Chang, M. et al. (Eds.) (2019). Proceedings of the 27th International Conference on Computers in Education. Taiwan: Asia-Pacific Society for Computers in Education

Supporting Knowledge Organization for Reuse in Programming: Proposal of a System Based on Function–Behavior–Structure Models

Kento KOIKE[a*], Takahito TOMOTO[b], Tomoya HORIGUCHI[c] & Tsukasa HIRASHIMA[d]
[a]*Graduate School of Engineering, Tokyo Polytechnic University, Japan*
[b]*Faculty of Engineering, Tokyo Polytechnic University, Japan*
[c]*Graduate School of Maritime Sciences, Kobe University, Japan*
[d]*Graduate School of Engineering, Hiroshima University, Japan*
*k.koike@t-kougei.ac.jp

Abstract: It is important to reuse knowledge acquired through problem-solving in programming. To reuse knowledge, it is effective to first understand differences between knowledge items and then to organize that knowledge. Therefore, we propose a method and develop a support system for facilitating knowledge organization in programming. We further examine a model of parts and problem-solving process of parts based on function–behavior–structure aspects. In this paper, we expand a previously developed system to propose an improved system that provides support based on feedback from these models. The aim of this system is to allow learners to consider behavior rather than thinking directly from function to structure.
Keywords: Programming education, knowledge organization, problem framing

1. Introduction

It is important to reuse knowledge acquired through problem-solving in programming. To reuse knowledge, it is effective to first understand the differences between knowledge items and then to organize that knowledge. Therefore, one of the goals in learning to program is to organize knowledge for reuse at an appropriate time.

Recently, the demand for programming education is increasing worldwide. Enhancing intelligent tutoring systems (ITSs) in programming education is therefore very important. For a computer to intelligently support such learning, it is desirable that it be adaptive to individual learning. In ITS research, learning effectiveness is characterized by (A) controlling features of the question or problem to be asked by indexing based on characteristics of target domains (e.g. Hirashima, Kashihara, & Toyoda, 1992; Horiguchi, Tomoto, & Hirashima, 2015), or by (B) making appropriate interventions such as feedback by grasping problem-solving processes based on explainable problem-solving models (e.g. Hirashima et al., 1992; Hirashima, Kashihara, & Toyoda, 1995).

Regarding (A), assuming that problem solving is directed toward acquisition of knowledge required for a solution, descriptions of the programming knowledge itself lead to indexing of the problem. Some studies have utilized structure–behavior–function aspects, combining each aspect to handle knowledge in parts and using them for knowledge descriptions. Taking the example of qualitative physics, behavior can be defined as "multiple input–output relationships with causality in the object structure," and function can be defined as "the terminology assigned representing the function to the behavior" (de Kleer & Brown, 1984). Based on this definition, qualitative physics makes it possible to express phenomena and problem solutions by combining and correcting parts. In ITS, many examples apply structure–behavior–function aspects to descriptions of learning objects (e.g. Kashihara, Hirashima, Toyoda, & Nakamura, 1992; Matsuda, Kashihara, Hirashima, & Toyoda, 1997). A premise of ontology engineering is that "all artifacts are composed of a combination of parts" (Sasajima, Kitamura, Ikeda, & Mizoguchi, 1996). By this, an object is defined as being composed of parts can be processed to convert input to output based on the user's requirements. This makes it possible to express part structures in terms of the relation between their smaller subparts. Furthermore,

ontology engineering has expanded the idea of qualitative physics, giving different definitions for conventional functions and behaviors. Specifically, “behavior” is defined as “state transitions of an object that changes with time, and the result of simulations of the object required by the user,” and “function” is defined as “the result of interpreting the function based on the goal of the entire system.”

Based on these ideas from ontology engineering, we considered the problem-solving process in programming using these three aspects (Fig. 1) (Koike, Tomoto, Horiguchi, & Hirashima, 2018b, 2018a). First, if the end-state is output from a given initial state of variables provided as input to the source code that is the structure, the difference between the input and the output provided by the structure can be observed as the behavior of the structure. Furthermore, the observed behavior can be interpreted as the function by assuming the goal (context) of the overall system. Therefore, we apply the concept of ontology engineering and express knowledge in programming by treating the structure–behavior–function aspects together as parts. Thus, we proposed a method for facilitating knowledge organizing in programming and developed a support system for that method.

Regarding (B), there have been many attempts to grasp problem-solving processes in programming in terms of the program comprehension or program understanding concepts from system engineering. A number of models have been proposed (e.g., Harth & Dugerdil, 2017; von Mayrhauser & Vans, 1995), but no models that aim at supporting learning have been proposed. In this research, by defining programming knowledge as parts, we approach various elements related to programming that have previously been considered tacit and clarify and organize each element independently of the programming language used. In this way, we try to construct a model of the problem-solving process using parts from the viewpoint of learning and formalize tacit knowledge.

