Research on Event-B based modelling and verification of PLC system

He Zhao, Bin Fang, Hui-jie Li, Pu Wang[1, 2]

[1] College of Electronic and Control Engineering, Beijing University of Technology, Beijing, 100124, China

[2]The Ministry of Education P.R.C. Engineering Research Center of Digital Community, Beijing, 100124, China

Keywords: Event-B modeling, formal method, PLC system

Abstract. In order to ensure the safety of equipment and persons, the rigorous requirements on the correctness and reliability of control program are always needed in industrial control system. The traditional program design methods are based on the realization of the functions and verified by the simulations and tests. Errors can only be founded during the simulation and test phase. And some vital errors cannot be tested, because these errors may cause damages to the equipment and persons. So the correctness and reliability of control program cannot be ensured. For these problems, the formal design methods emerged as the times require. The errors can be found in design level by the formalized-model. In other word, formal methods could detect the errors earlier, reduce the cost of development, and are suitable for the occasions that require rigorous requirements on the correctness and reliability. This paper discussed a kind of Event-B based formal method, using Rodin Platform, to model, refine and verify the automated production line control system.

Introduction

The main objective of formal methods is to help developers to build the reliable system. Formal methods are a cutting edge technology for developing the critical system, where high safety and security are needed. Mathematics is a basic foundation for formal logic that provides some way to discover potential errors at the early stage of development. Figure 1 shows that module testing and integration testing are not required due to formally verified system at the specification and design level. Formal methods verify the whole system at the early stage of system development during specification and design, and to prove the correctness of system [1]. For the automated production line, introduced later, some damaging errors cannot be tested, even though we have designed the exception handlers. Formal methods can identify these errors and verify the correctness of exception handlers at the modeling stage, instead of testing.

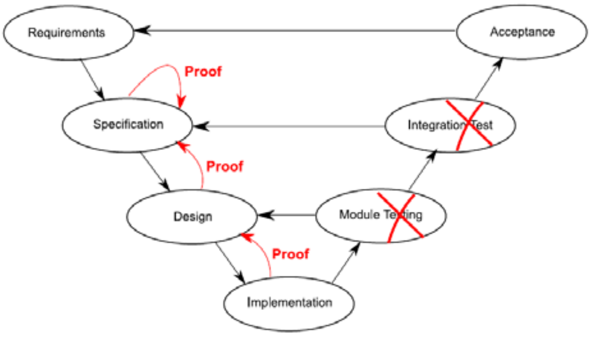


Fig.1 The V-model of safety-critical system development using formal methods

Event-B is a kind of formal methods based on predicate calculus and theorem proving, derived from B method, while B method, proposed by J -R Abrialb, is a kind of model-based formal modelling and verification method based on Z language. Comparing with B-method, Event-B is based on events, an important feature of Event-B which makes it suitable for periodic behaviour modelling. In addition, the dynamic part and static part are completely separated in Event-B model,

© 2015. The authors - Published by Atlantis Press

which makes the events interaction easier. The dynamic part of the model are considered in constantly perform reaction system with conditions in an interactive way, making the parallel behaviour and concurrent process modelling easier [2].

The theory and method of Event-B modelling

The theory of Event-B modelling. An event-B model usually consists of two parts: contexts, the static part, and machines, the dynamic part. And contexts contain carrier sets, constants, axioms, theorems. Axioms, considered as the attribute of carrier sets and constants, describe carrier sets and constants with theorems together. Similarly, machines include variables, invariants, theorems, variants and events. Invariants, considered as the attribute of the variables, describe variables with theorems together. An Event-B model can contain contexts only, or machines only, or both. Each context can be extended, while machines can be refined, these processes making the model more specific and increasing the function of model. In other word, an Event-B model can show more details and close to the actual situation by means of extending and refining. Moreover, machines can see one or several contexts, while the dynamic part contacting the static part. The relationship between contexts and machines showed in figure 2.

