Iyer, S. et al. (Eds.) (2022). Proceedings of the 30th International Conference on Computers in Education.
Asia-Pacific Society for Computers in Education

Monitoring of Learners' Activities in Software Structure Design Exercises

**Yasuhiro NOGUCHI[a*], Kanta INOUE[a], Satoru KOGURE[a],
Koichi YAMSHITA[b] & Tatsuhiro KONISHI[a]**
[a]*Faculty of Informatics, Shizuoka University, Japan*
[b]*Faculty of Business Administration, Tokoha University, Japan*
*noguchi@inf.shizuoka.ac.jp

Abstract: Learners acquire modeling skills from software design exercises using UML class diagrams by recognizing and solving errors by comparing their models with the requirement specifications. Teachers and teaching assistants must identify learners' impasse according to the learners' modeling activities and provide them appropriate feedback when they cannot solve their impasse. In this study, we constructed a system to observe learners' modeling activities during exercises. In an experiment, we also collected modeling activity data from learners and analyzed the patterns identifying learners' impasse based on their activities.

Keywords: Modeling exercises, software design, impasse detection, monitoring

1. Introduction

In recent years, as software has become larger and more complex, the importance of learning about software design has increased. A wide variety of learners, both university students and working professionals who are in computer science specialties, require to acquire software design skills (Hara et al., 2019; Krusche et al., 2020). In an exercise to acquire software design skills, learners create a model based on provided requirement specifications, and the learnes review their models each other. However, some learners experience an impasse in the process of creating their own models, and they have trouble reaching the milestone to be reviewed and refining their model based on the feedback. Before the exercise, knowledge of typical mistakes in learners' modeling process (Chren et al., 2019; Chourio et al., 2019) can be provided to learners. However, in the exercises, learners must improve their modeling skills to enable them to recognize and solve errors in their models by themselves. Therefore, teachers and teaching assistants (TAs) should follow learners' modeling activities, including their trial and error, and support them when they cannot break the impasse by themselves.

In such modeling exercises in actual classrooms, a small number of teachers/TAs often must support many learners, and it is difficult to provide one-on-one support for each learner's modeling activities. In addition, it is difficult for teachers to identify only from the learner's created model which part of the model caused an impasse and how the learner performed the trial and error. In this study, we constructed a system for monitoring the modeling activities of multiple learners during their exercises. In an experiment, we collected the modeling activity data of six learners and analyzed the pattern of learners' impasse based on their modeling activities.

2. Modeling Activity Monitoring System

Figure 1 shows an overview of our system. In the exercise, learners use the modeling tool Astah* (Change Vision, 2022) with our add-on software. The add-on software periodically records learners' class diagram in JSON format. The system aggregates these models, which are recorded periodically during the exercise via the network, and extracts the differences between recorded models to create a record of the learner's editing history in the modeling exercise.

The learner's editing history is categorized into three types of operations: create, modify, and delete. Additionally, these operations are classified by the elements to be edited: classes, attributes, methods, and relations (e.g., association, aggregation, composition, inheritance, dependency, etc.).

Moreover, each element has specific properties (e.g., an attribute has type, name, visibility, multiplicity, etc.). Figure 2 shows an example of a learner's modeling activities. In the extraction process, an element is categorized as “create” if it exists in the history at a particular moment but not in the previous point. The reverse case is categorized as “delete.” An element is classified as “modify” if the same location of the element is edited between one point and the previous point. For instance, in (2) of Figure 2, the role name “corners” was modified from “points” and the multiplicity “4” was created. The class “Circle” with an attribute “radius” and method “draw” were created. The association between “Point” and “Circle” was created with the multiplicity “1” and the role name “center.”

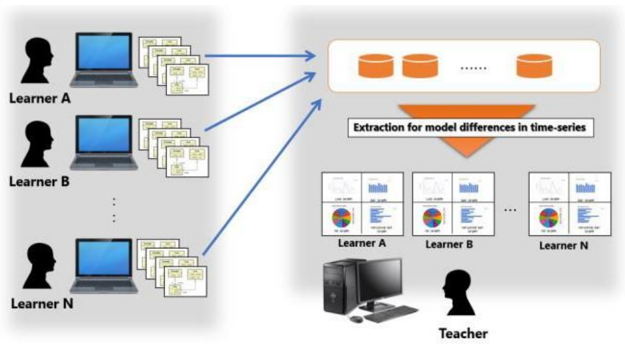

Figure 1. Overview of the monitoring system.

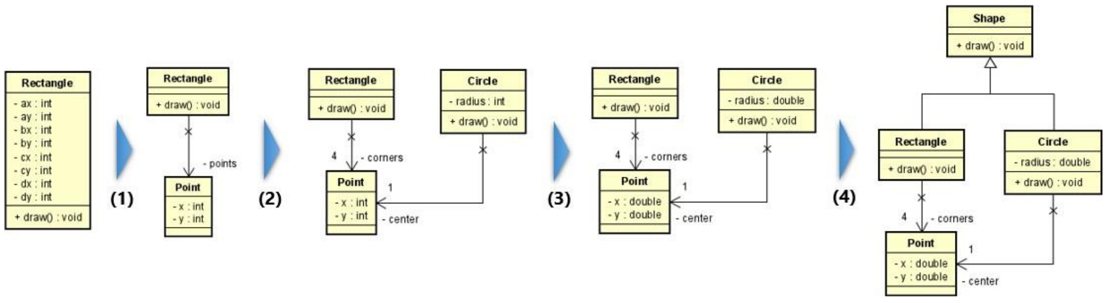

Figure 2. Learner’s modeling activity.