We expect realization of the following by our model of ITS for programming education:

- Adaptive presentation of problems
- Adaptive correction of incorrect judgments
- Grasping of errors in the overall learning process
- Estimation of learner knowledge states
- Encouragement of learning by trial-and-error

In this paper, we expand a previously developed system through these problem framings and propose an improved system. The proposed system has two new functions, one that provides feedback for grasping problem-solving processes and one for generating tasks on a criteria basis. The aim of this system is to allow learners to consider behavior rather than thinking directly from function to structure.

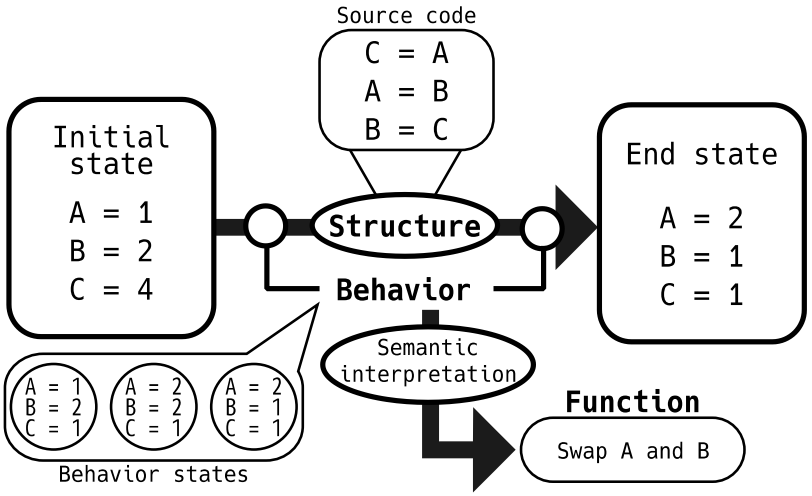

Figure 1. Problem-solving process in programming.

2. Method for Supporting Knowledge Organization and the Developed System

This section introduces the expandable modular statements (EMS) method, a learning method directed at knowledge organization that we previously proposed, and a newly developed knowledge organization support system for this method. Furthermore, we show a system configuration that addresses issues in the previous system.

2.1 Expandable Modular Statements Method

EMS has two aspects: construction and expansion (Fig. 2). First, parts that learners should construct are presented as a task, and the learners write code to construct chunks that include meaningful series as parts. Next, parts including the constructed parts are presented as a task, and learners construct larger parts that reuse the previously constructed parts.

In the example shown in Fig. 2, the "swap" parts are constructed by learner writing and combining "$c = a$," "$a = b$," and "$b = c$" in the "make swap" task. Then, in the "make sort" task, learner adding "if $a > b$" to the previously created "swap" expands the "sort" as larger parts. By repeating the construction, reuse, and expansion of such parts in this learning method, the learner is directed toward awareness of relations between each parts, and training for reuse.

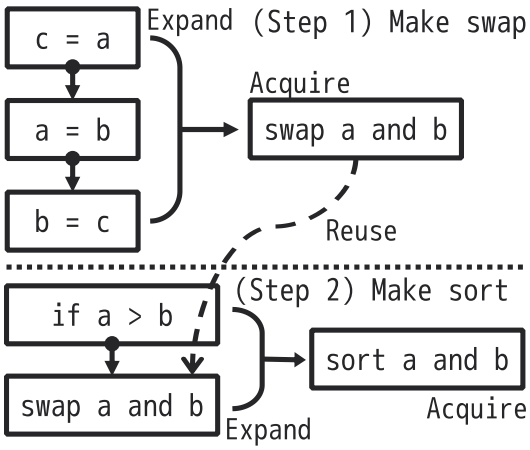

Figure 2. An example of EMS.

2.2 Previous Knowledge Organization Support System

We previously developed a system that supports the organization of knowledge use and EMS. Figure 3 shows the user interface of the previous system. In this user interface, the learner learns based on the EMS method through system assistance. Evaluations of this system showed that it significantly supports learning and code reuse (Koike, Tomoto, Horiguchi, & Hirashima, 2019).