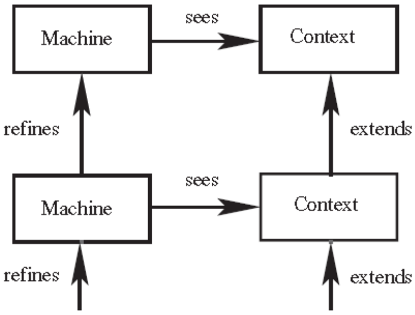


Fig.2: The relationship between Machine and Context

Event, contained in machine, is one of most important elements of Event-B model. Events describe the state and transfer in the model, representing what might happen in the actual system. Events are generally expressed as: evt any p where G (p, v) then S (p, v, v’) end. Among them, P means parameters, V means variables, G (p, v) represents the established conditions of event, and S (p, v, and v’) represents the behavior of event.

Proof obligation is another important concept in Event-B. Each model will be analyzed and proved, thus enabling us to establish that it is correct relative to a number of criteria. As a result, when the last model is finished, we shall be able to say that this model is correct by construction. The correctness criteria alluded to above will be made completely clear and systematic by giving a number of proof obligation rules, which will be applied to our models. After applying such rules, we shall have to prove these formally [3].

Rodin Platform. The Rodin Platform is an Eclipse-based IDE for Event-B that provides effective support for refinement and mathematical proof. The platform is open source, contributes to the Eclipse framework and is further extendable with plugins [4].The main functions of Rodin Platform include: Static checker (SC),Proof obligations generator(POG) and Provers. SC is responsible for the inspection of the grammar and generating error reports. POG is responsible for the generation of proof obligation needed during the model building and refinement process. Once the proof obligation generated, Provers will prove them automatically, and manually prove will be needed while Provers failed.

Modelling of automated production line based on Event-B

The automated production line. In following sections, we will introduce the automated production line model on Event-B. The automated production line is used to produce radiopharmaceuticals.

These radiopharmaceuticals are radioactive and may cause damage to the equipment and people, so the rigorous requirements on the correctness and reliability of control program are needed. Here, we will discuss the delivery part of system. The automated production line is PLC controlled by controlling a serie of cylinders, conveyors, electromagnets, and servo motors as the main implementing agencies to complete the fixtures sequentially delivered to the station. The automated production line is divided into lower and upper layers, connected by lifts. The lower delivers the processed fixtures by conveyors, and the upper, composed of the main channel and five manufacturing stations, delivers the fixtures to the corresponding stations. According to the function of the production line, we simplified the delivery progress describing by five kinds of cylinder. A class is responsible for the delivery progress between the lifter and the main channel. B class cylinders deliver the fixtures to the each stations form main channel. C class cylinders locate the fixtures on station. D class cylinders drive lifters, connecting the upper and lower layers. E class cylinders are responsible for the delivery progress between the lifter and conveyer. The entire delivery progress will be divided into 11 major steps, and the progress could be paused or restarted at any time.

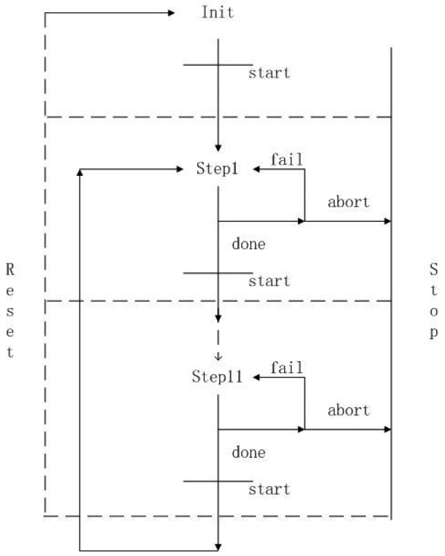
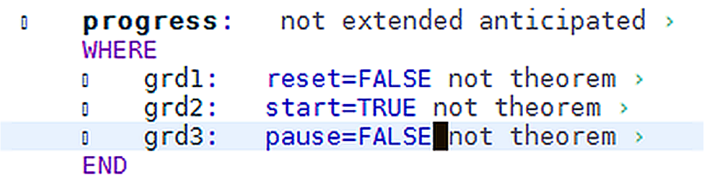


