ISBN 978-981-11-3671-9
Proceedings of 2017 the 7th International Workshop on Computer Science and Engineering (WCSE 2017)
Beijing, 25-27 June, 2017, pp. 1144-1149
doi: 10.18178/wcse.2017.06.200

Bounded Retransmission in iUML-B State Machine: a Case Study

Han Peng [1+], Chenglie Du [1] and Haobin Wang [2]

[1] College of Computer Science, Northwestern Polytechnical University, Xi'an, China
[2] College of Computer Science, Xi'an Aeronautical University, Xi'an China

Abstract. This paper presents a case study using the iUML-B state machine to model bounded retransmission protocol. The case is inspired by Abrial's treatment of this example to illustrate the practicality of modeling control flow using the iUML-B state machine. The case study contains several aspects of our approach: (1) divide the system model into component state machines and control flow state machines; (2) modeling different types of variables using the state machine; (3) refinement method based on iUML-B state machine.

Keywords: Event-B, control flow modelling, iUML-B state machine, bounded retransmission protocol

1. Introduction

Event-B [1] is a formal method which is evaluated from B method [2]. It uses simple symbols and structures to model the system, and is well suited for different areas, including distributed systems. But Event-B lacks the constructors to modelling the control flow of system. It usually uses state variables to control the order in which events occur. This makes it very difficult and complex to observe and analyze the control flow of the Event-B model. Although some methods of modelling control flow are proposed, such as ERS method [3], CSP || B method [4], and Flow method [5], these methods are not intuitive for software engineers.

iUML-B 6]is the graphical front end of Event-B, using "UML-like" class diagrams and state machines to describe the state and behaviour of the system. The iUML-B's graphical model can generate Event-B code directly on the Rodin platform [7] and automatically embeds the generated code into the context (set, constant, and axiom) and machine (variables, events, invariants). IUML-B has been applied to formal modelling in many fields. Fathabadi[8]used iUML-B state machine to establish the thread scheduling model of the many-core system. Hoang [9] used iUML-B to model and verify the behaviour of haemodialysis machine. The transform edges in the iUML-B state machine can be linked to events in the Event-B model to control the order in which these events occur. The principle is shown in Figure 1.

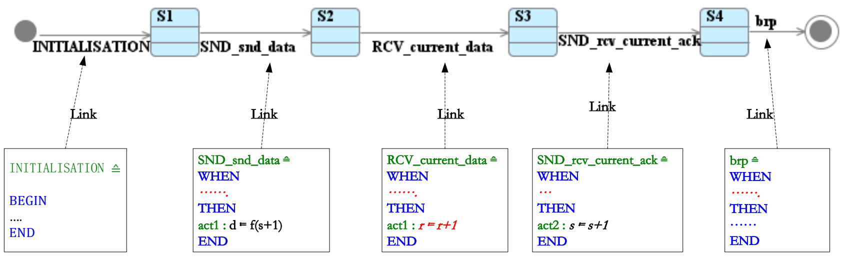
For example, if we want to specify the order in which events occur in the Event-B model like this:

INITIALISATION-> SND_snd_data-> RCV_current_data-> SND_rcv_current_ack-> brp,

we can first draw the iUML-B state machine shown in Figure 1, and then link each transition edge to an event to make the events in the Event-B model occur in the order specified in the iUML-B state machine .

This paper presents a case study that illustrates the method of modelling and refining the Evnet-B model's control flow using the iUML-B state machine. The case study was inspired by Abrial's treatment of bounded retransmission protocols [1]. The method is based on a step-by-step refinement of the Event-B model, where additional details are introduced at each refinement stage. Our method uses iUML-B instead of control variables in Event-B to manage the control flow of events in a clear and visible way.

