소프트웨어 기반 시스템을 설계하는 새로운 접근법으로서의 인지시스템공학

함 동 한*
*전남대학교 산업공학과

Cognitive Systems Engineering as a New Approach to Designing Software-Based Systems

Dong-Han Ham*
*Dept. of Industrial Engineering, Chonnam National University

Abstract

소프트웨어 기반의 시스템 설계과정에서 설계자가 고려해야 하는 요소들이 다양해지면서 시스템 설계가 점점 어려워지고 있다. 다양한 설계 요소들이 존재하지만 사용자의 특성 및 직무, 사용가능한 정보기술의 특성 등이 핵심적인 요소로 간주된다. 또한 정보기술이 발달하면서 인간과 시스템의 상호작용이 점점 인지적인 특징을 지니게 되었다. 따라서 사용성 높고, 효율적이면서 안전한 소프트웨어 기반의 시스템을 개발하기 위해서는 시스템 설계자가 사용자의 인지적인 요구사항 및 그들의 직무를 시스템 설계과정에서 체계적으로 다룰 수 있어야 한다. 그러나 소프트웨어 공학, 시스템 공학 및 인간-컴퓨터 상호작용 등에서의 전통적인 시스템 설계 방법은 이러한 설계자의 설계활동을 효과적으로 지원하는데 한계가 있었다. 그 대안으로 인지시스템공학(cognitive systems engineering; CSE)은 인간중심의 설계철학을 바탕으로 소프트웨어 기반의 복잡한 시스템 설계과정에서 설계자의 활동을 체계적으로 도와줄 수 있는 유용한 개념과 방법을 제공해주고 있다. CSE는 원래 사람이 실시간으로 감시 및 제어해야 하는 복잡한 사회기술적 시스템(예: 원자력발전소 및 공항관제소)의 분석, 설계 및 평가를 위해 태동한 학문이다. 그러나 CSE에서 제공하는 이론적 및 방법론적 프레임워크는 소프트웨어 기반의 시스템을 설계하는 데에도 유용하게 활용할 수 있는 충분한 가능성을 갖고 있다. 이 논문은 CSE의 근간을 이루는 핵심 개념 및 원칙을 고찰하고 소프트웨어 기반 시스템 설계에의 활용가능성 및 그 방안을 논의한다.

Keywords : Cognitive Systems Engineering, Cognitive Work Analysis, Software Requirements, Requirements Engineering, Socio-technical Systems Engineering

1. Introduction

Software has become the backbone of modern industrial systems (e.g. nuclear power plants (NPPs), air traffic control, and intensive care unit) [2]. As such software-based systems are increasingly complex, their design and evaluation is accordingly difficult [1]. There is a range of factors that system designers should consider through whole phases of development life cycle. Additionally, in comparison with traditional software development focusing on software program, design of software-based systems includes other design items as well as software program

† 이 논문은 2012년도 전남대학교 학술연구비 지원에 의하여 연구되었음(과제번호: 2012-0730)
† 교신저자: 함동한, 광주광역시 북구 용봉로 77 전남대학교 공과대학 산업공학과
M・P: 016-417-4607, E-mail: donghan.ham@gmail.com; dhham@jnu.ac.kr
2012년 7월 2일 접수; 2012년 9월 3일 수정본; 2012년 9월 4일 게재확정

For example, design of information systems operated in the control room of NPPs should be accompanied by the design of task procedure, training system, staffing, and alarm system.

Another main characteristic of software-based systems is that the interaction between systems and users is very cognitive [12]. This enforces system designers to take into account cognitive characteristics of users and to regard software-based systems as artificial cognitive systems. Designers need to have a viewpoint that human (natural cognitive system) and software-based systems (artificial cognitive systems) constitute joint cognitive systems (JCS). This point is also emphasized by a lot of usability and safety problems that were reported in the literature [6].