Fig.3: The structure of delivery system

Requirements document of delivery system. Event-B is a model-based formal modeling method, requirements document and refinement strategy are needed before the formalize-modeling. As mentioned earlier, Event-B model is usually divided into two parts, the dynamic and static, so the requirements could be divided into two parts similarly: Function Requirements (FUN) and Environmental Requirements (ENV). Function requirements include: FUN-1: Fixtures should be delivered to the corresponding station sequentially. FUN-2: The production line could discover the errors and retry the procedure, if an error occurs during transmission. FUN-3: The production line could automatically revert to the in the initial state of the corresponding step and stop the entire delivery program, while an error could not be handled by retries. FUN-4: The personnel need some control buttons which could start, pause and reset the production line at any time. Environmental requirements include: ENV-1: Each cylinder must be among the three states, pushed out, pulled in or hold. ENV-2: Due to the interference on the physical structure, class D and class E cylinders could not push out at same time. ENV-3: In the case of the presence of fixtures in the main channel, class A could not push out until class D cylinders have pushed out already. ENV-4: E class cylinders must be pushed out while the conveyor is working. ENV-5: Proximity sensors are used to detect the position of fixtures. They will turn on when fixtures are delivered to the corresponding location. ENV-6: Electromagnets can be energized only if fixtures are going to be delivered between the main channel and manufacturing stations.

Refinement strategy of Event-B model. Event-B is an incremental modeling method, the model could be more specific and functional by refinement. Refinement strategy is to describe the progress that the Event-B model becomes more specific. In this case, we first solve the control problem, namely FUN-4: The delivery system could start, pause and reset the production line at any time. Then, we focus on the delivery progress, describing the progress by several steps, namely FUN-1: Fixtures should be delivered to the corresponding station sequentially. Next, we will realize the errors handling, namely FUN-2 and FUN-3. We have already described the FUN-1 to FUN-4 in the current stage model so far, but the formalize-modeling is not finished yet. We have to connect the model with environment, namely connect the virtual information in model with the actual information in environment. The entire Event-B model of automated production line delivery system consist of 4 contexts and 14 machines, these will be introduced in the following sections.

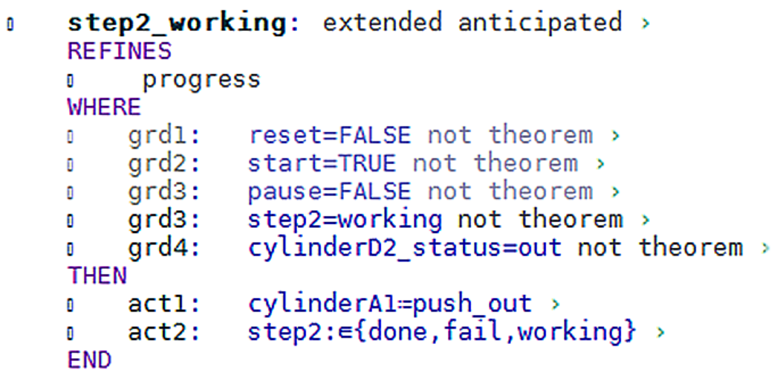
The initial model--mac0. In the initial model, mac0, three BOOL variables are used to describe the start, pause and reset signal. And then, the corresponding events are introduced representing the diffrent operating state of the delivery system. The most important event is event_progress standing for the working state of system, as can be seen in Figure 4. This event has no action in mac0, because we cannot see the detail of this stage and we will refine it in the following model. Furthermore, the property of this event is anticipated, that means this event is not divergent and need to be proved convergent in the following model. Note that proving the model convergent is to ensure the reachability of program.



```
progress:   not extended anticipated
WHERE
    grd1:   reset=FALSE not theorem
    grd2:   start=TRUE not theorem
    grd3:   pause=FALSE not theorem
END
```