[+] Corresponding author. Tel.: + 8613474113698; fax: +86(029)84252366.
E-mail address: hansbeng2016@gmail.com.



```
INITIALISATION ≙

BEGIN
....
END
```

```
SND_snd_data ≙
WHEN
......
THEN
act1 : d ≔ f(s+1)
END
```

```
RCV_current_data ≙
WHEN
......
THEN
act1 : r ≔ r+1
END
```

```
SND_rcv_current_ack ≙
WHEN
...
THEN
act2 : s ≔ s+1
END
```

```
brp ≙
WHEN
......
THEN
......
END
```

Fig. 1 Link iUML-B state machine with event

2. Problem description

2.1. Overview of bounded retransmission protocols

The purpose of bounded retransmission protocol (BRP) is to transfer sequential files from the sender to the receiver. After the transfer is complete, the recipient's file should be equal to the sender's original file. The sender should split the files that need to be sent into a series of data blocks and send the data blocks to the receiver in order. Once the recipient receives the data item, it stores it in its own file and sends an acknowledgement message to the sender on an acknowledgement channel. The sender will send the next data item after receiving this acknowledgement message. The principle of the BRP protocol is shown in Figure 2.

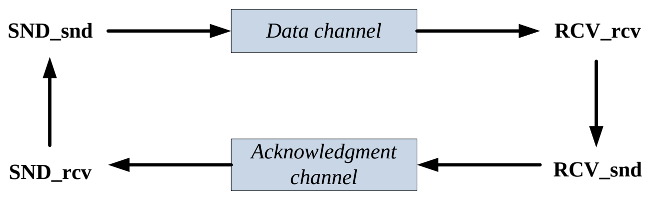


Fig. 2 The principle of bounded retransmission protocol

2.2. System Flow Requirements

We introduce the system control flow demand in the refinement of the system step by step . The control flow requirements of each layer are described as follows:

1) System abstract model (level 0)

- Introduced events: *SND_porgress*, *RCV_progress*, *brp*
- Flow requirements:
    - o *SND_porgress*, and *RCV_progress* can occur staggered. (*Flow0.1*)
    - o Only when *SND_porgress*, and *RCV_progress* have occurred, *brp* can occur (Flow0.2)

2) The first refinement (level 1)

- Introduced events: *SND_success*, *SND_failure; RCV_success*, *RCV_failure.*
- Flow requirements:
    - o When the system is initialized, only *SND_failure* or *RCV_success* events can occur (*Flow1.1*)
    - o *SND_failur* enables the *RCV_failure* event (*Flow1.2*)
    - o *RCV_success* enables the *SND_success* event (*Flow1.3*)

3) The second refinement (level 2)

- Introduced event: *RCV_rcv_current_data*
- Flow requirements:
    - o There will *be* 0 or more *RCV_rcv_current_data* events before the *RCV_success* event occurs. (*Flow 2.1*)

4) The third refinement (Level 3)

- Introduced events: *SND_snd_data*, *SND_timeout*, *SND_rcv_curr_ack*
- Flow requirements:

- The *SND_snd_data* event occurs one or more times before the *SND_success* event occurs. (*Flow3.1*)
- After the *SND_snd_data* event occurs, the *RCV_rcv_current_data* event, the *SND_rcv_curr_ack* event, the *RCV_success* event, the *SND_timeout* event, and the *SND_success* event may be enabled *(Flow3.2*).

2.3. System flow requirements classification

We divide the system control flow requirements into two categories.

1) Component control flow requirements

A component control flow requirement refers to the control flow requirements that involve only one participant in the system, such as *Flow 2.1* and *Flow 3.1*.

2) Combine event flow requirements

A combine event flow requirements refers to the demand for control flow that involves the interaction between multiple participants in the system. Such as *Flow0.1, Flow0.2, Flow1.1, Flow1.2, Flow1.3, Flow3.2* and so on.

3. Bounded retransmission protocol modelling using iUML-B state machine

3.1. General method

We classify the following requirements according to Section 2, and design the following types of state variables to control the behavior of a component or system:

1) State variables of Enumeration type, such as the status of the sender, the status of the receiver. Its definition is like: $r\_s \in \{working, success, failure\}$. Usually we can use this state variable to control the order of events of a component.

