Information Technology and Management (2020) 21:15–28
https://doi.org/10.1007/s10799-019-00299-9

Knowledge representation for computational thinking using knowledge discovery computing

Youngseok Lee[1] · **Jungwon Cho**[2]

Published online: 13 May 2019
© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract
Modern society needs to think of new approaches for solving problems with computing. Computational thinking is the process of abstracting and automating a variety of problems using computational technology. A system that expresses, manages, and processes knowledge such as computational thinking is called a knowledge-based system. This paper proposes to examine students' knowledge about computational thinking when they want to develop a Python project, and the correlation/association between these concepts. To achieve our goal, a field study was designed and data were collected from a computer programming lecture. Through this data analysis, we try to identify the factors through the correlation between data and clustering technique in order to express and discover the knowledge about the learner's computational thinking. For the verification of the factors identified, we analyzed the correlation between computational thinking and the pre- and post-test results of the LightBot. In addition to the regression analysis of the proven factors, the probability of the research model was analyzed through the structural equation to process the knowledge discovered. In this paper, we present various problems in the domain of programming education and analyze the means to diagnose and improve knowledge based on computational thinking by finding various problem-solving methods. To pre-examine the learner; he/she was diagnosed using a test paper and the LightBot execution test. We checked the learner's current knowledge state by analyzing the correlation between the test site and the results of the LightBot. To analyze the level of knowledge improvement of learners, we designed an experiment to analyze the correlation between learning and the actual test results through a system that applied the problem-solving learning method. An analysis of the experimental results demonstrated that there was a correlation between the test results for a learner and the pre-test results of the LightBot. Additionally, the group mean scores of the learners who learned as per the proposed technique were observed to be significant. During this process, we analyzed the effects of problem-solving and system application on academic achievement through factor analysis, regression analysis, and structural equation modeling. The ability to pinpoint various problem scenarios and solve problems more effectively using computational technologies will become more important in future. For this purpose, applying our proposed technique for deriving and improving knowledge based on computational thinking to software education will induce the interest of students and increase the learning effect.

Keywords Knowledge representation · Knowledge discovery · Problem solving · Computational thinking · Knowledge-based tutoring system

1 Introduction

A knowledge-based system is an artificial intelligence system that is used to solve complex problem, and it allows knowledge to be represented [1]. Knowledge-based systems are mostly used in problem-solving procedures, and they solve problems according to expert knowledge. In knowledge-based systems, it is important to represent and manage the knowledge of experts. You can learn how to represent your own knowledge through software education [2].

✉ Jungwon Cho
jwcho@jejunu.ac.kr

Youngseok Lee
yslee38@kangnam.ac.kr

[1] KNU College of Liberal Arts and Sciences, Kangnam University, 40 Gangnam-ro, Giheung-gu, Yongin-si, Gyeonggi-do 16979, South Korea

[2] Department of Computer Education, Jeju National University, 102 Jejudaehak-ro, Jeju-si, Jeju-do 63243, South Korea

In the era of the current Fourth Industrial Revolution, software is expected to become the center of value creation and innovation. In accordance with these changes, software education is being improved in many countries [3]. Recently, many universities are providing software education to all students so that they can design and create software using the knowledge of their major subject. This direction of software education improves logical and creative thinking through the process of ideation for problem-solving based on computational thinking, resulting in people coming up with their own solutions [4].

Computational thinking refers to the ability of solving problems through automated techniques by recognizing and abstracting problems; this recognition and abstraction of problems are based on the basic concepts and principles of computational technology. Through software training, computer specialists can help students shape their ideas using computational technology in their major subject field and thus enable them to communicate with experts in various fields. Computational thinking is computer-based thinking; therefore, it can be developed and improved through software education based on educational programming languages [5].

Although there are various types of educational programming languages for software education, it is desirable to choose languages that are easy to learn and extend to various types of application programs and additionally are of interest to non-computer-science students [6]. Python is relatively easy to learn programming, and its graphics processing capability is simple; therefore, it is appropriate to be learned initially by beginners [7]. Python is also useful to develop various apps and web forms; therefore, it is highly likely to be used as a programming language for convergence education. After analyzing the current situation both at home and abroad, Python can be judged to be a suitable educational programming language for computer specialists [8].

However, in many universities, software education for non-entrepreneurs is proceeding in a manner similar to computer-oriented programming language-oriented education and without considering specific educational goals and methods for the non-entrepreneurs [7]. Therefore, non-computer-science students face difficulties in learning to improve their computational thinking in the software curriculum. To develop problem-solving skills, students are required to identify, present, and investigate real-world problems [8]. Additionally, the problems should be structured so that learning takes place in a series of processes that implement problem-solving methods using an educational programming language as a tool [9].

In this paper, we present an example of problem-solving-based software education that can enhance the thinking ability of students while inducing interest in students to facilitate software education. We analyze how to diagnose and improve knowledge based on computational thinking through the process of presenting various problem scenarios and finding various problem-solving methods for learning. This study aims at analyzing the effects of these types of education on diagnosing and improving knowledge and analyzing the effects of this type of education on satisfaction of learners.

2 Related work

2.1 Knowledge discovery

Software education involves learning a programming language that allows one to communicate to the computer and learn to order and check the results of the computer as one thinks. Currently, we can distinguish professional programming education to train software developers and universal programming education to improve upon computational thinking and problem-solving ability. In this study, we focus on universal programming education to train talented people with computational thinking and problem-solving ability for the preparation of the future society of the Fourth Industrial Revolution era [9].

Programming finds ways to solve problems and resolve them in the course of solving a given problem; it gradually completes a task by iterating and combining these processes. From this perspective, various skills such as information processing ability, procedural thinking ability, problem-solving ability, logical thinking ability, and reasoning ability are improved through programming [10]. To resolve this problem, it is necessary to start with an easy problem that one can solve and to go on further so that one can solve various difficult and complicated situations [11].

To improve such computational thinking, it should be possible to utilize various knowledge on a computer basis. Such knowledge can be roughly classified into three types [12]. In the first type, there is domain knowledge comprising learning content objects, relationship information, explanation, examples, and problems between them. In the second type, there is teaching knowledge as a necessary strategy for the learning and teaching processes. Regarding the third type, it is the knowledge of learners accumulated from the knowledge of the instructor. If such knowledge is structured and stored in a repository, a system can be designed such that knowledge can be selected and problems can be correctly recognized and solved. Additionally, the knowledge stored in the knowledge repository should be configured such that the results of various processes can be compared [13].

