Training of Software Formal Modeling Using Visual Blocks for Actions and Guards of Extended Place/Transition Net

Akio Usuda
Division of Reliability-based Information Systems Engineering, Graduate School of Engineering, Kagawa University
2217-20 Hayashi-cho, Takamatsu-shi, Kagawa 761-0396, Japan

Ryoichi Ishigami
Department of Engineering and Design, Faculty of Engineering and Design, Kagawa University
2217-20 Hayashi-cho, Takamatsu-shi, Kagawa 761-0396, Japan

Tomohiko Takagi
Department of Engineering and Design, Faculty of Engineering and Design, Kagawa University
2217-20 Hayashi-cho, Takamatsu-shi, Kagawa 761-0396, Japan
E-mail: s21g457@kagawa-u.ac.jp, s19t301@kagawa-u.ac.jp, takagi@eng.kagawa-u.ac.jp

Abstract

Extended Place/transition Net (EPN) that is one of software formal modeling languages consists of the parts of PN and VDM++. The PN part deals with state transitions of software. On the other hand, the VDM++ part deals with actions and guards on the transitions, and requires skills of system engineers who construct EPN models. We propose an extended training technique of EPN-based modeling using visual blocks for the VDM++ part. The visual blocks implemented by Blockly will be useful to accelerate the trainees' understanding of syntactical aspects of VDM++, and thus they are introduced into each step of the existing training technique for EPN-based modeling. The effectiveness of the proposed technique is discussed through a preliminary experiment using our prototype tool.

Keywords: software modeling language, place/transition net, VDM, training

1. Introduction

Extended Place/transition Net (EPN) [1] is one of formal modeling languages that can be used to analyze, design, and test software from the abstracted viewpoint of its behavior. It consists of the parts of PN [2] and VDM++ [3]. The PN part deals with state transitions of software. On the other hand, the VDM++ part deals with actions and guards on the transitions.

System engineers will sometimes face the difficulty of using EPN, since it includes special syntax. To address this problem, some studies of training/learning support techniques for software modeling have been conducted. As the first of a series of the studies, [4] describes a training support technique for bug fixing in EPN-based modeling. In the technique, animated graphics were introduced so that trainees can intuitively understand the meaning of given faulty EPN models and the results of their bug fixing. After that, in order to focus on modeling itself, Ue et al. [5] modified the previous technique, and constructed a learning support technique for PN-based modeling. Additionally, in order to focus on EPN that has higher representation power than PN, we extend the previous technique, and proposed a training technique for EPN-based modeling [6]. The technique was designed for personal on-demand training, and consists of the following three steps (overview):

[Step 1] A trainer creates an exercise that includes software requirements (that is, questions in the exercise), correct EPN models that satisfy the software requirements (that is, correct answers in the exercise), and components that the correct/incorrect EPN models consist of.

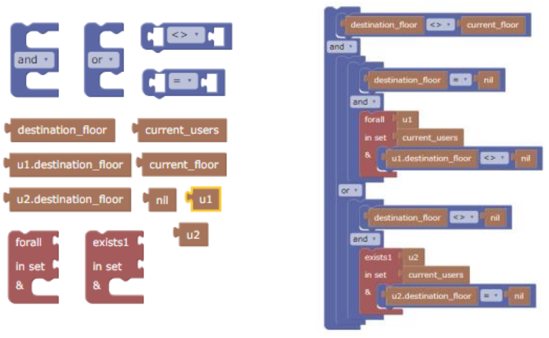
[Step 2] A trainee reads the software requirements, and tries to assemble his/her EPN model from the components.

[Step 3] The trainee's EPN model is checked according to the correct EPN models.

In order to support the trainee, Step 2 provides (i) VDM++ specifications that are generated from the trainee's EPN model and (ii) reference documents of VDM++. However, there is still room to improve the training for VDM++ part of EPN.

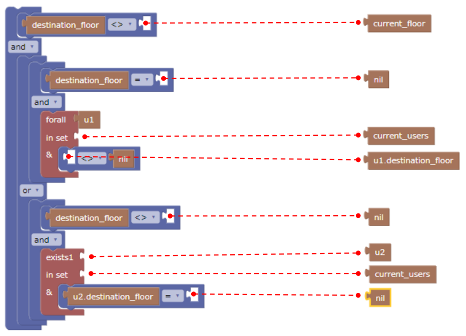
We propose an extended training technique of EPN-based modeling using visual blocks for the VDM++ part. The visual blocks that look like the pieces of a jigsaw puzzle are well-known and well-used in the field of programming education, and play an important role in visual programming. Existing visual programming support tools such as Blockly [7] can be used in our technique. The visual blocks will be useful to accelerate the trainee's understanding of syntactical aspects of VDM++, and thus they are introduced into each step of the previous training technique for EPN-based modeling.

This paper is organized as follows. The extended training technique is proposed in section 2, and then the effectiveness is discussed in section 3. Finally, we show conclusion and future work in section 4.