2) State variables of the Boolean type, such as the send enable bit, the receive enable bit, and so on. Its definition is like: $w \in BOOL$. We can use this variable to control the event order between multiple components.

3) Bounded numerical variables. Such as the number of messages sent by the sender and the number of messages that the recipient has accepted. Its definition is like: $s \in N$. Such variables usually involve only the state changes of a component, but are sometimes used to compare with other component counter variables. Such as *where r = s then*.

For the above three types variables that will affect the event order of Event-B model, We take the following measures:

1) We use the state variables of the enumeration type to model the event control flow of a component in the system and model it as an iUML-B state machine. The state nodes on the state machine are the value of these variables and the transition edged of the machine is associated with events that have changed the state.

2) We use the state variables of Boolean type to model the event enabling relationship between multiple components in the system and model it as an iUML-B state machine with only two states. The state of the state machine represent, respectively, that the Boolean amount is true and the amount of Boolean is false . The linking rule for the transition edges in the state machine is that if an event changes the value of a state variable, the event is linked to the Non-reflexive edge in the state machine; if an event does not change the value of a state variable, this event is linked to the reflexive edges of the state machine.

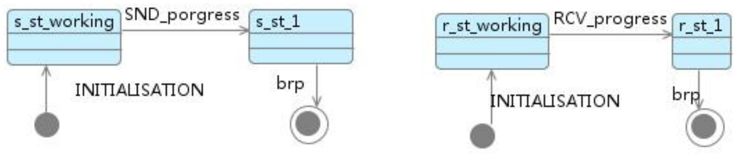
3) Bounded numeric variables are placed in the guard and action sections of the transform edge of iUML_B state machine.

The innovation of this approach is that it extracts a variety of control flows that affect the behavior of the system as a separate control flow, and then model each control flow as one state machine. The overall control flow of the system is the composition of these state machines, that is, the overall behavior of the system is a composed state machine, which is composed by separate components control flow automata one by one. Here we use this method to model the bounded retransmission protocol.

3.2. System modelling and refinement process

1) System abstraction model

The abstract model of the system is a combination of the sender component state machine and the receiver component state machine. The iUML-B state machine model of the sender and the receiver is shown in Fig. 3 (a) and Fig. 3 (b), respectively.



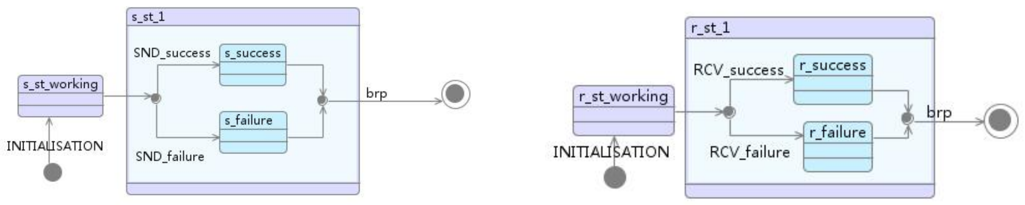
(a) SNDMachine0 (b) RCVMachine0

Fig. 3 System abstraction model

It is important to note that although *SNDMachin0* and *RCVMachin0* do not explicitly control the interleaving relationship between the sender and the receiver (*Flow1.1*) and the constraints on the *brp* event (*Flow1.2*). But the combination of these two state machines satisfies this requirement, that is, *SysMachine0* = *SNDMachin0* || *RCVMachin0* ⊨ *Flow1* ∧ *Flow2*, the underlying theory is the product of the state machine.

2) The first refinement

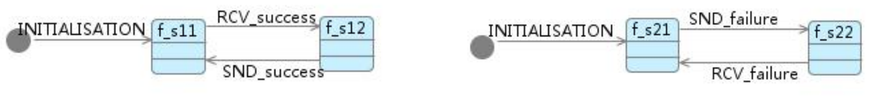
The first refinement introduces four events that refine the behavior of the sender and the receiver, respectively. We first mode the behavior of the sender and the receiver, as shown in Figure 4 (a) and Figure 4 (b).



