J Inf Process Syst, Vol.14, No.6, pp.1420~1430, December 2018
https://doi.org/10.3745/JIPS.01.0034

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

A Study on Business-Based Screen Design Techniques for Designing Efficient Applications

Tae-Woo Kim*, Sun-Yi Park*, and Jeong-Mo Yeo*

Abstract

To build a successful information system, design and development should be carried out from the enterprise perspective. A complicated business is represented in various ways as technology advances, and many development methodologies have been studied from the viewpoint of technology and development. Each domain is independently designed and developed from the enterprise perspective, but there would be inclusive parts due to the integrated process wherein the definition, design, and development of business are carried out, and the design is done based on the designer's experience. This study would like to address the technique of designing screens based on the business process of the applications derived from the business. It designs the screens that appear when actual applications are completed, including how the data transfer process in the derived business process is represented and operated on the relevant screens. It designs the screen which is displayed when the actual application is completed and how the data transfer process in the derived business process is represented and operated on the relevant screen. In addition, it designs the DFD representing the overall flow of data for each business to represent the movement procedure between screens in general. Through the design method proposed in this study, the client's requirement could be confirmed to reduce the cost for redevelopment, the problem of communication between designers and developers with various experiences could be reduced, and an efficient design procedure could be provided to persons who lack design experience.

Keywords

Applications, Business Processes, DFD, Screen Design, Screens

1. Introduction

The business complexity of a company is increasing with the advancement in IT technology [1], and its success is also related to the well-made information system. Performance would differ depending on how much complex business is involved in the information system. Many studies have been conducted to develop information systems successfully [2,3]. In addition, a fast, accurate information system should ideally be made as the development speed becomes faster and the development cycle is shortened in the present compared to the past. Many studies have been done on the design and implementation by architecture from the enterprise perspective in an effort to meet client requirements [4,5], as a result, various development methodologies have emerged. A lot of technological advancements

※ This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
Manuscript received November 19, 2018; accepted November 23, 2018.
Corresponding Author: Jeong-Mo Yeo (yeo@pknu.ac.kr)
* School of Computer Engineering, Pukyong National University, Busan, Korea (mtkim7895@pukyong.ac.kr, {psunyi, yeo}@pknu.ac.kr)

Copyright© 2018 KIPS

have been made as well from the application's point of view of the enterprise-wide architecture domains, and various development methodologies have been studied and applied to actual projects according to the development's characteristics and requirements [6,7]. Nevertheless, there are some difficulties in designing the application. From an enterprise perspective, the design method should lead to an application, and the database architecture is designed based on the business it is derived from. From the enterprise perspective, the design is done for each architecture domain, and existing business analysis procedures have been analyzed in relation to the application; note, however, that there is an inclusive part in the conversion process of the design for each architecture domain, with a part requiring the designer's experience-based design in the application design as well. Various knowledge-based designers and developers are involved in developing an application, and the experience-based design may cause difficulties in communication between designers or between designers and developers. In addition, the experience-based design elements pose many difficulties to persons who lack experience in developing applications in carrying out the design. To solve these problems, this study would like to define the application's operations based on the business derived through business analysis as a procedure for designing the application as well as design screens displaying the state of application where the defined process is completed.

2. Related Studies

2.1 Data Flow Diagram

The data flow diagram (DFD) is a tool primarily used in the structured analysis technique, which focuses on data to express changes applied by the information entered between components configuring a system and the result in the form of a network. The DFD is a top-down method which draws from large processes to small ones and the control and order, etc., of data are represented from a different point of view because it focuses on the data [8,9]. The DFD consists of four components: external objects, processes, arrows, and repositories. The external object is responsible for the input and output of data at the boundary of a DFD, representing the start and end of data generation in the process handling procedure. The process plays the role of a converter that converts input data into the desired data to output it in a system, which is represented by describing a process name or an actor performing the process in a circle. The flow of data represents the interface between the components of a DFD, and it is indicated by an arrow. The data repository is a file or a database system that stores data, and it performs data storage or access requests. The DFD uses four components to express in detail what users require from a higher level to a lower level and to derive the structure of the system. In the structural analysis technique, the DFD represents the overall appearance of the system from a data perspective.