In the system for utilizing such knowledge, an expert module, a teacher module, and learner module should be present. The expert module organizes the information that

contains the knowledge that the learner wants to learn from the instructor and manages the repository that stores it. The instructor module functions as a teacher; an example would be different learning scenarios such as guided free play, learning by doing, discovery learning, and mixed-initiative dialog, and a real person who selects and orders teaching styles and teaching contents. The learner module comprises the learner's personal information, knowledge about the current learner's area, and ability information about the learner's course. When such a knowledge-based system is constructed, the learner's current competency is evaluated and the gap between the current level and the target competency is analyzed.

After selecting the knowledge to be learned through this analysis and confirming the learning history and preference of the learner, the learning strategy is established. Further, the appropriate learning material is selected, and the educational information to be used for the learner is then transferred to the expert and learner modules. At this time, the expert module should have all the learning information about the learning topic. The ability of the learner model to provide all data on the learning topics needed to analyze and evaluate the learner's behavior is required [14].

In order that the learner module corrects the learning performance of the learner and to detect the error in the learning activity according to the progress of learning, the expert module estimates the cause of error and suggests alternative knowledge about the problem. The instructor module analyzes the proposed alternative knowledge and the learning strategy and preference of the learner. It further establishes a plan such as the order of learning and type of learning contents suited to the correct learning method. This information is transferred to the expert learner modules and further learning is conducted accordingly. When the current learning process is completed, the next learning process is selected through the learning performance diagnosis for current learning. Further, the learning strategy is established, and the procedure is presented to the learner.

2.2 Knowledge representation and reasoning

Knowledge representation and reasoning are techniques for analyzing or understanding complex problems and provide/propose appropriate solutions. In knowledge discovery, it is necessary to explain the knowledge in a form similar to the process of human thought and to formulate the expressed knowledge in a manner such that inferences can be made in accordance with various conditions.

Knowledge representation is related to automated reasoning. One of the primary purposes of explicitly expressing knowledge is the inference and assertion of new knowledge [15]. Advertisers often want to serve online advertisements to new consumers who are similar to their existing customers. The customer base contains similar groups of customers who share common tastes, interests or preferences (unsupervised segmentation), and we provide recommendations of similar products. Thus, whenever you see statements like, "People who like *X* also like *Y*" or "Customers with your browsing history have also looked at …," similarity is being applied [15].

Clustering analysis techniques can obtain a meaningful data structure without dictionary information for knowledge representation and reasoning [16]. Clustering analysis techniques are applicable to almost all types of data and are easy to apply because they do not need role definitions for variables [16]. Clustering is the task of grouping together similar objects into several groups. For instance, n observations comprising p variables are divided into several groups according to their similarity (proximity). A group of similar objects is called a cluster. Clustering aims to understand the structure of the entire data by understanding the nature of each group, from which a set of rules is derived [16].

To develop knowledge discovery and reasoning techniques similar to human problem-solving procedures, we used non-hierarchical clustering methods [17]. K-means clustering is a method of assigning each object to the closest center point [16]. First, we select K initial center points. Second, each object is assigned to the cluster with the closest center point, and the center point of the new cluster is then calculated. Third, the above steps are repeated until there is no change in the allocation of each object. Finally, the K clusters are formed [16].

However, it is difficult to measure the non-similarity distance satisfying certain condition and determine the weight value that is generally used. In the case of K-average cluster analysis, the results are poor if the setting of K is not appropriate [16]. One disadvantage is that the interpretation of the results is ambiguous because there is no purpose given in advance.

2.3 Knowledge base for computational thinking

Computational thinking is the process of thinking that includes defining and describing a problem so that a computer (person or machine) can perform it in an effective manner. An open-ended problem requires a comprehensive and meaningful solution based on several variables. To achieve this, we need problem decomposition, knowledge representation, modeling, and algorithms found through computational thinking [18, 19].

Computational thinking involves decomposing the decision process, taking into account the associated variables and all possible solutions, and choosing the correct decision based on the corresponding method and problem limits [20]. The general solution is to resolve the problem, identify the variables involved in representing the data, and generate the

algorithm [21]. Common answers are generalizations and abstractions that can be used to solve the variants of the problems that are derived from the original problem [20, 21].

Computational thinking involves the following features in the problem-solving process [22]:

- Analyze and logically organize data
- Data modeling, data abstraction, and simulation
- Organize user specific problem for computer help
- Identify, verify, and implement possible solutions
- Automate solutions through algorithmic thinking
- Generalize and apply this course to other problems.

Computational thinking is practical enough to be applied to real-world problems such as "when keys are lost", or "when determining the priority of pushed tasks". It is also applied in business, finance, engineering, and art [23, 24]. If a problem is solved first such as the problem-solving method of the computer; or if the problem is identified and structured, and an algorithm is introduced to solve it in a stepwise manner, it can be said that the problem is solved by computing accident [24]. For example, in the case when you are looking for a lost key: "If the key is not in the room, look inside the car. If it is not in the car, look in the coat pocket. If you cannot find it elsewhere, you can create a new key." The solution has a structure similar to the "If, else if, else" construct in computer programming languages [23, 24].

Computational thinking is attracting attention as a way of new creative thinking. Its role is to help people solve problems across all fields easily and using a new approach. Computational thinking facilitates in the collection and analysis of data and parallelizing problems to solve multiple problems simultaneously [25]. In the modern society where computers dominate our lives entirely, The ability to identify various problem situations and solve problems using computing technology will become increasingly important [26]. This is because computers are used in various fields such as in design programs that use software to operate businesses, talking with software subcontractors, and even expressing art with computer technology. In this situation, computerization of problems in various fields can be applied not only to more efficient and accurate solutions but also to other problems with similar types [27].

Problem-solving learning and problem-based learning are similar [28, 29]. It is a learning method in which learning is performed in a series of processes that present problems to the learners, search for solutions to the problems presented, and develop solutions through individual or collaborative learning [30]. Problem-solving learning or problem-based learning is a learner-centered learning environment and model in which learners are taught to solve the presented practical problems [31].

Students learn thinking strategies and domain knowledge together in this learning. This form of problem-solving learning originated in medical education and is now used in other fields as well [32]. The goal of problem-solved learning is to develop flexible knowledge and to acquire effective problem-solving skills, and to improve self-directed learning. Problem-solving is a form of active learning [33].