3. Experiment

In this experiment, we collected the modeling activity data of six learners in an exercise to create a model of a point-of-sale system in 60 minutes, and they recorded the time when they felt an impasse. We evaluated three impasse-detection rules based on the learners’ activity data that were customized from rules to detect programming exercise impasses (Yamashita et al., 2017):

- A learner did not change the model for more than 10 minutes.
- A learner deleted an element more than five times within 5 minutes.
- A learner deleted more than 15% of the diagram and it reverted to the previous model.

However, these rules did not adequately identify the learners’ impasse records. Figure 3 shows a learner's editing activities and number of elements in the diagram during the modeling exercise. The purple, red, and blue arrows show the time when a rule correctly identified the learner’s impasse record, the time only when the learner recorded an impasse, and the time when a rule was triggered at not impasse time, respectively.

Regarding the red arrowed time, we observed that the learner recorded the impasse; however, the learner focused on another area instead of refining the elements related to the impasse. Regarding the blue arrowed time, the learner deleted all the attributes in a class and replaced them. The extraction process should be improved to be categorized as “modify” even when the learner takes time to replace a chunk of elements. From the observed modeling activities, learners used various modeling methodologies (Torre et al., 2018; Störrle et al., 2018). One attempted to complete one class and then begin creating the next. Another attempted to create all the classes first and then detail them. Impasse-detection rules must be adjusted depending on the modeling methodology selected by the learner. In addition, some models created by the learners included major errors even when they did not experience an impasse. The rules should be designed based on the learners not having tool-supported systematic verification methods such as “build,” “run,” and “trace” in programming exercises.

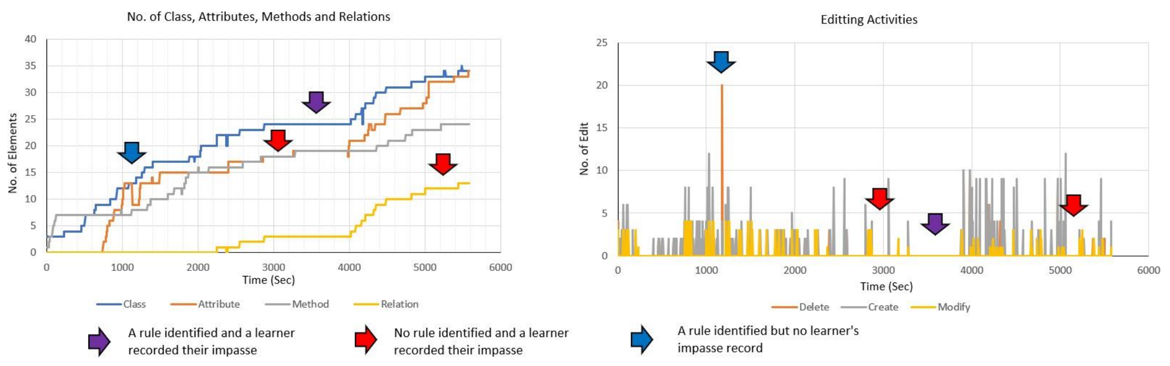

Figure 3. Number of elements and editing activities for a learner

4. Conclusion

In this study, we built a prototype to monitor and analyze learners' modeling activities. Using the prototype, we collected modeling activity data from six learners and analyzed learners' impasse based on their activities. From the learners’ activities, we could observe the characteristics of learners’ modeling activities and differences from programming exercises. In a future study, we will refine the impasse-detection mechanism based on these characteristics.

Acknowledgements

This work was supported by JSPS KAKENHI, Grant Numbers JP18K11566 and 22K12311.

References

Change Vision. Premier Diagramming, Modeling Software & Tools | Astah, https://astah.net, accessed on 2022/05/01.

Hara, S., Kayama., M., Nakao, T., Nagai., T, & Taguchi, N. (2019). A UML programming environment for ICT related subject at Junior High School. *In Proceedings of the 2019 The 3rd International Conference on Digital Technology in Education (ICDTE 2019)*. ACM, USA, 141–146. https://doi.org/10.1145/3369199.3369215

Krusche, S., Frankenberg, N., Reimer, L. M., & Bruegge., B. (2020). An interactive learning method to engage students in modeling. *In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET '20)*. ACM, 12–22. https://doi.org/10.1145/3377814.3381701

Chren, S., Buhnova, B., Macak, M., Daubner, L., & Rossi., B. (2019). Mistakes in UML diagrams: analysis of student projects in a software engineering course. In Proceedings of the 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET '19). IEEE Press, 100–109. https://doi.org/10.1109/ICSE-SEET.2019.00019

Chourio, P., Azevedo, R., Castro, A., & Gadelha, B. (2019). Most common errors in software modeling using UML. *In Proceedings of the XXXIII Brazilian Symposium on Software Engineering (SBES 2019)*. ACM, 244–253. https://doi.org/10.1145/3350768.3353820

Störrle, H., Baltsen, N., Christoffersen, H., & Maier, A. M. (2018). How do modelers read UML diagrams? Preliminary results from an eye-tracking study. *In Proceedings of the 40th International Conference on Software Engineering,* ACM, 396–397. https://doi.org/10.1145/3183440.3195025

Torre, D., Labiche, Y., Genero, M., Teresa., M. B., & Elaasar, M. (2018). UML diagram synthesis techniques: a systematic mapping study. *In Proceedings of the 10th International Workshop on Modelling in Software Engineering (MiSE '18)*. ACM, 33–40. https://doi.org/10.1145/3193954.3193957

Yamashita, K., Sugiyama, T., Kogure, S., Noguchi, Y., Konishi, T., Itoh, Y. (2017). An educational support system based on automatic impasse detection in programming exercises. *Proceedings of the 25th International Conference on Computers in Education*, 288-295.