our future research will concentrate on elaborating the modeling systems with various kinds of context data. We are working on extending this approach with modeling uncertainty in context-awareness. Developing a tool that allows to translate context-aware systems to Event-B model automatically is also one of our future aims.

**Acknowledgments.** This work is partly supported by the research project "Methods and tools for program analysis and their applications in education", No. QGTD.13.01, granted by Vietnam National University, Hanoi.

## References

1. B method web site. http://www.bmethod.com (2013)
2. Event-b and the rodin platform. http://www.event-b.org (2013)
3. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. Int. J. Ad Hoc Ubiquitous Comput. **2**(4), 263–277 (2007)
4. Benselim, M.S., Seridi-Bouchelaghem, H.: Extended UML for the development of context-aware applications. In: Benlamri, R. (ed.) NDT 2012, Part I. CCIS, vol. 293, pp. 33–43. Springer, Heidelberg (2012)
5. Colman, A.W.: Role oriented adaptive design. Ph.D. thesis, Swinburne University of Technology (2006)

6. Ejigu, D., Scuturici, M., Brunie, L.: An ontology-based approach to context modeling and reasoning in pervasive computing. In: Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops '07, pp. 14–19 (2007)
7. Henricksen, K., Indulska, J., Rakotonirainy, A.: Modeling context information in pervasive computing systems. In: Mattern, F., Naghshineh, M. (eds.) PERVASIVE 2002. LNCS, vol. 2414, pp. 167–180. Springer, Heidelberg (2002)
8. Indulska, J., Robinson, R., Rakotonirainy, A., Henricksen, K.: Experiences in using CC/PP in context-aware systems. In: Chen, M.-S., Chrysanthis, P.K., Sloman, M., Zaslavsky, A. (eds.) MDM 2003. LNCS, vol. 2574, pp. 247–261. Springer, Heidelberg (2003)
9. Kjaergaard, M.B., Bunde-Pedersen, J.: Towards a formal model of context awareness. In: First International Workshop on Combining Theory and Systems Building in Pervasive Computing 2006 (CTSB 2006) (2006)
10. Samulowitz, M., Michahelles, F., Linnhoff-Popien, C.: Capeus: an architecture for context-aware selection and execution of services. In: Zieliski, K., Geihs, K., Laurentowski, A. (eds.) New Developments in Distributed Applications and Interoperable Systems. IFIP, vol. 70, pp. 23–39. Springer, Heidelberg (2002)
11. Mostefaoui, S.: A context model based on uml and xml schema representations. In: IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2008, pp. 810–814 (2008)
12. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. In: Proceedings of the Workshop on Mobile Computing Systems and Applications, pp. 85–90. IEEE Computer Society (1994)
13. Shehzad, A., Ngo, H.Q., Pham, K.A., Lee, S.Y.: Formal modeling in context aware systems. In: Proceedings of The 1st International Workshop on Modeling and Retrieval of Context (MRC 2004) (2004)
14. Strang, T., Linnhoff-Popien, C.: A context modeling survey. In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England (2004)
15. Tran, M.H., Colman, A., Han, J., Zhang, H.: Modeling and verification of context-aware systems. In: Proceedings of the 2012 19th Asia-Pacific Software Engineering Conference, APSEC '12, vol. 01, pp. 79–84. IEEE Computer Society, Washington, DC (2012)