Computational Thinking 능력이 소프트웨어 개발에 미치는 영향에 관한 연구

박성빈[†] · 안성진[††]

요 약

최근 정부와 산업계에서는 국가경쟁력의 핵심요소로 강조되고 있는 소프트웨어의 경쟁력을 강화해야 한다는 요구가 증가하고 있으며 이를 위해서는 소프트웨어를 개발하는 소프트웨어 개발자들의 직무능력 향상이 필요하다고 할 수 있다. 본 연구결과 창의적 문제해결방법인 Computational Thinking이 소프트웨어 개발자들의 직무능력 향상에 긍정적인 영향을 준다는 점과 기존연구결과와는 다르게 소프트웨어 개발이라는 업무적 특성으로 인해 simulation, algorithms & procedures, parallelization 등이 소프트웨어 개발자들에게 영향력 있는 요소로 나타났으며 직무능력 향상을 위해서는 이에 대한 교육이 강조될 필요가 있다. 이중에서 특히 simulation이 가장 큰 영향을 주기에 이에 대한 강화가 필요하다고 할 수 있다.

주제어 : Computational Thinking, 소프트웨어 개발자, 직무능력향상, 시뮬레이션

A Study on the Effectiveness of Computational Thinking Ability on Software Development

Seongbean Park[†] · Seongjin Ahn[††]

ABSTRACT

There has been rising demand within the South Korean government and among various industries in recent years on the need for strengthening the competitiveness of the software industry, which they highlight as being a core element for national competitiveness. This strategy would require enhancing the job competency of software developers. This study's results show that computational thinking (CT), which is one of the more creative solutions to the problem, has a positive effect on the enhancement of software developers' job competency. Furthermore, the study finds that the software development tasks of simulation, algorithms and procedures, and parallelization (in that order) serve as influential elements for software developers, which differs from previous studies' findings. The South Korean government thus should emphasize education in these areas in order to enhance the nation's job competence. Because simulation, especially, has the greatest influence among these areas, there is a particular need to strengthen that area.

Keywords : Computational Thinking, Software developer, Job Competency, Simulation

† 정 회 원: 성균관대학교 교과교육학과 컴퓨터교육전공 박사수료
†† 종신회원: 성균관대학교 교과교육학과 교수(교신저자)
논문접수: 2015년 7월 22일, 심사완료: 2015년 8월 16일, 게재확정: 2015년 9월 18일

1. Introduction

In the current digital generation, software (SW) is variously used in people’ s daily lives as well as by different industries. SW is used in PCs and smart phones, and is utilized in automobiles, aircraft, and countless other scenarios. As the usage of software rises, its competitiveness has been recognized as a key element of manufacturing, which shows the importance of the SW industry even more[1].

Because SW is developed by humans, training experts is an important challenge for industry; education is essential for successfully carrying out such a task.

Consequently, there has been a trend for strengthening software education, along with the software coding frenzy that has spread across the globe[2]. In developed countries such as the United States, the United Kingdom, and Japan, software education is now provided as mandatory education for students; each government has created and offers software-related curricula[1][2]. The United States is considered to be the leading country in software education[1].

South Korea recognizes “software as the key factor driving national competitiveness in the creative economy of the twenty-first century.” The national strengthening of the SW industry, which is falling behind in global competition, needs to take place by expanding into different industries, fields of sciences, and society, as these are the creators of industrial innovation and values. The government has introduced policies for the “realization of a software-driven society ” ; and for education and training that will lead to a “convergent” society, several studies have been conducted on the application of Computational Thinking (CT) to the education curriculum[3][4][5][6][7][8].

Through these processes, CT-related studies consider programming to be the most effective activity for enhancing CT. Several empirical studies have been conducted to actively support these suggestions. In particular, an increasing number of studies have focused on developing evaluation tools for CT (and programming to enhance the same)[9][10], but they are limited by the fact that these studies targeted students (i.e., not practitioners) in the education process.

Furthermore, SW education and industry expert trainings by universities and their faculty members are systematically promoted according to the standards and guidelines of South Korea’ s National Competency Standards (NCS)[11]. There is a limitation, however, because they are configured based on programming, without an understanding of computer science in general.

