Regular paper
Journal of the KIECS. pp. 167-172, vol. 15, no. 1, Feb. 29. 2020, t. 99, pISSN 1975-8170 | eISSN 2288-2189
http://dx.doi.org/10.13067/JKIECS.2020.15.1.167

큰 규모 소프트웨어의 개발을 위한 모델링 기법의 방향성에 대한 연구

조민호*

A Study on the Direction of Modeling Techniques for the Development of Large Scale Software

Min-Ho Cho*

요 약

큰 규모의 소프트웨어 개발에서 모델링의 중요성은 이미 확인된 사항이다. 그리고 대부분의 프로젝트에서 모델링은 UML을 활용하여 수행되고 있다. 하지만 UML은 클래스를 중심으로 설계된 모델링 도구로서 공통모듈이나 컴포넌트, 프로세스 그리고 데이터의 모델링을 수행하기에는 복잡하고 기능이 부족하다. 이런 점을 보완하고자 컴퓨터 분야에서 개발되었던 구조적 방법론, 정보공학 방법론을 포함한 다양한 모델링 기법을 통합해서 사용하는 방안을 제시함으로써 실제 산업 현장에 도움이 되고자 한다. 이번 논문을 통하여 고객이 만족하는 소프트웨어의 개발을 위해서는 UML만 사용하지 않고 다양한 방법론의 기법을 사용하는 것이 유용함을 보이고자 한다.

ABSTRACT

The importance of modeling in large-scale software development has already been confirmed. In most of the projects, modeling is done using UML. However, UML is a class-based modeling tool, which is complicated and lacks in modeling common modules, components, processes, and data. To supplement this point, this paper will help the actual industrial field by suggesting the integration of various modeling techniques, including structural methodology and information engineering methodology developed in the computer field. Through this paper, I would like to show that it is useful to use various methodologies instead of using UML to develop software that satisfies customers.

키워드

software modeling, methodology, Information Engineering, UML, structural methodology
소프트웨어 모델링, 방법론, 정보 공학, UML, 구조적 방법론

Ⅰ. 모델링의 개념 및 분류

소프트웨어가 비즈니스에 중요한 환경으로 인식되면서, 소프트웨어는 많은 기능을 가지도록 대형화되고, 장기간에 걸쳐 사용되며, 지속적인 수정과 기능 개선이 이루어지고 있다[1]. 이런 소프트웨어는 많은 개발자가 오랜 기간에 걸쳐 개발해야 하는 특성이 있으며, 기능의 복잡성으로 인하여 하나의 언어가 아닌, 여러 관련 제품들이 통합되어 개발되는 특징을 보이고 있다[2]. 이런 환경에서 개발된 소프트웨어의 품질

* 교신저자 : 중원대학교 컴퓨터공학과
• 접 수 일 : 2019. 12. 17
• 수정완료일 : 2020. 01. 16
• 게재확정일 : 2020. 02. 15
• Received : Dec. 17, 2019, Revised : Jan. 16, 2020, Accepted : Feb. 15, 2020
• Corresponding Author : Min-Ho Cho
Dept. Computer System Engineering, JungWon University,
Email : chominhokr@jwu.ac.kr

을 보증하고 지속적인 관리를 위해서는 소프트웨어의 구성에 대한 관심이 좀 더 필요하다[3]. 이런 환경의 변화를 반영하여 최근에는 소프트웨어 모델링 분야가 실제적인 코딩보다 더욱 중요한 것으로 생각되고 있다[3].

소프트웨어 모델링은 그림 1과 같이 정의하고 요약할 수 있다. 즉, 소프트웨어를 원하는 관점에서 표현하는 것을 모델링이라고 한다. 모델링은 논리적 모델링과 물리적 모델링으로 나누어 생각할 수 있다.

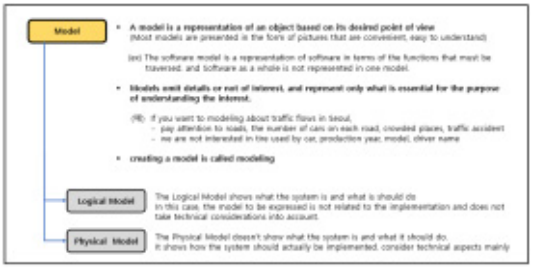

그림 1. 모델링의 개념 및 분류
Fig. 1 Modeling concepts and classification

모델링을 수행함에 있어 지켜야 하는 기본적인 원칙은 그림 2에 정리하였다. 요약하면 모델링은 추상화되고 정형화되어야 하며 Top-Down으로 계층적 분류를 통하여 단위 작업의 수준까지 분할하여야 하며, 실제 수행은 관련 도구(프로그램)를 사용한다.

