Journal of Digital Convergence
Vol. 17. No. 12, pp. 229-234, 2019

ISSN 1738-1916
https://doi.org/10.14400/JDC.2019.17.12.229

클라우드 모델 기반의 자산 효율성 평가

최한용
신한대학교 IT융합공학부 컴퓨터공학전공 교수

Cloud Model based Efficiency Evaluation of Asset

Hanyong Choi
Professor, Division of IT Convergence Engineering, Shinhan University

요 약 소프트웨어 시장은 모바일 시장 확장으로 다품종의 다변화된 서비스 요구를 갖고 있다. 이를 위해 도메인 시장의 정형화된 아키텍처 자산을 기반으로 설계 도메인으로 확장하여 다양한 앱을 생산하고자 한다. 아키텍처 설계 정보는 클라우드를 기반으로 자산을 등록하고 관리할 수 있는 방법을 연구 중이다. 그리고 이와 같은 자산 관리 소프트웨어 시스템은 효율성을 평가하기 위한 방법이 필요하다. 본 연구에서는 선행된 자산관리 시스템을 클라우드 기반으로 서비스하기 위해 아키텍처 자산의 효율성을 평가하기 위한 평가모델을 제안하고자 한다. 아키텍처 자산의 효율성을 평가하기 위해 아키텍처 자산의 도메인 특성 값을 반영하기 위한 방법을 사용 하였다. 효율성 평가는 SQuaRE Series의 ISO/IEC 25010 표준을 기반으로 클라우드 기반의 자산데이터의 효율성을 평가하기 위한 평가모델 부 특성을 구성하였다. 아키텍처 자산은 복합자산으로 구성되어 설계되었을 때 설계 도메인의 특성에 따라 평가 항목의 가중치를 부여한 부 특성의 필수적 평가요소와 선택적 평가요소를 적용하도록 하여 평가모델의 설계 유연성을 제공하도록 하였다.

주제어 : 자산, 품질평가, 효율성, 아키텍처, 클라우드

Abstract The software market has diversified service needs due to the expansion of the mobile market. To this end, the company intends to produce various apps by extending to the design domain based on the structured architectural assets of the domain market. In this study, we propose an evaluation model that can evaluate the efficiency for servicing assets that reflect the domain characteristics of architecture based on cloud. Based on the characteristics of ISO/IEC 25010 quality model of SQuaRE Series, a software evaluation standard, evaluation model sub-features for evaluating the efficiency of cloud-based asset data were constructed. When the architectural assets were designed as composite assets, they were designed to provide the flexibility of the evaluation model by applying the mandatory and optional evaluation elements of the sub-features that weighted the evaluation items according to the characteristics of the design domain.

Key Words : Asset, Quality Evaluation, Efficiency, Architecture, Cloud

1. 서론

소프트웨어 시장은 특정 산업 영역을 지원하기 위한 시장에서 새로운 산업구조를 설계하는 영역으로 진화하고 있다. 이는 생산방식의 변화요구만이 아니라 생산의 효율성을 고려해야만 하는 상황이다. 이를 해결하기 위한 방안으로 오픈소스를 기반으로 플랫폼을 개방하고 플랫폼을 활용하여 다양한 영역에 효과적으로 시스템을 적용

*This work was supported by the Shinhan University Research Fund, 2019.
*Corresponding Author : Han-Yong Choi(hychoi@shinhan.ac.kr)
Received October 22, 2019 Revised November 23, 2019
Accepted December 20, 2019 Published December 28, 2019

하려 하고 있다[1]. 그리고 다양한 요구사항은 기술적으로 수용해야 하며, 시스템 생산성 과정에서 효율성을 증가시켜 생산하기 위한 방법에 많은 어려움을 겪고 있다. 이를 위해 서비스의 일반화에 따라 기존 시스템 또는 자원의 효과적인 사용을 위해 많은 방법이 연구되고 개발되어 사용되고 있다. 그러나 분산 환경에서의 상호 연결에는 여러 가지 어려움과 문제점을 갖고 있다. 설계자들의 설계정보에 대한 고려보다는 물리적 네트워크 환경에 대한 요소를 중심으로 하고 있다. 시스템 개발 단계에서 추상화가 잘 이루어진 기본적인 자산의 활용을 위해 클라우드 모델을 기반으로 설계정보를 재사용할 수 있도록 하고자 한다.

