Identifying Domain-Specific Cognitive Strategies For Software Engineering

ABSTRACT

Due to the rapidly changing nature of today's work environment, software engineering (SE) students are required to have self-regulated learning (SRL) and problem solving skills. Previous research suggests that training students in the use of domain-specific cognitive strategies and using scaffolded instruction for strategy training improves students' SRL and problem solving task performance. In order to identify SE-specific cognitive strategies, we conducted a survey of advanced-level SE students. We then conducted a pre-test and post-test experiment to analyze the effectiveness of identified strategies in improving students' task performance. One control and two treatment groups were used during the experiment. The control group was not exposed to any strategies, while one treatment group was instructed verbally in the use of strategies and the other was trained using a newly developed scaffolded strategy training module. The results of the experiment demonstrate significant improvement in post-test task performance for both treatment groups, with a further increase in performance for those undertaking the scaffolded strategy training module.

CCS Concepts

•Social and professional topics → Software engineering education;

Keywords

Software Engineering; Learning and Education; Problem Solving; Self-Regulated Learning; Instructional Scaffolding

1. INTRODUCTION

Today's working environment is characterized by rapid change. Due to the evolving requirements within the IT market, it is crucial for software engineers to develop skills that allow them to acquire new knowledge, apply previously learned knowledge in unexpected situations, and to solve dynamic problems. In order to provide effective learning opportunities, educational institutions should not only facilitate the development of content expertise but also foster the development of problem solving and self-regulated learning (SRL) skills. This need is not only evidenced by reports from industry companies [16] but also by the fact that developing problem solving and SRL skills is a major educational objective in several countries [15].

Problem solving and SRL skills are widely applicable cognitive tools and involve the ability to acquire and use new knowledge, or to use pre-existing knowledge to solve novel problems [22]. The development of problem solving and SRL skills is important for students to progress from novice to experts within specific domains. The level of problem solving and SRL proficiency depends on domain-specific knowledge and strategies and how students adapt different strategies over time [13, 18]. Previous studies show that strategy training has had a positive impact on students' task performance [12]. Moreover, using instructional scaffolding further improves students' task performance and problem solving skills [7]. This underscores the fact that students should be trained to develop and use discipline-specific strategies to successfully solve problems and tasks related to a discipline- and specifically designed and targeted educational interventions can be offered to develop and improve strategy use and problem solving skills within a domain.

Despite extensive focus on the structure of software engineering degrees [11] and on experiences on designing and teaching software engineering courses [23], previous research lacks, to the best of our knowledge, the identification of learning strategies that are specific to software engineering (SE). There is a need for SE-specific strategies to be identified and articulated in order to assist students in developing effective problem solving skills.

This paper aims to identify SE-related strategies used by advanced software engineering students and to analyse how these strategies fit with various software development tasks. Howard-Rose & Winne [8] suggest that the nature of the tasks that students are asked to complete can influence the strategies learners use to achieve a solution. As different phases of the software development life cycle (SDLC) may require different cognitive and meta-cognitive skills, we align identified strategies according to the phases in software development process to identify task-based SE-specific strategies. The identified strategies can then be incorporated in the design of introductory and advanced level SE courses to support students' early and successful transition from novices to experts in SE domain. In addition, we will also examine if there is a significant difference in the performance

DOI:

of students who were taught to use the identified strategies and those who were not. For this purpose we will use verbal and scaffolded instruction methods to impart strategy knowledge. This will help us in analyzing the effects of these instructional methods on students' task performance.

The main contributions of this work are (a) identification of task based SE-specific strategies, which may in turn guide the design of SE curricula to assist students in developing problem solving and self-regulated learning skills, (b) development of a prototype tool for scaffolding the teaching of software engineering strategies related to requirement analysis, and (c) analysis of effectiveness of scaffolded strategy instruction in order to improve students' academic task performance.

2. RELATED WORK

Research has shown that training students on the use of cognitive and meta-cognitive skills and strategies results in significant gains in performance in a variety of content domains [4, 10, 12, 17].