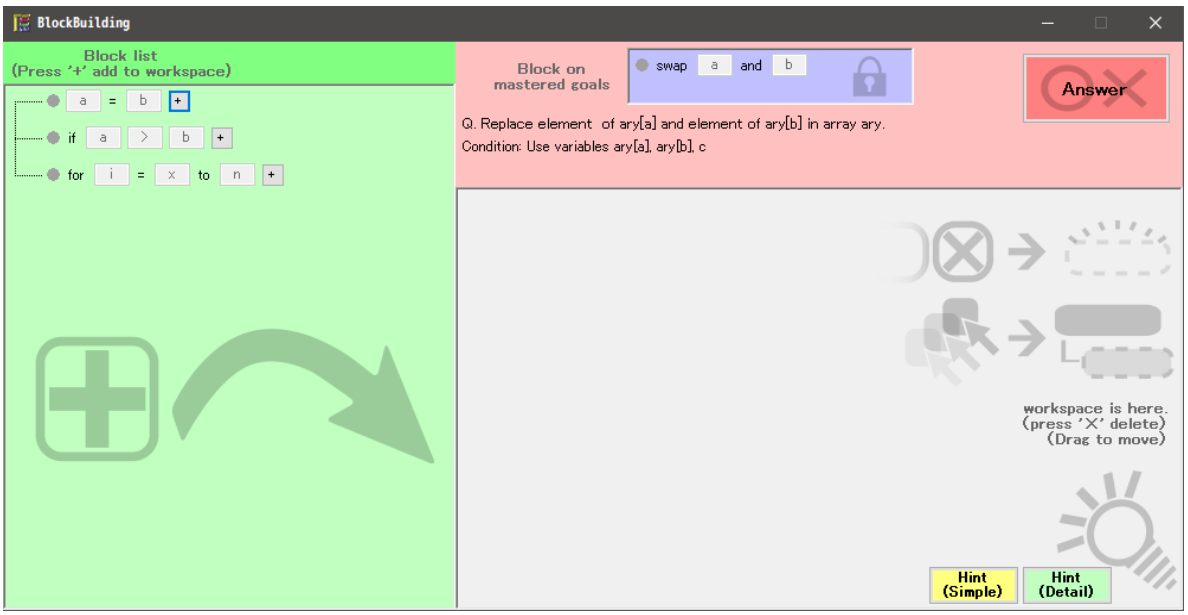

Figure 3. User interface of the previous system.

Figure 4(a) shows the configuration of the previous system, in which problems are presented stepwise in a certain task sequence, and the learner learns according to that sequence. When responding to tasks or pressing the hint button, the system provides feedback to direct the learner. The previous system, however, presents only task sequences that we have subjectively determined. Furthermore, the provided feedback only compares the structure of learner answers with the correct answer to point out deficiencies and excesses.

Therefore, we examined the definition of a parts model as knowledge and a model of the problem-solving process as related to the acquisition and expansion of parts, so that the system can better grasp and support learning (see Section 3). In the newly proposed system, these models are implemented in the previous system to address some issues in that system.

Figure 4(b) shows the configuration of the proposed system, in which tasks can be generated from a system parts database and task sequences can be automatically generated while considering differences between tasks. Based on the problem-solving process, learners can grasp in which processes errors occur when parts are incorrectly constructed or consider which part elements are insufficiently understood. This should provide more adaptive feedback.

