Case-Based Evaluating Assistant of Novice Programs

Hiroyoshi Watanabe, Masayuki Arai and Shigeo Takei
School of Science and Engineering, Teikyo University.
1-1 Toyosatodai Utsunomiya-shi Tochigi, 320-8551 Japan
E-mail: {hiro,arai,takei}@ics.teikyo-u.ac.jp

This paper presents a method of implementing an evaluating assistant system that supports teachers' evaluation work of students' programs using case-based reasoning. The target evaluation tasks are to judge whether a student's program satisfies the requirements of the given problem and to give advice for the student's program. The case-based evaluating assistant system compares a program submitted by a student with evaluation cases in the case-base. If some case matches the program, the system applies the judgment and advice on the case to the program. We implemented a case-based evaluating assistant system for novice programs written in an assembly language based on the proposed method. The implemented system was utilized for actual classes and the results showed that the system reduced the teachers' evaluation work drastically.

Keywords: **program evaluation, programming classes, supporting teachers, case-based reasoning**.

1 Introduction

This paper describes a method of supporting teachers' evaluation work of students' programs. In programming education, programming exercise courses play an important role, because writing programs is indispensable to learning programming. In programming exercise classes, however, the teacher's loads of evaluation tend to be very heavy because the teacher has to read so many programs and reports. We aim to implement a computer system as an evaluating assistant that supports the teachers' evaluation work.

There are two approaches to the evaluation of students' programs. The first one is to diagnose programs and give advice by knowledge-based program recognition [1]-[3]. Most of the systems based on the approach require a huge amount of knowledge on bugs, and it would be difficult to constitute the systems practically. The second approach is to support teachers' evaluation work [4]. This approach may not necessarily aim at automating the evaluation work. It aims at implementing practical systems by limiting computers' evaluation work. We took the second approach.

We consider the program evaluation work as collaboration between teachers and computer systems and propose an evaluating assistant model of programs. We also propose a framework of the case-based evaluating assistant system.

2 Target Task and Evaluating Assistant

2.1 Task of Program Evaluation

The target evaluation tasks of a student's program are the following two tasks: (1) the first task is judging whether a student's program satisfies requirements of the given problem. When teachers set problems, they have educational intentions about what students should learn, namely concepts, algorithms, instructions and so on. Teachers read students' programs to see whether the educational intentions are achieved. Therefore, teachers accept a student's program when the program satisfies requirements of the given problem. The first

task is defined on the assumption that students have to submit their programs over and over until their programs are accepted. (2) The second task is giving written advice. Teachers give advice to students whether they accept a program or not: teachers give advice about the reasons why the program is rejected, and advice about bettering the program even if the program is accepted.

2.2 Evaluating Assistant of Programs

Figure 1 illustrates an evaluating assistant in the electronic submission environment of programs. The evaluating assistant pre-evaluates submitted programs and a teacher can refer to the results when he or she evaluates the programs. If the teacher trusts the evaluating assistant, the results from the assistant can be sent to students directly. Such an evaluating assistant is expected to save a teacher a lot of time and energy.

The output of the evaluating assistant consists of evaluation results, their reasons and the degree of confidence. The evaluation results include the judgment of acceptability (*accept* or *reject*) and written advice. The degree of confidence is one of *surely*, *probably* or *unknown*. When the degree of confidence is *unknown*, evaluation results and their reasons are not given.

The evaluation results of the assistant are required to be always correct when the degree of confidence is *surely*. If so, the results of the assistant with *surely* confidence can be sent to students directly, in other words, teachers can trust the evaluating assistant.

The evaluating assistant should have the capability to learn. The final results of the teacher's evaluation are available for the learning. If the assistant is capable of learning, almost the same programs as ones which the assistant has evaluated incorrectly, are expected to be evaluated correctly in the future.

