SPECIFICATION AND VERIFICATION OF WORKFLOW APPLICATIONS USING A COMBINATION OF UML ACTIVITY DIAGRAMS AND EVENT B

Ahlem Ben Younes and Leila Jemni Ben Ayed
Research Unit of Technologies of Information and Communication (UTIC) ESSTT, Bab Manara, Tunisia

Keywords: Specification, Formal Verification, Validation, UML, Event B, Workflow Applications.

Abstract: This paper presents a transformation of UML activity diagrams (AD) into Event B for the specification and the verification of workflow applications. With this transformation, UML models could be verified by verifying derived event B models, automatically, using the B powerful support tools like B4free. The workflows is initially expressed graphically with UML AD and translated into Event B. The resulting model is then enriched with Invariants/Assertions describing functional properties of workflow models such as deadlock-inexistence. We present translation rules of UML AD into EventB, and we propose also a translation process of UML AD into EventB specifications based on the refinement technique of Event B to encode the hierarchical decomposition in UML AD. Also, we propose a solution to specify time in Event B, and by an example of workflow application, we illustrate the proposed technique.

1 INTRODUCTION

The workflow applications are characterized by a high complexity. Increasingly, they have to obey to realiabity, safety, timed requirements. Today, UML AD (Johason, 1998) are considered as an OMG standard notation in the area of workflow applications modelling (Dumas, and Hofstede, 2001). However, the fact that UML lacks a precise semantics is a serious drawback of UML-based techniques. Also, UML AD is not adapted to the verification of workflow applications. In previous works (Ben Younes and Jemni Ben Ayed, 2007,2008), we have proposed a specification and verification technique for workflow applications using a combination of UML AD and Event B. The work presented in this paper is a part of them. The proposed approach gives readable models and an appropriate formal method which allows verification of required properties (no_deadlock, liveness, fairness) to prove the correctness of the workflow specification. In this context, several solutions have been proposed. Some of them use model checking for the verification. Van der Aalst (Van der Aalst, 2000) proposed a technique which uses Petri nets for the verification of the correctness of workflow applications using a compositional verification approach. Karamanolis and al (Karamanolis and all, 2000) use process algebra for the verification of workflow properties. Our contribution, in this context, consists of using event B method and its associate refinement process and tools for the formal verification of workflow applications. The verification is based on a proof technique and therefore it does not suffer from the state number explosion occurring in classical model checking as in the cases of works in (Van der Aalst, 2000), (Guelfi and Mammar, 2005) and (Karamanolis and all, 2000). The Event B method (Abrial, 1996b) is a variant of the B formal method (Abrial, 1996a), proposed by Abrial to deal with distributed, parallel and reactive systems (Abrial, 1996b). B models provides an automatic proof which convince the user that the system is effectively correct and satisfies properties which are presented as invariants/assertion. The concept of refinement is the key notation for developing B models. The refinement of a formal model allows one to enrich the model in step by step approach. The last refinement gives the implementation machine which map directly to a programming language such as C or ADA. The strong point of B is support tools like as *AtelierB* (Clearsy,2001) or *B4free* (Clearsy,2004), an academic version of *AtelierB*. Most theoretical aspects of the method, such as the formulation of

Ben Younes A. and Jemni Ben Ayed L. (2010).
SPECIFICATION AND VERIFICATION OF WORKFLOW APPLICATIONS USING A COMBINATION OF UML ACTIVITY DIAGRAMS AND EVENT B.
In *Proceedings of the 5th International Conference on Software and Data Technologies*, pages 312-316
DOI: 10.5220/0003012003120316