Fig.4: event_progress

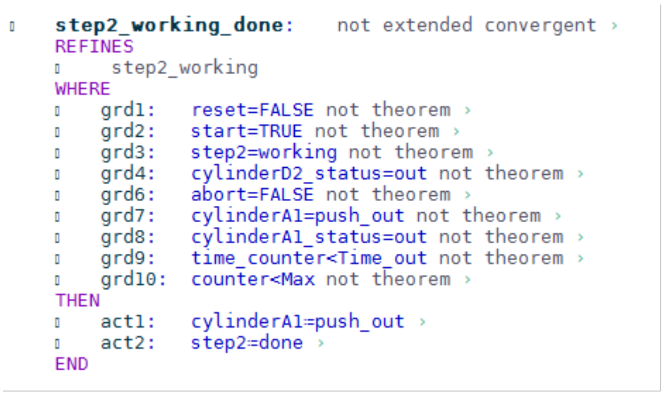
The first refinement--mac1. The first refinement is mainly to divide the event_progress into several progresses. We introduced conveyor, cylinders, steps, corresponding variables and variants in the model to describe each single step. This event has to be divided into 11 major steps, and the other events have to be refined or introduced with the new elements as well in this model. The Figure 5 shows the simplest step of progress, step2 working. Comparing with the event_progress in mac0, Figure 4, the each single step in mac1, here step2, shows more detail of progress. The step2_working is to push out cylinderA1 and change the status of step2 while step2 is working and cylinderD2 has already been out, according to ENV-3.



```
step2_working: extended anticipated
REFINES
    progress
WHERE
    grd1:   reset=FALSE not theorem
    grd2:   start=TRUE not theorem
    grd3:   pause=FALSE not theorem
    grd3:   step2=working not theorem
    grd4:   cylinderD2_status=out not theorem
THEN
    act1:   cylinderA1≔push_out
    act2:   step2:∈{done,fail,working}
END
```

Fig.5:step2_working

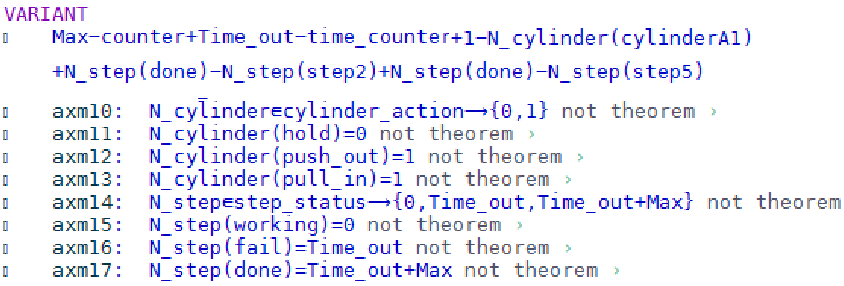
The second refinement to the seventh refinement—mac2 to mac7. We have already divided the progress into 11 major steps, but some of these events have non-deterministic actions and are anticipated. That means these events may not always execute the way we wanted, and we have to handle the exception. By introduced the timer and counter in model, we believe that the step has succeed if actions completed in time, or step has failed and needs to be retried automatically until the specified number of times that deadline of aborting program. We split each single step into three states: working, done and fail, so that we can handle the different status.



```
step2_working_done:    not extended convergent
REFINES
     step2_working
WHERE
    grd1:   reset=FALSE not theorem
    grd2:   start=TRUE not theorem
    grd3:   step2=working not theorem
    grd4:   cylinderD2_status=out not theorem
    grd6:   abort=FALSE not theorem
    grd7:   cylinderA1=push_out not theorem
    grd8:   cylinderA1_status=out not theorem
    grd9:   time_counter<Time_out not theorem
    grd10:  counter<Max not theorem
THEN
    act1:   cylinderA1≔push_out
    act2:   step2≔done
END
```