However, traditional software and systems engineering have proven inadequate to fully address the changed design environment and requirements. Human-computer interaction (HCI) and human factors engineering have failed to provide a good solution to these demands. As an alternative, a new discipline called cognitive systems engineering (CSE) has emerged in order to provide concepts, models, and frameworks that help designers cope with these new design demands in a more effective way. This paper introduces the basic concepts and principles of CSE and discusses how they can be applied to the design of software-based systems.

2. Overview of CSE

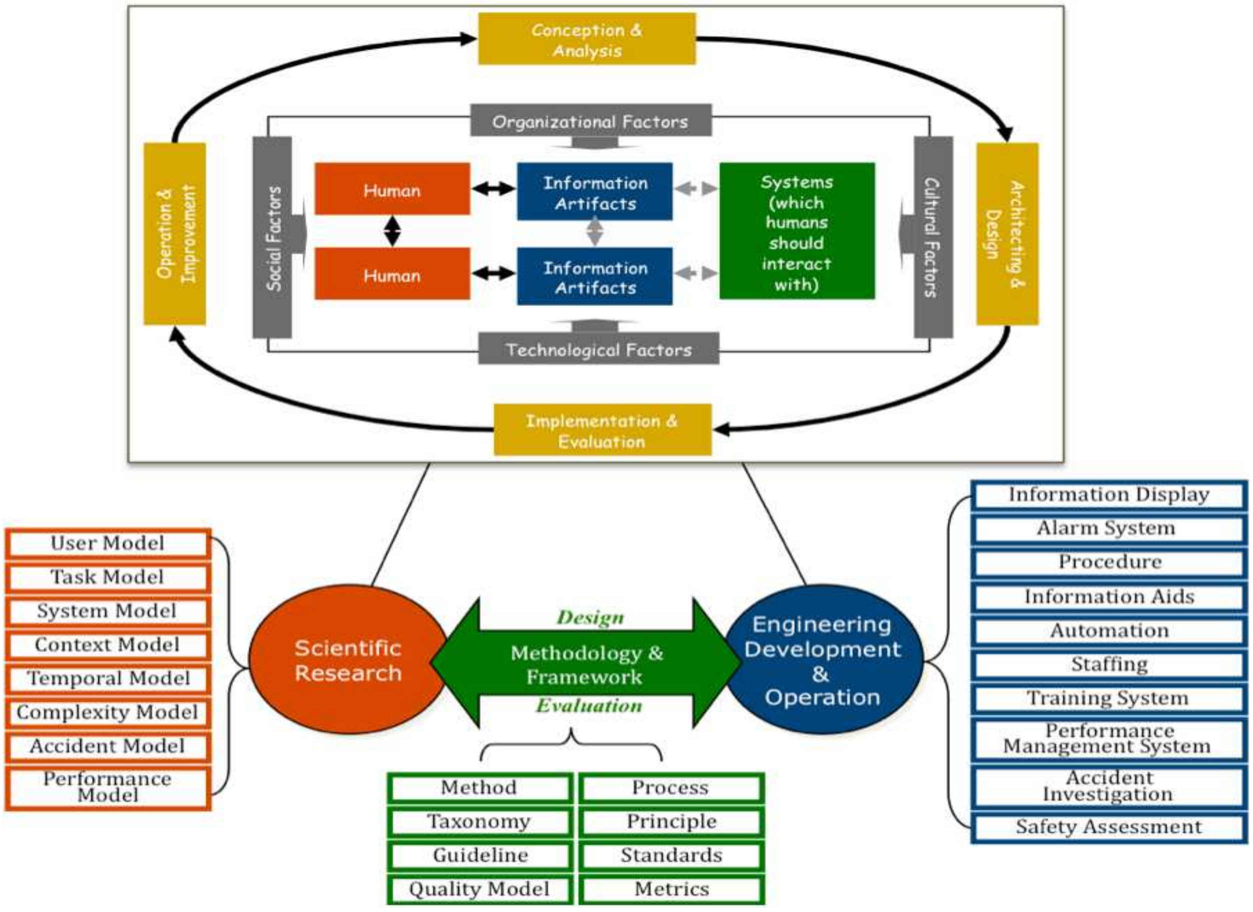
CSE was originally concerned with analysis, design, and evaluation of complex socio-technical systems that need to be monitored and supervised by human operators in real time [11]. It has provided concepts, models, and frameworks for analyzing, designing, and evaluating usable, safe systems for supporting human operators' cognitive activities. It is a new branch of systems engineering with the purpose of enhancing the performance of JCS, paying particular attention to human operators' cognitive performance. As software has become more and more a critical part of complex socio-technical systems, CSE is now regarded as one of viable disciplines for designing software-based systems.