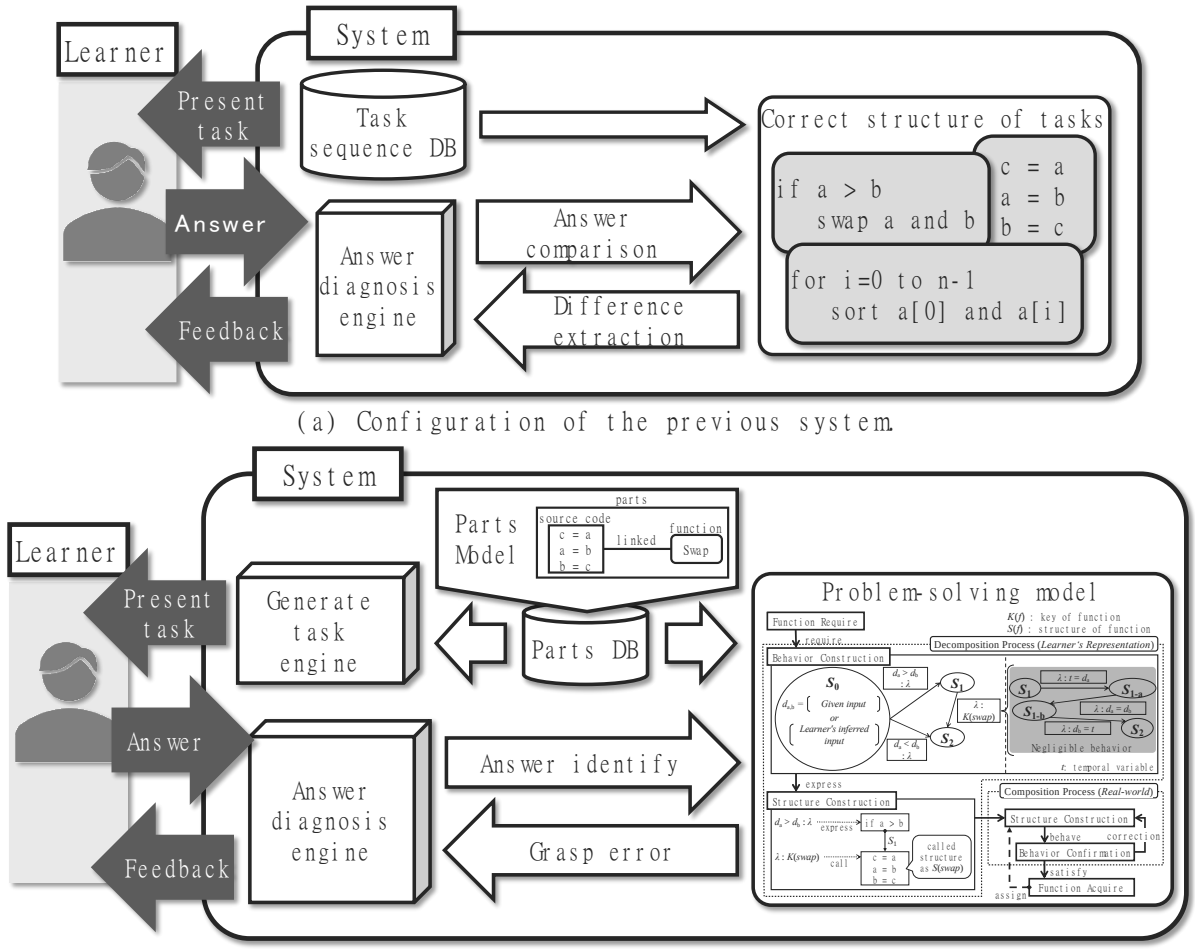

Figure 4. System configurations of the previous and proposed systems.

3. Parts Model and Problem-Solving Process Model

This section outlines the proposed parts model and a model for problem-solving processes based on that parts model.

3.1 Parts Model

To reuse a meaningful series of code as parts, learners must associate stored source code and its behavior. Source code behavior can be categorized as overall behavior or meaning attached to a target behavior according to the code's purpose. For example, in the source code "c = a," "a = b," and "b = c" in "swap," there are state transitions such as "change variable c to the value of a," "change variable a to the value of b," and "change variable b to the value of a." However, only the latter two commands are interpreted as the meaning of the "swap" function, and the behavior "change variable c to the value of a" is not focused upon.

Therefore, in this research we define "structure" as source code, "behavior" as the overall source code behavior, and "function" as meaning attached to behaviors toward the intended purpose. Figure 5 shows the relations between these three elements.

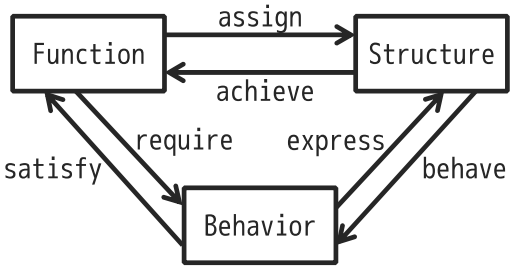

Figure 5. Parts model.

3.2 Problem-Solving Process Model

Figure 6 shows a model of the problem-solving process based on the parts definition. In this research, the problem-solving process related to parts is divided into two processes: "decomposition of required functions" performed as a representation of learners, and "composition of required functions" performed in the real world. In particular, we focus on the decomposition process with the aim of grasping the process by which learner answers are derived in the real world. Therefore, in this model, how each part element is derived in the decomposition process is modeled as a problem-solving process.

The model in Fig. 6 expresses the problem-solving process in a case assuming that the task "sort two variables (sort)" will be accomplished by expanding the previously coded "replace two variables (swap)." First, we derive the behavior required to achieve the "sort" function. In other words, to code the function, the state transition to be performed is derived based on input given explicitly or assumed by the learner. In state transitions, the state is denoted by S and the input is denoted by d. When transitioning from S_0 to the next state, transition conditions and operations are expressed in Fig. 6 as a rectangle on the edge between the current state and the next state S_n. The expression in the rectangle represents "transition condition: operation," and "λ" indicates no operation. For example, the state transition $S_0 \rightarrow S_1$ is performed only when $d_a > d_b$, while the state transition $S_1 \rightarrow S_2$ in the K(*swap*) operation is performed unconditionally. Here, K(*swap*) is the meaning of the index that recalls. the "replace two variables" structure and is used to omit the behavior to be represented by reusing the learner's existing knowledge. A learner not having this knowledge cannot achieve this task without assuming the detailed operation shown in the gray part in the upper right. But if the parts of *swap* are known in advance, it is not necessary to consider detailed behaviors.