(a) (b)

Fig. 1. Example of visual blocks and a block-structured guard.



(a) (b)

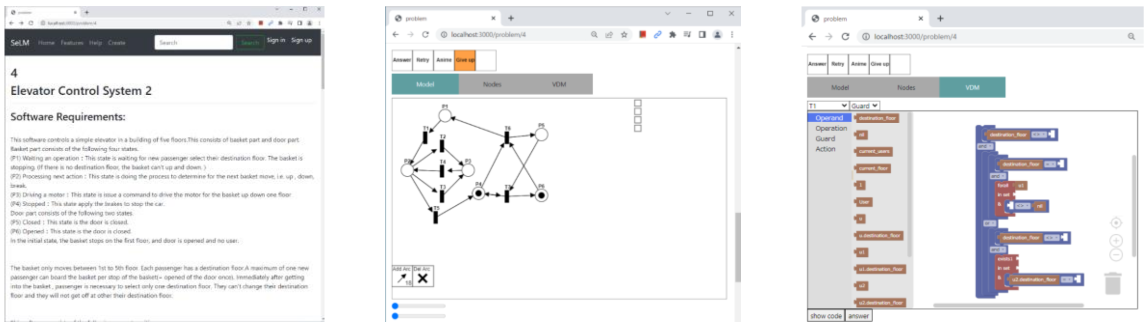
Fig. 2. Example of block-based components
(a fill-in-the-blank component and choice components).

2. Extended Training Technique

In this section, we propose an extended training technique of EPN-based modeling using visual blocks for the VDM++ part (hereinafter, the extended technique). Each step of the previous technique discussed in section 1 is extended as follows.

In Step 1, a trainer creates visual blocks, and then block-structured actions and guards (that is, the VDM++ parts of the correct EPN models). The visual blocks are chiefly classified into statements/expressions, operators, and operands. An example of the visual blocks that were implemented by Blockly is shown in Fig. 1 (a). The block-structured actions and guards are assembled from the visual blocks. An example of a block-structured guard that were implemented by Blockly is shown in Fig. 1 (b). After that, the trainer creates block-based components to give a trainee fill-in-the-blank questions with multiple choices. The block-based components are classified into fill-in-the-blank components and choice components. They are created by disassembling the block-structured actions and guards. The way of disassembling determines the degree of difficulty of the exercise. An example of a fill-in-the-blank component and choice components, which was created from Fig. 1 (b), is shown in Fig. 2 (a) and (b), respectively. Additionally, in order to increase the degree of the difficulty, the trainer will be able to create faulty (mutated) block-based components that lead a trainee to construct his/her incorrect EPN models. Note that the PN part and other materials of the exercise in the extended technique are the same as ones in the previous technique.

In Step 2, the way of constructing the PN part in the extended technique is the same as the way in the previous technique. If a trainee selects a transition to construct its action or guard in his/her EPN model, he/she receives a complete set of the block-based components. The trainee


SeLM Home Features Help Create Search Sign in Sign up

4

Elevator Control System 2

Software Requirements:

This software controls a simple elevator in a building of five floors.This consists of basket part and door part.
Basket part consists of the following four states.
(P1) Waiting an operation : This state is waiting for new passenger select their destination floor. The basket is stopping. (If there is no destination floor, the basket can't up and down.)
(P2) Processing next action : This state is doing the process to determine for the next basket move, i.e. up , down, break.
(P3) Driving a motor : This state is issue a command to drive the motor for the basket up down one floor.
(P4) Stopped : This state apply the brakes to stop the car.
Door part consists of the following two states.
(P5) Closed : This state is the door is closed.
(P6) Opened : This state is the door is closed.
In the initial state, the basket stops on the first floor, and door is opened and no user.

The basket only moves between 1st to 5th floor. Each passenger has a destination floor.A maximum of one new passenger can board the basket per stop of the basket(= opened of the door once). Immediately after getting into the basket , passenger is necessary to select only one destination floor. They can't change their destination floor and they will not get off at other their destination floor.

(a) Section of software requirements (b) Editor for the PN part (c) Editor for the VDM++ part

Fig. 3. Partial screenshots of the prototype tool.

tries to select an appropriate fill-in-the-blank component from the set. Additionally, the trainee tries to choose appropriate choice components from the set, and then tries to fill in the blanks with them. After that, the action or guard of the selected transition is immediately checked according to one in the correct EPN models. In this study, we use Blockly and are developing a prototype of a tool that supports the extended technique. Fig. 3 shows partial screenshots of the prototype tool that supports this step.

In Step 3, the PN part in the trainee's EPN model is checked according to one in the correct EPN models. Note that the constructing of the PN part, the constructing and checking of the VDM++ part are concurrently performed in Step 2 in the extended technique.

3. Discussion

We performed a preliminary experiment using prototype tools to discuss the effectiveness of the extended technique.