Therefore, two objectives will be discussed in this study. First, this study targets software developers in the lower seven job categories among the NCS’ s defined information technology (IT) development job “families” to determine if CT competency has a positive effect on enhancing job competency in carrying out a given task. In this study, the previous study target of students is expanded to include general software developers. Second, this study aims to determine if job competence can be enhanced by including advanced competency of CT into the existing education curriculum, which is only based on programming. This should serve as a useful resource for the development of software-related human resources (HR).

2. Theoretical Background

2.1 Computational Thinking

Wing (2008) suggested that CT can be defined as thinking like a computer scientist—such as in problem-solving activities, according to the basic concepts and rules of computer science—and it is a key concept for understanding system architecture and human behaviors. CT includes recursive thinking, abstract thinking, proactive thinking, procedural thinking, logical thinking, simultaneous thinking, analytical thinking, and strategic thinking[6].

CT is a repetitive, recursive form of thinking in which the solution to a complex and difficult problem is restructured into another already known problem through methods such as simulation. It uses abstract analytical thinking when solving very difficult problems, or building complex system architectures. From the perspective of preventing the worst situations through prevention, protection, error elimination, damage prevention, and repetition, it uses logical and procedural thinking, which is also a heuristic thinking method for quickly finding solutions. CT is a cognitive and justified tool that is obtained by fusing and complexifying the broad fields of computer science[6].

Furthermore, CT can be distinguished into the following nine competencies : 1) data collection, 2) data analysis, 3) data representation, 4) problem decomposition, 5) abstraction, 6) algorithms and procedures, 7) automation, 8) simulation, and 9) parallelization[5][6][12].

2.2 Software Developers

The environment of the software industry has rapidly changed in recent years; the technology and users' requests for software have become both complicated and diverse. The quality of HR that the industry needs has thus correspondingly become more specialized and diverse, as well. The support for software-related HR has not been provided smoothly, however, and there is no standard for systematic support of their growth. As such, there is a need for the input, growth, and cultivation of software-related HR[13][14].

The NCS was created as a national standard according to these needs. It was systematized by using the knowledge and technology that are required for performing tasks in industrial sites, according to various industry sectors and levels[11].

The SW development portion of the NCS was based on SW job competency standards, in which the job families of "IT services," "packaged SW," and "embedded SW" were combined into the job family of "IT development." These were further categorized into seven jobs according to their functional aspects, as shown in Table 1. As a result, employees in the job family of "IT development," according to the NCS, can be considered to be software developers[11][15].

<Table 1> NCS Classification System (Information Technology Section Criteria)

Category	Divition	Section (Job Family)	Subsection(Job)
20. ICT	01. IT	02. IT Development	01. SW Architecture 02. Application SW Engineering 03. Embedded SW Engineering 04. Database Engineering 05. Network Engineering 06. Security Engineering 07. UI/UX Engineering

* Source : National Competency Standards(NCS)

2.3 Relationship between Computational Thinking and Software Developers

Computer sciences researchers have recently defined CT as the fundamental cognitive ability for problem-solving in all academic areas, which is the most useful cognitive ability for solving complex, real-life problems. CT is cognitive ability that includes the understanding of problem-solving according to the basic concepts and principles of computer sciences, system architecture, and human behaviors. It can be strengthened by utilizing automatic performances through various computing devices; it also includes the ability to select and utilize appropriate computing devices to automatize abstraction abilities and abstract concepts in order to select and construct appropriate abstractive concepts for problem-solving[16].

Many previous studies have selected programming as the most effective activity for CT[9][10].

In other words, as a problem-solving ability based on the concepts and principles of computer sciences, CT can produce strategic knowledge that can be applied to common problem-solving situations; it is the ability to execute and automatize abstractive concepts or problem-solving strategies with the use of various computing devices. As such, the problem-solving abilities demanded by society can be expected to be enhanced by learning CT[16][17].

According to previous studies, because problems can be understood through abstraction and the automation of CT—and the necessary solutions can be provided for problem-solving in SW development—job performances in SW development can be expected to increase in the future[6].

3. Study Design

3.1 Research Target and Model

This study targeted 169 employees in the lower seven jobs of the IT development job family, as defined by the NCS . The survey was conducted over five days, from March 5th to March 9th of 2015. The questionnaires from ten participants were excluded from this study because they were not employed in the job family of IT development. Therefore, a total of 159 questionnaires were included in the analysis. The demographic analysis of the collected data is shown in Table 2.