(A) SNDMachine1 (b) RCVMachine1

Fig. 4 The first refinement model of the component state machine

If we only combine *SNDMachine1* and *RCVMachine1*, we cannot meet the flow control requirements of *Flow2.1, Flow2.2, and Flow2.3*, which involve the synchronization of the sender and the receiver .Therefore, we establish two flow automata *Flow-1* and *Flow1-2* to constrain the behavior of the system. As Figure 5 (a) and Figure 5 (b) shows.



(A) Flow1-1 (b) Flow1-2

Fig. 5 Flow state machine for the first refinement model

In Flow1-1 we specify the event after the *INITIALISATION* as *RCV_success* and the event after the *INITIALISATION* in *Flow1-2* is specified as *SND_failure*, which satisfies the demand *Flow2.1*. At the same time, let *SND_success* and *RCV_failure* appear after *RCV_success* and *SND_failure* respectively, which satisfies the requirements *Flow2.2* and *Flow2.3*.

3) The second refinement

The second refinement introduces only one event, *RCV_rcv_current_data*, which indicates that the receiver receives the current data. The corresponding flow requirement is *Flow 2.1*. We first add this event to the receiver component state machine, as shown in Figure 6.

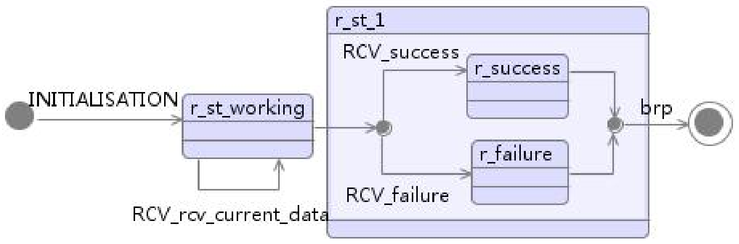


Fig. 6 Receiver component of the second refinement model -RCVMachine2

We found that *RCVMachine2* satisfies the flow requirement *Flow2.1*. So we no longer need to build a new flow state machine.

4) The third refinement

All of the three events introduced during the third refinement are the sender's events, along with three control flow requirements *Flow3.1* and *Flow3.2*. We first add the newly introduced events to the sender state machine, as shown in Figure 6. It can be seen that *Flow3.1* and some of the *Flow3.2* control flow requirements have been met by the sender component state machine.

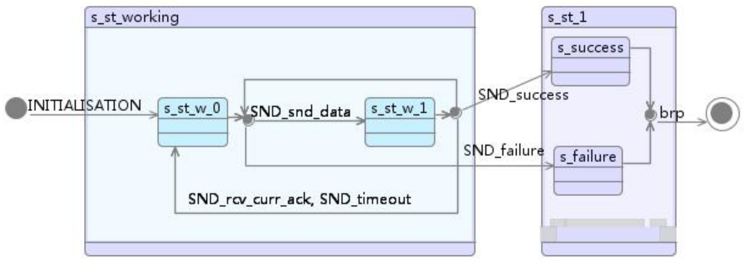


Fig. 7 The third refinement of the sender component state machine-SNDMachine3

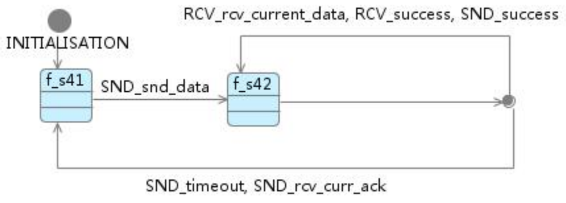

RCV_rcv_current_data, RCV_success, SND_success
INITIALISATION
f_s41
SND_snd_data
f_s42
SND_timeout, SND_rcv_curr_ack

Fig. 8 The third refinement of the control flow state machine

In order to model *Flow3.2*, we added *Flow3* state machine in the third layer refinement model. In *Flow3* we have specified that the *RCV_rcv_current_data* event or the *RCV_success* event can occur after the sender sends the *SND_snd_data*.