Locke et al. studied the effect of strategy training on performance in a management course. As per the study results, strategy training had positive effect on students' task performance [12]. Çalışkan et al. investigated the effects of strategy instruction on students' physics problem solving performance. Students were provided domain specific strategy instruction and the results of the experiment revealed that the strategy instruction had positive effects on students' performance and strategy use [4]. Labouvie-Vief & Gonda compared task performance of the elderly when trained to use strategies on a training and a different but related transfer task. Results showed significant improvement in task performance of elderly after strategy training on both the tasks [10].

Further, studies have shown the effectiveness of scaffolding strategy instruction in improving problem solving skills. For example, Ge & Land demonstrate the use of scaffolding in improving problem solving for ill-structured problems [7]. Rittle-Johnson & Koedinger demonstrated that students' mathematical problem solving was improved when contextual, conceptual and procedural knowledge was scaffolded [19].

To be able to assist SE student in developing domain-specific cognitive skills, We first need to determine SE-specific strategies. Although some authors have discussed the cognitive processes involved in different aspects of software development such as in design [9] and requirements analysis [1], but no domain-specific cognitive strategies were cited. The closest study is by Falkner et al. (2014), who identified successful and unsuccessful learning strategies specific to the discipline of computer science and aligned these strategies with Zimmerman's learning model [6]. Another study by Falkner et al. (2015) compared reflections provided by students in software development courses, with the aim of analysing the evolution of learning strategies from novice first year students to expert final year students. Results showed that discipline-specific learning strategies evolve over course years, but that scaffolding is required in different areas to help students in strategy development [5]. Both these papers are focused on learning strategies for computer science domain as a whole but not specifically on software engineering.

3. RESEARCH QUESTIONS

Our research design is driven by following research questions (RQ):

- **RQ1** What are the SE-specific cognitive strategies according to the key tasks in software development process?
- **RQ2** Is there a difference in task performance of students who were trained to use SE-specific cognitive strategies and those who were not?
- **RQ3** Is there a difference in task performance when students were trained using scaffolded strategy instruction and when students were trained using verbal instruction?

4. RESEARCH METHODS

To answer our research questions, we divided our research study into two phases.

In **phase A**, we conducted a survey as the primary data collection instrument to identify SE-specific cognitive strategies. The research survey was aimed at students who had completed an advanced level SE related course at a university. Before enrolling in this course, students are required to complete an introductory SE course that provides them with the basic knowledge about software processes and practices. In the third year course, students are required to complete an SE project delivering fully implemented software and related artifacts. The survey questionnaire was based on open-ended questions, aimed at soliciting information about students' strategy use when performing various SE tasks during the project.

Participation in the survey was voluntary and anonymous and we performed both qualitative and quantitative analyses on the survey responses. To answer RQ1, the data collected from the survey was subjected to the inductive way of thematic analysis [3], where a coding framework and the design for analysis was developed based on the data itself. We first performed open coding and then proceeded to axial coding, subsequently aligning our coding framework in order to compare identified strategies with the SDLC phases.

During the open coding process, we read through the textual responses to the survey and selected a block of text representing an idea or a concept e.g. one student mentioned that they compared meeting notes to ensure that requirements are not changed. Similarly another student mentioned that keeping meeting records helped them in identifying requirements conflict. We mapped both these textual references to the same concept i.e. 'comparing meeting notes to identify requirements conflict'. Then in axial coding process we mapped this identified cognitive strategy to requirements analysis task in SDLC as making sure that there are no requirements conflict is part of requirements analysis phase.

In **phase B** of our research, we asked students who had either completed or were enrolled in a software engineering related course to voluntarily participate in an experiment activity. We used pre-test and post-test research design and performed quantitative analysis on collected data to answer RQ2 and RQ3. In both pre-test and post-test, students were required to perform the task of extracting system requirements from a given textual description of a real-world process. For pre-test we used real-world process of 'initiat-

ing requisition request to HR', and in post-test we used the process of 'daily listing for newspaper delivery'.