proof obligations, are carried out automatically. The automatic and interactive provers are also designed to help specifiers to discharge the generated proof obligations. All of these points make B well adapted to large scale industrial projects (Behm,1998). However, B is still difficult to learn and to use.This is why we have proposed in our previous work (Ben Younes and Jemni Ben Ayed, 2007) an approach which combines the use of UML AD and Event B for the specification and the verification of workflow applications. The workflow is initially modeled graphically with UML AD (Step1). After that, the resulting graphical readable model is translated into Event B in incremental development with successive refinements (Step2). This refined model is enriched by relevant properties (no deadlock, no livelock, strong fairness, etc) (Step3) which will be proved using the *B4free* tool (Clearsy,2004) (step4). So, this allows one to rigorously verify semi-formal specifications in AD UML by analysing derived Event B models. On the other hand, we can use AD UML specifications as a tool to develop Event B specifications. In our works (Ben Younes and Jemni Ben Ayed, 2007,2008) , we have presented the translation process which uses the B method refinement and proposed translation rules for the basic concepts of UML AD (activity, Sequence of activities, choice (decision), loop parallel activities (fork and join) and atomic process) and also for dynamic invocations concept (Ben Younes and Jemni Ben Ayed, 2008) into Event B. In this paper, we discuss contribution of our proposed approach for the verification of workflow applications and we extend our work presented in (Ben Younes and Jemni Ben Ayed, 2007) by adding new translation rules for the synchronization in UML AD (event, send/receive concepts) into Event B. Also, we propose in this paper a solution to specify time in the event B method and derivation of temporal expressions in UML AD (timeout) into Event B. These translation rules give not only a syntactical translation, but also give a formal semantics using the Event B method semantics for the activity diagrams. In this context, there have been efforts for defining semantics for activity diagram in the works of Eshuis (Eshuis and al, 2001, 2004) and also the works of (Guelfi and Mammar, 2005). However, these works not consider the hierarchical decomposition of activities in UML AD, and suffer from the state number explosion. Moreover, in Eshuis (Eshuis and al, 2004, 2001) approach, no details are given about how time is defined. Although, the work of (Guelfi and Mammar, 2005) propose a systematic way for translating the semantic of timed activity Diagrams into the PROMELA input language of the SPIN model checker, but they no consider the hierarchical decomposition of activities in UML AD, no translation rules are given about the refinement in UML AD. Our contribution, in this context, consists of using Event B method and its associate refinement process to encode the hierarchical decomposition of activities in UML AD and tools for the formal verification of workflow applications. Moreover, in the refinement of B, it is not needed to re-prove these properties again while the model complexity increases. Notice that this advantage is important if we compare this approach to classical model checking where the transition system describing the model is refined and enriched like in SPIN model checker. This paper is structured as follows. Section 2 presents derivation rules of event, send/receive event and time in UML AD into Event B notation. By an example we illustrate our contribution in section3. Finally, a summary of our work concludes the paper

2 TRANSLATION FROM UML AD TO EVENT B

A- The translation of the send event action into Event B

In Event B, we translate the send of an internal event by the definition of new variable *v_Name_Evt* for each new internal event Name_Evt. This variable takes the value *TRUE* if the event occurs and *FALSE* in the other case.

B- The translation of the receive event action in UML AD into Event B

In Event B, we translate the receive event action by: The definition of a new boolean variable *v_Name_Evt* for each event Name_Evt generated by the environment. The definition of a new variable *hand;*the generation of an event *Detect_Evt*. The B event *Detect-Evt* allows all event (for example v-E) to have a random value. It simulates the event detection when the detection system has the control. The control is given alternatively to the detection system when *hand =1* and to the control system in the other cases.

C- The representation of the time in Event B

The timeout expressed in B, will impose alternation between the clock, the control system and the detection system. We use the variable *hand* and the control is given alternatively to the clock when

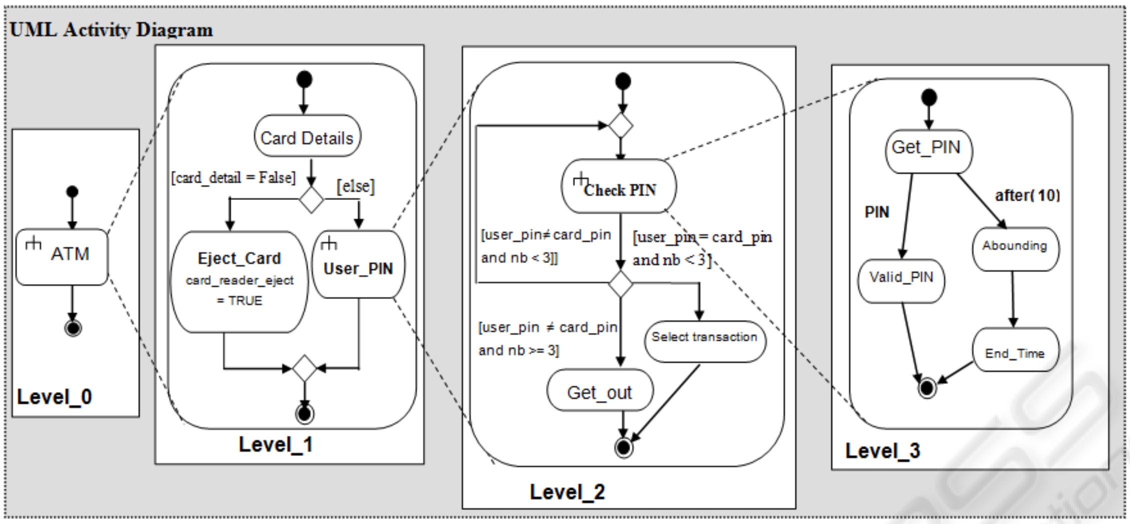
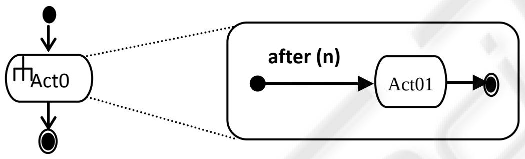