3.3. Proof obligations

In Table 1, we describe the statistics of the development in terms of proof obligations (and how many proof obligations were discharged) for each refinement step. This case study was carried out under the following conditions:

- Rodin v3.2

- Plug-ins: iUMLB State-Machines (for Rodin 3.x.x) 3.4.1, ProB (for Rodin3) 3.0.9.
- The control flow code was generated automatically

Table 1 Proof Obligations of BRP model

	Total	**Auto**	**Manual**	**Reviewed**	**Undischarged**
Brp0	0	0	0	0	0
Brp1	14	14	0	0	0
Brp2	13	12	1	0	0
Brp3	29	29	0	0	0

4. Conclusion

In this paper, we propose a method to build the Event-B control flow model using the iUML-B state machine, and prove the practicability of the method by bounded retransmission protocol case study. We designed several state variables to control the event order of the Event-B model and to design the corresponding automaton for each state variable. In addition, we define the component state machine and the flow state machine, respectively, to express the behavior of the component itself and the interaction between components. The experimental results show that most of the proof obligations can be discharged automatically, without further manual participation.

In the future, we will give the iUML-B state machine certain formal behavior semantics, such as labeled transition system or communication sequential process. In this way, we are able to convert the control flow modeling problem of the Event-B model into a controller synthesis problem.

5. Acknowledge

We are very grateful to Prof. Colin Snook and Dr. Thai Son Hoang of the University of Southampton for their valuable constructive suggestions in this research work.

6. References

[1] J.-R. Abrial, *Modeling in Event-B: System and Software Engineering*: Cambridge University Press, 2010.

[2] J.-R. Abrial, *The B-book: assigning programs to meanings*: Cambridge University Press, 1996.

[3] E. Alkhammash, M. Butler, A. S. Fathabadi, and C. Cîrstea, "*Building traceable Event-B models from requirements*," Science of Computer Programming, vol. 111, pp. 318-338, 2015.

[4] S. Schneider, H. Treharne, and H. Wehrheim, "*A CSP approach to control in event-B,*" presented at the Proceedings of the 8th international conference on integrated formal methods, Nancy, France, 2010.

[5] A. Iliasov, "*Use Case Scenarios as Verification Conditions: Event-B/Flow Approach,*" in Software Engineering for Resilient Systems - Third International Workshop, SERENE 2011, Geneva, Switzerland, September 29-30, 2011. Proceedings, 2011, pp. 9-23.

[6] M. Y. Said, M. Butler, and C. Snook, “A method of refinement in UML-B," Software & Systems Modeling, vol. 14, pp. 1557-1580, 2015.

[7] J. R. Abrial, M. Butler, S. Hallerstede, T. S. Hoang, F. Mehta, and L. Voisin, "*Rodin: an open toolset for modeling and reasoning in Event-B,*" International Journal on Software Tools for Technology Transfer, vol. 12, pp. 447-466, 2010.

[8] A. Salehi Fathabadi, C. Snook, and M. Butler, "*Applying an Integrated Modeling Process to Run-time Management of Many-Core Systems,*" in Integrated Formal Methods: 11th International Conference, IFM 2014, Bertinoro, Italy, September 9-11, 2014, Proceedings, E. Albert and E. Sekerinski, Eds., ed Cham: Springer International Publishing, 2014, pp. 120-135.

[9] T. S. Hoang, C. Snook, L. Ladenberger, and M. Butler, "*Validating the Requirements and Design of a Hemodialysis Machine Using iUML-B, BMotion Studio, and Co-Simulation,*" in International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z, 2016, pp. 360-375.