EAI Endorsed Transactions
on Context-Aware Systems and Applications

Research Article ICST.ORG

Formal Modeling and Verification of Context-Aware Systems using Event-B[★]

Hong Anh Le[1,*], Ninh Thuan Truong[2]

[1]Hanoi University of Mining and Geology, Duc Thang, Bac Tu Liem, Ha Noi, Viet Nam
[2]VNU - University of Engineering and Technology, 144 Xuan Thuy, Cau Giay, Ha Noi, Viet Nam

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 context constraint preservation. 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 scenario of an Adaptive Cruise Control system.

Received on 19 October 2014; accepted on 28 November 2014; published on 16 December 2014

Keywords: context awareness, refinement-based modeling, verification, Event-B

doi:10.4108/casa.1.2.e4

1. Introduction

Context awareness is a computing paradigm that makes applications responsive and adaptive with their environment. Context-aware systems potentially determine their behavior 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 [4]. In a narrow view, a context-aware system is somehow considered as an event-driven system, i.e. it receives events emitted by context changes and responds to these changes with the providing context knowledge.

Context-aware systems often use context rules to adjust the behavior if the circumstances are changed. Furthermore, the behavior of context-ware systems is 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 [4, 17]. 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.

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 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

[★]This is the extended version of the paper published in ICCASA 2013
[*]Corresponding author. Email: lehonganh@humg.edu.vn

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. 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. A set of translation rules are proposed to define context awareness components formally. It is a refinement-based method allowing to construct the system gradually (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: Section 2 provides some background of Context awareness and Event-B. In Section 3, we introduce an incremental approach to model a context-aware system by formalizing its components using Event-B notations and making use of its refinement. 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 Section 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 [2] 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 . A basic structure of an Event-B model consists of MACHINE and CONTEXT.

An Event B CONTEXT describes a static part where all the relevant properties and hypotheses are defined. A CONTEXT consists of carrier sets, constants, axioms. Carrier sets, denoted by s, are represented by their names, and are non-empty. Different carrier sets are completely independent. The constants c are defined by means of a number of axioms $P(s, c)$ also depending on the carrier sets s.

A MACHINE is defined by a set of clauses. A machine is composed of variables, invariants, theorems and events. Variables v are representing states of the model. Invariants $I(v)$ yield the laws that state variables v must always be satisfied. These laws are formalized by means of predicates expressed within the language of First Order Predicate Calculus with Equality extended by Set Theory. Events $E(v)$ present transitions between states. Each event has the form evt = any x where $G(x, v)$ then $A(x, v, v')$ end where x are local variables of the event, $G(x, v)$ is a guard condition and $A(x, v, v')$ is an action. An event is enabled when its guard condition is satisfied. The event action consists of one or more assignments. We have three kinds of assignments for expressing the actions associated with an event: (1) a deterministic multiple assignment ($x := E(t, v)$), (2) an empty assignment (skip), or (3) a non-deterministic multiple assignment ($x :| P(t, v, x')$).

To deal with complexity in modeling systems, Event-B provides a refinement mechanism that allows us to build the system gradually by adding more details to get more precise model. A concrete Event-B machine can refine at most one abstract machine. A refined machine usually has more variables than its abstraction as we have new variables to represent more details of the model. In superposition refinement, the abstract variables are retained in the concrete machine, with possibly some additional variables. In vertical refinement such as data refinement, the abstract variables v are replaced by concrete ones w. Subsequently, the connections between them are represented by the relationship between v and w, i.e. gluing invariants $J(v, w)$.

In order to check if a machine satisfies a collection of specified properties, Event-B defines proof obligations (POs) which we must prove. Some of the proof obligations relevant to discussion here are invariant preservation (INV). INV PO means that we must prove that invariants hold after event's execution. The proof obligation is as folows: $I(v), G(w, v), A(w, v, v') \vdash I(v')$, where $I(v)$ is invariant, $G(w, v)$ is guard of the event, $A(w, v, v')$ is assignment clauses of the event.