Once the required behavior is realized, the "Structure Construction" block then constructs the structure by combining parts to express the desired behavior. Parts corresponding to each behavior are combined, but "λ: K (*swap*)" reuses a structure from the learner's existing knowledge based on the *swap* index.

Next, the "Composition Process" block writes out the solved structure in the real world and verifies its behavior. If the structure behavior satisfies the function, the learner assigns the function to the structure and acquires it as knowledge. The learner's knowledge state can thus be expressed as a set of parts (Fig. 7). For example, in constructing the structure in the decomposition process in Fig. 6, the structure is recalled by indexing the parts from which the K(*swap*) function is created from the learner's existing knowledge. Furthermore, "sorting of two variables" added to the process in Fig. 6 is newly

added to knowledge as *K*(*2var_sort*). In the process in Fig. 8, a learner who has acquired "sorting of two variables" from the process in Fig. 6 next aims at coding "determine the minimum value in the array." In this way, knowledge is organized by expanding parts in stages.

The function name *K*(*name*) used for the index is only a label for distinguishing parts among learners; it has no other meaning.

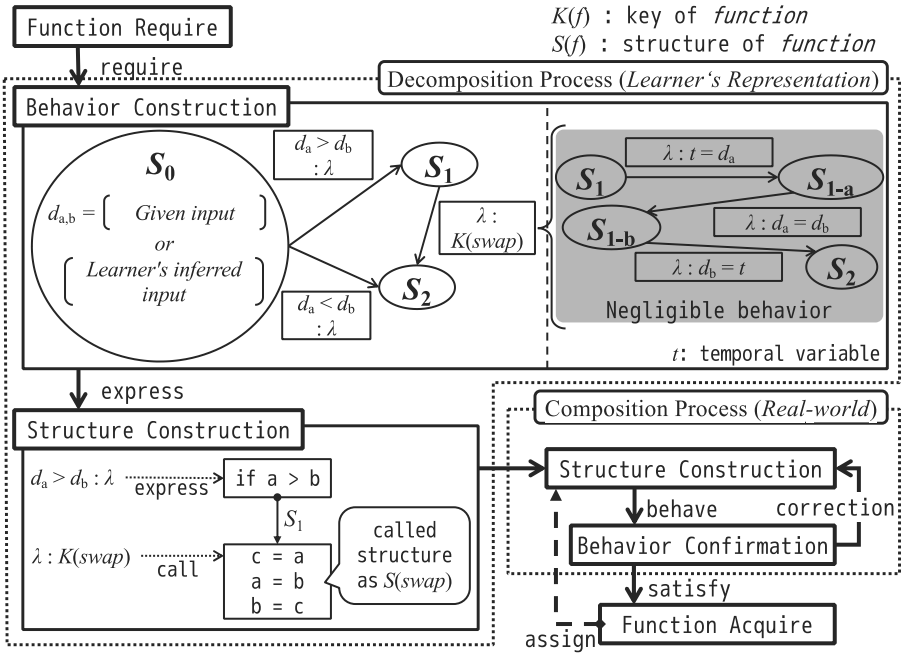

Figure 6. Model of the parts-based problem-solving process (the example of *2var_sort*).

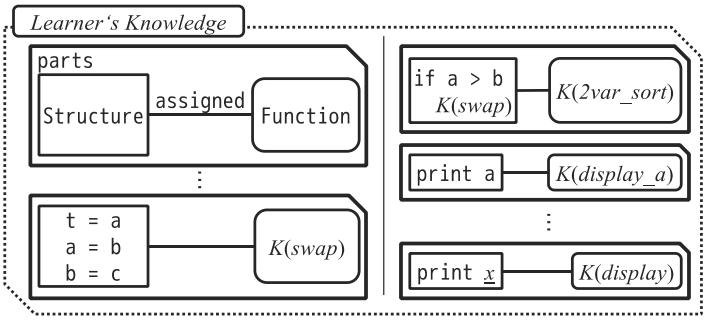

Figure 7. Expression of learner knowledge states.