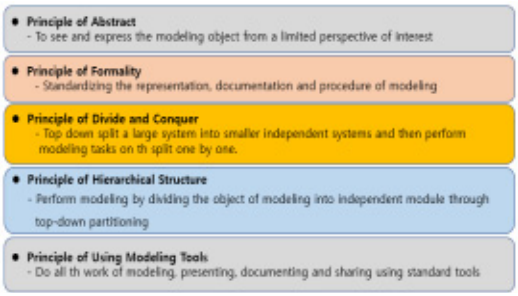

그림 2. 모델링의 기본 원칙
Fig. 2 Basic principles of modeling

소프트웨어의 모델링은 기본적으로 3가지 요소를 가지고 구성된다. 규약(Convention), 표현(Representation), 명세(Specification)가 모델링의 3요소이다[4]. 그림 3에 이에 대하여 요약하였다.

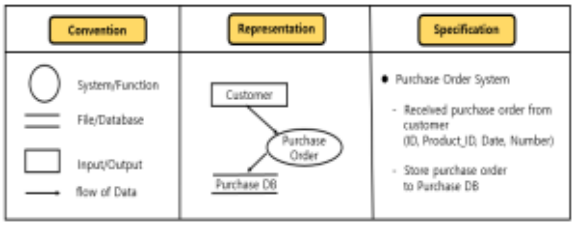

그림 3. 소프트웨어 모델링의 3요소
Fig. 3 3 element of software modeling

그림 3은 구조적 모델링을 대상으로 모델링의 3요소에 해당하는 예를 제시하고 있다. 마지막으로 소프트웨어 모델링을 수행하기 위해 필요한 관점을 정리하면 “기능관점”, “정보관점”, “동적관점”의 3개로 분리할 수 있고, 이것은 각각 기능모델링(Functional Modeling), 정보모델링(Information Modeling), 동적모델링(Dynamic Modeling)으로 표현 할 수 있다[1].

II. 컴퓨터 환경의 발전과 방법론의 관계

모델링 기법에 대한 평가를 위해서는 컴퓨터의 발전 역사와 각 시대에 따른 모델링 기법에 대하여 정리할 필요가 있다. 그림 4는 컴퓨터의 발전과 모델링 기법의 개발 과정을 정리한 것이다. 참고로 모델링을 수행을 목적으로 구체적으로 정의한 것이 방법론(methodology)이다.

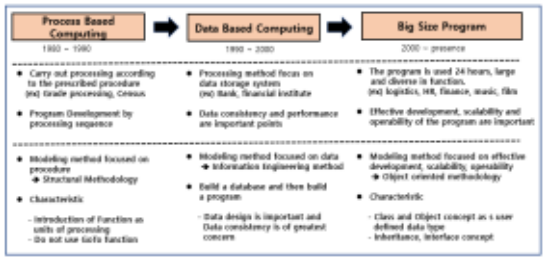

그림 4. 모델링 기법의 역사
Fig. 4 History of Modeling method

그림 4에서 나타내고자 하는 것은 모델링을 수행하기 위하여 개발된 방법론이 컴퓨터의 발전 단계와 맥락을 같이하며 발전해 왔다는 것이다.

구조적 방법론은 컴퓨터가 단위 작업의 연산에 사용되는 시절에 개발된 모델링 방법론이고[4], 이후에

컴퓨터가 데이터를 다루는 능력을 가지게 되면서 일반 비즈니스에 사용하기 위하여 개발된 것이 정보공학 방법론이다[5]. 이후에 개발해야 하는 프로그램이 커지고, 여러 명이 오랜 시간 개발해야 하는 환경이 도래하면서, 프로그램 작성의 편의성을 위하여 개발된 것이 객체지향 방법론인 것이다[6][7][10].

