Supporting the Acquisition of Programming Skills with Program Construction Patterns

Max Kesselbacher
Department of Informatics Didactics
University of Klagenfurt
Klagenfurt, Austria
max.kesselbacher@aau.at

***Abstract*—A central topic to computer science education is the training of novice programmers. Novice programming skills have already been mapped to hierarchical levels, and expert programming skills have been measured based on task performance. But didactical instructions and individual support to acquire expert programming skills have not been provided in detail. I propose the investigation of structural and semantic patterns in program construction sequences in an IDE-based *learning analytics* setting. I aim to provide a more fine-grained assessment of programming skills, to enable skill assessments during programming tasks, and to support the individual acquisition of programming skills.**

***Index Terms*—program construction, learning analytics, programming patterns, programming skills**

I. Problem Area and Related Work

With the increased widespread application of information technology systems, many countries establish informatics basic education in compulsory schools. Still a central topic to computer science education is the training of novice programmers. Winslow [1] summarized key differences between novice and expert programmers. Since then, a hierarchy of programming skill levels of novice programmers has been evaluated by Lister [2] based on neo-Piagetian theory. The formal-operational level of programming experts is not covered. From the research field of software engineering, the performance-based measurement of programming skills is an active field, especially measuring expert programming skills (see Bergersen et al. [3]). But it lacks didactic support on how higher individual programming skills can be achieved.

In the field of *learning analytics*, data of student programming processes are used to improve the teaching of programming. Early work by Jadud [4] used measures of compiler errors to identify problem students. The stream of programming events was used by Ihantola et al. [5] to measure assignment difficulty. Patterns of clustered program states reached during programming were used by Blikstein et al. [6] to predict student achievement. On a more fine-grained scale, Rivers et al. [7] analyzed students' learning curves regarding used syntax elements in programming assignments.

In this thesis, I aim to contribute with a detailed analysis of program construction sequences from a syntax and semantics point of view in order to identify patterns of program construction. The expected contributions are: a more fine-grained assessment of programming skills with program construction patterns; assessment of programming skills during programming in an IDE-based *learning analytics* setting; individual support in the acquisition of higher programming skills applicable to both novice and professional programmers.

II. Research Question and Methodology

The main research goal of the thesis is to support the acquisition of programming skills by a detailed investigation of how new program code is constructed. An underlying assumption is that patterns found in program construction sequences represent expert programming skills (in part also conjectured by Soloway [8]) and can be used to educate programmers of different skill levels. The main research question is:

MRQ. To what extent do structural and semantic patterns of program construction, be they positive or not, influence acquiring expert programming skills?

Sequential research steps are planned. First, I will review methods and scales for measuring programming skills, and existing hierarchies for programming skills, compiling them in a reference catalog. Next, I plan to capture programming sequences in an IDE-based *learning analytics* setting with the aim to identify micro patterns during program creation. In the taxonomy of Hundhausen et al. [9], programming data (editing actions and file snapshots) and additional physiological data (eye gaze and head movements) will be captured to assess programming behavior and program content in relation to eye movements. Qualitative (grounded theory) as well as quantitative (clustering) methods are planned for data analysis.

The next step is to relate the micro patterns to measured programming skills, investigating whether the usage of patterns can be attributed to certain levels of programming skills. Lastly, I plan to prepare didactical applications of identified micro patterns to determine whether the programming skills of individuals can be improved by the usage of those micro patterns attributed to higher levels of programming skills.

Structural patterns encompass the sequence of syntax elements used during program construction, and can be analyzed quantitatively and qualitatively. Semantic patterns are then found by qualitatively analyzing the structural patterns for their semantics in the forming program code. As an example, the bottom sequence shown in Figure 1 exhibits a structural pattern of *construction of control structure before data structure*, which might be related to lower programming skills [1].

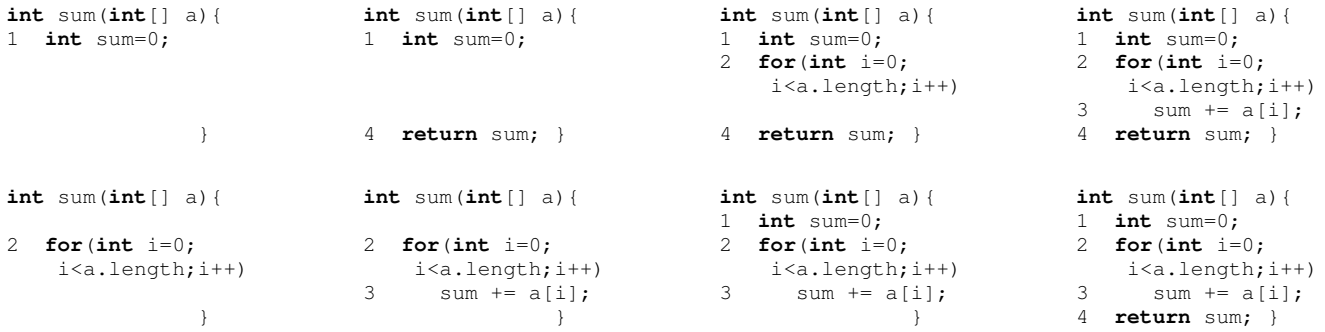

Fig. 1. Two sequences (top and bottom) of constructing a method in Java, computing the sum of an array of numbers. Numbers are used to show the additions in each step. On top, code for dataflow is constructed before the loop (sequence 1 – 4 – 2 – 3). On bottom, the loop is constructed before completing the function dataflow (sequence 2 – 3 – 1 – 4).