Fig.6 step2_working_done

Figure 7 shows the events, step2_working_done, reined by step2_working. It has already introduced timer and counter and setted the property of this event convergent. The convergent model is to ensure the reachability of program, which means the program will not be stuck by this progress. To prove the convergence, we have to introduce variant to describe it. Variant could be described by natural number or set. Here, we created variant by natural number shown in Figure 8 and the mappings of relevant elements.

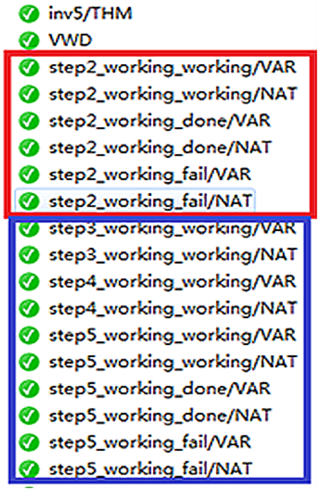


```
VARIANT
    Max-counter+Time_out-time_counter+1-N_cylinder(cylinderA1)

    +N_step(done)-N_step(step2)+N_step(done)-N_step(step5)

    axm10:  N_cylinder∈cylinder_action→{0,1} not theorem
    axm11:  N_cylinder(hold)=0 not theorem
    axm12:  N_cylinder(push_out)=1 not theorem
    axm13:  N_cylinder(pull_in)=1 not theorem
    axm14:  N_step∈step_status→{0,Time_out,Time_out+Max} not theorem
    axm15:  N_step(working)=0 not theorem
    axm16:  N_step(fail)=Time_out not theorem
    axm17:  N_step(done)=Time_out+Max not theorem
```

Fig.7 variant of mac3

Among them, Max is the maximum number of retries and counter records the current number of retries. Time_out is the maximum allowable time and time_counter represents timer. N_cylinder() and N_step() is mappings of cylinders and steps to natural number respectively.

The proof obligations are generated by Rodin Platform automatically, shown in Figure 8. VAR stands for the convergence of event and NAT indicates the existence. We have to put the relevant elements of event into the variant we created to prove the all VAR and NAT. The red part is generated by the convergent events and blue part by the anticipated events. That means the variant is not only making the events we wanted convergent, but also not making the other anticipated-events divergent. In other word, the variant is for the entire model.



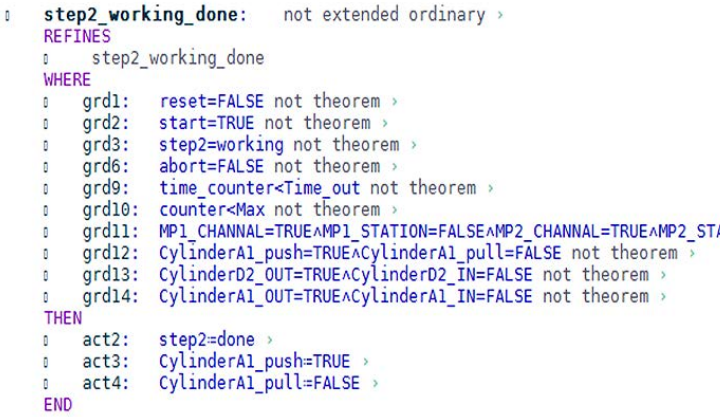
```
inv5/THM
VWD
step2_working_working/VAR
step2_working_working/NAT
step2_working_done/VAR
step2_working_done/NAT
step2_working_fail/VAR
step2_working_fail/NAT
step3_working_working/VAR
step3_working_working/NAT
step4_working_working/VAR
step4_working_working/NAT
step5_working_working/VAR
step5_working_working/NAT
step5_working_done/VAR
step5_working_done/NAT
step5_working_fail/VAR
step5_working_fail/NAT
```

Fig.8 proof obligation of step2_working in mac3

From mac2 to mac7, we refined each single event and proved the convergence to satisfy the requirements of FUN-2 and FUN-3.However, up to now, we just focus on the situation of with no fixtures on the delivery system, and we have to introduce the fixtures in the following models.