2.2 Use Case Scenario

Object-oriented analysis techniques have been proposed and developed due to the fact that they can accurately reflect real-world structures and increase productivity through reuse. When analyzing the client's requirements to design with an object-oriented analysis technique, the system's actors and use cases they perform are derived to draw a use case diagram, and a use case scenario is created for each

use case [10-12]. A use case scenario shows the flow and process of events for each use case by the information exchanged between the system and the actors or generated between the use cases. The use case scenario specifies the role performed by the relevant use case in the system and the details and includes the related use cases needed to perform the event. In addition, the details of event processing where the use case is executed are classified into basic flow items to separate and describe the procedures to be processed. In a use case, other situations except those represented by the basic flow are classified into an alternative flow, and the flow performed to handle the error, etc. generated in the system is classified into an exception flow to draw it [13,14]. Object-oriented analysis techniques visually present to the user through a use case and describe the details through a use case scenario. Because the use case scenario expresses the content of the actions performed by each use case, the flow of data is expressed through different procedures, and many procedures must be performed.

2.3 User Story

The agile development technique has been studied in order to be flexible in responding to changes in requirements and to pursue the development process's efficiency [15,16]. In order to express the client's requirements or software functions instead of documenting them in accordance with the efficiency pursued by the agile technique, a brief user story of one or two sentences is prepared. A user story is focused on a short developmental life cycle for the purpose of activating communication rather than detailed specifications, so a small number of persons implement and test it within a short time. By carrying out short, small-size development, it continues to reflect the client's requirements to pursue development. User stories perform analysis and development in small to large units due to the development characteristics. Because it reflects the changing requirements, all of the derived businesses are not transformed. In addition, there are difficulties in larger projects particularly grasping the overall flow of the system.

A use case scenario and a user story express the flow executed by a system in writing, so there may be difficulties in communication depending on the developer's extent of understanding. This study would like to visualize the overall flow of a system to achieve communication more clearly.

3. Business-Based Application Screen Design

This study would like to present a screen design procedure that converts business and element processes derived under the condition of assuming that the business and element processes are divided for derivation by applying four actions needed for an application, such as C (create), R (read), U (update), and D (delete) into screens based on the "product order system" [17] as a business defined from the enterprise perspective to develop information systems [18]. The screen design procedure consists of two major steps and designs the screens used by users and the DFD showing the overall data flow for each business.

3.1 Screen Design

In the screen design, the aim is to make a prototype which is the completed form of an application to check whether the client's requirement is exactly reflected to reduce the cost required to modify it

during or after development and communicate effectively with designers and developers possessing various kinds of know-how in visual information. The screen is designed by composing one or more business and element processes to reflect what is used in the actual environment and for convenient use by users. It is also composed of screens for users to use services and windows such as address search and warning for convenience in executing applications. For the design of screens, it first designs the information management business necessary to execute applications, and then the screen of the business actually executed using the constructed information [19,20].

In this study, we would like to describe the process of transforming the business process designed for employee registration into a screen design. Table 1 shows the employee registration process designed in process design.

The screen design represents the state of the screen where the relevant process is performed, and it is divided into three parts: data entered as information delivered on the relevant screen, events indicating the screen's operation, and data displayed on the screen.

Fig. 1 shows the employee registration screen of the employee's information management screens in the "product order system." For the screen's appearance, which component is used to represent information is pre-set, the information of the relevant component such as position and size is a visual representation, and one or more data sets are displayed on a screen. Thus, the data set information represented in the relevant component is displayed together.