Students were randomly assigned to three separate groups that we labeled as NS (No Strategy), TS (Textual Strategies) and SS (Scaffolded Strategies). NS was not exposed to any cognitive strategies. TS was explained the concept of cognitive strategies; how to use cognitive strategies to extract requirements and was provided with only the textual list of strategies to use during the task. SS was exposed to the scaffolded cognitive strategies. We created a computer-based animated strategy training module, where students viewed an animated tutorial guiding them in the use of strategies to extract system requirements from text. Previous research suggests that animations can be useful when delivering complex conceptual, procedural, contextual and problem solving knowledge [14, 24]. The design of our animated training module is in line with the guidelines provided by Weiss et al. [24]. We used subject matter classification, visual cues and textual narrations in our strategy training module as suggested by Weiss et al. Both TS and SS were exposed to the same list of strategies, (1) list all the system objects (2) list corresponding functions and attributes for each system object (3) determine all the system interactions (4) list all the data input/output (5) map specific verb phrases in text with functions or events.

Both pre-test and post-test descriptions were created in a way that total 25 system requirements could be identified using the strategies. Same number of requirements for both tasks gave us uniformity in difficulty level and assessment process. Students' task performance for both pre-test and post-test was assessed on the basis of software requirements assessment criteria (SRAC) developed by the researchers. SRAC has 5 items based on the above strategies and each item can have maximum score of 5 based on the number of correct requirements identified using each strategy, hence the maximum score can be 25. SRAC items are, (1) requirements for each entity have been extracted (2) requirements for each function of each entity have been extracted (3) requirements for each system interaction have been extracted (4) requirements for each data input/output have been extracted (5) requirements for each verb have been extracted.

5. ANALYSIS AND DISCUSSION

PHASE A: A total of 29 students participated in the survey. Although a medium sample size, 29 participants is sufficient as Bertaux [2, p.35] mentions that 15 is the minimum acceptable sample size for qualitative research and the average sample size for qualitative research is 22 [20].

All the identified strategies with corresponding number and percentage of responses are provided in Table 1. The table also provides grouping of strategies according to the phases in SDLC. We realize that the list in Table 1 is not an exhaustive list of SE-specific cognitive strategies and can be further improved.

During our analysis we noticed that besides articulating strategies that were developed specifically for SE, students also mentioned several strategies that were adapted to fit in SE context, which is consistent with the research by Falkner et al [6]. For example concept mapping is a general information visualization strategy, but here it was mentioned in SE context as 'conceptually mapping interactions among system entities and class objects to clarify design'. On the other hand 'creating prototypes to confirm requirements' is a strategy that is specifically developed for SE. Ability to select relevant strategies and adapt them according to a specific context are useful skills to have. Design for SE courses should incorporate teaching students about general strategies and how these strategies can be adapted to specific discipline tasks. The strategies provided in the list can be used as an example.

Maximum number of students mentioned strategies belonging to project management phase, which shows that students mostly focused on the project management task. As project management involves all aspects of software development and spans all phases of SDLC, it can be a possible reason for students' focus on project management. An interesting observation is that though most students used project management strategies, highest number i.e. 12 strategies were identified for requirements analysis and most of these strategies are adapted. As for this course students were required to extract and elicit requirements from textual project description and client, it appears that students found it easier to adapt general strategies for requirements analysis task such as comprehension and organization strategies.

Least number of students mentioned design strategies. Also, lowest number of strategies were identified for design along with testing phase. The less focus on design and testing suggests that students might have struggled with these tasks and further analysis is required to determine the reasons why these two areas were least focused. This provides us an idea about the challenging areas for the students and instruction on these areas should be supported with carefully designed scaffolded learning activities in order to promote strategic action in these units. Moreover, over all there are several strategies that are mentioned by only 1 student. By incorporating these strategies in SE course instruction, more students can be made aware of them, which in turn can assist students in improving their meta-cognitive and cognitive skills.

The most popular strategy cited by students is 'using online forums to learn coding solutions', which was mentioned by more than half of the participants.