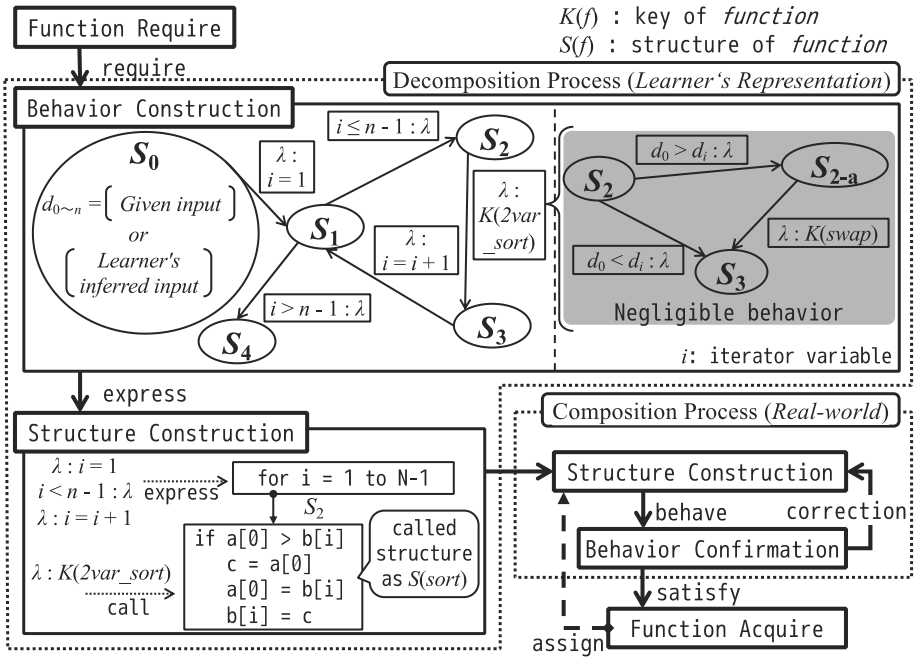

Figure 8. Model of the parts-based problem-solving process (the example of *search_min_value*).

4. Evaluation of Explanations Generated by the Model

We have developed a learning support system with a learning method that promotes acquisition of parts and confirmed its effectiveness. We also verified that system feedback effectively promotes learning. However, the previous system was unable to grasp behaviors. The previous system handles parts as knowledge only within the structure–function correspondence relationship, and system feedback could only point out insufficient points by comparing the structure developed by the learner and that of the correct answer. We consider that adding behavior elements to parts based on the proposed model enables feedback that was impossible in the previous system.

To verify the effectiveness of feedback from the proposed model, we performed an evaluation experiment for comparison with feedback from the conventional system.

4.1 Evaluation Method

Participants were 17 undergraduate and graduate students who have studied programming for at least three years in programming lectures. All participants had acquired basic concepts such as “for” and “if” statements, sorting algorithms, and functions.

The experiment first suggests possible learner errors and two possible explanations for that error (Table 1). The detail of explanation sets is omitted due to space limitations. Each set of explanations was evaluated by five questions (Table 2). One explanation is that from the previous system, and the other is obtained from the proposed model.

The method was evaluated by having participants answer questions using a five-scale response, with the explanation from the previous system as 1 and the model explanation as 5. Participants were also asked to describe the reason for their answers.

Table 1 *Example Explanation Set*

Explanation Set 3	
Problem	Make a function that sorts and outputs two variables
Possible learner error	if a > b a = b b = a print a print b
Explanation from previous system	In line 2, “c = a” is the correct answer.

Model explanation	This function is a combination of "sort two variables" and "output two variables." Therefore, "output two variables" is correct, but in "sort two variables," it is necessary to assign the value of *a* to a temporary variable before assigning the value of *b* to *a* in the second line.

Table 2 *Evaluation Questions*

Questions
(Understanding) When describing a learner's error, which explanation do you think would provide deeper understanding of the program?
(Develop) When describing a learner's error, which explanation do you think would best help the learner develop programs in the future?
(Explore) When describing a learner's error, which explanation do you think would best help learners identify errors in the future?
(Appropriate) When describing a learner's error, which explanation do you think is most appropriate for beginners?
(Difficult) When describing a learner's error, which explanation do you think is easiest to understand?

4.2 Evaluation Results

Table 3 summarizes the experimental results. The number of *N* indicates the total of each evaluation in four explanation sets by 17 participants. Table 3 showing that the model was highly evaluated in all responses to explanation sets 1 to 4.

Table 4 shows each explanation set and its average value. The number of *N* indicates participants in each evaluation. In the results for each explanation, the model explanation is generally highly evaluated in all questions.

These results suggest that the proposed model can provide more appropriate feedback that can be realized in the system, suggesting that effective support can be realized through learning of parts under the proposed model. However, this evaluation has mainly two limitations. One is unclear the reproducibility of learners' emotions in the actual learning environment. Another one is unclear the effect on learner skill.