그러나 다양한 요구를 반영하는 소프트웨어들은 단일화된 평가방법을 적용하고 동일하게 평가기준을 제시하여 평가하였다. 그리고 클라우드를 기반으로 하는 아키텍처 자산의 설계에서는 표준화와 재사용성에 효율성을 고려하여 아키텍처 자산을 정제해야만 한다[2-4]. 따라서 클라우드 기반의 제품 라인에서 재사용할 수 있는 자산을 기반으로 표준 자산 컴포넌트로부터 자산을 재사용하고 이를 평가하기 위한 모델이 필요하다.

본 논문에서는 선행된 자산관리 시스템을 클라우드 기반으로 확장하였다. 클라우드 기반의 표준화된 설계 도메인의 아키텍처 자산은 표준 자산과 도메인 응용 자산으로 설계 도메인의 유연성을 제공하기 위한 평가모델을 구성하였다. 소프트웨어 평가는 SQuaRE의 ISO/IEC 25010 표준을 클라우드 기반 평가 시스템에 반영하였고 효율성 영역에 반영하기 위한 부 특성을 기준으로 구축하였다. 아키텍처 자산은 설계 도메인의 특성을 반영하여 유연성 있는 설계를 반영하기 위해 각 자산의 특성에 따라 부 특성에 가중치를 부여한다. 부 특성은 필수적 요소와 선택적 요소에 따라 가중치를 갖고 평가에 적용하였다. 본 논문의 구성은 2절에서 자산 구성과 품질을 위한 기반연구, 3장에서는 효율성 평가구성 항목과 평가방법 그리고 4장에서는 결론에 대하여 기술하였다.

2. 기반연구

2.1 자산의 메타데이터와 구성 방법

자산 시스템은 기본적인 아키텍처를 구성하기 위해 메타모델을 이용하여 설계정보를 표현하였다. 메타 모델로 표현된 설계정보는 표준화된 아키텍처 자산과 도메인 확장 자산으로 구성하였다.

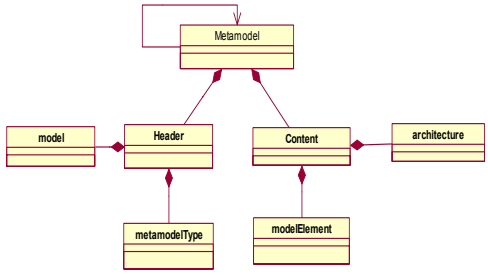

Fig. 1. Meta Model

Fig. 1은 표준화된 아키텍처의 설계정보를 구성하기 위한 XMI 메타모델이다[1,5]. 그리고 Fig. 2는 표준화 영역의 클라우드에서 아키텍처 자산을 표현한 메타모델을 이용하여 합성된다. 아키텍처 정보와 관계정보는 도메인 아키텍처 설계영역에 포함되고 설계자 응용도메인 아키텍처를 표현할 수 있도록 메타 모델을 정의하였다[1,5]. 사용자 특성을 반영한 응용 도메인에 대한 아키텍처를 설계하기 위해 자산을 재사용 하여 새로운 도메인 아키텍처를 구성한다.

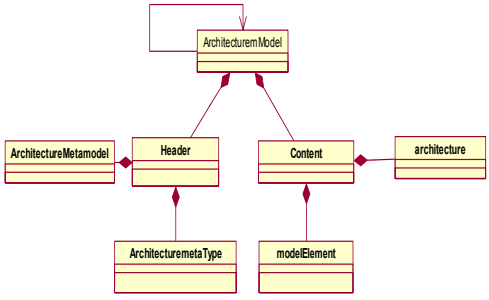

Fig. 2. Architecture Structure

기존 연구에서 사용하는 아키텍처는 공개표준 명세서인 RAS를 이용하여 명세하였다[6]. RAS 명세방식은 재사용의 일관성을 위하여 설계 자산들의 아키텍처, 내용 그리고 설명에 대한 내용을 나타내고 있다. 서로 다른 특성을 갖는 각 자산은 다음과 같이 명세하였다. 표준 영역의 아키텍처는 자산 명세를 하기 위한 필수적인 구성항목과 요소들을 포함하고 있다. 그리고 각 설계자의 의도를 반영한 도메인 자산은 자산의 특성에 대한 추가적 의미 때문에 Core RAS의 확장하는 기능으로 표현하였다. 기존 연구에서는 자산을 계층적으로 분류하여 설정하였고 아키텍처의 특성에 따라 분류되는 방법을 적용하였다.