> Whenever I got stuck while coding I asked my peers or searched different forums. It is easy and it is fast. (Q3, Student 11)

It emphasizes the importance of virtual learning spaces and online sources as an important learning resource. On online forums, students can quickly find worked solutions for recurring coding and technical problems. This calls for the need for instructional mechanisms in SE courses, where quick and targeted guidance should be provided when students most need it.

Phase wise analysis reveals that there are many areas within each development phase, which were not mentioned at all. For example, in requirements analysis most strategies belong to requirements gathering or requirements discovery but areas like requirements management or how to specify requirements were not covered. Similarly, in design the focus was on visual representation or system modeling but strategies regarding how to come up with multiple design solutions or how to devise good design were not mentioned. In development, managing and debugging code; in testing, test planning and management; and in project management, risk management and resource planning were not covered. Some students mentioned facing difficulties related to some

Table 1: SE-Specific Cognitive Strategies Mapped To Software Development Tasks (Total Responses: 29)

Strategies	Responses	% Responses
Requirements Analysis	15	51.7
List System Entities, Classes, Data Inputs and Outputs, Business And System Events	11	37.9
List Corresponding Functions And Attributes For Each System Entity And Class Object	9	31.0
Determine User And System Interactions	8	27.5
List All Data, Variable And Control Dependencies	8	27.5
Create Prototypes To Clarify Requirements	7	24.1
Ask Questions What, When, Where, How, Who About Each System Entity, Class Object, User And Interaction	5	17.2
List Alternate Actions And States For Each System Role And Object	5	17.2
Map Specific Verb Phrases In Project Description with Functions or Events	4	13.7
List Do And Don't , Should And Shouldn't Scenarios	3	10.3
Identify Specific Words That Refer to Data or System Attributes In The Text Such As Speed, Volume, Space	2	6.8
Compare Meeting Notes To Identify Requirement Conflicts	2	6.8
Ask If-Then-Else Questions To Identify Alternate Scenarios For Each Entity, Class Object And User	1	3.4
Design	12	41.3
Compare Strengths and Weaknesses Of Design Alternatives	9	31.0
Conceptually Map Interactions Among System Entities, Class Objects To Clarify Design	8	27.5
Improve Design By Explaining And Prototyping	7	24.1
Determine Action Flows For Each System Role For Designing Interfaces	6	20.6
Create Map Of The Relationship Between Requirement, Scenarios, Contexts To Visualize Information For Design	6	20.6
Determine State Flows For Each Entity, Class Object To Visualize Information For Design	2	6.8
Graph Data And Control Dependencies To Clarify Design	1	3.4
Development	17	58.6
Use Online Forums To Learn Coding Solutions	15	51.7
Improve Knowledge Sharing By Using Collaborative Working Tools	7	24.1
Partition Development Tasks On The Basis Of Functions, Events	4	13.7
Create Tests Before Code	2	6.8
Use Standard Design Patterns	1	3.4
Use Comments To Explain and Understand Code	1	3.4
Develop Mock-Ups To Evaluate Solution	1	3.4
Dry Run Algorithm Before Coding To Check Correctness	1	3.4
Testing	13	44.8
Test Quality Attributes On Minimum, Maximum and In Between Measurements	10	34.4
Use Peer Reviews To Detect Bug Before Coding	8	27.5
Trace Tests With Requirements To Ensure Comprehensive Tests	8	27.5
Use Standards For Test Cases To Manage Test Quality	4	13.7
Perform Root-Cause Analysis To Find Improvement Areas	3	10.3
Describe Pre and Post Conditions For The Scenarios For Testing	1	3.4
Perform Exploratory Testing To Understand System	1	3.4
Project Management	20	68.9
Manage Tasks And Evaluate Performance By Using Scheduling Charts	14	48.2
Break Down Tasks In To Smaller Tasks To Manage Work	14	48.2
Create Task Dependencies For Scheduling	11	37.9
Group All Functionally Related Elements to Simplify Task	10	34.4
Compare Requirements To Assess Complexity	10	34.4
Categorize Tasks On The Basis Of Complexity	9	31.0
Conduct Periodic Meetings To Evaluate Task Accomplishment	8	27.5
List One-line Tests For Estimation	1	3.4
Create Naming Conventions To Manage Document and Code Versions	1	3.4
Use Sample Project Documents To Learn How To Develop Documents	1	3.4
Use Standard Templates For Documentation	1	3.4

of these areas such as one student mentioned that responsibilities among group members were not properly defined, and students worked on different tasks on 'as needed' basis.