Through group collaboration, students find out their current level of knowledge, what information they need to solve the problem, and how to approach it. The instructor should develop students' understanding, while at the same time, instilling confidence in students to challenge the faculty and encouraging them for the same. Problem-solving learning represents a paradigm shift that attempts to escape the traditional teaching–learning philosophy centered on lectures. The components that guide problem-solving learning are very different from traditional teaching methods [34].

Problem-solving learning can improve the ability to solve problems while understanding them, solving problem-solving plans, practicing, revising, and re-practicing practical methods. In this study, we propose the following features of the problem that are required for the software education based on the problem-solving learning based instructional programming language education stage in various previous studies [34].

- Problems that describe or solve real-world situations
- Problems with more than one solution
- Problems that can transfer knowledge
- Motivating and challenging issues
- Problems with expansion and disassembly

Generally, the programming process is similar to the process of problem-solving. Therefore, this study analyzes the research on problem-solving learning in various fields and applies it to software education.

3 Experimental design

3.1 The process of knowledge discovery

The knowledge discovery process is interactive, iterative, and involves the following critical steps [35]. To discover knowledge, the domain of the knowledge to be found is searched, and the data is transformed after preprocessing based on the data sample [35]. Based on this, data mining techniques are applied to analyze patterns, leading to the discovery of knowledge through verification [35]. Figure 1 shows a common knowledge discovery process [35].

In this paper, the knowledge representation process for the proposed knowledge discovery process is shown in Fig. 2. Based on the general knowledge discovery process,

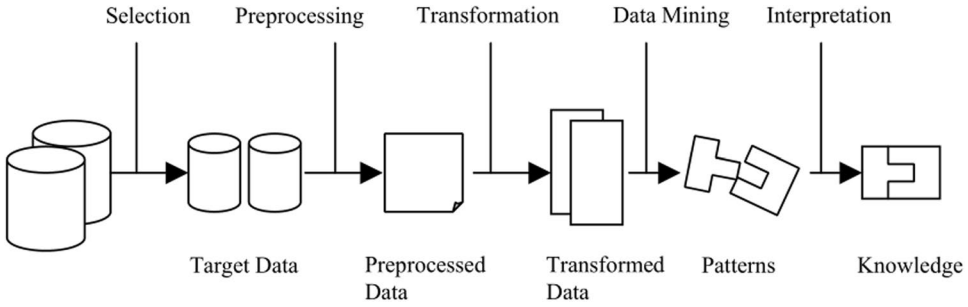

Fig. 1 Knowledge discovery process

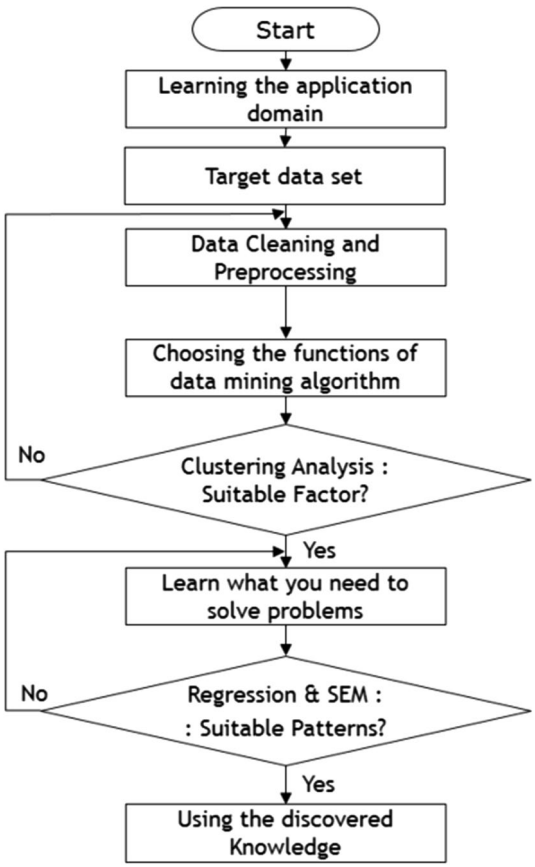

Fig. 2 The proposed knowledge discovery process

we propose a knowledge-based tutoring system and determine a knowledge discovery technique to utilize it.

We collected data from learner's data and questionnaires using the proposed system. The data collected result from the measurement of the computational thinking abilities of 151 students who attended computer programming in the first semester of 2018, which was held at the beginning of the semester [34]. An analysis of the clustering experimental results show that computational thinking appears as an element of pattern, automation, abstraction, and algorithm.

The computational thinking score of these four domains is distributed from 0 to 12 points, K-means clustering is conducted to analyze the discovery and relevance of knowledge on computational thinking. A K-means clustering was conducted to analyze the discovery and relevance of knowledge in computational thinking. The clustering analysis results shows that computational thinking has four elements: pattern, automation, abstraction, and algorithm, and a computational thinking score ranging from 0 to 12 was assigned to each of these four domains. The results are shown in Tables 1 and 2.

Table 1 Initial cluster centers of *K*-means clustering analysis

	Cluster			
	1	2	3	4
Initial cluster centers				
Computational thinking	10	3	9	6

Through this analysis, we try to identify the factors through correlation between the data and clustering techniques to express and discover the knowledge about the learner's computational thinking. For the verification of the factors identified, we analyzed the correlation between computational thinking and the pre- and post-test results of the LightBot. In addition to the regression analysis of the proven factors, the probability of the research model was analyzed through the structural equation to process the knowledge discovered.

3.2 Proposed knowledge-based tutoring system

Figure 3 shows the structure of a system that extracts knowledge suitable for the learner to provide knowledge-based tutoring, provides customized learning, and evaluates it based on the learning results. To achieve this, the knowledge structure database for customized and problem-solving learning is included as a main feature, additionally including the structure of the learning system.

The interface module provides the learning contents and evaluation paper provided by the system in a form that can be used by the learners. It interprets the response results of the learners and returns them to the system [36].