그러므로 특정 방법론은 특정 환경을 위하여 개발된 것임을 기억해야 한다. 비록 UML이 나중에 개발되어서 앞에서 개발된 방법론을 참고하였기 때문에 그들의 기능을 많이 포함하였다고 해도 특정 상황에서는 특정 환경을 위해 개발된 방법론의 유용성에는 미치지 못하는 경우가 많다. 이번 연구는 이런 상황을 전제로 해서 현재의 시점에서 어떤 방법론을 어떤 환경에서 어떻게 적용하는 것이 바람직한가에 대하여 논하고자 한다[8]. 그림 5에 앞에서 설명한 내용을 정리하였다

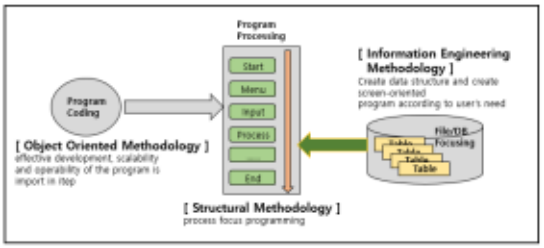

그림 5. 방법론의 적용 범위
Fig. 5 Scope of methodology

III. 소프트웨어 모델링을 위한 가이드

그림 5에서 정리한 것과 같이 오늘날의 소프트웨어는 업무처리를 위한 절차가 많고, 다루어야 하는 데이터의 종류도 많고, 데이터와 연관된 메서드(함수)도 많아서 개발자에게 어려운 점이 있다. 더군다나 한번 개발된 소프트웨어가 비즈니스의 중심으로 사용되며 기능의 확장과 보완, 개선이 지속적으로 발생하는 특징을 가진다. 이런 환경에 효과적으로 대처하기 위해서는 개발해야 하는 소프트웨어의 모델링에 보다 많은 노력을 기울여야 할 필요가 있다. 소프트웨어의 모델링을 위해 고려하여야 하는 관점을 그림 6과 같이 정리하였다[11].

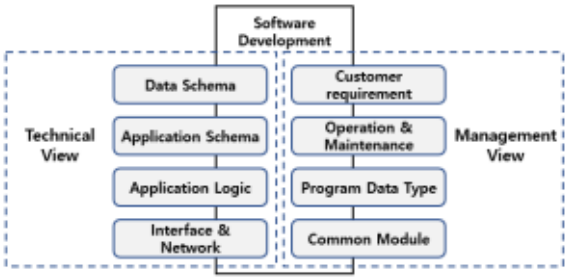

그림 6. 소프트웨어 개발에서 고려할 요소
Fig. 6 Factors to consider in software development

그림 6은 현대 소프트웨어의 환경에서 개발할 때 고려해야 하는 요소를 정리하고 있다. 데이터 설계는 프로그램이 사용해야 하는 데이터의 종류와 구조에 대한 부분을 담당하는 부분으로 대부분의 경우에 데이터베이스관리시스템(DBMS)을 사용하게 된다. 구조설계 부분은 개발해야 하는 프로그램을 기능을 중심으로 Top-Down으로 분리하여 소프트웨어의 전체 구조를 정의하는 역할을 한다. 프로시저 설계는 구조 설계에서 식별된 단위 모듈의 기능과 다른 단위 모듈과의 관계, 데이터베이스와의 관계를 정리하는 부분이다. 인터페이스설계는 사용자가 접하는 화면의 구성에 대한 부분을 정리하는 부분이다. 이상과 같이 데이터 설계, 구조설계, 프로시저 설계, 인터페이스 설계는 개발할 소프트웨어의 기술적 혹은 기능적 부분을 담당하게 된다.

다음으로 고려해야 하는 것은 개발할 소프트웨어의 관리적 시각이다. 그림 6의 요구사항은 개발할 소프트웨어에 대한 사용자의 요구사항을 파악, 정리, 합의하는 과정을 말하는 것이다. 유지/보수는 개발된 소프트웨어의 관리 및 기능 확장을 위해 고려해야 하는 요소에 대하여 정리하는 부분이다. 프로그램 자료형은 개발할 프로그램의 규모가 크기 때문에 데이터와 메서드의 종류가 많아지므로 이것을 효과적으로 관리하기 위하여 사용자정의 자료형을 선언해서 사용하는 방법에 대한 것이다. 과거에는 배열과 구조체를 주로 이용하였지만, 최근에는 클래스를 사용하는 경우가 많다. 마지막으로 공통모듈은 컴포넌트기반개발(Component Based Development)과 관점지향 프로그래밍(Aspect Oriented Programming)에서 다루는 것으로 개발할 소프트웨어의 기능 중에서 여러 곳에서 반복적으로 사용되는 경우에 미리 개발하여 개발자가

공유함으로써 소프트웨어의 관리와 기능 확장, 유지보수에 드는 노력을 줄일 수 있는 것이다. 대표적인 경우가 데이터베이스관리 시스템을 연결하는 부분 및 화면의 공통부분이다

그림 6에서 정리한 부분들이 소프트웨어를 모델링하기 위하여 고려해야 하는 부분이다. 그러면, 이들을 효과적으로 모델링해서 원하는 소프트웨어를 개발하기 위해서는 어떤 모델링 방법이 적당한가에 대하여 생각해 보자.