> There was no role assignment and accordingly no responsibility division. So the plan, tracking and evaluation was ineffective. (Q5, Student 4)

Some students mentioned that they faced difficulty in managing requirements early on and thus encountered difficulties in managing the schedule.

> Trying to identify all the requirements up front was quite difficult, so we kept adding requirements as we thought of them all throughout the project and that affected our timeline. (Q1, Student 9)

To overcome the difficulties mentioned above i.e. effective requirements gathering; controlling timeline; and team management, adaptive scaffolding should be used in SE content and strategy instruction, where personalized guidance and feedback should be provided to the students. Moreover, SE courses should incorporate varied tasks and assignments so that students can get exposure to all areas of software engineering.

Although, several students mentioned using reviews to detect bugs before implementation, which is a good proactive strategy but one student quoted using peer reviews as a re-

Table 2: Descriptive Statistics For The Groups

Measurements	Groups	n	M	SD
Pre-Test	NS	10	11.50	1.780
	TS	10	11.50	2.173
	SS	10	12.20	1.814
Post-Test	NS	10	11.90	1.792
	TS	10	15.50	2.550
	SS	10	19.90	2.079

active testing strategy only.

> We mainly used peer reviews to find issues as things that we developed did not work as intended. (Q3, Student 16)

This emphasizes the importance of informing students about not only which strategies to use but also when, where and how to effectively use those strategies.

For students to successfully transition from being novice to expert, it is important for students to have the ability to reflect on their own performance and adapt strategies accordingly [21]. Among various SE strategies indicated by students, we did not find many strategies that suggest the use of reflection. SE students using cognitive strategies is positive but at the same time reduced focus on the use of reflection strategies is worrying. We suggest including post-project analysis activities in SE courses to help students in improving their reflective skills.

PHASE B: The NS, TS and SS students' responses to the pre-test and post-test were analyzed using SPSS v23. Descriptive statistics and mixed between-within subjects analysis of variance (two-way mixed ANOVA) were performed to compare pre-test and post-test scores on the SRAC (task performance) within each group and between NS, TS and SS. Descriptive statistics including mean and standard deviation for pre-test and post-test of each group are provided in Table 2. There were no outliers, as assessed by boxplot. The data was normally distributed, as assessed by Shapiro-Wilk's test of normality ($p > .05$). There was homogeneity of variances ($p > .05$) and covariances ($p > .05$), as assessed by Levene's test of homogeneity of variances and Box's M test, respectively.

When Table 2 is examined in the pre-test measurement of the SRAC, it can be seen that the mean of the NS (M=11.50) and TS (M=11.50) groups is equal but mean of the SS group is slightly higher (M=12.20). According to the results of the two-way mixed ANOVA, the difference between the NS, TS and SS pre-test means was not statistically significant ($p > 0.05$), $F(2, 27) = 0.438$, $p = 0.650$, partial $\eta^2 = 0.031$. These results indicate that the task performance of the students before they started in the experimental study was very similar in all three groups.

When Table 2 is studied in the post-test measurement of the SRAC, the mean of the SS (M=19.90) is highest and the mean of TS(M=15.50) is higher than the mean of NS (M=11.90). When two-way mixed ANOVA was conducted to test the treatment main effect on the post-test means of the TS and SS, it was determined that for TS (4.00 ± 1.12 mmol/L, $p < .05$) and SS (7.70 ± 1.00 mmol/L, $p < .05$) both, task performance was statistically significantly greater in post-test than pre-test. But there was no statistically significant difference in task performance of non-treated NS (4.00 ± 0.22 mmol/L, $p = 0.104$) within pre-test and post-test ($P > .05$). See Table 3.

