Journal of Education and Practice
ISSN 2222-1735 (Paper) ISSN 2222-288X (Online)
Vol.17, No.3, 2026

Construction of a Self-Regulated Learning Process Model for Programming Problem-Solving

Haonan Hu
College of Education, Zhejiang Normal University
Jinhua,321004,Zhejiang,China
E-mail: huhaonan20000923@163.com

Abstract

With the popularization of computer education, programming ability has gradually become an important learning literacy in the information age. However, in actual learning, students often encounter difficulties in task comprehension, insufficient strategic planning, and low debugging efficiency when solving programming problems. The theory of self-regulated learning provides an important theoretical perspective for understanding how learners actively plan, monitor, and reflect on their learning processes.Based on self-regulated learning theory, this paper systematically analyzes the dimensions and processes of self-regulated learning in programming problem-solving through literature analysis and theoretical integration. On this basis, combining the general problem-solving model and the domain characteristics of programming tasks, a self-regulated learning process model for programming problem-solving is constructed, which includes five stages: problem understanding, problem decomposition, coding and debugging, reflection and summarization, and consolidation and improvement.This model reveals the self-regulating mechanisms of learners in programming problem-solving from four dimensions: cognition, metacognition, behavior, and motivation, providing a theoretical basis for programming teaching design and the development of intelligent learning systems.

Keywords: Programming Education;Self-Regulated Learning;Problem Solving;Learning Process Model

DOI: 10.7176/JEP/17-3-02
Publication date: March 28th 2026

1. Introduction

With the rapid development of information technology, programming ability has gradually become an essential basic competency for individuals in the information age, and its importance has become increasingly prominent. Against this background, many countries and regions have incorporated programming education into the basic education curriculum system to promote the popularization and implementation of programming education.

However, programming learning often presents a high threshold for beginners. Relevant studies have shown that programming learning is not merely a process of mastering program syntax and logical structures, but also requires learners to possess comprehensive abilities such as complex problem analysis and abstract generalization. This leads to students being prone to understanding deviations, excessive cognitive load, and other problems in the process of programming learning, which affects learning outcomes (Qian & Lehman, 2018).

From the perspective of the nature of the learning process, programming problem-solving is a typical complex cognitive activity. Learners need to repeatedly cycle and continuously advance through multiple links, such as understanding task requirements, planning problem-solving strategies, writing programs, and debugging programs. This process not only relies on learners' solid grasp of programming knowledge, but also requires them to have good learning strategies and self-regulation abilities—self-regulated learning specifically refers to a dynamic process in which learners actively regulate various aspects such as cognition, metacognition, motivation, and behavior around established learning goals to plan, monitor, and evaluate their own learning activities. As a key core competency in the field of programming learning, self-regulation ability is directly related to the efficiency and quality of learners' programming problem-solving, which is specifically reflected in various links of learners' process of solving programming problems, such as goal setting, process monitoring, behavior control, and reflection and summary (Prasad & Sane, 2024). Especially in the online learning environment, self-regulated learning ability is a key factor for students to achieve efficient programming learning. Existing studies have confirmed that students with strong self-regulated learning ability often show better learning outcomes in programming learning (Loksa et al., 2022).

The theory of self-regulated learning provides an important theoretical perspective for analyzing the behavioral rules and cognitive mechanisms of learners in complex learning tasks. Zimmerman proposed that self-regulated learning is a core form of learners' active participation in learning, and defined it as a cyclic process consisting of three stages: forethought, performance, and self-reflection. In the context of programming learning, students need to continuously cycle and advance between understanding tasks, formulating strategies, writing code, and debugging programs, and this process is highly consistent with the cyclic structure of self-regulated learning.

Although there is a natural internal connection between self-regulated learning and programming problem-solving, existing studies mostly focus on the summary of programming learning strategies or the design of teaching intervention programs, and there is a lack of systematic and in-depth analysis of the internal structure and mechanism of action of self-regulation in the process of programming problem-solving, resulting in an obvious research gap. Based on this, this paper, taking the theory of self-regulated learning as the foundation, constructs a self-regulated learning process model for programming problem-solving by systematically analyzing the specific process of programming problem-solving and its self-regulation dimensions, aiming to provide a solid theoretical reference and support for programming education-related research and teaching practice.