2.2. Context-aware systems

The term "context-aware" was first introduced by Bill Schilit [15], he defined contexts as location, identities of objects and changes of those objects to applications that then adapt themselves to the context. Many works have been focused on defining terms of context awareness. Abowd *et al.* [3] defined a context aware system is a system that has the ability to detect and sense, interpret and respond to aspects of a user's local environment and to the computing devices themselves. Context-aware systems can be constructed in various methods which depend on requirements and conditions of sensors, the amount of users, and the resource available on the devices. A context model defines and stores context data in a form that machines can process. Baldauf *et al.* [4] summarized several most relevant context modeling approaches such as key-value, markup scheme, graphical object oriented, logic based and ontology based models. Chen [7] also defined three different approaches to achieving contextual data as follows

- Direct sensor access: The systems directly gather contextual data from built-in sensors. This approach does not require any additional layer but it is just suitable for simple cases not for distributed systems.
- Middle-ware infrastructure: It introduced layered architecture to separate business logic and user interfaces of the system. The system is extensible because it does not have to modify if sensors access changes.
- Context server: It allows multiple clients to use same resources. This approach is based on client-server architecture. Collected contextual data is stored in a so-called context-server. Clients use appropriate network protocols to access and use this data.

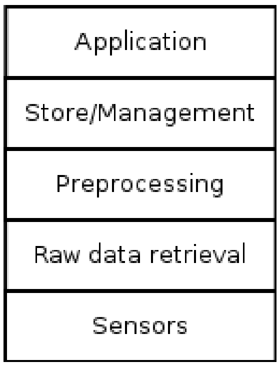


Figure 1. A layered conceptual framework for context-aware systems [7]

Figure 1 illustrates a layered conceptual framework for context-aware systems. The first layer consists of physical or virtual sensors which are able to capture context data from the environment. The second layer is able to retrieve data from sensors using providing API. Before storing the data in the fourth layer, it can be preprocessed in the third layer which is responsible for reasoning and interpreting contextual information. Finally, the application layer actually implementing reactions to different events which are raised by context changes.

In this paper, we focus on a context-aware system which directly use contextual data from physical sensors. The system senses many kinds of contexts in its working environment such as position, acceleration of the vehicle and/or temperature, weather, humidity, etc.. 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 behavior 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 very 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.

3.1. Set representation of context awareness

Firstly, we introduce a simple structure of context-aware systems consisting of five components depicted in Figure 2. A basic operation of the system is that if there is any change from the environment that can be detected using sensors, it sends events to the 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 preserve the constraints.

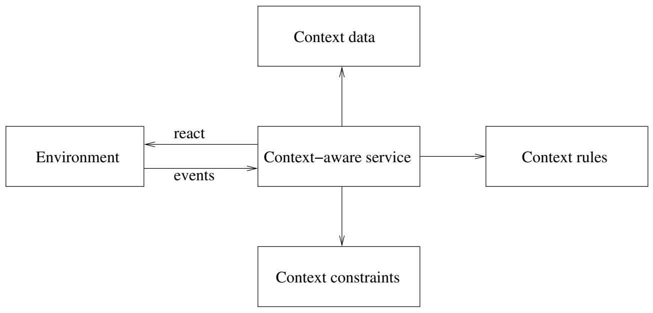


Figure 2. A simple context-aware system

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.

Definition 2 (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.

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 event e occurs in condition C then do action A. Hence, we present definitions for each element $r, r \in R$ as follows

Definition 3 (Context rules). A context rule is used for reasoning and describing the behavior of context-aware

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

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

systems. It specifies the response actions when a specific event is raised at any condition. Thus, the context rule is denoted 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. Note that, the event e should be included in the event set E.

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

Definition 4 (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.

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 Subsection 3.1. We present transformation rules between a context-aware system and an Event-B model as follows:

- Rule 1: Context data is presented by a set of context entities and their relations. The context entities be treated as a collection of sets and constants. Recall that, an Event-B context consists of set, constant and axioms clauses. The axioms clauses list various predicates of constants in the first order logic formulas. Hence, we can formalize directly a context data by an Event-B context.