Several disciplines have influenced the development of CSE, which include human factors engineering, HCI, cognitive psychology, artificial intelligence, information science, and software engineering. To solve a practical design and evaluation problem, CSE has borrowed concepts and methods from those disciplines and unified them by employing systems engineering viewpoint. Although CSE can be applied to any kind of problems where information and knowledge processing is critical issue, we can summarize the typical research topics of CSE as follows:

- Analysis of cognitive works in complex systems
- Design of human-computer interfaces in complex systems
- Operation of safety-, mission-critical complex systems (designing automation that human can easily understand, designing training systems, minimizing human errors and enhancing system safety, and accident analysis and investigation)
- User interface design and usability for electronic consumer products and software
- Human-computer cooperative information system
- Cognitive issues in product innovation and knowledge management
- Computer supported cooperative work

3. CSE Approach to Software-Based Systems

Figure 1 shows a model of CSE research activities. As stated previously, all of the research activities of CSE concern with analysis, design, and evaluation of complex socio-technical systems. The upper part (rectangle) of Figure 1 shows that these systems are generally composed of humans, information artefacts, and (technical) systems that humans should interact with. In these systems, there are several types of interactions: human-human interaction, human interaction with information artefacts, human-systems interaction through information artefacts. All of these interactions can be explained in the context of JCS.



<Figure 1> Model of CSE Research Activities

To develop these interactions, systems designers need to consider four different factors in a collective manner, which include technological factors, social factors, organizational factors, and cultural factors. And this development can be explained from the perspective of a life cycle that includes conception & analysis, architecting & design, implementation & evaluation, and operation & improvement.

Like other engineering disciplines, CSE research activities can be categorized into three groups: scientific research, engineering development and operation, and methodology and framework connecting the two activity groups. Scientific research activities of CSE aim at developing various theoretical models that can explain JCS. Typical models include user (cognitive) model, task model, work domain model, and context model. Engineering development activities are concerned with developing actual information artefacts comprising JCS, which contain information display, database, procedure, automation, and so on.

However, the design of information artefacts should be interpreted as a part for developing three types of interactions (human-human, human-artefact, human-systems). Methodology and framework development activities bridge the gap between scientific models and design items [3-4]. They attempt to offer a practical set of processes, methods, guidelines, and so on to develop information artefacts based on the theoretical models found in scientific research activities; thus they make the process of developing information artefacts more systematic.

CSE approach to designing software-based systems needs to be considered in the context of Figure 1. From the perspective of CSE, design of software-based systems implies that several items should be simultaneously designed in order to optimize the performance of JCS. In other words, CSE approach emphasizes the concept of JCS that could not be found in other approaches.

Development of high-quality JCS needs several kinds of models as a basis for development, as shown in the left of Figure 1. To help designers to make use of the models as well as to build them, CSE offers comprehensive frameworks or methodologies and problem-oriented methods and techniques.

The first and primary problems that we should address when applying CSE to the design of software-based systems are what to analyze and how to analyze them-requirements identification and specification [5]. CSE approach leads designers to identify several analysis dimensions that conceptually constitute JCS and thus to identify design requirements on the basis of these dimensions.