2. Theoretical Basis

2.1 Self-Regulated Learning Theory

The theory of self-regulated learning is one of the important research directions in the field of learning sciences. This theory emphasizes that learners do not passively accept knowledge in the learning process, but continuously regulate their learning behaviors through active activities such as planning, monitoring, and reflecting. The self-regulated learning model proposed by Zimmerman is regarded as one of the most representative theoretical frameworks in this field. Zimmerman argued that self-regulated learning is a continuous cyclic process, in which learners optimize their learning strategies through continuous planning, implementation, and reflection, thereby improving learning outcomes. Based on this viewpoint, Zimmerman divided the learning process into three interrelated stages: the forethought stage, the performance stage, and the self-reflection stage (Zimmerman, 2000). This model emphasizes the cyclic nature of the learning process, that is, after completing a learning task, learners will adjust their subsequent learning according to the reflection results.

Table 1. Zimmerman’s Self-Regulated Learning Model

Stage	Main Activities
Forethought Stage	Goal setting, task analysis, strategy planning
Performance Stage	Task implementation, strategy application, self-monitoring
Self-reflection Stage	Outcome evaluation, causal analysis, strategy adjustment

In addition, scholars such as Pintrich further analyzed self-regulated learning from the perspective of learning regulation dimensions, arguing that learners mainly regulate their learning through the following four aspects in the learning process (Pintrich, 2000). In the cognitive dimension, learners process learning content through various cognitive strategies, such as understanding, organizing, and integrating information; in the metacognitive dimension, learners need to monitor and regulate their own learning process, such as planning learning steps or checking learning outcomes; in the behavioral dimension, learners support learning activities by regulating learning behaviors or the learning environment; in the motivational dimension, learners maintain learning motivation through goal setting, self-efficacy, and emotional regulation.

Table 2. Perspectives on Learning Regulation Dimensions

Self-Regulation Dimension	Meaning
Cognitive Regulation	Comprehension and information processing of learning content
Metacognitive Regulation	Monitoring and control of the learning process
Behavioral Regulation	Management of learning behaviors and learning environment
Motivational Regulation	Maintenance of learning goals and motivation

The above theories provide an important framework for analyzing learning behaviors in complex learning tasks. In the context of programming learning, students not only need to master programming design knowledge, but also continuously plan, monitor, and reflect in the process of solving problems. Therefore, the theory of self-regulated learning provides an important theoretical basis for understanding the programming learning process.

2.2 The Process of Programming Problem-Solving

Although the general problem-solving process has certain commonalities across different disciplines, programming problem-solving presents obvious domain-specific characteristics due to factors such as programming language expression, algorithm design, and computer execution mechanisms. Therefore, many studies have conducted specialized analyses on the problem-solving process in program development or programming learning.

Combining the characteristics of program development tasks, Deek integrated problem-solving models and proposed a six-stage model for programming learning (Deek & Turoff, 1999), including formulating the problem, planning the solution, designing the solution, translating the solution, testing the solution, and presenting the solution. Among them, the three stages of "planning—design—translation" correspond to the strategy planning, algorithm design, and code implementation processes in problem-solving, respectively, while "presenting the solution" emphasizes summarizing and expressing the solution after the problem is solved.

From the perspective of self-regulated learning, Loksa and Ko further proposed a six-stage cognitive-behavioral model of programming problem-solving (Loksa & Ko, 2016), including problem reinterpretation, analogy retrieval, solution search, solution evaluation, solution implementation, and post-implementation evaluation. This model refines the programming problem-solving process into a series of observable cognitive and behavioral activities, and points out that learners often need to continuously backtrack and iterate between these stages.

The programming problem-solving process has two prominent characteristics. First, programming tasks usually require detailed decomposition of complex problems and gradual implementation of solutions through modularization. Second, during program operation, computers can provide real-time feedback, which requires learners to continuously modify and optimize the code during program testing and debugging, thus forming a cyclic and iterative problem-solving process.