The eighth refinement to the tenth refinement—mac8 to mac10. We detect the position of fixtures by the proximity sensors installed in the main channel and each station, introduced the sensors in ctx3, and refine the relevant events in mac8. Electromagnets are used to limit the fixtures, can be energized only if fixtures are going to be delivered between the main channel and manufacturing stations (ENV-6). This part is completed in mac9. And there are some pre-processing operations while fixtures have arrived the corresponding stations, cylinders action in the second and fourth station and the alignment operation of slide tables driven by servo motors in the first, third and fifth station. We simplify the progress and finish it in mac10.

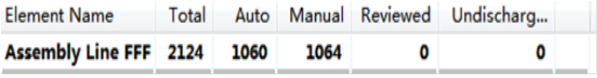
The eleventh refinement to the thirteenth refinement—mac11 to mac13. We have almost finished all the requirements, but the modeling is not over yet. A complete Event-B model should be a closed model, from the controller to environment and feedback, so we have to connect the signal in controller to the signal in the real signal in environment. We finish this part in mac11 to mac13.Figure 9 shows the step2_working_done in mac13, and the variables begun with capital letters are the actual signals in environment.



```
step2_working_done:    not extended ordinary ›
REFINES
      step2_working_done
WHERE
    grd1:   reset=FALSE not theorem ›
    grd2:   start=TRUE not theorem ›
    grd3:   step2=working not theorem ›
    grd6:   abort=FALSE not theorem ›
    grd9:   time_counter<Time_out not theorem ›
    grd10:  counter<Max not theorem ›
    grd11:  MP1_CHANNAL=TRUE∧MP1_STATION=FALSE∧MP2_CHANNAL=TRUE∧MP2_ST
    grd12:  CylinderA1_push=TRUE∧CylinderA1_pull=FALSE not theorem ›
    grd13:  CylinderD2_OUT=TRUE∧CylinderD2_IN=FALSE not theorem ›
    grd14:  CylinderA1_OUT=TRUE∧CylinderA1_IN=FALSE not theorem ›
THEN
    act2:   step2≔done ›
    act3:   CylinderA1_push≔TRUE ›
    act4:   CylinderA1_pull≔FALSE ›
END
```

Fig.9 step2_working_done in mac13

The requirements delivery system have already been satisfied in the model, we still have to prove all proof obligations generated by the model to ensure the correctness of model. Figure 10 shows the statistics of all proof obligations.



Element Name	Total	Auto	Manual	Reviewed	Undischarg...
Assembly Line FFF	**2124**	**1060**	**1064**	**0**	**0**

Fig.10 the statistics of all proof obligations

Conclusion

In order to ensure the safety of equipment and people, the rigorous requirements on the correctness and reliability of control program are always needed in PLC system. Comparing with the traditional methods, formal methods are easier and earlier detecting the errors, especially the errors are hardly tested in PLC system. This paper discussed a kind of Event-B based formal method, to model, refine and verify the automated production line control system and improved the reliability. This method is a reference to other similar programing.

Acknowledgements

This work is supported by the Beijing Laboratory For Urban Mass Transit.

References

[1] Springer Using Event-B for Critical Device Software Systems.2013

[2] J-R Abrial Event-B language [J].RodinDeliverable, 2005.3(1):2-254

[3] J-R Abrial Modeling in Event-B system and software engineering[C] Cambridge University Press 2010

[4] http:/www.event-b.org/[EB/OL]