Table 3 *Overview of Results (1: Conventional Explanation – 5: Model Explanation)*

N = 68	Understanding	Develop	Explore	Appropriate	Difficult
Average	4.26	4.38	4.37	4.13	4.06
Positive	55	56	58	48	50
Negative	9	6	7	7	11

Table 4 *Results of each Explanation Set (ES)*

N = 17	Understanding	Develop	Explore	Appropriate	Difficult
ES-1	4.12	4.59	4.24	4.12	4.12
ES-2	4.47	4.47	4.59	4.06	3.65
ES-3	4.41	4.18	4.12	4.12	4.12
ES-4	4.06	4.29	4.53	4.24	4.35

5. Proposed Knowledge Organization Support System

5.1 Overview

As described in Section 2, we proposed a system for knowledge organization and have developed a support system for it. This section extends this support system and proposes a system that implements the model outlined in Section 3 (Fig. 9).

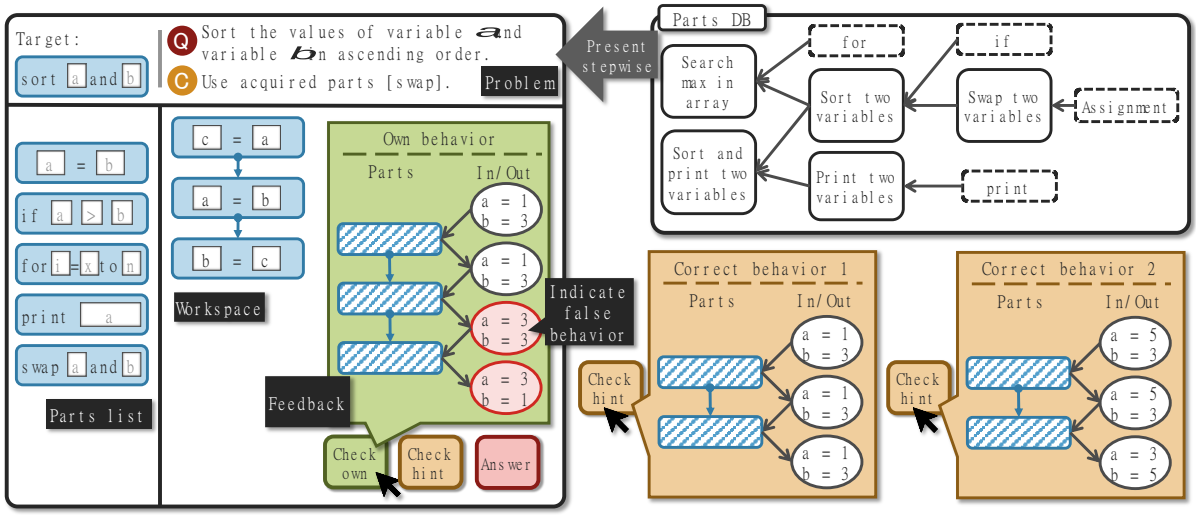

Figure 9. Interface of the proposed system and operation examples.

The proposed system generates behaviors from learner-constructed parts by grasping models of parts and problem-solving processes and furthermore deriving correct behaviors from the structure of parts contained in the system. It is thus possible to provide explicit feedback regarding behavior of the learner's structure and differences with the correct behavior. The goal is to make learners more conscious of implicit behavior and to organize knowledge based on knowledge differences.

In addition, this system improves the fixed task sequence of the previous system, and can adaptively generate task sequences based on the inclusion relation of parts. Consequently, task presentations that were previously subjective are now based on certain criteria.

5.2 Learning and Feedback

This section describes the specific learning flow, using the interface of the proposed system shown on the left side of Fig. 9 as an example. In that example, the learner has already learned how to swap values. First, a problem statement is presented to the learner, and the learner constructs the target structure by combining parts in the parts list in the construction area while following the constraints indicated by the conditions. When the behavior confirmation button is pressed, behavior of the learner's structure is generated as feedback, and errors are indicated in comparison with the behavior generated from the structure of the correct parts, which are stored in the system. When the correct behavior button is pressed, the correct behavior with blank parts in some datasets is presented as shown at the lower right of Fig. 9. These two forms of feedback give learners an opportunity to think about how their own structure should behave.

In the previous system, learning was linked only to the relation between function and structure. The proposed system, however, is expected to transform learning by including behavior from this feedback.

5.3 Task Presentation

As shown at the upper right of Fig. 9, tasks in the proposed system are presented stepwise, aiming toward a final goal for parts based on inclusion relations in a parts database stored in the system. If parts from other sequences are to be synthesized as subsequent parts, the system presents a task from the root of that sequence, promoting the acquisition of other parts to be synthesized. For example, after solving "Sort two variables" task shown the left side of Fig. 9, "Search max in array" task is presented shown the Fig. 10.

However, this research aimed at organizing knowledge already stored for reuse. Therefore, there is no acquisition of primitive parts that the system considers unnecessary, as shown by the dotted line at the upper right of Fig. 9.