Table 3: Group-Wise Comparison Within Pre-Test And Post-Test

Groups	Mean Diff.	Std. Error	P-Value
NS	-4.000	0.221	0.104
TS	-4.000	1.125	0.006
SS	-7.700	1.001	0.000

*The mean difference is significant at the 0.05 level.

Table 4: Pair-Wise Comparison Between Groups For Post-Test

Groups		Mean Diff.	Std. Error	P-Value
NS	TS	-3.60	0.967	0.003
	SS	-8.00	0.967	0.000
TS	NS	3.60	0.967	0.003
	SS	-4.40	0.967	0.000
SS	NS	8.00	0.967	0.000
	TS	4.40	0.967	0.000

*The mean difference is significant at the 0.05 level.

Furthermore, there was statistically significant difference in post-test task performance between groups, $F(2, 27) = 34.318$, $p = 0.000$, partial $\eta^2 = 0.718$. Post-test task performance was statistically significantly greater in SS (8.00 ± 0.97 mmol/L, $p = 0.000$) and TS (3.60 ± 0.97 mmol/L, $p = 0.003$) compared to NS. Task performance in TS was statistically significantly lower than SS (4.40 ± 97 mmol/L, $p = 0.000$). See Table 4. Therefore, the results can be seen to be in favour of the SS group.

In our experiment we trained students on relevant SE-specific strategies to extract requirements from a textual description. In light of the data analysis, it has been determined that use of the identified cognitive strategies has had positive effect on SE students' task performance. This is consistent with the results of previous research that examined the effects of strategy training on task performance [12]. Moreover, students' task performance was further improved when use of strategies was taught using scaffolded instruction as compared to the verbal explanation only, which is again in accordance with the previous research [7]. These findings have many implications for SE pedagogy. First and foremost, students should be taught domain-specific as well as general cognitive strategies. Second, to improve students' ability to transfer strategic action and cognitive skills to different tasks and problems, scaffolding should be used to teach how to use these strategies effectively i.e. declarative and procedural strategic knowledge should be followed by conditional knowledge, which is about when and where to apply these strategies.

6. CONCLUSIONS

Pressley & Hilden argued that students should develop strategies according to the contextual influences in a particular discipline [18]. In this paper we present cognitive strategies that are adapted or specific to SE context. Identified strategies can help guide the design of SE courses in order to provide awareness about strategies and enable their

effective use. When students are more aware of their cognitive strategic action, their meta-cognitive skills can also improve, which enable them to better monitor and reflect on their learning and performance and develop and adapt strategies accordingly. Hence, for assisting SE students in becoming effective problem-solvers and self-regulated learners, SE curricula should incorporate teaching of these strategies.

Our study found that students used general cognitive strategies such as mapping and questioning, and articulated them in SE context. It would be useful to identify more general strategies that can be specifically adapted to SE tasks. The identified strategies can help guide the adaptation and design of more strategies specific to SE domain. Mayer (1992) suggested that students' problem solving proficiency depends on the development of domain-specific strategies [13]. Our experiment shows that training students to use identified strategies improves students' SE task performance, which is in line with previous research on effect of strategy instruction on performance [12]. Another important aspect of strategic action is transfer. By training students on the use of strategies, students' transfer of cognitive skills can be improved, allowing them to apply strategies on other complex problems and tasks.

As suggested by Ge & Land, our experiment also shows that using scaffolding to teach strategy use is very effective. Therefore, instructional scaffolding for strategy training, especially adaptive scaffolding providing personalized instructional experience should be used in SE courses to assist students in improving their cognitive and meta-cognitive skills [7]. The identified list of SE-specific cognitive strategies is not complete and additional research is warranted to identify more strategies. We suggest creating a knowledge base of adapted and SE-specific strategies; and example applications of suitable strategies on various scenarios. SE students, instructors and researchers can refer to this knowledge base when learning or designing SE courses or conducting research experiments.

7. REFERENCES

[1] Agarwal, R., Sinha, A. P., and Tanniru, M. Cognitive fit in requirements modeling: A study of object and process methodologies. *Journal of Management Information Systems 13*, 2 (1996), 137–162.