Furthermore, in order to determine the sub-competencies of CT—which are necessary for performing each job in the IT development field—a research model was constructed, with the independent variable set as the sub-competency of CT and the dependent variable set as the amount of CT needed for job performances.

<Table 2> General Characteristics of Subjects

(N=159)

Division	Items	Frequency	%
Work Experience	Less than 3 years	26	16.4
	3-6 years	32	20.1
	7-9 years	31	19.5
	10-12 years	21	13.2
	13 years and above	49	30.8
Job Duty	SW Architecture	41	25.8
	Application SW Engineering	62	39.0
	Embedded SW Engineering	4	2.5
	Database Engineering	13	8.2
	Network Engineering	8	5.0
	Security Engineering	13	8.2
	UI/UX Engineering	18	11.3

3.2 Validity and Reliability Analysis of Measurement Tools

The objective of this study is to find the necessary level of CT for the job performance of software developers in order to suggest implications for the enhancement of their ability. The data was collected and the investigation tools were configured to find relationships between the job family of the IT development field, as defined by the NCS, and the sub-competencies of CT. Both the validity and the reliability of the survey questions were analyzed to examine the effectiveness of the investigation tool. In order to test the validity of the measured questions, an exploratory factor analysis was performed to find unidimensionality for the questions regarding the competencies comprising CT in each IT development job. A Cronbach' s α value was utilized to test for reliability in order to test the internal consistency of the measurement tool in the reliability analysis.

The results of the exploratory factor analysis and the reliability analysis are shown in Table 3. First, the exploratory factor analysis found that it was loaded as a single factor for each job in the IT development field . The load value was shown to be at least above 0.501, and the variance ratio (which indicates explanatory power) was shown to be at least above 42.4 percent. In addition, the Cronbach' s α value was shown to be at least above 0.847 as a result of the reliability test. The validity and reliability of the survey questions thus were shown to be good, and they were used in the analysis.

4. Empirical Analysis

4.1 Analysis Method

In this study, multiple regression analysis was performed to determine the CT sub-competencies needed for the performance of each job in the IT development field.

<Table 3> Result of Exploratory Factor Analysis and Reliability Analysis

Division	Loadings						
	SW Architecture	Application SW Engineering	Embedded SW Engineering	Database Engineering	Network Engineering	Security Engineering	UI/UX Engineering
Data Collection	0.649	0.654	0.667	0.728	0.725	0.724	0.686
Data Analysis	0.729	0.715	0.657	0.673	0.738	0.686	0.748
Data Representation	0.620	0.614	0.652	0.644	0.708	0.673	0.501
Problem Decomposition	0.761	0.753	0.737	0.577	0.786	0.767	0.697
Abstraction	0.793	0.761	0.738	0.764	0.763	0.769	0.703
Algorithms & Procedures	0.711	0.777	0.760	0.630	0.833	0.802	0.795
Automation	0.532	0.676	0.585	0.657	0.763	0.702	0.670
Simulation	0.670	0.698	0.704	0.595	0.754	0.663	0.723
Parallelization	0.578	0.613	0.652	0.553	0.648	0.744	0.645
Computational Thinking	0.757	0.747	0.702	0.658	0.689	0.637	0.733
Eigenvalue	4.688	4.945	4.724	4.236	5.509	5.162	4.819
Variance ratio(%)	46.884	49.449	47.242	42.356	55.088	51.616	48.190
Chronbach's α	0.871	0.885	0.875	0.847	0.909	0.895	0.879

The conformity assessment used for the regression model was as follows. The ratio for the sum of the regression square in the overall sum of squares was determined as the coefficient of determination, and was labeled "R2." The range for the coefficient of determination is an error between 0 and 1. Because the coefficient of determination is closer to 1, this indicates that the explanatory power is greater for explaining the change in the dependent variables, according to the independent variables of the multiple linear regression model currently considered. The F-test that appears in the variance analysis table shows the overall change of the dependent variables, which is shown by the total sum of squares dissociated into the sum of regression squared and the sum of squared errors ; their relationship is summarized in the table. By testing the significance of the regression coefficient, whether or not each independent variable is significant in explaining the dependent variable can be tested. Furthermore, the significance of the regression model itself that the current researcher is considering can be tested through the variance analysis table.