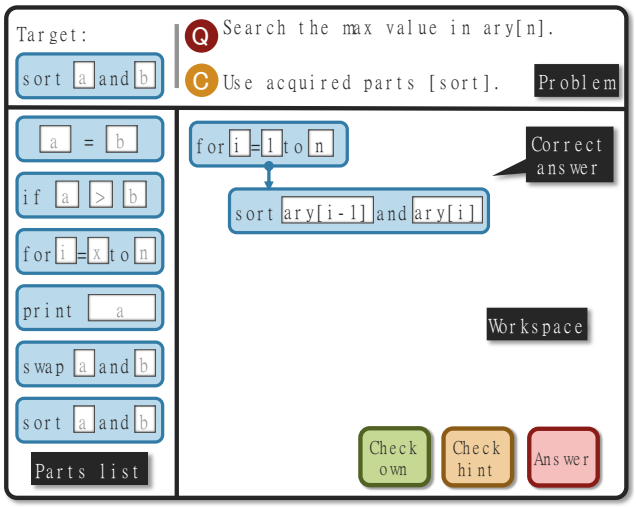

Figure 10. An example of task presenting in the proposed system.

6. Discussion and Future Works

We supported knowledge organization by previously proposing a knowledge organization method aiming at improved reusability of knowledge and by developing and evaluating that system. However, there were issues in problem presentation and feedback in that system. Therefore, we aimed at improvement by using models for parts and for the examined problem-solving process.

This paper proposed a system that improves task presentation and feedback functions by implementing these models in the system. Specifically, we added functions for generating sequences based on certain criteria from a parts database and for suggesting and teaching behaviors. The proposed system can be expected to support knowledge organization while allowing learners to focus on part behaviors.

In future work, we will implement the proposed system and perform evaluation experiments using that system.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Numbers JP18K11586, JP15H02931, JP16K12558, and JP17H01839.

References

de Kleer, J., & Brown, J. S. (1984). A qualitative physics based on confluences. *Artificial Intelligence*, *24*(1), 7–83. https://doi.org/https://doi.org/10.1016/0004-3702(84)90037-7

Harth, E., & Dugerdil, P. (2017). Program Understanding Models: An Historical Overview and a Classification. In *Proceedings of the 12th International Conference on Software Technologies - Volume 1: ICSOFT,* (pp. 402–413). SciTePress. https://doi.org/10.5220/0006465504020413

Hirashima, T., Kashihara, A., & Toyoda, J. (1992). Providing problem explanation for ITS. In *International Conference on Intelligent Tutoring Systems* (pp. 76–83). Springer.

Hirashima, T., Kashihara, A., & Toyoda, J. (1995). A Formulation of Auxiliary Problems and Its Evaluations. In *Proceedings of 7th World Conference on Artificial Intelligence in Education: AI-ED95* (pp. 186–193).

Horiguchi, T., Tomoto, T., & Hirashima, T. (2015). A framework of generating explanation for conceptual understanding based on “semantics of constraints.” *Research and Practice in Technology Enhanced Learning*, *10*(1), 1.

Kashihara, A., Hirashima, T., Toyoda, J., & Nakamura, Y. (1992). Advanced explanation capabilities for intelligent tutoring systems: The explanation structure model (EXSEL). *Systems and Computers in Japan*, *23*(12), 93–107. https://doi.org/10.1002/scj.4690231209

Koike, K., Tomoto, T., Horiguchi, T., & Hirashima, T. (2018a). Proposal of a Framework for Stepwise Task Sequence in Programming. In *International Conference on Human Interface and the Management of Information* (pp. 266–277).

Koike, K., Tomoto, T., Horiguchi, T., & Hirashima, T. (2018b). Proposal of an Adaptive Programming-Learning Support System Utilizing Structuralized Tasks. In *Workshop proceedings of the International Conference on Computers in Education ICCE 2018* (pp. 278–287).

Koike, K., Tomoto, T., Horiguchi, T., & Hirashima, T. (2019). Proposal of the Expandable Modular Statements Method for Structural Understanding of Programming, and Development and Evaluation of a Learning Support System. *Transactions of Japanese Society for Information and Systems in Education*, *36*(3), 190–202. (*in Japanese*)

Matsuda, N., Kashihara, A., Hirashima, T., & Toyoda, J. (1997). An Instructional System for Behavior-Based Recursive Programming. In *Proceedings of Artificial Intelligence in Education 97* (pp. 325–330).

Sasajima, M., Kitamura, Y., Ikeda, M., & Mizoguchi, R. (1996). A representation language for behavior and function: FBRL. *Expert Systems with Applications*, *10*(3–4), 471–479.

von Mayrhauser, A., & Vans, A. M. (1995). Program comprehension during software maintenance and evolution. *Computer*, *28*(8), 44–55.