Table 1. Employee registration process in business process

Business/process name	External input or DB access	Internal input/ Output or showing data	Transfer motion	DB access or internal function	External output or internal function output	Next process
Employee Registration	-	{Employee View Information1}:I	<Select Department> = {{ Department View Information1}}-L;	{Employee View Information1}. {deptNo, DeptNm} = {Department View Information1}.{deptNo, deptNm}-sys;	-	-
	R(Dept, Emp); > {{Department Information1}}	{Department View Information1}}	c(Regular)? <Select Manager> = c(Regular)?{{Manager View Information}}-L;	{Employee View Information1}. {mgrNo, mgrNm} = {Manager View Information1}.{empNo, empNm}-sys;	-	-
	-	{{Manager View Information1}}	[Register]	*Employee Input Check*{{employee View Information1}};	Error Object Name? Error Object Name: {employee Information1}	Object Name? "Employee Input Error"; : "Employee Register Save";
	-	-	[Cancel]	-	-	x"Employee Information Manager";
Employee Input Error	Error Object Name	/* Message to re-enter invalid object names*/	[Confirm]	-	-	xreturn;
Save Employee Registration	{Employee Information1}	-	[Yes]	{empNo}-sys;>{Employee Information}-C(Emp); > {Employee Information1}.deptNo?{{ Work Basic Details}={{Employee Information1}. {empNo, deptNo, hireDate},'991231'} -sys; > { Work Basic Details}-C(wrkDtl);}	-	x"Confirm Employee Registration"
	-	-	[No]	-	-	xreturn;
Confirm Employee Registration	-	/* Message to inform you that you have successfully saved the entered employee information*/	[Confirm]	-	-	x"Employee Registration"x"Manage Employee Information";

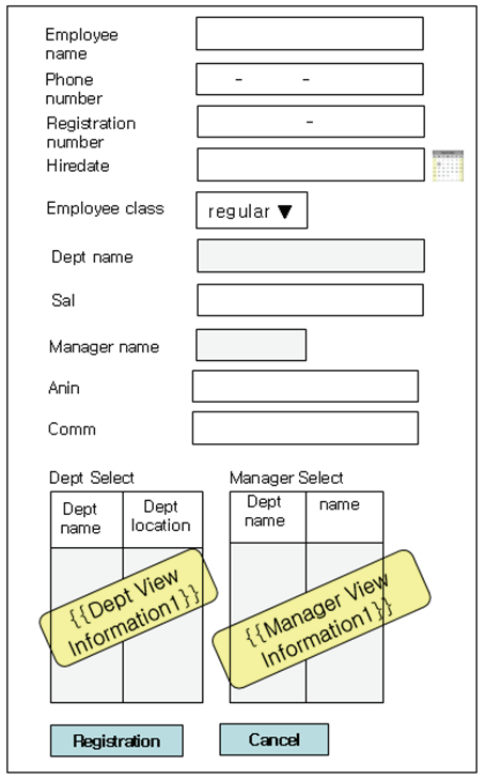


Fig. 1. Screen for employee registration.

As information displayed on the screen, data input is shown when the screen starts or new data is displayed when an event occurs. The screen's input data is divided into internal and external, the internal input data refers to the data displayed after internally calculating when accessing a database to import data or a specific event occurs, or the data to be entered by users; external data refers to the data when receiving data after opening another screen or window or temporarily moving from the screen currently in progress to another screen or window to undergo a series of processes. For the input, the entered information is specified with an "in" symbol. In the creation process of the information management process needed for the application's operation as in Fig. 2, there is no input value when initially executing the screen because users make an empty input. In this case, it enters "none", which means that there is no input data when the screen is initially operated; if input is made several times, "in" is appended with a number to indicate several inputs. When there is an input, it should be expressed for the relevant input information, and the information entered by users is expressed by an "I (input)" symbol after the entered data. In addition, more than two pieces of information are connoted into a data set to express because the relevant input information is continuously used in the future. It also details which types of information initially construct the connoted information. The detailed information about the data set specifies first which columns make it up as indicated by double curly brackets if a result of more than two rows is required at a time separates the information that requires input in connection with the data architecture, or gets stored as data but not displayed on the screen; for the data accessed by the database to import for the source of the relevant data, a query such as the database's table and condition to import the relevant data is specified.

1. in : none > in1, in2, in3
   1) in1 : {**Employee view information1**}:I
   2) in2 : R(dept, Employee); > {{**dept information1**}} > {{**dept view information1**}}
   3) in3 : R(Employee, workDtl, dept, w(Employeeclass=='G' AND End Date='991231')); > {{**manager information1**}} > {{**manager view information1**}}
   - {**Employee view information1**} : {*Employeename, basic address, detail address, email id, email address, *phone number, *registration number, *hiredate, *Employeeclass, {deptno, deptname}<(<dept select>), c(regular)?(sal, {managerno, managername}<(<manager select>)), c(contract)?(anin, comm), c(temporary)?(comm, perWrkHh)}
   - {{**dept information1**}} : {{*dept no, *deptname_asc1, dept location, *phone number, dept manager class, regular dept manager no, contract dept manager no}}
   - {{**dept view information1**}} : {{*deptno, *deptname#_asc1, dept location#, *phone number, dept manager class, regular dept manager no, contract dept manager no}} <= {{**deptinformation1**}}
   - {{**manager information1**}} : {{*Employeeno, *deptname_asc1, *name_asc2}}
   - {{**manager view information1**}} : {{*Employeeno, *deptname#_asc1, *name#_asc2}} < {{**manager information1**}}

Fig. 2. Input on the employee registration screen.