Table 2 ANOVA of *K*-means clustering analysis

	Cluster		Error		F	Sig.
	Mean square	*df*	Mean square	*df*		
ANOVA						
Computational thinking	148.856	3	.310	147	479.557	.000

The ANOVA table indicates which variables contribute the most to your cluster solution
Variables with large F values provide the greatest separation between clusters

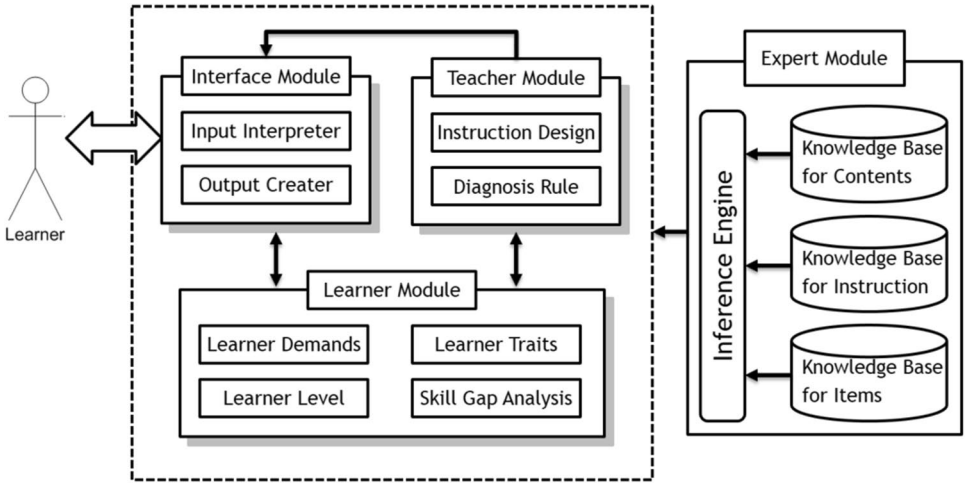

Fig. 3 Proposed knowledge-based tutoring system

The learner module extracts learner characteristics and level from the competency-based learner model and suggests appropriate learning contents and evaluation items for the individual. The diagnostic module of the learner module processes the learner results received from the interface module and generates the learner model based on the instructor and learner knowledge. To re-analyze the learner information, the evaluation is conducted, and the appropriate items necessary for the learner's level of education are observed and presented [36].

The teacher module provides a choice of personalized learning to enhance the learner's vulnerable characteristics. This is based on the learner's characteristics and level extracted from the learner module. Evaluation items are extracted considering academic level and characteristics [36].

The expert module manages the knowledge bases of contents, questions, and teaching methods. The inference engine determines the method by which the items can be extracted based on the knowledge base of contents. It is possible to determine the degree of difficulty of the items generated by each rule through the knowledge base of the items [36].

3.3 Application example of problem solving method of proposed system

Figure 4 shows the result of summarizing the learning procedures to perform problem-solving learning. After

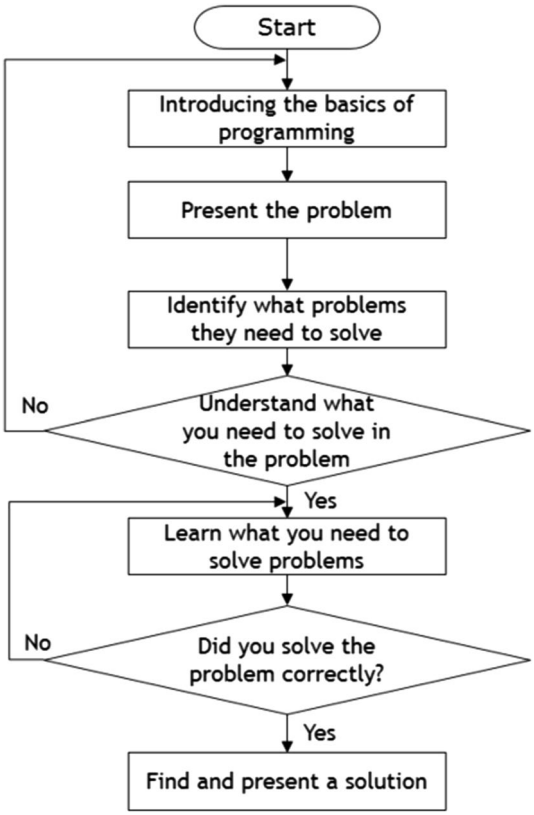

Fig. 4 Procedure for problem-solving learning

introducing the basics of programming, the problem and its background are presented to the students [37]. Students identify the problems they need to solve, what exactly they need to solve in the problems and determine if they can be solved. If the student does not understand the problem, he/she can go back to understanding the basic element; if he/she can identify and judge the problem, the learning contents and additional items to solve are learned [38].

If a problem can be solved, its solution is summarized and presented. However, if it cannot be solved, then it is extracted and studied again. Each problem is arranged considering class objectives, problem contents, and class time. At the beginning of a class, easy and simple problems are presented; students are able to solve complex problems gradually. Thus, the problem-solving learning process is repeated [37]. The basic lecture plan is similar to the programming education for non-computer specialists proposed in the current study; however, the contents of education are modified so that problem-solving techniques can be applied [38].

3.4 Experimental result

During problem-solving learning, students developed solutions for various general and coding problems through Python programming. In this process, students used the proposed system and received points for each completed task. We set this score as the system score. In addition, students attempted to solve various problems on LightBot, which is an easy coding-based Flash game that teaches the concepts of programming. The LightBot execution result is a record of students' attempts to solve such problems. LightBot will be described in more detail in Sect. 4.

To analyze the knowledge based on the computational thinking ability for the problem-solving learning thus configured, the results using the system, the results of the post-test of the LightBot performed at the end of the semester, and the results of the analysis of the correlation with the final academic performance are shown in Table 3.

The correlation between the performance on the system and actual academic performance showed a very strong positive correlation with a value of .733 ($p < 0.01$). Therefore, higher the system score, higher the actual academic performance. The results of the light bot performance test conducted at the end of the semester showed that the correlation between the system performance was .444 ($p < 0.01$) and the correlation with the academic performance was .389 ($p < 0.01$) Therefore, the results of LightBot, which can be evaluated in the process of execution, can be said to have an academic correlation. Therefore, the use of the proposed system for problem-solving learning based on computational thinking is highly related to the actual academic performance, and it can be concluded that the students' computational thinking ability is improved.

Table 3 Correlation between system scores and post-test of the LightBot and actual academic performance

	System score	Academic performance	Light Bot execution result
System score			
Pearson correlation	1		
Sig. (2-tailed)			
N	151		
Academic performance			
Pearson correlation	.733**	1	
Sig. (2-tailed)	.000		
N	151	151	
LightBot execution result			
Pearson correlation	.444**	.389**	1
Sig. (2-tailed)	.000	.000	
N	143	143	143

**Correlation is significant at the 0.01 level (2-tailed)

Table 4 Independent sample test conducted for the total learner

Elements	N	Avg.	SD	t	*p*
Light Bot execution result					
Pre	150	16.09	3.311	−5.511	.000
Post	143	18.03	2.667		

Consequently, the results of the lecture and the results of the lecture through the LightBot are highly related to each other. The results of the independent sample test are shown in Table 4.