We created two types of exercises on the subject of an elevator control system, and set them on the old prototype tool (discussed in [6]) and the new prototype tool (discussed in section 2), respectively. After that, three trial users, that is, three undergraduate students in our laboratory tackled the exercises in the order of using the new one and the old one. In the three, the two are beginners of EPN, and the rest is researching on an EPN-related field. Before the tackling of the exercises, we made oral explanations about EPN and how to use the prototype tools to the three. During the tackling, we observed the behavior of each of the three. Also, we added oral explanations, if one had any questions. After the tackling, the three answered our questionnaire. After the first trial user finished tackling, we fixed some bugs and improved the description of software requirements. As the result of this preliminary experiment, we found the following:

- In the new one, the trial users seemed to need more detailed software requirements, and seemed to think that the degree of the difficulty of the exercises is relatively high. In other words, the new one will give more practical training, since the trial users seemed to need a deeper understanding of software requirements to construct their EPN models.
- In the new one, the trial users seemed to be able to intuitively understand the syntactical rules and structures of VDM++ to some extent by the visual blocks that were implemented by Blockly. Also, Blockly seemed to have improved the usability of our prototype tool.
- In the new one, the early checking of each action and guard in Step 2 seemed to help the trial users to understand and construct.
- The preparation of the exercises in the old one needed a great amount of our efforts, but it was smaller than the amount of our efforts to prepare the exercises in the new one.

We will need the improvement of the prototype tool and additional experiments with many trial users to reveal the effectiveness more clearly.

4. Conclusion and Future Work

In this paper, we proposed an extended training technique of EPN-based modeling using visual blocks for VDM++ part. In the step of creating an exercise, a trainer creates block-based components for actions and guards in order to give a trainee fill-in-the-blank questions with multiple choices. In the step of tackling the exercise, each action or guard of a selected transition in a trainee's EPN model is immediately checked according to one in trainer's correct EPN models. We performed a preliminary experiment using prototype tools, and found that the extended technique will give more practical training, will help trainees to understand to some extent, will require a greater amount of trainer's efforts to prepare exercises, and so on.

In future study, we need to discuss (1) the effectiveness of giving more detailed software requirements to trainees, (2) training techniques based on abstracted software requirements, and (3) techniques of supporting the preparation of exercises. Furthermore, the visual blocks will be useful also to construct EPN models in actual software development, and may be introduced into EPN-based software modeling tools.

Acknowledgements

A part of this work was developed from the graduation thesis written by Mr. Yuya Morinishi at Kagawa University. This work was supported by JSPS KAKENHI Grant Number JP17K00103.

References

1. T. Takagi and R. Kurozumi, Prototype of a Modeling Tool to Convert between Extended Place/Transition Nets and VDM++ Specifications, *Proceedings of International Conference on Artificial Life and Robotics*, ALife Robotics, Oita, Japan, pp.157-160, Jan. 2019.
2. N.G. Leveson and J.L. Stolzy, Safety Analysis Using Petri Nets, *IEEE Transactions on Software Engineering*, Vol.13, No.3, IEEE, United States, pp.386-397, Mar. 1987.
3. J. Fitzgerald, P.G. Larsen, P. Mukherjee, N. Plat and M. Verhoef, *Validated Designs for Object-Oriented Systems*, Springer-Verlag London, 2005.
4. T. Takagi, S. Morimoto, Y. Ue and Y. Imai, Animated Graphics-based Training Support Method and Prototype Tool for Bug Fixing of Extended Place/Transition Nets, *Journal of Robotics, Networking and Artificial Life*, Vol.5, No.4, pp.278-282, Mar. 2019.
5. Y. Ue and T. Takagi, Learning Support Technique of Software Visual Modeling Using Place/Transition Nets, *Proceedings of International Conference on Artificial Life and Robotics*, ALife Robotics, Oita, Japan, pp.751-754, Jan. 2020.
6. T. Takagi and A. Usuda, A Technique for Learning Software Modeling Using Extended Place/Transition Net and Its Prototype Tool, *Journal of Robotics, Networking and Artificial Life*, Vol.9, No.1, pp.81-86, June 2022.
7. Blockly, https://developers.google.com/blockly (accessed in Dec. 12, 2022).

Authors Introduction

Mr. Akio Usuda

He received the B.S. degree from Kagawa University in 2021. He is a master's student in the Graduate School of Engineering at Kagawa University. His research interests are in software engineering, particularly software design.

Mr. Ryoichi Ishigami

He is an undergraduate student in the Faculty of Engineering and Design at Kagawa University. His research interests are in software engineering, particularly software quality control.

Dr. Tomohiko Takagi

He received the B.S., M.S. and Ph.D. degrees from Kagawa University in 2002, 2004 and 2007, respectively. He became an assistant professor in 2008, and a lecturer in 2013 in the Faculty of Engineering at Kagawa University. Since 2018 he has been an associate professor in the Faculty of Engineering and Design at Kagawa University. His research interests are in software engineering.