The action means a process procedure generated by an event on the screen and represents a processing procedure to be performed in detail. The event is generated by a component or an input device such as mouse and keyboard, etc., an action could be performed internally, and the data is transferred and moved to another window or screen.

Since the action represents a procedure necessary for actually performing business, it includes various processing procedures, and multiple actions may be performed complexly to be represented. The action represents the processing with a ">" symbol when the relevant event occurs and with a ">>" symbol when the window displayed on the current user's screen or the screen is closed and data gets delivered and moved.

2. <dept select> = {{**dept view information1**}}-L; > {**Employee view information1**}.{deptno, deptname} = {**dept view information1**}.{deptno, deptname}-sys;
3. c(regular)?<manager select> = c(regular)?{{**manager view information1**}}-L; > {**Employee view information1**}.{manager no, manager name}={**manager view information1**}.{Employeeno, name}-sys;
4. [registration] > *Employee check*({**Employee view information1**}); > object name of error? Object name of error:{**Employee information1**} > object name of error?"Employee input error"; : "Employee registration manager";
   - *Employee check*({**Employee information1**})/object name of error : Check that the values of the objects are invalid (for example, a required input), if it has error then return to name of the object in the first error object, and function that returns null if all are valid
   - {**Employee information1**} : {*Employee name, basic address, detail address, email Id, email address, *phone number, *registration number, *hiredate, *Employeeclass, deptno, sal, managerno, anin, comm, comm, perWrkHh} <= {**Employee view information1**}

Fig. 3. Event on the employee registration screen.

Fig. 3 shows the actions on the "employee registration screen" they are represented by a "[]" symbol when clicking a component to generate an event; data transfer or internal function call is also indicated, and data is connoted into a set unit to express. For a function, its function name and parameters are briefly indicated like data, representing a definition of the function on the first use. Since performing an action expresses complex business, it is also represented if performed in accordance with various conditions, and the condition for an action is separated with "c()?" if it corresponds to the condition, it is divided into "true" and "false" to execute according to the command after the "?" if true and write a command to be executed if false by separating with a ":" symbol after the true command if false. If there is no execution when false, it is represented by omission, and the condition of the database is expressed with "w()."

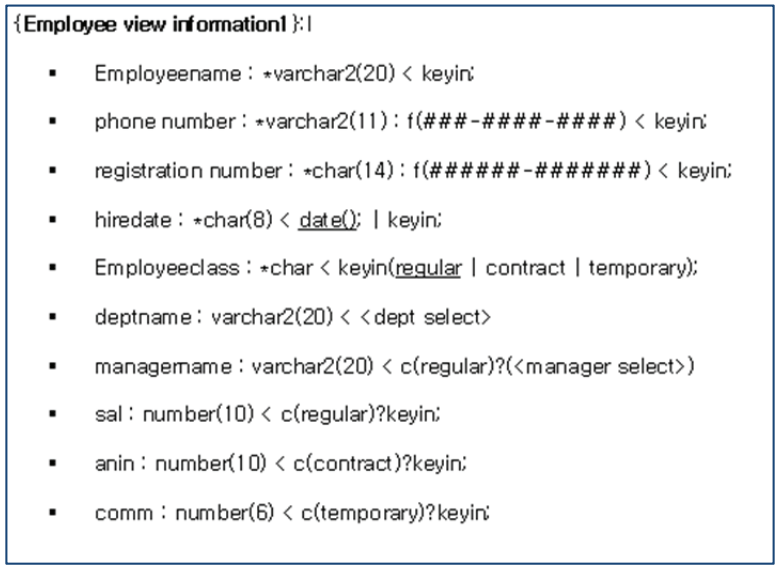


Fig. 4. Data expressed on the employee registration screen.