The t test result should be chosen such that the same variance is not assumed, and a one-way test is conducted to confirm the learning performance. The results of the proposed system are as follows. The average score is 16.09 (standard deviation 3.311) and 18.23 (standard deviation 2.667). Thus, it can be seen that effective learning is achieved and there is a significant difference (significance probability $p < 0.001$) with 95% reliability. In other words, there was a difference between the results of the pre-diagnosis and the post-diagnosis, and it was confirmed that the learning outcomes were better when using the computational thinking-based learning recommendation system for the suggested problem-solving learning.

4 Case study

4.1 Discovering learner's knowledge

To measure the computational thinking power of students after applying software education for the validity of the

proposed software education model, we used the Korean intellectual educational information service and the computational thinking ability test [34]. The results of this study are as follows. It is recommended that learners use this pretest evaluation to diagnose the levels of early learners. However, since this test is a pen and pencil test that can be used to measure computational thinking, it may be difficult to measure the automation element of computational thinking through actual abstraction; therefore, we use a tool called LightBot.

LightBot is a programming puzzle game that uses a game mechanism based on programming concepts [39]. It allows the game learner to get a real understanding of basic control flow concepts such as procedures, loops, and conditional statements [40]. The learner uses mouse-click commands to move the robot on the tiles, thereby solving each level and understanding the concepts necessary for computational thinking [39, 40]. LightBot can be used by anyone older than 8 years of age and is used as a learner diagnostic tool because it is suitable for diagnosing the initial learning ability of computer specialists [40].

- Light Bot Step 1: Basic Step

Basic game rules are explained and executed in the following manner. When the user clicks on each step, the LightBot move one step forward to the square cell and move to the destination. Arrows are a way to succeed when a light bot is activated and reaches its final point by arranging a one-shot, left-turn, right-turn, jump, and an activity to turn on a light appropriately.

- Light Bot Step 2: Procedure Step

After executing the basic step, a procedure step with a slightly complex task is presented. Procedures can be understood in terms of functions. Separating a repetitive task as a procedure allows simplification of the code by calling the procedure whenever required. This enables to isolate repeating patterns throughout the process.

For example, suppose there exists a pattern: 'forward > forward > forward > light on > right turn' that repeats twice. Here, it is possible to solve the problem by recording the repeated operation in the procedure area called PROC1 and calling the procedure name twice from the MAIN area.

- Light Bot Step 3: Loops

At this stage, the user can learn repetition. If P1 is placed in the PROC1 procedure area to call itself back, one can perform infinite iterations through the recursive function. This mission is performed in 3D space through the LightBot game; therefore, the user can naturally learn functions that can teach the elements of abstraction and automation while cultivating a sense of space. LightBot can increase mission difficulty with commands such as move, turn left, turn right, jump, fire on, function 1, and function 2. The difficulty of the mission is increased by raising thinking power and not by learning new concepts. This is an appropriate tool to improve computational thinking skills, even for those with computer skills.

At the beginning of the semester, we unraveled the computational thinking point to diagnose computational thinking ability and made it possible to solve the LightBot with a brief description of 60 min. The computational thinking ability presented in Tables 1 and 2 consists of four detailed elements (pattern, automation, abstraction, and algorithm), and its score is defined by the combined score of the four elements. We sought to understand the students' knowledge status by analyzing the correlation between the four elements, the students' computational thinking (CT), and the LightBot pre-tests conducted at the beginning of the semester. Table 5 shows the result of analyzing the correlation between the results of the computational thinking ability test and LightBot.

As a result of the correlation analysis, the pattern, automation, abstraction, and algorithm were found to have a weak but significant correlation ($p < 0.05$). Particularly, the results of the diagnosis of the computer and LightBot showed a positive correlation of .383 ($p < 0.01$) and .325 ($p < 0.01$), respectively. Therefore, higher the results of the computational thinking ability, higher the diagnostic results of the LightBot, and vice versa.

4.2 Design of research model

In this study, we analyze the process of diagnosis and improvement of knowledge of computational thinking for problem-solving learning. The purpose of this paper was to investigate the effect of problem-solving process on academic performance. For this purpose, the following four variables were selected to measure the diagnosis and improvement of the knowledge of computational thinking for problem-solving learning [37, 38].

- Problem-solving learning
- Lecture composition
- Lecture participation
- Academic performance.

Among these four variables, the data were extracted from the actual homework of students, midterm exam, and final exam. Questionnaires on problem-solving, lecture composition, and lecture participation were also distributed to the students. The hypotheses to be revealed in this study are as follows.

Table 5 Correlation between the results of the computational thinking ability test and LightBot

	Pattern	Automation	Abstraction	Algorithm	CT	LightBot
Pattern						
Pearson correlation	1					
Sig. (2-tailed)						
N	151					
Automation						
Pearson correlation	.281**	1				
Sig. (2-tailed)	.000					
N	151	151				
Abstraction						
Pearson correlation	.273**	.248**	1			
Sig. (2-tailed)	.001	.002				
N	151	151	151			
Algorithm						
Pearson correlation	.193*	.182*	.319**	1		
Sig. (2-tailed)	.018	.025	.000			
N	151	151	151	151		
Computational thinking						
Pearson correlation	.583**	.698**	.709**	.639**	1	
Sig. (2-tailed)	.000	.000	.000	.000		
N	151	151	151	151	151	
LightBot						
Pearson correlation	.253**	.325**	.247**	.179*	.383**	1
Sig. (2-tailed)	.002	.000	.002	.028	.000	
N	150	150	150	150	150	150

**Correlation is significant at the 0.01 level (2-tailed)

*Correlation is significant at the 0.05 level (2-tailed)

Hypothesis 1 Problem-solving learning will affect students' academic performance.

Hypothesis 2 Problem-solving lecture composition using learning contents recommendation system will affect students' academic performance.

Hypothesis 3 Interactions among students, faculty, and the system will affect students' academic performance.

The study model is based on the relationship between variables and prior studies and it is shown by Fig. 5.