In order to examine if the multiple linear regression model is valid (which was selected to explain the relationship between the explanatory variable and the response variable), and to determine how much this relationship is sufficiently explained, the total sum of squares, which shows the overall change, was divided into the sum of regression squared and the sum of the squared errors . The relationship summarized thus is the calculation method for finding the p value according to variance analysis. The hypothesis of this study was tested based on this multiple regression analysis model.

4.2 Sub-Competency for Each Job Affecting CT

The results of the regression analysis are shown in Table 4. The regression models, with the dependent variables being the overall CT that is needed for job performances, were all analyzed as being statistically significant, as follows: SW architecture (R2=0.486, F=15.670, $p<0.001$), application SW engineering (R2=0.490, F=15.907, $p<0.001$), embedded SW engineering (R2=0.420, F=11.993, $p<0.001$), database engineering (R2=0.401, F=11.063, $p<0.001$), network engineering (R2=0.395, F=10.809, $p<0.001$), security engineering (R2=0.356, F=9.156, $p<0.001$), and user interface / user experience (UI/UX) engineering (R2=0.487, F=15.738, $p<0.001$).

The observations for the sub-competencies that affect the necessary level of overall CT for each job were as follows: first, in the overall CT needed for job performance of SW architecture, the competencies of abstraction (B=0.170, t=2.189, $p<0.05$), algorithms and procedures (B=0.163, t=2.439, $p<0.05$), and simulation (B=0.217, t=3.002, $p<0.01$) were shown to have statistically significant positive effects.

Second, in the overall CT needed for job performance of SW engineering, only the simulation competency (B=0.264, t=3.135, $p<0.01$) was shown to have a statistically significant positive effect.

Third, in the overall CT needed for job performance of embedded SW engineering, the competencies of algorithms and procedures (B=0.176, t=2.200, $p<0.05$) and simulation (B=0.296, t=3.624, $p<0.001$) were shown to have statistically significant positive effects.

Fourth, in the overall CT needed for job performance of database engineering, the competencies of abstraction (B=0.241, t=3.401, p<0.001), simulation (B=0.215,t=3.304, p<0.001), and parallelization (B=0.153, t=2.204, p<0.05) were shown to have positive effects, but data representation (B=-0.164, t=-2.288, p<0.01) was shown to have a significantly negative effect. In addition, among the sub-competencies, the abstraction competency was shown to have the greatest effect.

Fifth, in overall CT needed for job performance of network engineering, no sub-competencies had significant effects at the significance level of 5 percent. At the significance level of 10 percent, however, the following competencies were shown to have significantly positive effects: data collection (B=0.146, t=1.669, p<0.1), automation (B=0.147, t=1.852, p<0.1), and simulation (B=0.143, t=1.679, p<0.1). Among these, the effect of automation competency was shown to be relatively higher.

Sixth, in overall CT needed for job performance of security engineering, the competencies of automation (B=0.147, t=2.048, p<0.05), simulation (B=0.191, t=2.314, p<0.05), and parallelization (B=0.199, t=2.228, p<0.05) were shown to have significantly positive effects. Among these, the effect of simulation

<Table 4> Sub-competency Effecting on the Need for Overall Computational Thinking

Division	Dependent Variable													
	SW Architecture		App. SW Engineering		Embedded SW Engineering		Database Engineering		Network Engineering		Security Engineering		UI/UX Engineering	
	B	t-value	B	t-value	B	t-value	B	t-value	B	t-value	B	t-value	B	t-value
Constant	0.692*	2.103	0.484	1.433	0.883**	2.593	1.266***	3.537	1.195***	3.923	1.294***	3.927	0.333	0.926
Data collection	0.054	0.768	-0.109	-1.552	0.064	0.857	0.138	1.945	0.146	1.669	-0.013	-0.150	-0.078	-0.940
Data analysis	0.104	1.327	0.153	1.898	-0.062	-0.752	0.116	1.513	0.135	1.557	0.095	1.092	-0.014	-0.147
Data Representation	-0.040	-0.630	0.105	1.523	0.120	1.610	-0.164**	-2.288	-0.040	-0.478	0.100	1.317	0.104	1.316
Problem Decomposition	0.060	0.860	0.129	1.840	0.122	1.610	-0.059	-0.892	0.084	1.004	0.065	0.820	0.158*	2.078
Abstraction	0.170*	2.189	0.103	1.347	0.030	0.373	0.241***	3.401	0.058	0.704	-0.160	-1.866	0.036	0.467
Algorithms & Procedures	0.163*	2.439	0.151	1.840	0.176*	2.200	0.061	0.949	0.053	0.589	0.105	1.274	0.267**	3.231
Automation	0.111	1.707	0.106	1.559	0.024	0.321	0.022	0.331	0.147	1.852	0.147*	2.048	0.154*	2.086
Simulation	0.217**	3.002	0.264**	3.135	0.296***	3.624	0.215***	3.304	0.143	1.679	0.191*	2.314	0.107	1.243
Parallelization	0.045	0.695	0.002	0.036	0.051	0.689	0.153*	2.204	0.014	0.191	0.199*	2.228	0.202*	2.213
R2	0.486		0.490		0.420		0.401		0.395		0.356		0.487	
F-value(p)	15.670***(0.000)		15.907***(0.000)		11.993***(0.000)		11.063***(0.000)		10.809***(0.000)		9.156***(0.000)		15.738***(0.000)	

