Using UML Activity Diagrams and Event B for the Specification and the Verification of Workflow Applications

Ahlem Ben Younes and Leila Jemni Ben Ayed

Research Unit of Technologies of Information and Communication (UTIC)
ESSTT, Tunisia

Abstract. This paper presents a specification and formal verification technique for workflow applications using UML Activity Diagrams (AD) and Event B. The workflow application is initially modeled graphically hierarchically with UML AD, then the resulting model is translated into Event B in order to check the correctness of workflow models (such as no deadlock, no livelock, fairness,.) automatically, using the B support tools. In this paper, we discuss the contributions and by an example of workflow application, we illustrate the proposed technique.

1 Introduction

Workflow modelling needs a language that is intuitive and easy to use. Activity diagrams of UML [1] provide a good option. Today, UML AD are considered as an OMG standard notation in the area of workflow applications modelling [3]. 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 this paper, our goal is to provide a specification and verification technique for workflow applications using UML AD which give readable models and an appropriate formal method which allows verification of required properties (such no deadlock, liveness, fairness) to prove the correctness of the workflow specification. Our contribution 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 [10] and [11]. In our previous work [2], we have proposed an approach which combines the use of UML AD and Event B for the specification and the verification of workflow applications. Hence, a semi-formal specification in UML AD could be verified by analysing derived Event B models. 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 [6] (Step4). In our works [2][13], we have presented the proposed translation rules for the

Ben Younes A. and Jemni Ben Ayed L. (2009).
Using UML Activity Diagrams and Event B for the Specification and the Verification of Workflow Applications.
In *Proceedings of the 7th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems*, pages 149-155
DOI: 10.5220/0002202001490155

basic concepts of UML AD (activity, Sequence of activities, choice (decision), loop, parallel activities (fork and join)) and for dynamic invocations concept [13] into Event B. In this paper, we discuss contribution of the proposed approach for the verification of workflow applications. 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 addition, in this paper, by an example of workflow application '' The Production Company'', we illustrate the proposed technique and the feasibility of our approach.

This paper is structured as follows. Section 2 discusses related work to ours. Section 3 presents a brief overview of the Event B method. Section 3 describes our approach for the translation of hierarchical decomposition in UML AD into a hierarchy of Event B models. Section 5 discusses the interest of this approach for the verification and the validation of workflow applications. Then an example illustrating our approach is given. Finally we conclude and give an overview of our future work.

2 Related Work

Modeling of Workflow Applications. Some related works have proposed to use Petri nets as a standard language for workflow modelling [15][12]. Considering classical Petri nets are not powerful enough for modelling workflows, Van Der Aalst and al have elevated it to high level Petri nets by adding time, colour, and hierarchy [12]. The problem with this is that still Petri net is not an easy language for modeling workflows. Moreover, there are not many results available with high level Petri nets. Today, UML AD is considered as an OMG standard notation in the area of workflow applications modelling [3]. Eshuis et al. [14] argue that Petri Nets may be unable to model workflow activities accurately without extending its semantics and this drawback has been addressed in UML activity diagrams.

Formal Verification of Workflow Applications. Van Der Aalst and al [10] discuss how to use Petri Net to model and analyse workflow processes . Karamanolis and al [11] use process algebra for the verification of correctness of workflow. In our works, our goal is to provide a specification and verification technique for workflow applications using UML AD which give readable models and an appropriate formal method which allows verification of required properties (such no deadlock) to prove the correctness of the workflow specification. Indeed, the main problem with UML activity diagrams is that they have no formal semantics. In this context, there have been efforts for defining semantics for activity diagram in the works of Eshuis [7]. However, these works not consider the hierarchical decomposition of activities in UML AD, and suffer from the state number explosion. 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. In addition, Event B allows the use of arbitrary natural number using the $:\in$ operators. The possibility of using arbitrary natural numbers allows to deal with all the possible case for activity/process description and modeling. Notice that this is almost impossible in model checking techniques [10][7][11], where a fixed value for the natural numbers is required. Usually the state number explosion problem arises when this natural number increase.