Therefore, when analyzing the programming learning process or designing programming teaching support strategies, special attention should be paid to learners' learning activities in task decomposition and debugging iteration. From the perspective of self-regulated learning, a systematic analysis of the programming problem-solving process helps to deeply understand the cognitive and behavioral characteristics of learners in programming learning, and provides a theoretical basis for the subsequent construction of learning support model*s*.

3. Self-Regulated Learning Dimensions in Programming Problem-Solving

Programming problems usually have a certain degree of complexity, and learners need to go through multiple stages such as problem understanding, algorithm design, program implementation, and debugging when completing programming tasks. In this process, learners need to continuously conduct cognitive processing and at the same time monitor and adjust their own learning process. Therefore, analyzing the programming problem-solving process from the perspective of self-regulated learning helps to more comprehensively understand the behavioral characteristics of learners in programming learning. Combined with the theory of self-regulated learning, the self-regulatory activities in programming problem-solving can be summarized into four dimensions: cognition, metacognition, behavior, and motivation. These four dimensions interact with each other and jointly affect learners' learning performance in programming problem-solving.

3.1 Cognitive Dimension

The cognitive dimension mainly involves the process of learners' understanding and information processing of programming tasks. In the process of programming problem-solving, learners need to analyze the problem requirements and convert the problem into a program logical structure. This process usually includes cognitive activities such as problem representation, algorithm design, and program structure planning.

Cognitive regulation refers to the process in which learners actively correct their understanding, make up for knowledge gaps, and integrate external feedback with existing knowledge according to the progress during task execution to optimize subsequent strategies. In programming problem-solving, this process is manifested in the targeted use of relevant programming language knowledge, algorithm and data structure concepts, and the realization of transfer and application in different problem scenarios. For example, in the task decomposition stage, students need to judge which data structure best meets the needs; in the coding stage, they need to select and combine appropriate grammatical structures to realize functions.

3.2 Metacognitive Dimension

Metacognition is the individual's self-evaluation, self-awareness, and self-regulation of their own cognitive processing process. In the context of programming problem-solving, metacognition mainly includes monitoring task requirements and one's own understanding state, planning and adjusting problem-solving strategies, and reflecting and seeking help when encountering difficulties.

Metacognitive regulation is the process in which learners actively set goals, make plans, and continuously monitor the effectiveness of strategies, evaluate progress and results during task execution. In programming, this is reflected in actively checking the problem meaning and problem-solving goals in the problem understanding stage; in the coding stage, regularly reviewing code logic and test results to judge whether they deviate from expectations; in the debugging stage, reflecting on the causes of errors and adjusting plans.

3.3 Behavioral Dimension

Behavior refers to the specific, observable actions taken by learners to achieve learning goals, including time management, resource utilization, code writing and debugging, etc. In programming problem-solving, behavior not only includes the planning and execution of tasks, but also every input, modification, and operation performed during coding and debugging.

Behavioral regulation is the process in which learners actively optimize and adjust specific actions according to the achievement of goals and external feedback during task execution. In the programming context, this is manifested in learners modifying the code structure, adjusting the debugging sequence, changing the variable design method, or trying alternative implementation schemes when the solution is blocked, based on external feedback such as system prompts, debugging outputs, compiler errors, or test case results. By continuously using external information to guide action adjustment, learners can continuously optimize strategies at the execution level and improve the efficiency and quality of problem-solving.

3.4 Motivational Dimension

Motivation is the internal force that drives learners to initiate, maintain, and complete learning activities, covering psychological factors such as learning interest, task value, and self-efficacy. In programming problem-solving, motivation directly affects the time, energy, and persistence invested by students.

Motivational regulation is the process in which learners maintain or enhance learning motivation by adjusting goal orientation, emotional state, and self-expectation. In programming tasks, this is manifested in maintaining a sense of accomplishment by setting phased small goals when encountering debugging difficulties, or actively seeking optimization schemes to pursue higher-level achievements after completing part of the tasks.