- Rule 2: Each event that is emitted by the environment component is represented by an Event-B event. For example: A context-aware system uses a sensor for detecting Wind speed. The sensor regularly detects the environment and this event is sent to the core service. Then, it is then formalized by an event: *detectWind*.

- Rule 3: Since context rule structure has the form of ECA, each rule $r = \langle e, a, c \rangle$ is mapping to an Event-B event. Where e is event that the system senses or received from its environment, c is the additional conditions for reasoning. More precisely, conjunction of e and c are guards of Event-B event while a is mapped to the body of the event (see Table 1). All these events are included in either Event-B abstract machines or a refined ones.

- 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. For example, more sensors are attached in the system to get various kind of context data. The system also has more context rules to handle with these data. The updated system still has to satisfy context constraints which has been established. Therefore, it requires to have a suitable modeling method for incremental development. As we have described in Subsection 3.2, a context-aware system is translated to an abstract Event-B model. It is apparently suitable for modeling the initial stage of a context-aware system. 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 already know that Event-B provides both superposition refinement and vertical refinement. In the former, the abstract variables are retained in the concrete machine, with possibly some additional variables, hence it is suitable for modeling a context-aware system which is often extended by adding new sensors.

- Static part: For example, when a new sensor is added to the system, we may have to deal with new context data types. The context data can be an extension of the old one. Applying Rule 1, we formalize it as a new Event-B context which extends the ones in abstract model.

- Dynamic part: We begin with abstract machines to model the general behavior of the very beginning system, after that we refine these machines by concrete ones to represent new requirements of the systems. For instance, adding new sensors usually generate new appropriate events. Hence, the system also needs more rule to react to these events. In the refined machines, new added variables can refer to new context data elements. The events of a new refined machine can refine the abstract ones to describe the system more precisely. With the dynamic part, we need to prove that a new model in the increment step needs to satisfy the context constraints including ones defined in the early stages. That proof

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

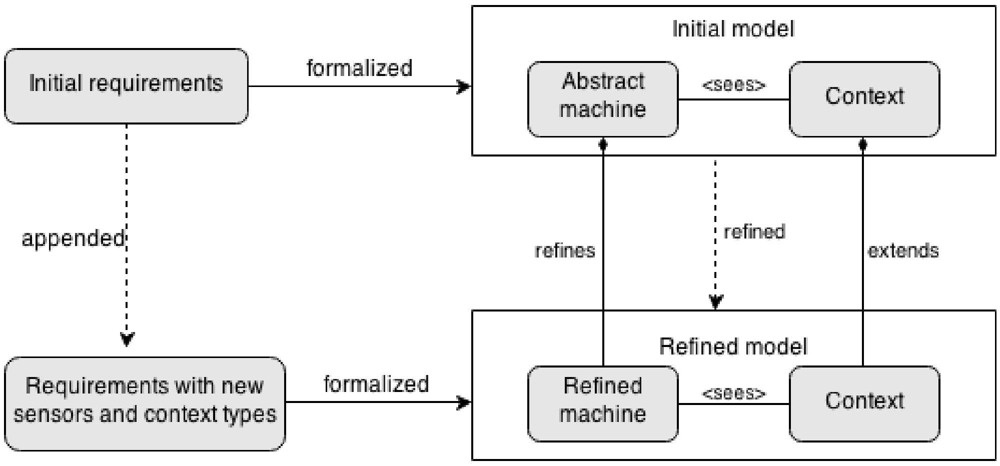


Figure 3. Incremental modeling using refinement

can be achieved by using *INV* proof obligations which states that invariant preservation PO can be checked at any refinement. According to Rule 4 in Section 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.

The proposed incremental modeling method is illustrated in Figure 3.

4. A case study: 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 the modeling method presented in Section 3.2 and then we verify context constraint preservation properties of the system.