3 The Event B method

We use the B method [5] and its event based definition [9] to formalize UML AD models of workflow application.

Event B Models. An Event B model is composed of a set atomic events described by particular generalized substitution (**ANY**, **BEGIN** and **SELECT**). Each event *Evt* is fired if the guard *P* associated to this event is true. For the purpose of this paper, we will only use the **SELECT** substitution *Evt*= **SELECT** *P* **THEN** *G* **END**. Moreover, a B model contains a set of properties i.e invariants, liveness, safety and reachability properties which can be prove during the development thanks to the embedded proof system associated to B and the tool supported by *B4free* [6]. Finally, B models can be refined into other B models which can be enriched by new events and new properties.

Design with Event B. A set of events is described to define a transition system that allows to represent the workflow application to be specified. In the case, of a workflow application described by several sub-system (sub-process), our approach uses the refinement technique to introduce the events of the composed automate (workflow). Each system is then described progressively by refinement in an incremental way. Robustness and reacheability were expressed and checked according to the B method. Moreover, in the refinement, 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.

Finally, a strong point of the B method is that the B support tools like *B4free* [6] provide utilities to discharge automatically the generated proof obligations (of the invariant preservation and the refinement correctness). Analyzing the non-discharged proof obligations with the B support tools is an efficient and practical way to detect errors encountered during the specification development.

4 The Translation Process from UML AD into Event B

The proposed translation process, uses the refinement process of Event B to describe composition of AD: to each decomposition level of an activity (workflow process), which corresponds to a subactivity in UML AD notation, is associated an Event B refinement. In our approach, each subactivity Act0, composed, for example, of two activities Act01 and Act02, is translated into an abstract Event B model and one refinement: **ModelLevel0** and **RefLevel1**. The abstract model **ModelLevel0** is associated to the abstract level (Level0) (the AD containing the subactivity Act0) and contains only event *EvtAct0* associated with the subactivity Act0. The second model (**RefLevel1**) is a refinement of the first one and corresponds to the second level of decomposition (**Level1**) (the AD of the subactivity Act0 describing the execution of the activities Act01 and Act02). Two new events *EvtAct01* and *EvtAct02* associated with the two activities Act01 and Act02 are added in the refinement. These events carry the semantics of the execution of the two activities Act01 and Act02. The new events are fired and when they are completed, the refined event *EvtAct0* is fired. The

firing order of the events is determined by introducing a decreasing variant [2] that represents the control pass in the UML semantic (The token)[1]. A variant is a natural number, which decreases to 0. In practice, this variant corresponds to a decreasing enumeration of action states in UML AD.

In [2] [13], we have proposed translation rules for the concepts of UML AD (activity, Sequence of activities, choice (decision), loop, parallel activities (fork and join), atomic process, and dynamic invocation) into Event B.

5 Validation and Verification of UML AD Model

The most important interest of the proposed translation of UML AD into Event B is to allow the formal verification of functional/structural properties (safety, no deadlock, etc) of workflow applications specified in UML AD, using a powerful support tool like *B4free* [6].

Our translation approach is based on the refinement of Event B to encode UML AD hierarchical decomposition of activities. A subactivity Act0 (process) is described by an initial state and a final state. It is refined into a sequence of basic events which lead from the initial state to the final one. The refinement preserves all the properties of the initial activity Act0. This process is repeated until basic events are reached. In this case, the validation process is completed. First, this allows to ***validate an activity/process***. The final sequence of events shows that there is a sequence of basic elements implementing the upper abstract activity. Then, the activity is validated: If an activity is validated (feasible) then its objective is realisable[9]. Second, this allows to ***validate a conception and hierarchical decomposition***. If some proof obligations related to the basic events cannot be proved, in the B resulted models, then, we can assert that some of basic events are missing and/or wrongly specified and therefore, the conception shall be update and/or completed. If all proof obligations related to the basic events are proved, then the hierarchical decomposition is correct. This ensures completeness properties. Compared to classical model checking verification techniques, where the transition system describing the model is refined and enriched with properties to be checked again, the advantage of using Event B is that it is not needed to re-prove again verified properties in the refined model while the model complexity increases.