*p<0.05, **p<0.01, ***p<0.001

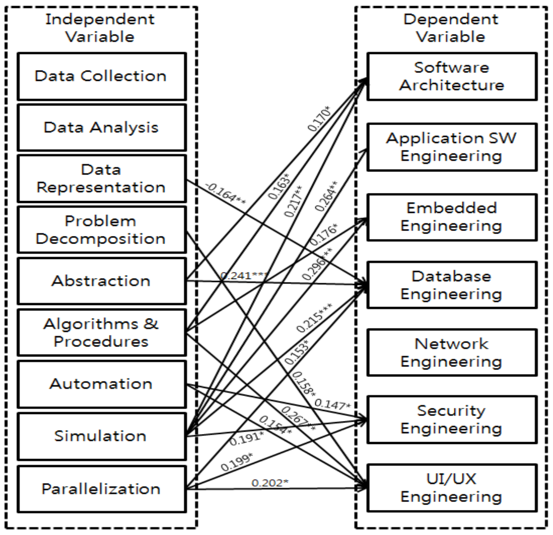
competency was shown to be relatively higher.

Finally, in overall CT needed for job performance of UI/UX engineering, the competencies of problem analysis (B=0.158, t=2.078, $p<0.05$), algorithms and procedures (B=0.267, t=3.231, $p<0.01$), automation (B=0.154, t=2.086, $p<0.05$), and parallelization (B=0.202, t=2.213, $p<0.05$) were shown to have significantly positive effects.

Among these effects, the effect of algorithms and procedures was shown to be relatively high. These study results do not highlight the importance of abstraction and automation, which previous studies on CT considered to be the core abilities. These results are therefore quite different from those of previous studies. Interviews were conducted with a number of the survey participants in order to investigate the cause for these results; the interviews found that these study results may be attributed to the working patterns of software developers. When software developers approach their jobs, rather than analyzing by abstracting and architecting the development target from the beginning, they tend to be more focused on increasing productivity. They may 1) receive already abstracted analyses and architected results; 2) recycle and reuse accumulated experience, expertise, and outcomes from previous job performances; and 3) receive solutions from senior workers. Because they performed simulation, which examines if previous products may be applied to new jobs, this sub-competence can be considered to be more important than abstraction.

4.3 Relationship between CT and Software Development Competencies

The regression analysis results shown in Table 4 show that the sub-competencies of CT have a positive effect on the enhancement of job competence. Among the IT development job family, as defined by the NCS, six jobs (with the exclusion of network engineering) were shown to have a significance level of 5 percent; and all seven jobs in the IT development job family were shown to have a significance level of 10 percent. Therefore, CT is necessary for the job competence enhancement of SW developers. The sub-competencies needed for each job were different; but the results of the regression analysis show that the CT sub-competencies with a significance level of 5 percent, which positively affected each job, were shown in the following order: simulation (five jobs), algorithms and procedures (three jobs), and parallelization (three jobs). Figure 1 summarizes these study results.



<Figure 1> Sub-competency for Each Job Effecting on Computational Thinking

5. Conclusion

This study found that CT is necessary for enhancing the job competence of software developers. Unlike previous studies, which found abstraction and automation competencies to be most important, this study found that simulation, algorithms and procedures, and parallelization were important, in that order, for the enhancement of software developers' job competence.