그림 6에서 개발할 소프트웨어의 모델링에 고려해야 하는 부분에 대한 정리가 마무리 되었다. 이제는 각 모델링 부분에서 사용할 수 있는 모델링 방법을 정하고자 한다. 이때, 특정 방법론에 치우친 것이 아닌, 지금까지 개발된 많은 방법론의 기법들 중에서 유용한 기법들을 모아서 종합적으로 고려하여 개발할 소프트웨어의 구성 부분과 어떻게 연관이 되는지를 그림 7에 정리하였다.

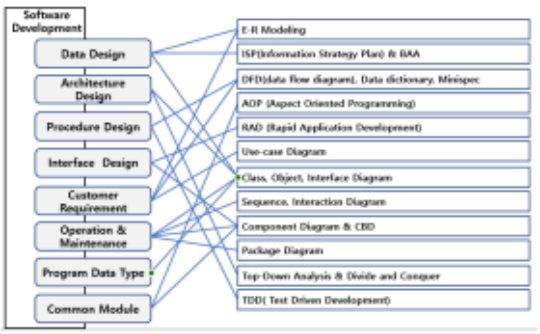

그림 7. 소프트웨어 개발요소와 기법의 연관관계
Fig. 7 Relationship between software development factors and techniques

그림 7은 소프트웨어 개발의 발전 과정에서 개발되었던 구조적방법론, 정보공학방법론, 객체 지향 방법론 외에도 XP(eXtream Programming), Agile과 같은 경량 객체지향 방법론[9]와 스프링 프레임워크에서 제공하는 관점지향 프로그래밍 기법 그리고 테스트 중심의 개발을 강조하는 테스트중심 개발과 개발된 소프트웨어의 소스를 기능의 변화 없이 정리하는 리펙토링에서 사용되는 기법을 모아서 소프트웨어 모델링의 각 단계에 매칭시킨 것이다.

오늘의 소프트웨어는 프로그램 제작, 모듈 구성이나 데이터 관리, 프로그램을 구성하는 자료형의 정의 등에 특정 언어나 제품, 기법만을 사용하여 개발할 수 없다. 이유는 이미 설명한 바와 같이 우리가 다루는 소프트웨어가 대규모, 많은 인원이 오랫동안 개발해야 하며 지속적으로 기능의 개선되는 추가되면서 발전해 나가야 하기 때문이다. 이런 이유로 특정 환경을 고려하여 개발된 방법론에 의존하는 것은 좋은 결과를 얻기 어렵다. 그런 점에서 그림 7은 다양한 방법론의 기법들을 모아서 개발하고자 하는 소프트웨어의 특정 부분에 맞는 것을 매핑한 것으로 실제 사용하는 경우 좋은 결과를 기대할 수 있다.

이해를 위하여 그림 7의 내용에 대해 보충설명을 하면 다음과 같다.

데이터설계 부분은 데이터베이스에 관련된 부분은 기존에 사용하는 객체관계 모델링과 정규화를 그대로 적용하므로 큰 문제가 없을 것이다. 다만, 정보공학 방법의 비즈니스영역분석(BAA)과 정보전략계획(ISP) 개념을 추가로 적용한다면 프로세스와 데이터의 연관관계에 대한 명확한 설정이 가능하므로 필요한 데이터를 정확하게 식별하여 비즈니스에 도움이 되는 데이터베이스의 구성이 가능하다. 데이터설계 부분에서 프로그램에서 다루는 데이터 설계 부분은 동시에 취급해야 하는 데이터와 메서드를 묶어서 처리하기 위하여 클래스 다이어그램을 제작하고 상속과 인터페이스를 이용하여 프로그램이 사용하는 데이터의 트리구조를 구성하여 개발자가 공유하는 과정이 필요하다. 이것을 통하여 프로그램의 제작이 쉬워지고 읽기 쉬운 코드의 작성이 가능하며, 유지보수가 편리한 환경을 만들 수 있다.

구조설계부분은 이미 기존에 사용하던 것과 동일하게 탑-다운 분석, 분할정복 기법을 적용하지만 대규모 프로그램을 제작한다는 전제가 있으므로 프로그램에서 필요한 사용자 정의 자료형을 선언해서 사용하는 부분을 보완해야 한다.