Figure 1: The UML AD model of the workflow ATM application.

hand=2 and to the system in the other cases. In UML AD, temporal event is specified with the after keyword (see Fifure1). The after (n) event expression, where n is a positive integer, means that n time units after the source of the edge was entered a timeout is generated. The formulation of the timeout in Event B is based on some derivation rules that we already introduced.



We propose to drift the timeout by: the definition of a integer variable Timeout which represents the time of the next generation of the event timeout; the definition of the event Evt_Init for initializing the variable Timeout with the value of the current time and the duration n (Exemple n =10 time unit); the definition of the B event Evt_Wait. Initially, hand=0. The event Evt_Init sets this variable to 1, and initializes the variable timeout. This passes the control to Evt_wait or Evt_Act01(Associated to the activity Act01). If (time < timeout)) then the variable hand passes to 1 (event Evt_Wait) to increment the time (event tick) but if (time>= timeout) this (Evt_Act01) is allows the following activity Act01 to be execute.

3 EXAMPLE

We illustrate the proposed technique over an example of a workflow application, which we have implemented using Event B tool *B4free*. This application represents a simplified **ATM login**. **Step1:** Initially, we describe this workflow application using UML. The resulting model is composed of three decomposition levels (See Figure 1). **Step2:** In the second step, by the application of the translation process and using the translation rules. Three refinement steps which correspond to each level of three level of decomposition in the UML AD model are necessary. **Step3 (Verification and Validation of the ATM Login application):** The **ASSERTIONS** clause contains liveness properties expressing that there is ***no deadlock.*** This property is ensured by asserting that the disjunction of all the abstract events guards implies the disjunction of all the concrete events guards. This guaranties that the new events can be fired *(no deadlock*). In each new refinement, we add this property**.** The **INVARIANT** clause allows to express the safety properties (called safety invariant) and the typing information (Typing invariant). Each refined model is enriched by relevant properties (safety, liveness, ect) which will be proved using the *B4free* tool. These properties shall remain true in the whole model and in further refinements: It is not needed to re-prove again verified properties in the refined model while the model complexity increases. It is the advantage of using *B4free* tool. For example:

- The safety property that the system ejects the card reader only if the card details are false: this property is added in the resulting refined model **Ref1_ATMcard** (associated to the **LEVEL1**) in the clause INVARIANT as follows:

```
REFINEMENT  Ref1_ATMcard
REFINES ATMcard
INVARIANT
   /* Safety properties*/
( card_reader_eject = TRUE =>  card_detail= FALSE )
```

- The temporal property T1 (the system should not be continuously open for more than 10 seconds without the even PIN present) can be proved by adding the safety invariant, in the resulting refined model **Ref3_ATMcard** (associated to the **LEVEL3** in UML AD model), which expresses that if the system is in the node Abounding (*pin_state =1*), then necessarily the deadline has arrived:

```
REFINEMENT  Ref3_ATMcard ………………….
REFINE Ref2_ATMcard
INVARIANT   /*Temporal properties*/
            (pin_state = 1) ⇒  (time >= timeout) /* T1*/
```