6 Application to the Example of Production Company Application

Step 1. Initially, we describe the production company using UML AD by employing a refinement technique, as it presented in figure 1.

Step2 and Step3. By the application of the translation process and using the translation rules [2][13], the initial UML AD model is translated into B event in a set of property preserving refinements. Three refinement steps which correspond to each level of three level of decomposition in the UML AD model (Figure 1) are necessary.

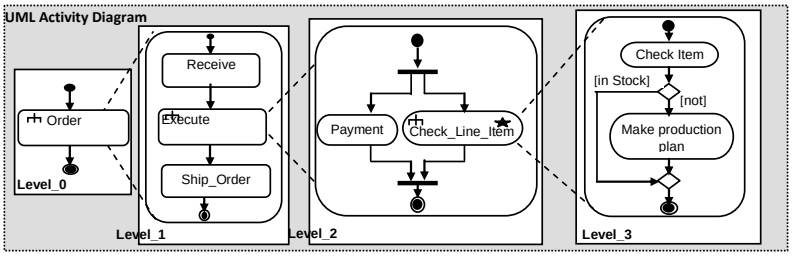

Fig.1. The UML AD model of the Production Company.

Following figure1, the activities Receive_order, Ship_order, Payment, Chek_Line_Item and Make_production_plan correspond to basic process/tasks of production company application. To illustrate this decomposition, initially we below give the first refinement level for subactivity Order_Processing.

REFINEMENT Ref1_Ord_Pro
REFINES Order_Processing
VARIABLES order_state
INVARIANT order_state ∈ 0..3
ASSERTIONS *order_state =0 ∨ order_state =1 ∨ order_state =2 ∨ order_state =3*
INITIALISATION *order_state :=3*
EVENTS
Evt_Order_Processing = **SELECT** *order_state= 0* **THEN** *skip* **END** ;
Evt_ReceiveOrder= **SELECT** *order_state= 3* **THEN** *order_state:=2* **END** ;
Evt_ExecuteOrder= **SELECT** *order_state= 2* **THEN** *order_state:=1* **END** ;
Evt_shipOrder= **SELECT** *order_state= 1* **THEN** *order_state:=0* **END**
END

The variable *order_state* play the role of variant ensuring the right events firing order: The variable *order_state* is decreased by each firing of events *Evt_ReceiveOrder*, *Evt_ExecuteOrder*, *and Evt_ShipOrder*, corresponding to the activities Receive_order, Execute_Order, and Ship_Order. The variant describes the precedence constraints.

In following, we give below the second refinement level for subactivity Execute_Order. In the *Evt_InitChekLineItem* event, the expression *nb_item :∈ NAT* allows to initialize the loop variant with any natural number corresponding to the arbitrary iteration of Chek_Line_Item activity. Then, the event *Evt_ChekLineItem is* fired *nb_item* times. The variant *nb_item* decreases from its arbitrary initial values to 0. The decreasing variant *nb_item* describes the events interleave and prevent that an event is fired infinitely (an event will be infinitely crossed in detriment of others). **The strong fairness (no livelock) properties** are expressed by the events interleave.

REFINEMENT Ref2_Ord_Pro

REFINES Ref1_Ord_Pro

VARIABLES *pay_state, check_state, nb_item*

INVARIANT *pay_state ∈ 0..1 ∧ check_state ∈ 0..1 ∧ nb_item ∈ NAT*

ASSERTIONS

order_state =0 ∨ order_state =1 ∨ order_state =2 ∨ order_state =3

=> order_state =0 ∨ order_state =1 ∨ (order_state =2 ∧ check_state=1) ∨ (order_state =2 ∧ check_state=0∧ nb_item ≠0)∨(order_state =2 ∧

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.