4. A Self-Regulated Learning Process Model in Programming Problem-Solving

In summary, programming problem-solving not only follows the phased logic of the general problem-solving process, such as "problem representation—plan formulation—plan execution—result evaluation", but also has obvious domain-specific characteristics. For example, it is highly dependent on the ability of problem decomposition and abstraction, as well as the frequent "implementation—debugging—iteration" cycle that occurs during the implementation process. Therefore, it is necessary to introduce the Self-Regulated Learning theory on the basis of the general problem-solving model to conduct contextual modeling of the programming learning process.

The three-stage model of self-regulated learning proposed by Zimmerman divides learning activities into the Forethought phase, Performance phase, and Self-reflection phase. It emphasizes that learners set goals and plan strategies before the start of the task, conduct self-control and monitoring during the task execution, and perform evaluation and regulation after the task is completed. This model provides an important theoretical framework for understanding learners' cognitive and regulatory behaviors in complex tasks.

Combining the three-stage structure of SRL and integrating the previous analysis of the programming problem-solving process, this paper contextualizes and refines the problem-solving activities in programming learning, and constructs a programming problem-solving process framework based on the self-regulated learning theory. This framework divides programming problem-solving into five stages: problem understanding, problem decomposition, coding and debugging, reflection and summary, and consolidation and improvement, which are respectively mapped to the three stages of SRL.

Among them, "problem understanding" and "problem decomposition" mainly correspond to goal setting and strategy planning in the Forethought phase of SRL; "coding and debugging" reflects self-control and self-monitoring in the Performance phase; "reflection and summary" and "consolidation and improvement" correspond to self-evaluation and adaptive regulation in the Self-reflection phase. In each stage, self-regulatory strategies related to the learning process are further identified to clarify the entry points for teaching support and learning intervention.

4.1 Problem Understanding

Programming tasks are usually presented in the form of problem descriptions, functional requirements, or example input and output. The problem understanding stage requires learners not only to accurately understand the semantic meaning of the task but also to identify key conditions, clarify the relationship between input and output, and judge whether the problem is suitable for solution through computer programs. This stage corresponds to the "problem reinterpretation" process in the Loksa and Ko model, and its core goal is to form a clear problem representation, laying a cognitive foundation for subsequent solution design. Since this stage occurs before the formal execution of the task and mainly involves goal understanding and task preparation, it belongs to the forethought phase of Self-Regulated Learning.

4.2 Problem Decomposition

Programming problems usually have high complexity and need to be handled in a modular manner. Problem decomposition is an important component of computational thinking. Learners need to split the overall task into multiple manageable sub-goals and initially consider the implementation method of each sub-goal, such as selecting appropriate data structures and control structures. This stage actually constitutes the preliminary planning of the solution and is an important transition from task understanding to algorithm conception. Therefore, problem decomposition is still an activity of strategy planning in the forethought phase of SRL.

4.3 Coding and Debugging

In the coding and debugging stage, learners need to convert the previously formed algorithmic ideas into specific program code and obtain feedback by running the program. Due to the strict grammatical and logical constraints of program execution, errors can usually be detected in the running process. Therefore, debugging activities become an important part of programming problem-solving. Learners need to locate and correct problems by reading error information, tracking the program execution process, and designing test cases. In this process, learners need to continuously pay attention to the task goals and monitor the effectiveness of their own operation strategies, which mainly reflects the characteristics of the performance phase in SRL.

4.4 Reflection and Summary

After completing the task, learners need to systematically review the entire problem-solving process and results, including judging whether the task goals are achieved, analyzing the causes of errors encountered, and evaluating the effectiveness of the adopted strategies. Through reflecting on the problem-solving process, learners can identify successful experiences and areas for improvement, and gradually form stable problem-solving strategies. This stage emphasizes metacognitive evaluation and attribution analysis, which is an important part of the self-reflection phase in SRL.