Typical analysis dimensions claimed by CSE frameworks are as follows [1, 6-7, 11-12]:

- Human users' cognitive capabilities and limitations (how they think, learn remember, use technology, interpret environment, form goals, work in teams)
- Structural, behavioural, and functional characteristics of work domain (system) that users should interact with
- Tasks that users should conduct
- Strategies that users use to achieve a task
- Collaboration works to achieve a task
- Functionalities of available information technology

With regard to the problem of how to analyze, the order of analyzing the above analysis dimensions should be specified in consideration of semantic relations among workproducts resulting from the analysis of each dimension. There are only a few CSE frameworks that address both of the problems (what to analyze and how to analyze) in an integrated manner. Of those, cognitive work analysis (CWA) framework has gained much attention in that it is based on ecological approach and offers several tools for modelling each analysis dimension. The next section explains CWA framework and describes two examples on the application of CWA to software-based systems.

4. Application of CWA

CWA is a formative analysis framework to human work analysis that predicts how works can be done, rather than how works should be done or how works are actually done. The purpose of CWA is to identify intrinsic work constraints shaping human goal-directed, adaptive behaviour.

4.1 Key Concepts and Phases of CWA

CWA advocates that human work analysis should be a prerequisite for designing software-based systems. It emphasizes that the purpose of human work analysis should be to identify intrinsic work constraints influencing the degree of freedom of human work activities. These intrinsic constraints are interpreted as design requirements. To analyze these constraints, CWA claims that analysis activities should consist of five phases. Table 1 explains the purpose of each of the five phases and modelling tools that can be effectively used in each phase.

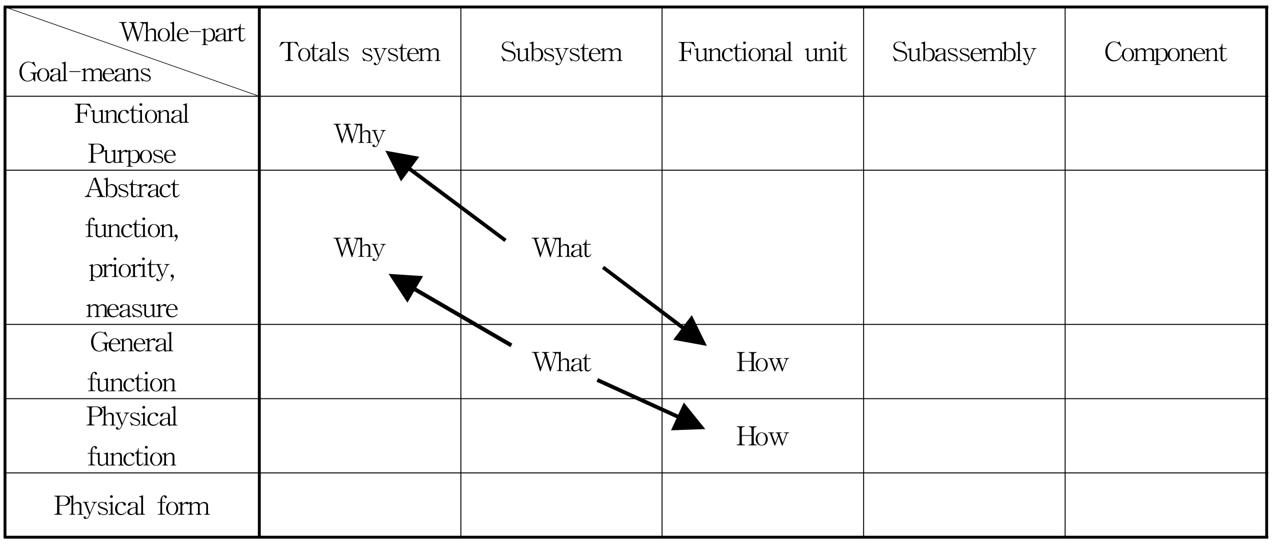
<Table 1> Summary of CWA Phases

Phase	Purpose of analysis	Modelling tool
Work domain analysis	To identify purposes of systems, and functions and their relations designed to achieve the purpose	Abstraction hierarchy
(Control) Task analysis	To Understand characteristics of tasks needed to perform the identified functions in terms of knowledge and cognitive process	Decision ladder
Strategy analysis	To understand how to perform the identified tasks in more detailed way	Information flow map
Social organization analysis	To understand the social organizational relations between actors (humans or automation)	All of the above three
Human competency analysis	To understand cognitive competencies that human need to have to perform the tasks	Skill-Rule-Knowledge taxonomy