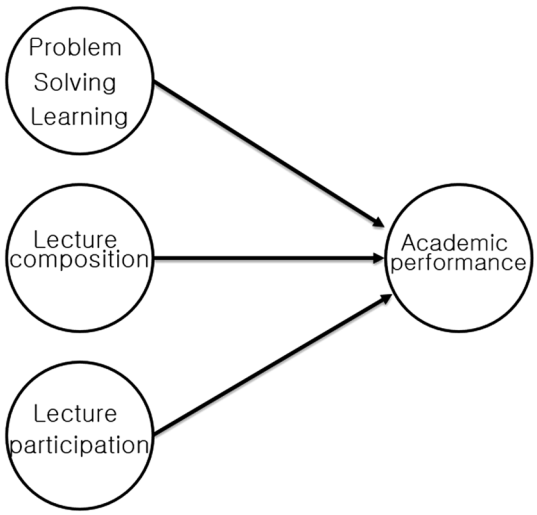

Fig. 5 Research model

4.3 Sampling and data collection

We conducted a satisfaction survey to prove a study applying a computational thinking-based learning system for the proposed problem-solving learning. We conducted this survey with the use of a system in the computer programming course having lectures from March to June 2018. The data analysis process of this study is as follows. First, the reliability of the measurement tool was tested using Cronbach's coefficient.

Second, descriptive statistics and cross-correlation matrices between the variables, which are the basic data of the structural equation modeling evaluation, were calculated. In the structural equation modeling analysis, if the normal

distribution assumption is severely violated, it may lead to distorted results in the maximum likelihood method. The kurtosis and the degree of kurtosis were investigated to verify that each variable follows a normal distribution, and the versatile normal distribution was verified [39].

Third, the fit and the model of the model were estimated. Fourth, the direct and indirect effects of variables were confirmed. Fifth, the path coefficient and significance of the final model were tested and the magnitude of the total effect was analyzed.

Amos 18.0 was used for the analysis of this study and statistical significance was verified at significance level of .05. Table 6 shows the definitions of the variables and the actual variables used in this study.

The results of this study are as follows. First, the results of the study on the attitude and interaction were analyzed and the appropriate questionnaire items were extracted. The total reliability of the extracted questionnaire items was very high (.914). The results of reliability analysis are shown in Table 7.

A total of 20 questionnaires were constructed. Reliability of all the questionnaires exceeded .9, indicating a very high reliability. The results of the factor analysis for each item are shown in Tables 8 and 9.

4.4 Measurement and result

To analyze the structural equation model, there is a two-step approach is used. In this approach, the measurement and structural models are analyzed as a single model, and the validity is verified in the one-step approach and measurement model stage. In this study, we analyzed the structural

Table 6 The definitions of variables and related studies

Variable	Operational definition	Related study
Problem-solving learning	The degree of functional friendliness of problem-solving learning	Paik [28], GAO [41], Jaechoon [42]
Attitude toward proposed system	The degree to which the participant has interest in lecture and proposed system	Youngseok and Jungwon [36], Ku [33], Jaechoon [42]
Interaction	The degree of process and method that are related to each other in communication between teacher and learner, user and system	LIU [43], Junghwan [44],

Table 7 Reliability analysis results

Item	Mean	SD	N	Scale mean if item deleted	Scale variance if item deleted	Corrected item-total correlation	Squared multiple correlation	Cronbach's alpha if item deleted
No. 1	3.93	.849	151	74.15	116.659	.721	.661	.906
No. 2	3.49	1.404	151	74.58	111.525	.577	.536	.911
No. 3	4.07	.892	151	74.01	116.393	.697	.604	.906
No. 4	4.20	.766	151	73.87	118.177	.711	.680	.907
No. 5	3.79	.803	151	74.28	120.895	.513	.465	.911
No. 6	4.12	.757	151	73.95	119.365	.645	.551	.908
No. 7	3.56	1.004	151	74.51	115.092	.673	.664	.907
No. 8	3.64	.989	151	74.43	113.927	.744	.769	.905
No. 9	3.70	.958	151	74.37	114.342	.749	.764	.905
No. 10	3.51	.979	151	74.56	114.514	.722	.741	.905
No. 11	3.84	.895	151	74.23	116.259	.701	.614	.906
No. 12	3.55	.862	151	74.52	116.064	.743	.649	.905
No. 13	4.40	.750	151	73.67	117.783	.753	.727	.906
No. 14	4.24	.846	151	73.83	118.006	.646	.605	.908
No. 15	3.97	.752	151	74.11	118.855	.682	.618	.907
No. 16	3.41	1.041	151	74.66	124.132	.232	.223	.918
No. 17	3.81	.700	151	74.26	137.929	− .476	.375	.927
No. 18	4.60	1.200	151	73.47	116.411	.492	.383	.912
No. 19	3.70	1.063	151	74.37	115.662	.604	.474	.909
No. 20	4.54	.746	151	73.54	126.077	.237	.238	.916

Table 8 KMO and Bartlett's test

Kaiser–Meyer–Olkin measure of sampling adequacy	.922
Bartlett's test of sphericity	
Approx. Chi square	1827.303
df	190
Sig.	.000

relations of the research model using the two-step approach of Anderson and Gerbing [45], which analyzes the structural equation model.

The conformity metrics of the measurement model were also based on the model conformance. The minimum and maximum values of root mean square error of approximation (RMSEA), standardized root mean square residual (SRMR), Tucker–Lewis index (TLI) and comparative fit index (CFI) were analyzed [45]. In this study, the number of cases is less than 250, and the number of measurement variables is between 12 and 30. Therefore, it can be estimated that SRMR satisfies the criteria of model conformity by satisfying the condition of having a value of .080 or less, CFI of .950 or greater, TLI of .950 or greater, and RMSEA of less than .080.

The model consistency of confirmatory factor analysis was estimated using the maximum likelihood estimation method. The results showed that SRMR = .021, TLI = .955, CFI = .958, and RMSEA = .077 (LO: .064, HI: .090). Since the number of cases is less than 250 and the number of measurement variables is between than 12 and 30, the SRMR satisfies the condition of .021 or below .08. CFI and TLI with values of .958 and .955, respectively, satisfy the condition of being .92 or higher. Additionally, RMSEA satisfies the criteria of model conformity by satisfying the condition of being greater than .065 but below .080. The standardized and non-standardized coefficients of the set measurement models are shown in Tables 10, 11, 12 and 13.

Table 11 shows the estimation results of the model.

The covariance and the correlation coefficient between the potential variables seen in the set measurement model Fig. 3 are analyzed in Tables 12 and 13.