프로시저설계부분은 UML의 어떠한 방법으로도 쉽고 정확하게 표현하기 어렵다. 유스케이스 다이어그램의 확장 기능으로 정리하면 너무 복잡하게 표현되는 단점이 있고, 유스케이스를 적용하여 글로 정리하면 전체를 한 번에 파악하기 어려운 문제가 있다. 더구나, 구조설계를 통해 식별된 단위 기능이 상호 어떻게 연결되는 지에 대하여 정확하게 표현하는 것이 중요한데, 이런 점에 대한 표현이 UML에서 명확하지 않

다. 그래서 비록 오래전에 개발된 방법이기는 하지만 DFD를 이용하면 모듈자체의 기능 정의 외에도 다른 모듈, DBMS와의 관계를 쉽게 표현할 수 있으며, 추가로 서술해야 하는 상세 내용은 미니스펙을 이용하여 유스케이스와 동일하게 서술할 수 있다. 설명에 대한 이해를 위하여 DFD를 이용하여 작성된 예를 그림 8에 제시한다.

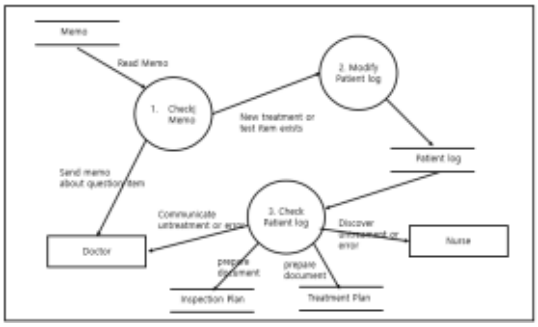

그림 8. DFD의 제작 사례
Fig. 8 DFD example

그림 8은 단위 모듈을 식별하고 있으며(“도서주문접수”, “도서주문처리”, “도서대금청구”), 외부 시스템(사용자, 다른 시스템)과의 관계 그리고 데이터베이스와의 처리 절차를 쉽게 파악할 수 있도록 해준다. 그런 점에서 객체지향프로그램의 개발에서 단위 모듈의 식별은 DFD를 사용하는 것도 좋은 방법이라고 생각한다.

인터페이스 설계는 RAD기법을 적용하여 개발해야 하는 프로그램의 프로세스 중에서 핵심이 되는 프로세스를 대상으로 화면을 구성하고 화면과 화면의 연결을 보여줌으로써 사용자의 요구사항과 개발될 시스템에 대한 기대치를 조정할 수 있다. 현재 사용하고 있는 UML에서는 이런 부분에 대한 고려가 되어 있지 않으므로 개발될 시스템에 대한 기대치를 조정하고 숨어있는 사용자의 요구사항을 발견하기 위하여 RAD 기법을 사용하는 것이 효과적이라고 생각한다.

요구사항 부분은 객체지향 방법론에서 사용하는 유스케이스 다이어그램 기법이 유용하다고 생각하며, XP, Agile에서 사용하는 사용자스토리기법을 추가하면 좀 더 좋은 결과를 얻을 수 있을 것이다.

유지보수부분은 현대의 소프트웨어 개발에서 가장 중요한 부분으로 공통적인 부분에 대한 개발과 공유 그리고 개발하기 전에 테스트 계획을 세우고 개발하는 자세가 필요하다. 추가로 최근의 소프트웨어는 한 개의 언어로 개발되는 경우가 거의 없고 다양한 라이브러리와 외부 자원을 연계하여 개발하므로 이들의 버전 관리와 설치 제어를 위한 별도의 과정이 필요하다. 그 외에 프로그램 자료형은 객체지향 프로그램의 존재 이유이므로 클래스/객체 다이어그램을 통하여 프로그램에서 필요한 자료형을 선언하고 공통자원으로 활용하는 것이 필요하다. 공통모듈은 이미 설명한 컴포넌트기반개발과 관점지향 프로그램의 내용을 참고하면 된다.

Ⅳ. 결론

소프트웨어를 개발함에 있어서 모델링은 개발할 소프트웨어의 가치를 높이고 비즈니스에 도움에 되도록 하기 위한 필수적인 과정이다. 그래서 모델링을 정형화한 방법론은 컴퓨터의 발전과 함께 지속적으로 개발되고 적용되어 왔다. 구조적 방법론에서 정보공학 방법론, 객체지향 방법론 그리고 컴포넌트기반개발, Agile/XP 방법론에 이르기까지 다양한 방법론이 개발되고 적용되었다. 그런데, 현재 시점에서 소프트웨어를 개발하는 대부분의 프로젝트는 UML을 활용하는 것으로 인식되고 있다. 하지만, UML은 객체지향 환경에는 유용한 기법이지만 프로그램 개발에 필요한 모든 분야에서 유용한 방법을 제공하는 것은 아니다.