Detailed explanation of the modelling tools is beyond the scope of this paper. In this paper, only abstraction hierarchy (AH) is described briefly, as it is related to two examples of CWA application to be explained later. AH is a multi-level knowledge representation framework for modelling the functional structure of a particular work domain or system [11]. Although there is no absolute rule for the number of abstraction levels, five levels are usually employed for analyzing complex socio-technical systems or software-based systems. Figure 2 explains the five abstraction levels and properties represented by each level. One important feature of AH is that it is defined by means-ends relationships between adjacent levels, with higher levels describing purpose-related functional information and lower levels describing more physically implemented information [11]. AH is differentiated from the hierarchical levels characterized by physical part-whole decomposition that is a typical of engineering analysis methods. Another advantage of using AH as a modelling tool for system analysis is that it is device-independent, event-independent, and psychologically relevant [13]. Thus it can be widely used for any kind of work domain or system. Figure 3 shows abstraction-decomposition space that combines AH dimension and part-whole dimension. CWA advocates that designers need to examine the invariants of a work domain (system), which can influence the degree of freedom of human workers' activities, by using the matrix shown in Figure 3.

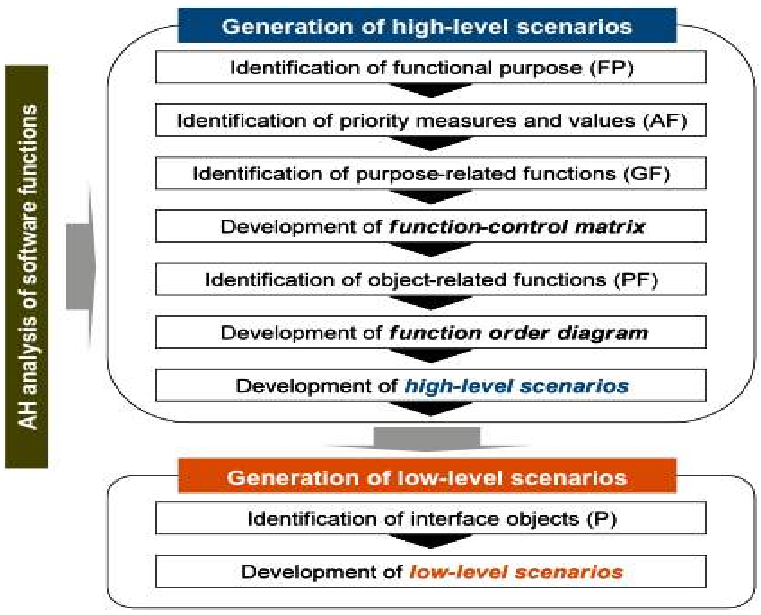
Means-Ends relations	Property represented
Functional purpose	the purpose for which the system was designed
Abstract function	the causal structure of the process in terms of mass, energy, information or value flows
General function	the basic functions that the system was designed to achieve
Physical function	the characteristics of the components and their interconnections
Physical form	the appearance and spatial location of those components

<Figure 2> Five Levels of AH



Whole-part / Goal-means	Totals system	Subsystem	Functional unit	Subassembly	Component
Functional Purpose	Why				
Abstract function, priority, measure	Why	What			
General function		What	How		
Physical function			How		
Physical form					

<Figure 3> Abstraction-Decomposition Space


AH analysis of software functions
Generation of high-level scenarios
Identification of functional purpose (FP)
Identification of priority measures and values (AF)
Identification of purpose-related functions (GF)
Development of function-control matrix
Identification of object-related functions (PF)
Development of function order diagram
Development of high-level scenarios
Generation of low-level scenarios
Identification of interface objects (P)
Development of low-level scenarios

<Figure 4> Generation of Scenarios Based on Abstraction Structure of Software Functions

4.2 Two Examples of CWA Application

Application of CWA to the design and evaluation of software-based system has been increasing. Here, two examples are introduced, which focused on the analysis of a work domain by the use of AH.