Table 9 Factor analysis result

Questions of questionnaire		Factor			
		1 (problem-solving learning)	2 (lecture composition)	3 (lecture participation)	4 (classroom environment)
No. 10	Have you improved your ability to self-program through problem-solving-based learning?	.850			
No. 8	Did problem-based learning help me improve my computational or logical thinking skills?	.835			
No. 9	Could problem-solving-based learning be unique to your programming experience or method?	.807			
No. 7	Problem-solving Did you become interested in learning and motivated to learn while you were learning?	.771			
No. 1	To what extent do you think you have gained new and useful knowledge through this course?	.608			
No. 11	Did you actively participate in classes that use problem-based learning?	.591			
No. 19	How much do you think academic achievement has been achieved?	.524			
No. 12	What if I score a test?	.517			
No. 17	The progress of lectures, personal assignments, programming exercises, etc..	− .458			
No. 13	What do you score on the teaching method?		.780		
No. 15	How do you rate your credit assessment method?		.764		
No. 14	What do you score on communication with professors?		.757		
No. 4	How do you rate your content?		.723		
No. 6	How about a variety of case-based problem-solving lectures?		.584		
No. 18	Recommended to take this course to colleagues or juniors		.573		
No. 5	What if I score a task?		.496		
No. 2	Compared to expectation of the semester….:			.675	
No. 20	How long have you been attending the lecture?			.651	
No. 3	What do you score the lessons you learned?			.620	
No. 16	What if you grade your classroom environment?				.780

Table 10 Confirmatory factor analysis model compliance index (n = 151)

Model	NPAR	χ^2	*df*	SRMR	TLI	CFI	RMSEA		
							AVE	LO90	HI90
CFA model	67	309.326	164	.021	.955	.958	.077	.064	.090

Table 11 Regression weights

			Estimate	SE	C.R.	*p*
Academic performance	←	Problem-solving learning	.115	.068	2.689	.041
Academic performance	←	Lecture composition	.076	par_22		
Academic performance	←	Lecture participation	.199	.109	1.822	.048

Table 12 Covariance result

			Estimate	SE	C.R.	*p*
Problem-solving learning	↔	Lecture composition	.492	par_15		
Lecture participation	↔	Problem-solving learning	.457	.075	6.126	***
Lecture participation	↔	Lecture composition	.484	par_17		

***Significance at $p < 0.01$ respectively

Table 13 Correlations result

			Estimate
Problem-solving learning	↔	Lecture composition	.723
Lecture participation	↔	Problem-solving learning	.738
Lecture participation	↔	Lecture composition	.779

Covariance determines whether there is any correlation between the two variables and degree of correlation. Correlation coefficient enables to know whether the discriminant validity is satisfied between the two factors. The correlation coefficient is between .40 and .60 for the linguistic expression according to the correlation coefficient, and the correlation is high when the correlation coefficient is between .60 and .80. All the loads between the variables were above .457. It was found that the potential factors that were originally measured were appropriately measured by satisfying convergent validity.

4.5 Discussion

The purpose of this study was to analyze the process of diagnosis and improvement of knowledge based on computational thinking for problem-solving learning; and to analyze the structural relation between problem-solving, lecture composition, and lecture participation. For this purpose, the reliability and validity of the research model and measurement tools were verified and the structural relationship of the research model was analyzed. Statistical tests revealed that the causality between these potential variables was significant and the validity of the study model was verified.

We analyzed students' previous knowledge state using the computational thinking ability test and the LightBot game. The correlation between computational thinking and the sub elements pattern (0.583), automation (0.698), abstraction (0.709), and algorithm (0.639) were significantly high. The analysis also showed a high correlation between computational thinking and the LightBot game (0.383), which is a useful tool to learn and understand computer programming concepts and computational thinking.

In addition, the analysis of the students' group average scores according to the proposed technique showed that the scores significantly affect the LightBot test and the actual academic performance. This study investigated how to represent and discover students' problem-solving ability as knowledge, as well as how problem-solving learning, which enhances this knowledge, affects students' academic performance. For each knowledge representation technique, K-means clustering and various statistical analyzes were conducted on the results of the application of the proposed system.

In this study, how problem-solving learning affects students' academic performance was studied. The results showed that problem-solving learning directly affects students' academic performance. The effect of the potential variable was .135, which showed a statistically significant effect at significance level .001. The results of this study show that the ability of students to solve problems by improving their own computational skills and logical thinking ability by applying various problem-solving methods has been improved to improve their grades.

Additionally, it can be said that the structure of the lecture applying the learning content recommendation system

based on the problem-solving learning directly affects the academic achievement of the students. Likewise, the interaction between students, faculty, and the system can affect students' academic performance. However, it is difficult to say if the physical environment of the classroom affects the actual academic performance. However, it is necessary to consider the facility environment of the classroom for the psychological incentives and effects of the students.

5 Limitations and conclusions

Computational thinking is emphasized as the most important ability for everyone, regardless of their major subject, to prepare for the current era of the Fourth Industrial Revolution. In the future, it is predicted that people with talent who have a superior ability of computational thinking ability and problem-solving ability based on humanistic literacy are needed.

Software education as a universal education for this purpose should be structured so that it can be interesting to solve the problem using computational technology, but many universities still do not exceed the limits of programming language education.

In this study, we have proposed software education as a problem-centered form based on the Python language, which is widely used as an educational programming language in college. We solved the problem by using a solution according to the presented problem scenario. Consequently, it was found that students' knowledge acquisition and academic achievement improved. Additionally, the correlation between the problem-solving process and academic performance of the actual students was analyzed to obtain relevant results.

In future, we plan to research on ways to improve computational thinking power and problem-solving ability based on this study, and on the effectiveness of software education by applying the software education subject to real-world problems in non-major areas. In the future, I will study research on ways to improve computational thinking power and problem solving ability based on this, and study on effectiveness of software education by applying software education subject to real life problems by non-major areas.