4.5 Consolidation and Improvement

The consolidation and improvement stage is a further adjustment and transfer on the basis of reflection. Learners deepen their understanding by conducting targeted exercises or variant tasks, and transfer the strategies and knowledge formed in the original task to new scenarios, so as to realize the transformation from solving a single task to developing stable abilities. For example, learners can deepen their understanding by solving structurally similar programming problems, optimizing the original program, or trying different algorithm implementations. This stage reflects the cycle of "reflection—regulation—re-practice" in SRL and is an important extension of self-regulated learning.

Table 3. A Self-Regulated Learning Process Model in Programming Problem-Solving

Self-Regulated Learning Phase	Programming Problem-Solving Stage	Self-Regulated Learning Strategies
Forethought Phase	Problem Understanding	Activate relevant prior knowledge and experience; analyze task semantics; identify input, output, and constraints; check the completeness of understanding through self-questioning; develop motivation toward the task.
	Problem Decomposition	Decompose the task into sub-goals; preliminarily select appropriate algorithms and data structures; evaluate whether sub-goals sufficiently cover the task requirements; strengthen confidence in completing the task.
Performance Phase	Coding and Debugging	Translate algorithmic logic into program code; execute the program to obtain feedback; identify and locate errors through debugging and testing; continuously monitor the execution process.
Self-Reflection Phase	Reflection and Summary	Evaluate the effectiveness of the problem-solving process; analyze causes of errors; summarize effective strategies; record possible improvements.
	Consolidation and Improvement	Apply learned strategies in variant tasks; deepen understanding through additional practice; transfer successful experiences to new learning contexts.

5. Discussion

The self-regulated learning model for programming problem-solving constructed in this paper has the following theoretical significance.

First, the model explains learning behaviors in programming problem-solving from the perspective of the learning process, providing a new analytical framework for understanding the difficulties in programming learning.

Second, the model provides a reference for programming teaching design. For example, targeted teaching support strategies can be designed at different learning stages, such as providing task analysis guidance in the problem understanding stage and error diagnosis support in the debugging stage.

In addition, the model also provides a theoretical basis for the design of intelligent learning systems. For example, in an intelligent programming learning system, different forms of learning support can be provided according to the stage where the learner is located.

6. Conclusion

Based on the theory of self-regulated learning, this paper systematically analyzes the process of programming problem-solving and constructs a self-regulated learning process model consisting of five stages. The research shows that programming problem-solving is not only a process of knowledge application, but also a self-regulatory process in which learners continuously plan, monitor, and reflect.

Future research can combine real teaching scenarios, further verify the application effect of the model in programming learning through teaching experiments, and explore the design method of intelligent learning support systems based on this model.

References

Qian, Y., & Lehman, J. (2018). Students' misconceptions and other difficulties in introductory programming: A literature review. ACM Transactions on Computing Education, 18(1), 1-24.

Prasad, P. & Sane, A. (2024). A self-regulated learning framework using generative AI and its application in CS educational intervention design. In Proceedings of the 55th ACM Technical Symposium on Computer Science Education, pp.1070–1076.

Loksa, D., Margulieux, L., Becker, B.A. & Ko, A.J. (2022). Metacognition and self-regulation in programming education: theories and exemplars of use. ACM Transactions on Computing Education, 22(4), pp.1–31.

Zimmerman, B.J. (2000). Attainment of self-regulation: a social cognitive perspective. In M. Boekaerts, P.R. Pintrich & M. Zeidner (eds.), Handbook of self-regulation. San Diego: Academic Press, pp.13–39.

Pintrich, P.R. (2000). The role of goal orientation in self-regulated learning. In M. Boekaerts, P.R. Pintrich & M. Zeidner (eds.), Handbook of self-regulation. San Diego: Academic Press, pp.451–502.

Deek, F.P. & Turoff, M. (1999). A common model for problem solving and program development. IEEE Transactions on Education, 42(4), pp.331–336.

Loksa, D. & Ko, A.J. (2016). The role of self-regulation in programming problem solving process and success. In Proceedings of the 2016 ACM Conference on International Computing Education Research, pp.83–91.