One example is concerned with requirements specification based on AH concept [10]. One reason for low productivity of software development is that requirements specifications that designers have to work with are not tailored to support their problem solving methods and strategies [9]. Software development is a cognitively complex process that needs a series of problem-solving tasks. Specifications used in problem-solving tasks are made to provide assistance in this process. Thus it can be said that requirements specifications in software development process should help software designers solve their problems effectively. However, cognitive psychologists have claimed that the representation of the problem provided to problem solvers can affect their performance. The representations available to problem solvers can either degrade or enhance performance. If their content, structure, and form are not built reflecting the cognitive characteristics of problem solvers, they can degrade the problem solving performance.

Requirements specifications in software development can be regarded as the representations of the problem domain and tasks for which software should be developed. Therefore, requirements specifications should be constructed in a way that they reflect the cognitive characteristics of software designers as problem solvers.

However, a requirements specification can be used by multiple users. Additionally, they have different mental models of the problem and tasks they have to deal with, and their problem solving methods and strategies can be various. Thus, good requirements specifications should support all the possible methods and strategies that can be employed by their multiple users. Leveson [10] pointed out that one main reason why many software engineering tools and environments are not readily accepted or easily used is that they imply a particular mental model and force potential users to accept through problems using only one or a very limited number of strategies that are usually preferred by the designer of the tool. Thus she claimed that AH concept can be effectively used for specifying requirements in software development because the AH-based representation of a problem domain is a domain knowledge representation that can be used for any tasks and strategies. She called this approach, which applies AH concept for requirements specifications, as 'intent specifications'.. To demonstrate the use of the intent specification approach, Leveson [10] applied AH concept to identifying and specifying design requirements of Traffic Alert and Collision Avoidance System to be used in US air traffic control system and found that this approach can be a useful way for supporting software development activities.

Another example is about the evaluation of software usability using scenarios organized by AH concept [8]. Scenario-based analysis has been much used in software engineering community. In spite of its widespread applications, it has two critical drawbacks. First, there is no systematic process of generating and using scenarios. Second, scenario-based analysis has a problem of incompleteness in dealing with the functions of software. To resolve these problems, Kwon et al. [8] proposed a new method for generating and using scenarios organized in terms of abstraction structure of software and conducted a case study for testing the applicability of the method in a word processing program developed by a Korean company. Figure 4 shows the structure of the proposed method.

As shown in Figure 4, they claimed that two types of scenarios need to be differentiated. High-level scenarios are concerned with the semantics of tasks that are characterized by the functional features of software. More specifically, the semantics of tasks contain task goals, task information structure organized by menu structure, and task sequences. For example, 'printing a file in a PDF format' can be a high-level scenario. In this scenario, the main issue is what functions of software can be used to achieve the goal of a task. The task 'printing a file in a PDF format' can be done by two functions: 'save as' and 'print'. Low-level scenarios are about detailed operations to achieve a task that are related to visible user interface objects of software. In the scenario above, the function 'print' for the task 'printing a file in an PDF format' can be done by three ways: executing a menu element, pressing a shortcut key, and clicking a button. This method claimed that, to generate high-level scenarios, function-control matrix and function order diagram should be developed based on the abstraction structure of software functions. Function-control matrix explicitly shows the relations between the functions at the two abstraction levels: physical function and generalized function. Function order diagram represents the temporal relationships between functions needed to achieve a task.

This method consists mainly of three phases: analysis of the abstraction structure of software functions, generation of high-level scenarios by using function-control matrix and function order diagram, and generation of low-level scenarios by associating interface objects with the functions identified in the high-level scenarios. In the context of Figure 1, this example can be classified to a scientific study for developing a system and task model, as well as a methodological study for developing a method and process for usability evaluation.

5. Conclusion

This paper introduced CSE as a new discipline for developing software-intensive systems. Complex socio-technical characteristics of software-based systems need a new design framework that supports designers' activities with human-centred perspective. CSE advocates that designers need to have the concept of JCS to implement cognitively well-engineered software-based systems. CSE has developed several comprehensive frameworks and problem-oriented methods and techniques for developing an effective JCS. Of those, CWA is regarded as one of the most promising frameworks. We outlined the CWA and explained how to apply CWA to the problem of designing software-based systems.