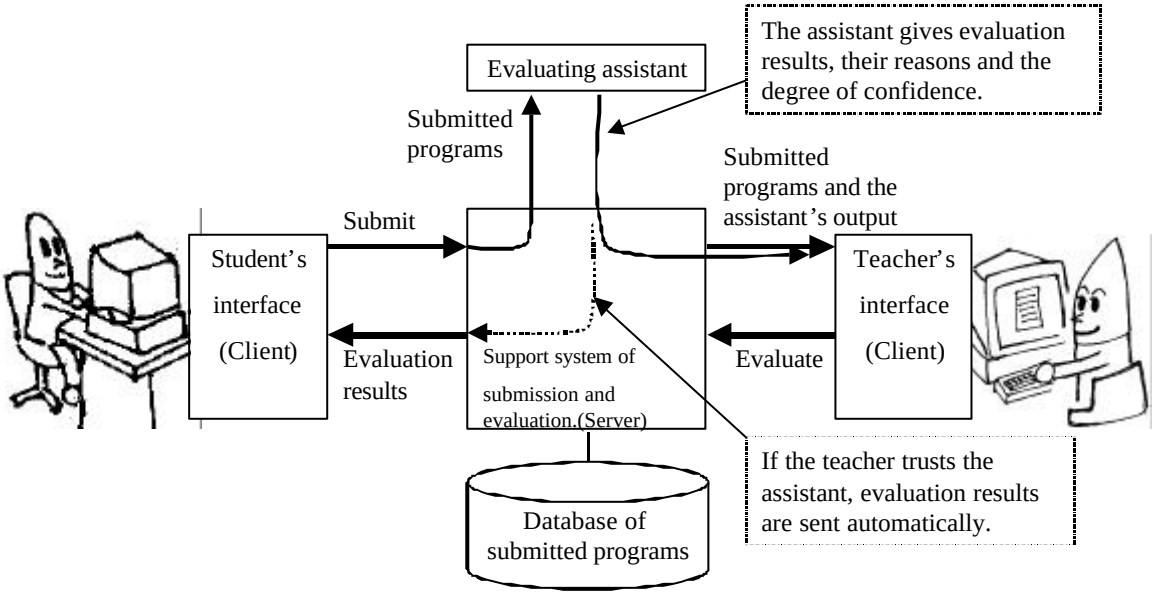

Figure 1 Evaluating assistant of programs

3 Case-based Evaluating Assistant

The case-based reasoning approach is one of the best approaches to implement an evaluating assistant described in Section 2.2. Case-based reasoning systems make use of stored past cases directly in solving newly presented problems [5]. The case-based evaluation of programs is defined as "if some evaluation case of a program whose implementation is the same as the newly given program, then the evaluation results on the case are applied to the given program".

3.1 Representation of Cases

A case for the case-based evaluating assistant consists of retrieval information, problem description, solution description and maintenance information.

(1) The retrieval information includes problem identification that the program is written for and features of the program. The features of the program depend on target programming languages. For example, numbers of if-statements, while-statements and other statements are available in the case of language C.

(2) The problem description in the domain of a program evaluation task is a program list itself. A program list should be represented as a normalized form [2], or a generalized form [6], because there are many variations of program lists for the same implementation.

(3) The solution description includes the judgment of acceptability and written advice.

(4)The maintenance information includes a teacher's name and the date of adding or updating the case.

3.2 Processes of Case-based Program Evaluation

The processes of case-based program evaluation are the following:

(1) Problem analysis: The retrieval information is extracted from a student's program list.

(2) Case retrieval: Cases are retrieved using information generated by analyzing a given program. Cases that have no possibility of matching the given program should be pruned here.

(3) Evaluating and selecting cases: Evaluating cases is the process of matching a given program against cases. The purpose of the process is to investigate whether the given program has the same implementation as the cases, or not. All candidates of cases are evaluated and the best match case is selected. The method of matching programs depends on the target programming languages.

(4) Applying and adapting cases: If there is a case that matches the given program, the judgment of acceptability on the case is applied to the given program. In addition, advice sentences on the case are available for the given program, although the sentences should be adapted for the given program. If no case matches the given program, the judgment and advice is not generated.

3.4 Case-base Maintenance

The maintenance of the case-base is performed using teacher's final evaluation results. One of the most important maintenance tasks is adding new cases when the evaluation results of the assistant are different from the teacher's. New cases are also added when the confidence of the evaluating assistant is not *surely*. More advanced maintenance, e.g., generalizing, specializing and forgetting cases [7], may be needed in order to refine the case-base.

4 The Evaluating Assistant System for Assembly Language Programs

Based on the proposed idea, we implemented a case-based evaluating assistant system for novice programs written in an assembly language [6]. The target assembly language is CASL which is adopted in examinations for information-technology engineers certified by the Japanese ministry of international trade and industry.

4.1 Implementations Depending on The Target Language

In this section, implementations depending on the target language CASL are described.

(1) Evaluating the program's action: Before the case-based program evaluation, the assistant system tests the action of a submitted program using prepared sample data. Only programs executed correctly are evaluated by case-based reasoning [6].

(2) Case representation: Although cases are represented in the form described in Section 3.1, no features except for a program ID are used for the retrieval information. A program list in a case is represented in

CASL itself, or its generalized form that we defined [6].

(3) Case retrieval: The implemented system retrieves all cases whose problem ID is the same as a given program. That is to say, the system does not prune candidate cases.