4.1. Initial description

ACC controls car's speed is based on the driving conditions which are enhanced with a context-aware feature such as target detection. The ACC system use a sensor to detect target in front of the car. The car has a maximum speed which is initially set to a value. If the car does not detect a target then ACC increase the speed, other wise decreases the speed with constant amount. If the car is stopped and there is no target detected then it is resumed with initial speed.

The ACC must conform to a context constraint such that the speed is always in safe range, i.e the speed is less or equal to the maximum speed.

4.2. Modeling ACC system

In this scenario, there are three sensors, following the approach presented in Section 3, we specify the initial system with one abstract machine and one context, namely *ACC_M*0 and *Target*.

- Applying Rule 1, context *Target* represents context data received from the target detection sensor. More precisely, the information sent by the sensor let us know whether there is an object which is currently in front of the car. We thus formalize the context entities as a set *TARGET_DETECTION* which is equivalent to *BOOL* set.

- The sensor will periodically send context data to the system by emitting events. We need to check if context data contains the information of obstacle objects. Hence, we specify two Event-B events *TargetDetected*, *TargetUndetected* correspondingly.

- Context rules are used for specifying the system behavior which depends on the context. We attract three rules from the initial description and translate them to three corresponding events following Rule 3.

- Applying Rule 4, the context constraint is translated to invariant *CXT_CSTR*.

Moreover, a variable *speed* in the model specifies the speed of the car. The abstract context and abstract machine Event-B specification of the ACC system are partially illustrated in Figure 4 and Figure 5 respectively.

CONTEXT Target

CONSTANTS

$TARGET\_DETECTION$

MAX_SPEED

INC

AXIOMS

axm1 : $TARGET\_DETECTION = BOOL$

axm2 : $MAX\_SPEED \in \mathbb{N}$

axm3 : $INC < MAX\_SPEED$

axm4 : $INC \in \mathbb{N}$

END

Figure 4. Abstract context of initial ACC system

Strengthen guards: With the initial description, the generated *INV* POs of the translated Event-B model are failed to prove. More precisely, with event *TargetDetected*, PO *TargetDetected/CXT_CSTR/INV* is generated as follows:

$speed < MAX\_SPEED \land target = FALSE \vdash speed + INC\_SPEED < MAX\_SPEED$

It is failed if $MAX\_SPEED + INC\_SPEED < speed < MAX\_SPEED$ before event execution.

To make the model more precise, we strengthen the context rules by adding more conditions. i.e. the event guards have more clauses (Figure 6)

4.3. Refinement: Adding weather and road sensors

At this stage, the system has more sensors to be aware more precisely of the current context situations. Weather and road sensors are attached to the system. Similarly to target detection sensor, they send the context data periodically to the system. Context rules of the system are also extended for reacting to new added sensors as follows: "When a car travels in a raining condition or sharp bend, ACC reduces car's speed". With new sensors, the system need to fulfil

MACHINE ACC_M0

SEES Target

VARIABLES

speed

target_det

INVARIANTS

inv1 : $speed \in \mathbb{N}$

inv2 : $target\_det \in TARGET\_DETECTION$

inv3 : $speed \le MAX\_SPEED$

EVENTS

Initialisation

begin

act1 : $speed := MAX\_SPEED$

end

Event *TargetDetected* $\widehat{=}$

when

grd1 : $target\_det = TRUE$

then

act1 : $speed := speed - INC$

end

Event *TargetUndetected* $\widehat{=}$

when

grd1 : $target\_det = FALSE$

then

act1 : $speed := speed + INC$

end

END

Figure 5. Abstract machine of initial ACC system

the constraint such as "The speed can not be equal to maximal speed if it is raining or the road is sharp".