III. Planned Experiments

Two types of experiments are planned to resolve the main research question. The first type is the collection of programming data of test subjects solving exemplary programming problems. The collected data includes measured programming skill, interactions with instrumented IDEs, sequences of code changes, eye gaze and head movements. This type of experiment is repeated to assess predictive validity of the identified patterns. Considered subjects include school students, university students with technology-related focus and professional programmers. The populations vary in their programming experience, which makes it possible to assess potential variance in programming skills. The considered programming languages are *Scratch* and *Java*. Differences of pattern usage in block-based and text-based programming are evaluated by analyzing similar syntactical elements.

The second type is a comparative experiment. Teaching introductory *Java* programming at university level, a class with adapted teaching material that incorporates micro patterns is compared to traditional teaching. Positive changes in the experimental group regarding the used micropatterns suggest that the acquisition of programming skills can be improved.

IV. Preliminary Results

A preliminary trial with the *Scratch 2* programming environment was conducted, capturing the program construction sequences of upper secondary school students. Aggregated metrics based on block type usage were found to strongly correlate to the programming skills attributed to the students, and coarse patterns were identified with association rules mining. The results show that programming skills are measurable from program construction sequences, supporting the assumption that programming skill manifests in such sequences.

From literature review, I identified continuous measurement instruments [3] and hierarchical divisions [2] to measure programming skills. I plan to evaluate both approaches for relating programming skills to pattern usage.

The next planned research steps are: instrumenting IDEs for *Scratch 3* and *Java*, developing representative programming examples, incorporating eye-tracking based metrics.

V. Summary

The aim of my thesis is to identify patterns of program construction and use them to support the acquisition of programming skills. An underlying assumption is that patterns used by experts to construct new program code are part of expert programming skills. I will investigate patterns in block-based (*Scratch*) and text-based (*Java*) programming languages. I plan to complete the thesis in Spring 2021, following four research steps: 1) Creating a reference catalog of measurement procedures for programming skills, 2) Capturing program construction sequences and identifying patterns in them, 3) Investigating the relation of patterns to measured programming skills, 4) Didactical application of patterns to evaluate individual acquisition of expert programming skills.

Acknowledgment

I want to express my gratitude to my supervisor Andreas Bollin, who guides me in growing as a researcher.

References

[1] L. E. Winslow, "Programming Pedagogy – A Psychological Overview," *ACM SIGCSE Bulletin*, vol. 28, no. 3, pp. 17–22, 1996.

[2] R. Lister, "Toward a Developmental Epistemology of Computer Programming," *11th Proceedings - WiPSCE '16*, pp. 5–16, 2016.

[3] G. R. Bergersen, D. I. Sjøberg, and T. Dybå, "Construction and validation of an instrument for measuring programming skill," *IEEE TES*, vol. 40, no. 12, pp. 1163–1184, 2014.

[4] M. C. Jadud, "Methods and Tools for Exploring Novice Compilation Behaviour," in *Proceedings of the 2006 ACM Conference ICER '06*, 2006, pp. 73–84.

[5] P. Ihantola, J. Sorva, and A. Vihavainen, "Automatically detectable indicators of programming assignment difficulty," in *Proceedings of the 15th Annual Conference SIGITE '14*, 2014, pp. 33–38.

[6] P. Blikstein, M. Worsley, C. Piech, M. Sahami, S. Cooper, and D. Koller, "Programming Pluralism: Using Learning Analytics to Detect Patterns in the Learning of Computer Programming," *Journal of the Learning Sciences*, vol. 23, no. 4, pp. 561–599, 2014.

[7] K. Rivers, E. Harpstead, and K. Koedinger, "Learning Curve Analysis for Programming," in *Proceedings of the 2016 ACM Conference ICER '16*, 2016, pp. 143–151.

[8] E. Soloway, "Learning to program = learning to construct mechanisms and explanations," *CACM*, vol. 29, no. 9, pp. 850–858, 1986.

[9] C. D. Hundhausen, D. M. Olivares, and A. S. Carter, "IDE-Based Learning Analytics for Computing Education: A Process Model, Critical Review, and Research Agenda," *Critical Review, and Research Agenda. ACM Trans. Comput. Educ*, vol. 17, no. 26, pp. 1–26, 2017.