(4) Case evaluation (program matching): The program matching process aims at making consistent correspondences of instructions, labels and registers between a case and a student's program [6]. If the following condition1 is met, the case matches the given program.

- **Condition 1**: All instructions of the case correspond to instructions of the given program, and all instructions that correspond to nothing do not affect to the program's action.

Especially, if the following condition is satisfied, it is called a "perfect match":

- **Condition 2**: Instructions of the case and the given program correspond one-to-one and the differences of the order of corresponding instructions are trivial.

If the best match case meets condition2, *surely* is assigned as the degree of confidence. If the best match case meets condition1 but not condition2, *probably* is assigned. In the other cases, that is, when no case meets the condition 1, *unknown* is assigned.

4.2 Experimental Results

The implemented assistant system was utilized for actual classes of the CPU and assembly language course at our university in 1999. Seventy-three sophomore students in the department of computer science took this course. Problems presented in classes of the course are the following: (P1) select bigger of the two given integers, (P2) sum the given N integers, (P3) select the maximum of the given N integers, (P4) rotate N bits to the right and (P5) check the correspondence of "(" and ")".

Table 1summarizes results of using the assistant system. The following are found from Table 1:

- The values of (f) show that the implemented case-based assistant system achieves sufficiently high accuracy of judgments. Furthermore, the accuracy of the case-based assistant system satisfies the requirements described in Section 2.2, because it is a hundred percent in cases of *surely* confidence.
- The values of (g) show that the ratios of available advice without modifying are not as high as the accuracy of judgments, although it is fairly high.
- Because teachers do not need to evaluate the acceptability when the case-based assistant system outputs evaluation results with surely confidence, it is estimated that the system reduces the teachers' evaluation work by percentages shown as (h). In other words, by using the assistant system, the evaluation work of teachers is reduced by 60 to 90 percent depending on problems.

These results demonstrate that the case-based assistant system is very effectual in reducing teachers' evaluation work. Still, there is room for improvement in the capability to generate written advice.

Table 1 Evaluation data of the assistant system based on practical use in classes

	P1	P2	P3	P4	P5
(a)Cases saved in the case-base	15	10	29	34	38
(b) Submitted programs	119	119	140	156	157
(c)Programs rejected by checking their action	44	39	44	54	79
(d)Programs evaluated by the assistant system with *surely* confidence	62	72	72	67	46
(e)Programs evaluated by the assistant system with *probably* confidence	8	3	5	5	0
(f)Judgment accuracy of the assistant system (%)	100 (100)	100 (100)	97.4 (100)	100 (100)	100 (100)
(g)Ratio of available advice generated by the assistant system without modifying (%)	92.9 (100)	69.3 (72.2)	88.3 (90.3)	66.7 (64.2)	91.3 (91.3)
(h) $((d)/((b)-(c)))\times 100$ (%)	92.7	90.0	75.0	65.7	59.0

( ) : values for programs evaluated by the assistant system with *surely* confidence only.

5 Conclusions

We have proposed a concept of a program evaluation assistant and a method of implementing the assistant by case-based reasoning. Based on the method, we implemented a system for a simple assembly language CASL and used it in actual classes; the results demonstrated that the system reduced teachers' evaluation work drastically. We plan to improve the modification functions of advice sentences (written advice) and the method of the case-base maintenance. This research was supported in part by the Japanese Ministry of Education Grant No.11680400 and No.12780293.

References

[1] Adam,A. and Laurent,J.P., "LAURA, A System to Debug Student Programs, Artificial Intelligence", vol.15, pp.75-122 (1980).

[2] Johnson,W.L., "Understanding and Debugging Novice Programs", Artificial Intelligence, vol.41, pp.51-97 (1990).

[3] Ueno,H., "Concepts and Methodologies for Knowledge-Based Program Understanding ― The ALPUS's Approach", IEICE Trans. Inf. & Syst., vol.E78-D, no.2, pp.1108-1117 (1995).

[4] Konishi,T., Suyama,A. and Itoh,Y., "Evaluation of Novice Programs Based on Teacher's Intention", Proc. of ICCE95, pp.557-566 (1995).

[5] Kolodner,J. ," Case-Based Reasoning", Morgan Kaufmann Publishers,Inc. (1993).

[6] Watanabe,H., Arai,M. and Takei,S. , "Automated Evaluation of Novice Programs Written in Assembly Language, Proc. of ICCE99, vol.2, pp. 165 - 168 (1999).

[7] Watanabe,H., Okuda,K. and Yamazaki,K., "Methods for Adapting Case-bases to Environments", IEICE Trans. Inf. & Syst., vol.E82-D, no.10, pp.1393 - 1400, (1999).