Refined model: Following the method presented in Section 3.3, context *Weather_Road* extending context *Target* represents context data of two new sensors. We use two constant sets *RAINING* and *SHARP* to represent context data from sensors. Machine *ACC_M1* is concrete

EVENTS

Event $TargetDetected \mathrel{\widehat{=}}$

when

$\mathbf{grd1} : target\_det = TRUE$
$\mathbf{grd2} : speed > INC$

then

$\mathbf{act1} : speed := speed - INC$

end

Event $TargetUndetected \mathrel{\widehat{=}}$

when

$\mathbf{grd1} : target\_det = FALSE$
$\mathbf{grd2} : speed < MAX\_SPEED - INC$

then

$\mathbf{act1} : speed := speed + INC$

end

END

Figure 6. Events with strengthened guards

machine specifying the behavior of the system updated with new requirements.

Since two new rules are appended to the requirements, two corresponding events are added for this machine. The first one representing a new added rule is not extended. This event *RainSharp* describes the behavior of the system when sensors send data indicating that it is raining or the road is sharp. While the second one *TargetUndetected* refines event of the abstract model. The context constraint is formalized as an invariant *cxt_ct*. The Event-B specification of the refined machine and extended context is partially illustrated in Figure 7 and Figure 8 respectively.

MACHINE ACC_M1

REFINES ACC_M0

SEES Weather_Road

VARIABLES

$\mathbf{isRain}$
$\mathbf{speed}$
$\mathbf{target\_det}$
$\mathbf{isSharp}$

INVARIANTS

$\mathbf{inv1} : isRain \in RAINING$
$\mathbf{cxt\_ct} : isRain = TRUE \lor isSharp = TRUE \Rightarrow speed < MAX\_SPEED$
$\mathbf{inv3} : isSharp \in SHARP$

EVENTS

Initialisation

begin

skip

end

Event $TargetUndetected \mathrel{\widehat{=}}$

extends $TargetUndetected$

when

$\mathbf{grd1} : target\_det = FALSE$
$\mathbf{grd2} : speed < MAX\_SPEED - INC$
$\mathbf{grd3} : isRain = FALSE$
$\mathbf{grd4} : isSharp = FALSE$

then

$\mathbf{act1} : speed := speed + INC$

end

Event $RainSharp \mathrel{\widehat{=}}$

when

$\mathbf{grd1} : isRain = TRUE \lor isSharp = TRUE$

then

$\mathbf{act1} : speed := speed - INC$

end

END

Figure 7. Refined machine for ACC system

```
CONTEXT  Weather_Road
EXTENDS  Target
CONSTANTS
        RAINING
        SHARP
AXIOMS
        axm1 : RAINING =
            BOOL
        axm2 : SHARP = BOOL
END
```

Figure 8. Extended context for ACC system

4.4. Verifying the system's properties

The system should always satisfy context constraint preservation properties. It means that the context constraint is preserved before and after using the context rules to adapt context changes. With proposed method, context constraints are translated to invariant clauses. Consequently, we prove the system's correctness by proving proof obligations of such invariants.

The proof obligations (PO) for these invariants of both abstract and refined machines as follows:

- Machine *ACC_M0*: "*TargetDetected/ctx_ct1/INV*" (Figure 3) and "*TargetUndetected/ctx_ct1/INV*"
- Machine *ACC_M1*: "*TargetUndetected/ctx_ct/INV*" and "*RainSharp/ctx_ct/INV*"

These POs are generated and proved automatically with the Rodin tool as illustrated in Figure 9. Hence, the ACC system always satisfies predefined context rules.

5. Related work

Many papers have been proposed for modeling and verifying context-aware systems with various approaches. Key-value data structure [13, 15] has been used in early works as the simplest method for modeling context awareness. This method exposes many problems since it lacks of interoperability, representation and reasoning mechanism.

Most research efforts that are based on mark-up scheme model have defined and extended markup languages. Henricksen *et al.* [10] proposed to represent contextual data by *Comprehensive Structure Context Profiles* (CSCP). Indulska *et al.* [11] 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 following the graphical model approach to model contextual data. Mostefaoui [14] presented a three-layered data model for context. Benselim and Hassina [5] 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.* [16] introduced a formal modeling method in context aware systems using OWL. Ejigu *et al.* [9] 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.