In the refinement **Ref3_ATMcard**, the event *Tick* maintains the control and allows time advance. In this way, we avoid the *Livelock* problem in the construction of a resulting B system. By the application of the translation rule for the time (see section 4.2), we use the variable *hand* (see section 4.2.B) and the control is given alternatively to the clock when *hand=2*, to the system when *hand= 0*, and to the detect system when *hand=1*. The variable *hand* describes the events interleave and prevent that an event is fired infinitely (an event will be infinitely crossed in detriment of others).

```
REFINEMENT  Ref3_ATMcard
………………….;
VARIABLES
hand, time, evt_pin, pin_state, timout …..
INVARIANTS
 hand ∈ {0,1,2} ∧ evt_pin ∈ BOOL∧ time ∈ N∧ timeout∈ N
∧ pin_state∈{0,1,2,3 }     /* the state variable pin_state is associated to the
composed activity   Chek_PIN*/
INITIALISATIOIN
hand:= 0 || evt_pin:=FALSE|| time :=0|| pin_state:=3……

EVENTS
Dect_Evt=   SELECT hand =1  THEN hand := 0 || evt_pin :∈ BOOL
END;
Tick=  SELECT hand =2 ∧ pin_state=2
       THEN hand :=1|| time := time+1   END;
Evt_GetPin=  SELECT hand=0∧ pin_state=3  ….
             THEN hand :=1|| pin_state :=2|| timeout :=  time+ 10
              END;
Evt_Wait_Pin =  SELECT hand=0∧ pin_state=2 ∧ evt_pin = FALSE
                  ∧   time < timeout…
                 THEN hand := 2  END;
Evt_ValidPin= SELECT hand=0∧ pin_state=2 ∧ evt_pin = TRUE ∧
                 time < timeout
               THEN pin_state :=0
              END;
Evt_Abounding= SELECT hand=0∧ pin_state=2 ∧ evt_pin = FALSE
              ∧   time >=timeout
             THEN pin_state := 1  END;
```

4 CONCLUSIONS

In this paper we have proposed a specification and verification technique for workflow applications using UML AD and Event B. We have extended our work (Ben Younes and Jemni Ben Ayed, 2007,2008) and have proposed new derivation rules for the synchronization in UML AD (event, send/receive concepts) into Event B. Also, we propose in this paper a solution to specify time in the event B method and derivation of temporal expressions in UML AD (timeout) into Event B. In our approach, variants are defined to ensure the correct firing order of events in these models. Also, decreasing variants are defined to solve livelock problem. These translation rules give not only a syntactical translation, but also a formal semantics using the Event B method semantics for the UML AD. Currently, we are working on the implementation of this approach. Another thing needed to be mentioned is that we just formalize the subset of UML activity diagrams. For instance, object flows do not be included in our model. However, our approach is also suitable for formalizing it.

REFERENCES

Johason.R, I. Jacobson, and G.Booch, 1998. “The Unified Modelling Language reference Manual” .Addison-Wesley,.

Ben Younes, A and L Jemni. Ben Ayed , 2007 “ Using UML Activity Diagrams and Event B for Distributed and Parallel Applications”. In 31st Annual IEEE International Computer Software and Applications Conference (COMPSAC 2007), Volume 1.

Dumas, M. and ter Hofstede, A. H. M., 2001 “ UML activity diagrams as a Workflows Specification language ”. In UML2001 page 76-90. Spinger-Verlag.

Clearsy, 2001. System Engineering Atelier B, Version 3.6,.

Abrial. J. R, 1996a “The B Book. Assigning Programs to Meanings”. Cambridge University Press.

Clearsy 2004, “B4free,” Available at http:// www.b4free.com.

Eshuis, R., and Wieringa R., 2001. A formal semantics for UML Activity Diagrams – Formalising workflow models, Technical Report. Twente, Dept. Of Computer Science.

Abrial J-R., 1996b.” Extending B without changing it” (for developing distributed systems)”. In H Habrias, editor, First B Conference.

Van der Aalst, W. M. P., 2000 “Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques”, in Business process management: models, techniques, and empirical studies. Lecture Notes in Computer Science 1806, Springer-Verlag.

Karamanolis, C., Giannakopoulou D., Magee, J., and S. M. Wheater, 2000 "Formal verification of workflow schemas," University of Newcastle, Tech. Rep.

Ben Younes, A and Jemni. Ben Ayed, L .2008 "From UML Activity Diagrams to Event B for the Specification and the Verification of Workflow Applications". In 32st Annual IEEE International Computer Software and Applications Conference (COMPSAC 2008).

Eshui R., and Wieringa. R. 2004. Tool Support for verifying UML Activity Diagram, IEEE transaction on software Engineering , vol 30 , N°7;

Guelfi N, and Mammar A. 2005. ''A Formal Semantics of Timed Activity Diagrams and itsPROMELA Translation''. In the 12th Asia-Pacific Software Engineering Conference (APSEC'05).