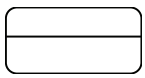
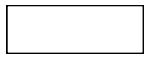
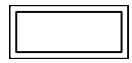
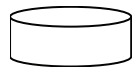
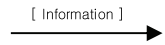
The data of the screen is represented by linking the database's information in terms of data expressed on the screen considering the relation with the data architecture from the enterprise perspective. Fig. 4 shows the "employee view information 1" information represented on the screen. The data name, size, type, etc. are created the same as the physical database's information, and the condition or format representing the data is separated with the "f()" symbol to indicate together. The data of the screen is divided into two types the data whose value is entered by users and the data received from the database or another window or screen and the information entered by users is specified with a "Keyin" after the data information. If the relevant data is not the user's data, it indicates the source of the relevant data.

3.2 DFD Design

The DFD shows the overall flow of a screen for each business, which expresses the screen design contents for each business in contracted form. The DFD shows the order of executing business and element processes of the relevant business on the screen for each business unit and briefly indicates the data transferred when moving between screens and the data displayed on the screen.

A rounded rectangle is divided by a line to enter the screen name at the top and indicate the screen contents displayed on the screen at the bottom; if there are many screen contents, they are contracted for entry. In addition, a color is added to express the figure if only the administrator is allowed to access the screen. A rectangle means an internal function, and the internal function name is entered. When expressing an internal function, the function name and the passed transfer value are entered together. A double rectangle is a system function indicating a case wherein the system creates and changes data on its own. When accessing a database to import or record data, the database is represented by a magnetic disk model. For database access, the database query statement is briefly expressed with a pseudo-code for developers' easy understanding later. The movement of information or screens caused by an event is expressed by an arrow, and the object generating the data or event is indicated on the arrow (Table 2).

Table 2. Representation of objects in DFD

	Screen or Window	Internal function	System function	Database	Event
Expression elements					

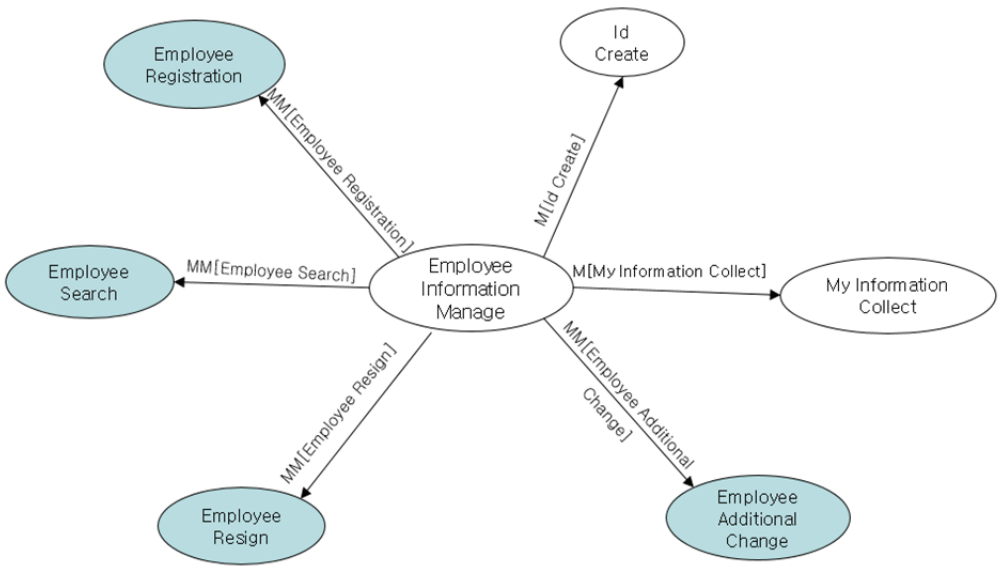


Fig. 5. Comprehensive DFD of employee management.

Fig. 5 shows the overall data flow of the business and element processes derived from the "employee information management" business, which expresses a comprehensive data flow if displaying all the data on a screen is difficult; the lower data flow is expressed as Fig. 6.