The **INVARIANT** clause allows expressing robustness properties. For Example, in the third refinement level ***Ref3_Ord_Pro*** for subactivity Chek_Line_Item, *Evt_PlanPro* is fired if the quantity in stock is deficient (*stock = FALSE). (*The variable *stock* is used to know if the quantity in stock is deficient or not)

REFINEMENT Ref3_Ord_Pro

REFINES Ref2_Ord_Pro

VARIABLES *item_state, stock*

INVARIANT $item\_state \in 0..2 \wedge stock \in BOOL$.....

Step4. Validation of the production company application.

Table 1. Summary of proofs, all Proof Obligations generated (nOp) have been proved (Pr= 100%).

Model	nOp	Auto	%Pr
Order_Processing	0	0	100%
Ref1_Ord_Pro	4	4	100%
Ref2_Ord_Pro	9	9	100%
Ref3_Ord_Pro	14	14	100%
TOLAL	27	27	100%

The table 1 illustrates the obtained results on our case study. The resulting Event B specification has been proven totally and then the initial UML AD model of our **production company** application is validated.

7 Conclusions

In this paper, we have presents a Event B based approach to reasoning about workflow applications. We show how an Event-B model can be structured from UML Activity diagrams (UML AD) and then used to give a formal semantic to UML AD which supports proofs of their correctness. More precisely, we propose a solution that uses the refinement in Event B to encode the hierarchical decomposition of activities in UML AD. The refinement in Event B allows to go from one abstract level to less abstract one (may be a program). Validation can be performed at any development stage and particularly at early development step allowing saving at development. Finally, this approach is tool supported. Indeed, The B4free is used to illustrate this approach. We are aware that the presented case study is simple, but it shows its feasibility and the possibility to scale up since the developed approach is generic. Currently, we are working on the implementation of this approach. In future works, we envisage the validation of transformation rules, and studying the correctness of the approach.

References

1. J.R, I. Jacobson, and G.Booch. The Unified Modelling Language reference Manual. In Addison- Wesley, 1998.
2. A. Ben Younes and L Jemni. Ben Ayed. 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. Beijing, China. 24-27 July 2007.
3. M. Dumas and A.H.M ter Hofstede. UML activity diagrams as a Workflows Specification Language. In *UML2001* page 76-90. Spinger-Verlag, 2001.
4. Clearsy. System Engineering Atelier B, Version 3.6, 2001.
5. J.R. Abrial. The B Book. Assigning Programs to Meanings. In Cambridge University Press, 1996.
6. JClearsy, "B4free," Available at http://www.b4free.com, 2004.
7. R. Eshuis, R. Wieringa. A formal semantics for UML Activity Diagrams – Formalising workflow models, Technical Report CTIT-01-04. Twente, Dept. Of Computer Science, 2001.
8. P. Behm, P. Desforges, and J.-M. Meynadier. METEOR: An Industrial Success in Formal Development. An invited talk at the 2nd Int. B conference, LNCS 1939, April 1998.
9. J-R Abrial. Extending B without changing it . In H Habrias, editor, First B Conference, Nantes, France, 1996.
10. W.M.P. van der Aalst. Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques. LNCS 1806, Springer-Verlag, 2000.
11. C. Karamanolis, D. Giannakopoulou, J. Magee, and S. M.Wheater. Formal verification of workflow schemas. University of Newcastle, Tech. Rep., 2000.
12. W.M.P. van der Aalst. The application of Petri nets to workflow management. The Journal of Circuits, Systems and Computers, 8(1):21–66, 1998.
13. A. Ben Younes and L Jemni. Ben Ayed. 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), July 2008,
14. R. Eshuis and R. Wieringa. Comparing Petri Net and Activity Diagram Variants for Workflow Modelling – Lecture Notes in Computer Science (LNCS), Germany, 2003.
15. A. Oberweis, R. Sch¨atzle, W. Stucky, W. Weitz, and G. Zimmermann. INCOME/WF: A Petri net based approach to workflow management. In H. Krallmann, editor, *Wirtschaftsinformatik '97*, pages 557–580. Springer, 1997.