Besides, Kjaergaard *et al.* [12] proposed a CONAWA calculus that provides mechanism for modeling and interwovenning sets of context-information. However, this approach has some limitations such as probabilistic context information modeling and verification of the system is not discussed yet.

More recently, Tran *et al.* [18] introduced a ROAD4Context framework which is based on Role-Oriented Adaptive Design (ROAD) [8] 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.

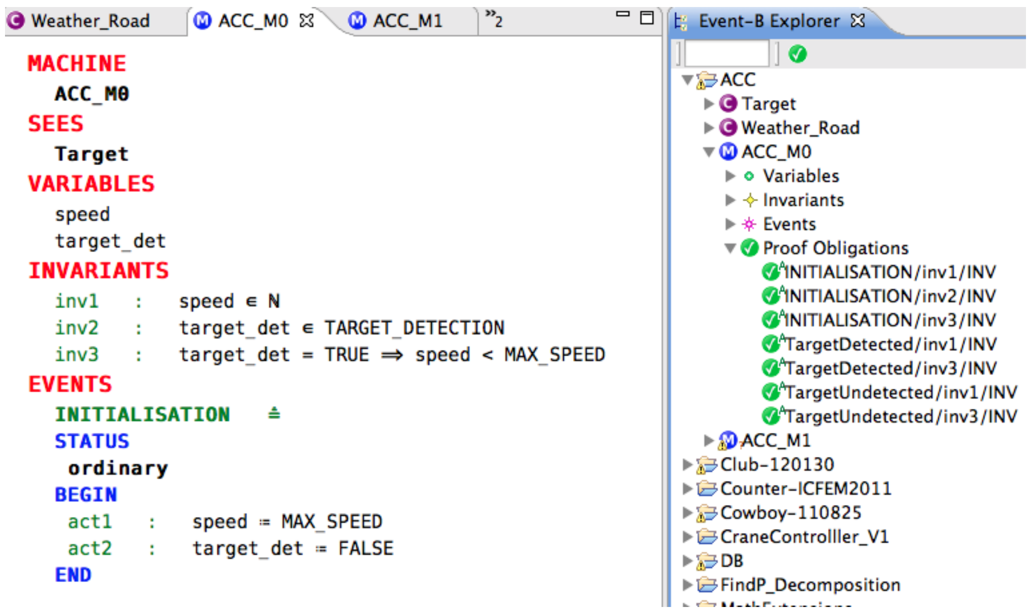
In comparison to these works, our method is different as we use Event-B as a modeling method. The advantages of our method are that a context-aware system can be modeled naturally by Event-B notations because the ECA form of context rules is mapped directly to an event. The Event-B refinement allows to develop a context-aware system gradually and ensures the correctness in each refinement stage. After the modeling, the verification process does not require any more translation step. Context constraints are proved mathematically by proving proof obligations which are automatically generated and proved in the supporting tool Rodin.

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 refinement-based approach to model and verify such systems. The advantages of our approach are natural representation of context-aware concepts to model and

Table 3. Proof of context constraint preservation

$target\_det = TRUE \Rightarrow speed < MAX\_SPEED$ $target\_det = TRUE$ $speed > INC$ $\vdash$ $target\_det = TRUE \Rightarrow speed - INC < MAX\_SPEED$	TargetDetected/ctx_ct1/INV



```
MACHINE
  ACC_M0
SEES
  Target
VARIABLES
  speed
  target_det
INVARIANTS
  inv1  :  speed ∈ ℕ
  inv2  :  target_det ∈ TARGET_DETECTION
  inv3  :  target_det = TRUE ⇒ speed < MAX_SPEED
EVENTS
  INITIALISATION  ≙
  STATUS
   ordinary
  BEGIN
   act1  :  speed ≔ MAX_SPEED
   act2  :  target_det ≔ FALSE
  END
```

```
▼ ACC
  ▶ Target
  ▶ Weather_Road
  ▼ ACC_M0
    ▶ Variables
    ▶ Invariants
    ▶ Events
    ▼ Proof Obligations
        INITIALISATION/inv1/INV
        INITIALISATION/inv2/INV
        INITIALISATION/inv3/INV
        TargetDetected/inv1/INV
        TargetDetected/inv3/INV
        TargetUndetected/inv1/INV
        TargetUndetected/inv3/INV
  ▶ ACC_M1
▶ Club-120130
▶ Counter-ICFEM2011
▶ Cowboy-110825
▶ CraneControlller_V1
▶ DB
▶ FindP_Decomposition
```