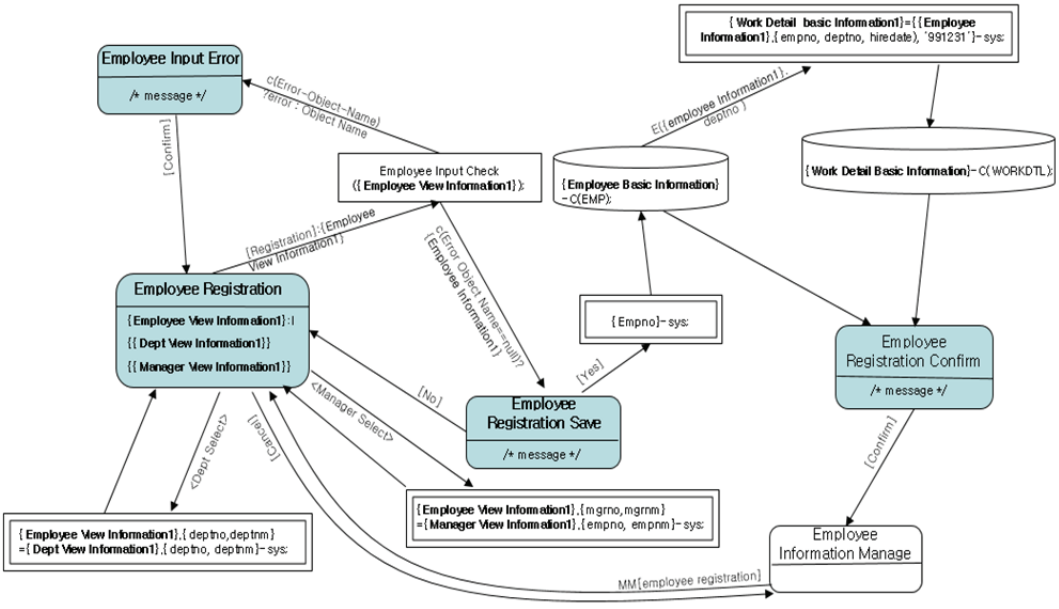


Fig. 6. Employee registration DFD of employee management.

4. Comparison with Other Studies

In this chapter, other methods that have been studied to design an application successfully are compared with the method proposed in this study. The comparison component compared the elements that should be expressed for application design (Table 3).

Table 3. Comparison with other studies

	Use case scenario	User story	Data flow diagram	This study
Consideration from the perspective of enterprise architecture	△	△	△	O
Analyzing hierarchical classifications	△	X	O	△
Expression of I/O data	O	O	O	O
Expression of data movement	O	△	O	O
Expression of database access	O	O	△	O
Expression of function usage	O	O	△	O
Expression of multidimensionality	O	X	O	X

O=apply, △=partial apply, X=not apply.

5. Conclusion

Developing an information system successfully requires designing and developing it from the enterprise perspective as well as being able to express fully the business in the application architecture based on the derived business.

This study derived the process procedure for applications to perform the business based on the defined business, presented the prototyped screens based on such, and proposed how the process defined through data representation and transfer on the screen is converted. The use of this method is expected to reduce the cost required for redevelopment in advance by reviewing the client's requirements and minimize possible problems through efficient communication between designers and developers with diverse experiences. In addition, persons who lack design experiences could also carry out the application design effectively.

In the future, a study should be done on the class design that derives objects for application development based on the screen design presented by this study.

Acknowledgement

This work was supported by a Research Grant of Pukyong National University (2017 year).

References

[1] B. S. Park, K. S. Yang, and H. S. Kim, "A study on the risk factors for successful enterprise architecture implementation," *Journal of the Korea Society of IT Services*, vol. 5, no. 3, pp. 1-23, 2006.

[2] C. Batini, E. Nardelli, and R. Tamassia, "A layout algorithm for data flow diagrams," *IEEE Transactions on Software Engineering*, vol. 12, no. 4, pp. 538-546, 1986.

[3] M. Tsikerdekis, "Persistent code contribution: a ranking algorithm for code contribution in crowdsourced software," *Empirical Software Engineering*, vol. 23, no. 4, pp. 1871-1894, 2018.

[4] N. A. Panayiotou, S. P. Gayialis, N. P. Evangelopoulos, and P. K. Katimertzoglou, "A business process modeling-enabled requirements engineering framework for ERP implementation," *Business Process Management Journal*, vol. 21, no. 3, pp. 628-664, 2015.