The concepts and modeling tools of CWA would be effectively used to design usable and safe software-intensive systems, together with traditional software and systems engineering methodologies. CWA framework emphasizes ecological approach to analyzing systems, which begins from work domain analysis through task analysis to users' cognitive analysis. Thus work domain analysis results usually becomes the bases for designing software-intensive systems. The detailed users' task analysis results offer a good understanding of user's task requirements. This implies that CWA framework can be particularly useful for requirements analysis and modeling and systems quality evaluation in the phases of systems development life cycle.

Although CSE has much potential as a new approach to designing software-based systems, there is still a lack of application case studies. Therefore the application of CSE frameworks and methods to various kinds of software-based systems remains a further research. As another future research topic, more detailed methodological process should be developed to make it easier to apply CSE to real design problems.

6. References

[1] Bisantz, Ann M., Emile Roth, Bart Brickman, Laura Lin Goesbee, Larry Hettinger, and James McKinney. (2003), "Integrating Cognitive Analyses in a Large-Scale System Design Process.", International Journal of Human-Computer Studies. 58(2): 177-206.

[2] Boehm, Barry. (2008). "Making a Difference in the Software Century.", IEEE Computer. 41(3): 32-38.

[3] Carroll, John (Eds.). (2003). HCI Models, Theories, and Frameworks, Morgan Kaufmann.

[4] Diaper, Dan and Neville Stanton (Eds.). (2004). The Handbook of Task Analysis for Human-Computer Interaction, Lawrence Erlbaum.

[5] Ernst, Neil, Greg Jamieson, and John Mylopoulos. (2006). "Integrating Requirements Engineering and Cognitive Work Analysis: A Case Study.", Proceedings of the 4th Annual Conference on Systems Engineering Research.

[6] Hollnagel, Erik and David Woods. (2006). Joint Cognitive Systems: Foundation of Cognitive Systems Engineering, CRC Press.

[7] Hori, Shinichiro, Kim Vicente, Yujiro Shimizu, and Isao Takami. (2001). "Putting Cognitive Work Analysis to Work in Industry Practice: Integration with ISO13407 on Human-Centred Design.", Proceedings of the Human Factors and Ergonomics Society 45th Annual Meeting.

[8] Kwon, Gyuhyun, Dong-Han Ham, and Wan Chul Yoon. (2007). "Evaluation of Software Usability Using Scenarios Organized by Abstraction Structure.", Proceedings of European Conference on Cognitive Ergonomics.

[9] Lauesen, Sauren. (2002). Software Requirements. Addison-Wesley.

[10] Leveson, Nancy. (2000). "Intent Specifications: An Approach to Building Human-Centred Specifications.", IEEE Transactions on Software Engineering. 26(1): 15-35.

[11] Rasmussen, Jens, Annelise Pejtersen, and L. P. Goodstein. (1994). Cognitive Systems Engineering, John & Wiley Sons.

[12] Vicente, Kim. (1999). Cognitive Work Analysis, Lawrence Erlbaum Associates.

[13] Vicente, Kim. (2006). "Cognitive Engineering: A Theoretical Framework and Three Case Studies.", International Journal of Industrial and Systems Engineering. 1(2): 168-181.

저 자 소 개

함 동 한

현재 전남대학교 산업공학과 조교수. 인하대 산업공학과 공학사, KAIST 산업공학과 공학석사 및 공학박사를 취득하였음. 2001 ~ 2005년 ETRI 선임연구원 재직. 2005 ~ 2012년 영국 미들섹스대학교 공학 및 정보과학부 종신연구교원 재직.

연구 분야는 인지시스템공학, 지식서비스공학, 서비스과학, 인간-컴퓨터 상호작용, 시스템 안전공학 등

주소: 광주광역시 북구 용봉로 77 전남대학교 공과대학 산업공학과