이번 연구에서는 소프트웨어 개발을 위해 필요한 모델링이 무엇이며, 어떻게 발전되어 왔는지에 대해 요약한 후에, 오늘날의 소프트웨어가 가지는 특성을 정리하였다. 즉, 여러 명의 개발자가 오랜 기간 개발하며, 개발된 소프트웨어는 비즈니스의 핵심으로 오랜 기간 사용되며 지속적으로 확장/개선되어야 하는 특징을 제시하였다. 이제 이러한 특징을 가지는 소프트웨어의 모델링에 대한 8가지 관점을 그림 6에 제시하고 각 관점별로 가장 유용한 방법들을 선정하여 연결시켜보았다. 그림 7 참조.

이번 연구는 현대의 소프트웨어 환경이 특정 모델링 기술에만 의존하는 것이 바람직하지 않다는 전제하에 필요한 기능별로 바람직한 방법을 찾아서 매칭시킴으로써 실무에서 모델링을 수행하는 담당자들에

게 도움이 되는 정보를 제공하고자 한다. 지면 관계상 각 방법에 대한 상세한 설명을 제시하지 못한 점은 아쉽지만, 개발자나 설계자의 관점이 아닌 고객이 만족하는 소프트웨어의 개발이라는 관점에서 모델링을 바라보는 시각을 가지게 되기를 바란다.

감사의 글

이 논문은 "2019년 가을철학술대회 우수논문" 입니다.

References

[1] M. Despa, "Comparative study on software development methodologies," *Database Systems Journal*, vol. 5, no. 3, Apr. 2014, pp. 37-56.

[2] W. Royce, "Managing the development of large software systems : Concepts and techniques," *IEEE WESCON*, vol. 26, no. 8, 1970, pp. 1-9.

[3] S. Woodward, "Evolutionary project management," *Computer*, vol. 32, no. 10, 1999, pp. 39-57.

[4] A. Y. Aleryani, "Comparative Study between Data Flow Diagram and Use Case Diagram," *Int. J. of Scientific and Research Publications*, vol. 6, issue. 3, Mar. 2016. pp. 124-127.

[5] A. Badio, "Entity-Relationship modeling revisited," *ACM SIGMOD Record*, vol. 33, no. 1, Mar. 2004, pp. 77-82.

[6] M. Kellner, R. Madachy, and D. Raffo, "Software Process Simulation Modeling : Why? What? How?," *J. of Systems and Software*, vol. 46, no. 2/3, Apr. 1999, pp. 91-105.

[7] B. G. Song and Y. S. Yu, "A Design and Impelmentation of Software Architecture for IPC in Vehicles Using Modeling Methodology," *J. of the Korea Institute of Electronic Communication Sciences*, vol. 7, no. 6, 2012, pp. 1567-1572.

[8] Z. Chi-long, T. Hong-lei and S. Wei, "Integrated Software Engineering Methodology," *2009 International Forum on Information Technology and Applications*, Chengdu. China, May 2009, pp. 694-697.

[9] R. Fojtik, "Extreme Programming in development of specific software," *Procedia Computer Science*, vol. 3, 2011, pp. 1464-1468.

[10] C. B. Shim, S. H. Jung, and K. J. Kim, "Object - Oriented Modeling based on UML for Integrated Manufacturing Management System using Web," *J. of the Korea Institute of Electronic Communication Sciences*, vol. 5, no. 6, Dec. 2010, pp. 602-612.

[11] B. Song and Y. Yu, "A Design and Implementation of Software Architecture for IPC in Vehicles Using Modeling Methodology," *J. of the Korea Institute of Electronic Communication Sciences*, vol. 7, no. 6, Dec. 2012, pp. 1567-1574.

저자 소개

조민호(Min-Ho CHO)

1989년 인하대학교 산업공학과 졸업(공학사)

HP Korea, Openwave, SK C&C등 산업체 근무

2003년 숭실대학교 대학원 컴퓨터공학과 졸업(공학박사)

2012년~ 중원대학교 컴퓨터공학과 교수

※ 관심분야 : 소셜네트웍, 소프트웨어공학, 데이터마이닝 및 빅데이터, 머신러닝