References

1. Rajagopalan C, Baldev R, Kalyanasundaram P (1996) The role of artificial intelligence in non-destructive testing and evaluation. Insight 38(2):118–123
2. Techopedia (2018) Knowledge-based system (KBS). https://www.techopedia.com/definition/7969/knowledge-based-system-kbs. Accessed 30 Aug 2018
3. García-Peñalvo FJ, Cruz-Benito J (2016) Computational thinking in pre-university education. In: Proceedings of the fourth international conference on technological ecosystems for enhancing multiculturality. ACM, pp 13–17
4. KERIS (2016) Research report KR 2016-4. http://lib.keris.or.kr/search/detail/CATLAB000000012086. Accessed 25 Sept 2017
5. Ater-Kranov A et al (2010) Developing a community definition and teaching modules for computational thinking: accomplishments and challenges. In: Proceedings of the 2010 ACM conference on Information technology education. ACM, pp 143–148
6. Jin SH, Shin S (2013) Case study and needs analysis on convergence education in engineering colleges. J Eng Educ Res 16:29–37
7. Python Software Foundation (2017) Python about. https://www.python.org/about/. Accessed 5 July 2018
8. Guo P (2014) Python is now the most popular introductory teaching language at top U.S. universities, BLOG@CACM. http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-us-universities/fulltext. Accessed 12 June 2018
9. Lee HW, Min HR, Yi KW (2008) A study on the improvable proposal of general education curriculum of engineering college—a case of Seoul National University. J Eng Educ Res 11(3):24–32
10. Kim SH (2015) Analysis of non-computer majors' difficulties in computational thinking education. J Korean Assoc Comput Educ 18(3):15–23
11. Duncan C, Bell T (2015) A pilot computer science and programming course for primary school students. In: Proceedings of the workshop in primary and secondary computing education. ACM, pp 39–48
12. Sung Y (2017) Development of SW education model based on HVC learning strategy for improving computational thinking. J Korean Assoc Inf Educ 21(5):583–593
13. Park SH (2016) Study of SW education in university to enhance computational thinking. J Digit Converg 14(4):1–10
14. Jeong I (2017) Study on the preliminary teachers' perception for the development of curriculum of the robot-based software education in the universities of education. J Korean Assoc Inf Educ 21(3):277–284
15. Jeon Y, Kim T (2015) The design and application of an experience-driven online software class based on creative problem solving for cultivating the creative personality of the elementary informatics-gifted students. J Korea Elem Educ 26(4):477–494
16. WIKIPEDIA (2018) Knowledge representation and reasoning. https://en.wikipedia.org/wiki/Knowledge_representation_and_reasoning. Accessed 28 July 2018
17. Tan PN, Steinbach M, Kumar V (2013) Data mining cluster analysis: basic concepts and algorithms. Introduction to data mining, pp 487–533
18. Liu D-R, Ke C-K (2007) Knowledge support for problem-solving in a production process: a hybrid of knowledge discovery and case-based reasoning. Expert Syst Appl 33(1):147–161
19. Jung HY (2014) An empirical study on information liberal education in university based on IT fluency and computational thinking concept. J Korea Soc Comput Inf 19(2):263–274
20. Chen G, Shen J, Barth-Cohen L, Jiang S, Huang X, Eltouhky M (2017) Assessing elementary students' computational thinking in everyday reasoning and robotics programming. Comput Educ 109:162–175
21. Kim K, Kim H (2014) A case study on necessity of computer programming for interdisciplinary education. J Digit Converg 12(11):339–348
22. Kim CW (2010) The application of computer education in the revise curriculum. J Educ Res Inst 12(1):41–55
23. Grover S, Cooper S, Pea R (2014) Assessing computational learning in K-12. In: Proceedings of the 2014 conference on innovation & technology in computer science education. ACM, pp 57–62
24. Jun S (2017) Design and effect of development-oriented model for developing computing thinking in sw education. J Korean Assoc Inf Educ 21(6):619–627

25. Atmatzidou S, Demetriadis S (2016) Advancing students' computational thinking skills through educational robotics: a study on age and gender relevant differences. Robot Auton Syst 75:661–670
26. Basawapatna A et al (2011) Recognizing computational thinking patterns. In: Proceedings of the 42nd ACM technical symposium on computer science education. ACM, pp 245–250
27. Seo J (2017) A case study on programming learning of non-SW majors for SW convergence education. J Digit Converg 15(7):123–132
28. Paik S (2017) The effects of educational programming language with PBL (problem based learning) on logical thinking ability and problem solving ability in elementary school environments, Master thesis. Korea National University of Education, Chung-Buk
29. Myung HJ (2014) Effects of software special classon programming and creative problem solving capability, Master thesis. Hanyang University, Seoul
30. Kim B, Jeon Y, Kim J, Kim T (2016) Development and application of real life problem solving lesson contents based on computational thinking for informatics integrated-gifted elementary school students' creativity. Korean J Teach Educ 32(1):159–186
31. Ku J, Jeon Y, Kim T (2016) The development and application of lesson contents based on the CT-CPS framework for improving the creative problem solving ability of elementary informatics gifted students. J Korea Elem Educ 27(2):339–357
32. Ki JY (2018) A study on UX design process lecture based on modified PBL (problem-based learning). J Korea Converg Soc 9(1):117–131
33. Ku JH (2017) Designing an app inventor curriculum for computational thinking based non-majors software education. J Converg Inf Technol 7(1):61–66
34. Lee Y, Cho J (2017) The influence of Python programming education for raising computational thinking. Int J u- and e- Serv Sci Technol 10(8):59–72
35. Dhiman AK (2011) Knowledge discovery in databases and libraries. DESIDOC J Libr Inf Technol 31(6):446–451
36. Lee Y, Cho J (2015) Personalized item generation method for adaptive testing systems. Multimed Tools Appl 74(19):8571–8591
37. Lee Y (2018) Python-based software education model for non-computer majors. J Korea Converg Soc 9(3):73–78
38. Lee Y (2018) Analyzing the effect of software education applying problem-solving learning. J Digit Converg 16(3):95–100
39. Yaroslavski D (2018) LightBot. http://LightBot.com/. Accessed 20 June 2018
40. Gouws L, Bradshaw K, Wentworth P (2013) First year student performance in a test for computational thinking. In: Proceedings of the South African Institute for Computer Scientists and Information Technologists conference. ACM, pp 271–277
41. Gao S et al (2016) Factors affecting the performance of knowledge collaboration in virtual team based on capital appreciation. Inf Technol Manag 17(2):119–131
42. Jo J et al (2016) A study on factor analysis to support knowledge based decisions for a smart class. Inf Technol Manag 17(1):43–56
43. Liu H et al (2014) A model for consumer knowledge contribution behavior: the roles of host firm management practices, technology effectiveness, and social capital. Inf Technol Manag 15(4):255–270
44. Junghwan L, Munkee C, Hwansoo L (2015) Factors affecting smart learning adoption in workplaces: comparing large enterprises and SMEs. Inf Technol Manag 16(4):291–302
45. Anderson JS, Prussia GE (1997) The self-leadership questionnaire: preliminary assessment of construct validity. J Leadersh Stud 4(2):119–143

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.