[5] Korea Database Agency, *The Guide for Data Architecture Professional*. Seoul: Korea Database Agency, 2013.

[6] J. H. Huh and K Seo, "Design and test bed experiments of server operation system using virtualization technology," *Human-centric Computing and Information Sciences*, vol. 6, no. 1, pp. 1-21, 2016.

[7] J. H. Huh, "PLC-based design of monitoring system for ICT-integrated vertical fish farm," *Human-centric Computing and Information Sciences*, vol. 7, no. 20, pp. 1-19, 2017.

[8] S. K. Yun, M. G. Park, and Y. J. Choi, "Automatic extraction of abstract components for supporting model-driven development of components," *KIPS Transactions on Software and Data Engineering*, vol. 2, no. 8, pp. 543-554, 2013.

[9] H. J. Jung, "A proposal of software quality measurement model and testcase on the basis of ISO/IEC 25010," *Journal of Korean Institute of Information Technology*, vol. 9, no. 10, pp. 197-205, 2011.

[10] D. H. Na and K. W. Jung, "UML-based enterprise architecture framework," in *Proceedings of E-Biz World Conference*, 2003, pp. 589-599.

[11] S. Jung, D. Lee, E. Kim, C. Chang, and J. Yoo, "OOPT: an object-oriented development methodology for software engineering education," *Journal of KIISE*, vol. 44, no. 5, pp. 510-521, 2017.

[12] C. Lee and C. Youn, "Dynamic impact analysis method using use-case and UML models on object-oriented analysis," *Journal of KIISE*, vol. 43, no. 10, pp. 1104-1114, 2016.

[13] M. Rahimi and J. Cleland-Huang, "Evolving software trace links between requirements and source code," *Empirical Software Engineering*, vol. 23, no. 4, pp. 2198-2231, 2017.

[14] S. Mirri and P. Salomoni, "Collaborative design of software applications: the role of users," *Human-centric Computing and Information Sciences*, vol. 8, article no. 6, 2018.
[15] M. Abi-Antoun, D. Wang, and P. Torr, "Checking threat modeling data flow diagrams for implementation conformance and security," in *Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering*, Atlanta, GA, 2007, pp. 393-396.
[16] S. Y. Lee and H. S. Yong, "Distributed development and evaluation of software using agile techniques," *The KIPS Transactions: Part D*, vol. 16, no. 4, pp. 549-560, 2009.
[17] J. Yeo, S. Park, and J. Myoung, *Useful Database Oracle Center in Practice*. En-core, 2016
[18] S. Y. Park, T. W. Kim and J. M. Yeo, "Study on the business process design method for designing applications," in *Proceedings of the 12th KIPS International Conference on Ubiquitous Information Technologies and Applications (CUTE)*, Taichung, Taiwan, 2017.
[19] M. Adler, "An algebra for data flow diagram process decomposition," *IEEE Transactions on Software Engineering*, vol. 14, no. 2, pp. 169-183, 1988.
[20] G. S. Kwuan, "Process methodology of designing user interface in enterprise portal," *Journal of the HCI Society of Korea*, pp. 1160-1166, 2008.

Tae-Woo Kim https://orcid.org/0000-0002-0819-0174

He received B.S. and M.S. degrees in School of Computer Engineering from Pukyong National University in 2015 and 2017, respectively. Since March 2017, he is with the School of Computer Engineering from Pukyong National University as a PhD candidate.

Sun-Yi Park https://orcid.org/0000-0002-4391-6479

She received B.S. degrees in School of department of electronic computing from pukyong National University in 1991 and received M.S. degrees in School of Computer Science and Engineering from Busan University of Foreign Studies in 1996 and Ph.D. degrees in School of Electronic Commerce Cooperation Process from Pukyong National University in 2012. She is with the School of Computer Engineering from Pukyong National University as Adj. Professor.

Jeong-Mo Yeo https://orcid.org/0000-0002-4873-5669

He received B.S. degree in School of the Department of Electronic Engineering from Donga University in 1980, M.S. degree in School of the Department of Electronic Engineering from Busan National University in 1982, and Ph.D. degrees in School of Computer Science and Electronic Engineering from Ulsan University in 1993. Since March 1986, he is with the School of Computer Engineering from Pukyong National University as Professor.