Figure 9. Checking properties in Rodin

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 which can be done by incorporating new theory plug-ins [6]. 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 project no. QG.12.50 granted by Vietnam National University, Hanoi (VNU).

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] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, HUC ’99, pages 304–307, London, UK, UK, 1999. Springer-Verlag.

[4] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. *Int. J. Ad Hoc Ubiquitous Comput.*, 2(4):263–277, jun 2007.

[5] M. Benselim and H. Seridi-Bouchelaghem. Extended uml for the development of context-aware applications. In R. Benlamri, editor, *Networked Digital Technologies*, volume 293 of *Communications in Computer and Information Science*, pages 33–43. Springer Berlin Heidelberg, 2012.

[6] M. Butler and I. Maamria. Practical theory extension in event-b. In *Theories of Programming and Formal Methods*, volume 8051, pages 67–81. Springer Berlin Heidelberg, 2013.

[7] H. Chen. *An Intelligent Broker for Context-Aware Systems*. PhD thesis, University of Maryland, 2004.

[8] A. W. Colman. *Role oriented adaptive design*. PhD thesis, Swinburne University of Technology, 2006.

[9] D. Ejigu, M. Scuturici, and L. Brunie. An ontology-based approach to context modeling and reasoning

in pervasive computing. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pages 14–19, 2007.

[10] K. Henricksen, J. Indulska, and A. Rakotonirainy. Modeling context information in pervasive computing systems. In *Proceedings of the First International Conference on Pervasive Computing*, Pervasive '02, pages 167–180, London, UK, UK, 2002. Springer-Verlag.

[11] J. Indulska, R. Robinson, A. Rakotonirainy, and K. Henricksen. Experiences in using cc/pp in context-aware systems. In *Proceedings of the 4th International Conference on Mobile Data Management*, MDM '03, pages 247–261, London, UK, UK, 2003. Springer-Verlag.

[12] M. B. Kjaergaard and J. Bunde-Pedersen. Towards a Formal Model of Context Awareness. In *First International Workshop on Combining Theory and Systems Building in Pervasive Computing 2006 (CTSB 2006)*, 2006.

[13] C. L.-P. Michael Samulowitz, Florian Michahelles. Capeus: An architecture for context-aware selection and execution of services. In K. ZieliÅĎski, K. Geihs, and A. Laurentowski, editors, *New Developments in Distributed Applications and Interoperable Systems*, volume 70 of *IFIP International Federation for Information Processing*, pages 23–39. Springer US, 2002.

[14] S. Mostefaoui. A context model based on uml and xml schema representations. In *Computer Systems and Applications, 2008. AICCSA 2008. IEEE/ACS International Conference on*, pages 810–814, 2008.

[15] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *In Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 85–90. IEEE Computer Society, 1994.

[16] A. Shehzad, H. Q. Ngo, K. A. Pham, and S. Y. Lee. Formal modeling in context aware systems. In *In Proceedings of The 1 st International Workshop on Modeling and Retrieval of Context (MRCâĂŹ2004)*, 2004.

[17] T. Strang and C. Linnhoff-Popien. A context modeling survey. In *In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England*, 2004.

[18] M. H. Tran, A. Colman, J. Han, and H. Zhang. Modeling and verification of context-aware systems. In *Proceedings of the 2012 19th Asia-Pacific Software Engineering Conference - Volume 01*, APSEC '12, pages 79–84, Washington, DC, USA, 2012. IEEE Computer Society.