These study results are in line with Moursund's suggestion (2009) that for problem-solving, the interactions between various elements in the solution of a problem should be considered, and it should be conducted in a system through modeling or simulation[7][18]. This also corresponds to the objectives of "modeling and simulation" courses that are currently being taught by prominent universities abroad. The SW education conducted in SW-related departments of domestic universities, however, or in software-related HR schools (including faculty training), are oriented toward programming.

It thus is difficult to consider "modeling and simulation" as an independent subject. This study has shown that including "modeling and simulation" courses in the software-related departments of domestic universities, as well as in software-related HR schools, could be greatly effective in leading toward the software-driven society that the South Korean government aims for.

By studying the employees of the IT development job family, this study has determined that "modeling and simulation" courses are necessary, although these study results will also need to be verified through more specific empirical research.

Reference

[1] Park, H. M. (2014). *State of global software education and trend of education tools.* Korea Internet & Security Agency.

[2] Park, M. U. (2014). *State of domestic education in programming and future of the software industry.* DIGIECO.

[3] Ministry of Science, ICT and Future Planning(2013). *Software (SW) innovation strategy.*

[4] Moon, G. S. (2013). *On the Direction of the Application of the Concepts of Computational Thinking for Elementary Education.* International Journal of Contents, 13(6).

[5] Korea Foundation for the Advancement of Science & Creativity (2014). *Research for introducing Computational Thinking into primary and secondary education.*

[6] Kwon, J. I. (2014). *A Study on the Effectiveness of Computational Thinking based Teaching and Learning on Students' Creative Problem Solving Skills.* Ph.D. dissertation, Sungkunkwan University Graduate, Seoul: Korea.

[7] Choi, S. Y. (2011). *An Analysis of "Informatics" Curriculum from the Perspective of 21st Century Skills and Computational Thinking.* The Journal of Korean Association of Computer Education, 14(6).

[8] Choi, J. W. (2015). *Puzzle-Based Algorithm Learning Model for Improving Computational Thinking for Informatics Gifted Students.* Ph.D. dissertation, Graduate School of Korea National University of Education, Chung-Buk, Korea.

[9] Kim, B. S. & Kim, J. H. (2012). *The Development and Implementation of an Algorithm Instructional Material through the Problem Solving on the KOI Final Test of Elementary Students.* JOURNAL OF The Korean Association of information Education, 16(1), 11-20.

[10] Lee, E. K. (2013). *Computer Education Curriculum and Instruction : Creative Programming Learning with Scratch for Enhancing Computational Thinking.* The Journal of Korean Association of Computer Education, 16(1).

[11] Koo, J. K. (2014). *Understanding "National Competency Standards" for constructing conditions for ability-driven society.* Human Resources Development Service of Korea.

[12] Computer Science Teachers Association (2011). *Computational Thinking in K-12 Education.*

[13] Lee, B. M. (2008). *SW job performance standards overview.* National IT Industry Promotion Agency.

[14] Bae, Y. K. (2006). *Robot programming education model in ubiquitous environment for enhancement of creative problem-solving ability.* Ph.D. dissertation, Graduate School of Korea National University of Education, Chung-Buk, Korea.

[15] National IT Industry Promotion Agency(2012). *SW Job Cometency Standards.*

[16] Lee, E. K. (2009). *Robot Programming Teaching and Learning Model to Enhance Computational Thinking Ability.* Ph.D. dissertation, Graduate School of Korea National University of Education, Chung-Buk, Korea.

[17] Greg Michaelson (2015). *Teaching Programming with Computational and Informational Thinking By Greg Michaelson.* School of Mathematical and Computer Sciences, Heriot-Watt University, 5(1).

[18] Moursund, D. (2009). *Computational Thinking.* Retrieved from IAE-pedia.

박 성 빈

1998 중앙대학교 졸업(학사)
2007 중앙대학교대학원졸업(석사)
2011 ~ 현재 성균관대학교 대학원 컴퓨터교육과 박사과정
관심분야: SW교육, Computational Thinking, Software development
E-Mail: gerbera4@skku.edu

안 성 진

1988 성균관대학교 정보공학과(학사)
1990 성균관대학교 정보공학과(석사)
1998 성균관대학교 정보공학과(박사)
1990 ~ 1995 KIST/SERI 연구원
1996 정보통신기술사
1999 ~ 현재 성균관대학교 컴퓨터교육과 교수
관심분야: SW교육, 정보윤리, 정보보안
E-Mail: sjahn@skku.edu