[2] Bertaux, D. From the life-history approach to the transformation of sociological practice. *Biography and society: The life history approach in the social sciences* (1981), 29–45.

[3] Braun, V., and Clarke, V. Using thematic analysis in psychology. *Qualitative research in psychology 3*, 2 (2006), 77–101.

[4] Caliskan, S., Selcuk, G. S., and Erol, M. Effects of the problem solving strategies instruction on the students' physics problem solving performances and strategy usage. *Procedia-Social and Behavioral Sciences 2*, 2 (2010), 2239–2243.

[5] Falkner, K., Szabo, C., Vivian, R., and Falkner, N. Evolution of software development strategies. In *37th International Conference on Software Engineering (ICSE)* (2015), vol. 2, IEEE, pp. 243–252.

[6] Falkner, K., Vivian, R., and Falkner, N. J. Identifying computer science self-regulated learning strategies. In *Innovation & technology in computer science education (ITiCSE)* (2014), ACM, pp. 291–296.

[7] Ge, X., and Land, S. M. Scaffolding students' problem-solving processes in an ill-structured task using question prompts and peer interactions. *Educational Technology Research and Development 51*, 1 (2003), 21–38.

[8] Howard-Rose, D., and Winne, P. H. Measuring component and sets of cognitive processes in self-regulated learning. *Journal of Educational Psychology 85*, 4 (1993), 591–604.

[9] Jeffries, R., Turner, A. A., Polson, P. G., and Atwood, M. E. The processes involved in designing software. *Cognitive skills and their acquisition 255* (1981), 283.

[10] Labouvie-Vief, G., and Gonda, J. N. Cognitive strategy training and intellectual performance in the elderly. *Journal of Gerontology 31*, 3 (1976), 327–332.

[11] Laird, L. Strengthening the "engineering" in software engineering education: A software engineering bachelor of engineering program for the 21st century. In *29th International Conference on Software Engineering Education and Training (CSEET)* (2016), IEEE, pp. 128–131.

[12] Locke, E. A., Frederick, E., Lee, C., and Bobko, P. Effect of self-efficacy, goals, and task strategies on task performance. *Journal of applied psychology 69*, 2 (1984), 241.

[13] Mayer, R. E. *Thinking, problem solving, cognition*, 2nd ed. New York : W.H. Freeman, 1992.

[14] Mayer, R. E., and Moreno, R. Animation as an aid to multimedia learning. *Educational psychology review 14*, 1 (2002), 87–99.

[15] OECD. Pisa 2012 assessment and analytical framework. mathematics, reading, science, problem solving and financial literacy.

[16] Patrick, C., Peach, D., Pocknee, C., Webb, F., Fletcher, M., and Pretto, G. The wil report: A national scoping study. *Final Report to the Australian Council for Teaching and Learning, ACEN* (2009).

[17] Pressley, M. *Cognitive strategy instruction that really improves children's academic performance.* Brookline Books, 1990.

[18] Pressley, M., and Hilden, K. Cognitive strategies. *Handbook of child psychology*.

[19] Rittle-Johnson, B., and Koedinger, K. R. Designing knowledge scaffolds to support mathematical problem solving. *Cognition and Instruction 23*, 3 (2005), 313–349.

[20] Sandelowski, M. Sample size in qualitative research. *Research in nursing & health 18*, 2 (1995), 179–183.

[21] Schön, D. A. *The reflective practitioner: How professionals think in action*, vol. 5126. Basic books, 1983.

[22] Sternberg, R. J. *Thinking and Problem Solving*, vol. 2. Academic Press, 2013.

[23] Thomas, C., and Berkling, K. Redesign of a gamified software engineering course. In *International Conference on Interactive Collaborative Learning (ICL)* (2013), IEEE, pp. 778–786.

[24] Weiss, R. E., Knowlton, D. S., and Morrison, G. R. Principles for using animation in computer-based instruction: Theoretical heuristics for effective design. *Computers in Human Behavior 18*, 4 (2002), 465–477.