2.2 자산의 복잡성과 품질

클라우드를 기반으로 하는 자산관리 시스템의 자산 복잡성은 논리적인 복잡도 측정을 제공하는 척도를 기반으로 한다[7]. 이때 각 자산의 특징 값을 표현하고 있는지 평가할 수 있다. 그래프 이론을 바탕으로 클라우드에서 제공하는 표준화된 아키텍처 영영의 일반적인 자산의 복잡도 측정을 할 수 있다. 그리고 설계자의 설계 의도가 반영된 아키텍처 자산은 아키텍처 하부에 설계된 표준 자산의 특성을 반영하여야 측정해야 하므로 이를 반영하여 전체 자산의 복잡성을 측정할 수 있다[1,5].

자산 연관 척도에 의해 자산 내부 항목에 있는 각 기본 자산의 요소와 다른 기본 자산 요소들 간의 연관성을 측정할 수 있다. 이 값은 표준 아키텍처 자산과 다른 아키텍처 자산이 공유하는 도메인 특성이 반영된 자산보다 많으면 척도 값이 커진다. 척도 값은 커진다는 것은 자산이 파생된 자산들과 연관성을 많이 갖고 모듈성이 떨어지는 특징을 가진다고 볼 수 있다.

소프트웨어의 품질을 평가하기 위한 방법 중 국제 표준은 ISO/IEC 25000 시리즈인 SQuaRE이다[8]. SQuaRE 시리즈는 ISO/IEC 9126 과 ISO/IEC 14598 시리즈가 소프트웨어 품질과 품질관리에 대한 국제 표준으로 서로 상호 보완적이다[9-11]. 그러나 이 두 가지 표준은 초기에 생명주기를 서로 다르게 나타내고 있어, 이 두 표준을 적용하여 사용하는데 불일치에 대한 어려움을 갖고 있다. 그래서 이 문제를 개선하기 위해서 SQuaRE 시리즈를 구성하여 표준으로 사용한다[12].

3. 자산의 효율성 평가

3.1 효율성 평가 모델

선행 연구에서 아키텍처 자산의 효율적 구성과 관리를 위해 표준화 영역을 중심으로 관리하고 자산의 효율성을 평가하기 위한 방법을 연구하였다. 그리고 이 시스템에서 사용하는 아키텍처 자산의 품질 평가를 위해 국제 표준을 이용하고자 한다.

본 연구에서는 Fig. 3에 제시된 바와 같이 ISO/IEC 25010에서 언급하고 있는 표준 구성안을 이용한다[7,9]. 아키텍처 자산을 평가하기 위해 ISO/IEC 기준에는 자산 기능을 충족시키는 능력과 관련된 소프트웨어의 특성과 특징으로 사용하고 있기 때문에 이 방법을 이용하여 아키텍처 자산의 평가모델을 적용하였다[13,14].

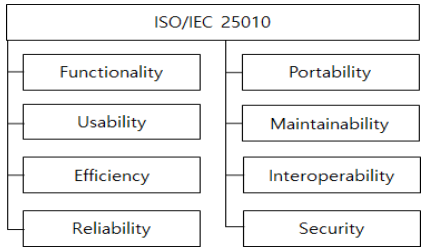

Fig. 3. ISO/IEC 25010 Configuration

이때 클라우드 기반의 아키텍처 모델은 정형화된 자산을 클라우드로 표준화하여 설계 요구자의 효율성과 상호 운용성을 만족 시키고 재사용 효율성을 갖게 된다. 그러므로 기존의 연구결과에 제시된 것처럼 복잡도를 감소시킬 수 있는 결과를 얻을 수 있다. 본 연구에서는 효율성, 상호 운용성, 유지보수성의 세 가지 특성을 기반으로 클라우드 기반의 아키텍처 자산의 효율성을 평가하고자 한다. 효율성은 시간반응성, 요소활용, 기억용량을 대상으로 하고 상호운용성은 공존성, 상호 운용성을 대상으로 한다. 그리고 유지보수성은 모듈성, 재사용성, 분석가능성, 시험가능성을 효율성 부 특성으로 지표를 설정하였다.

3.2 클라우드 기반 자산의 구성

클라이언트 기반에서 자산 구성한 것과 다르게 클라우드 기반으로 자산을 구성하였을 경우 각 자산은 모든 클라이언트 설계자의 표준을 제시하고 동일 설계자산을 갖게 된다. 또한 각 자산은 동일한 평가 프로세스를 통과한 검증된 자산만을 공유하여 응용 도메인에 적합한 설계영역으로 확장할 수 있게 된다. 자산의 효율성을 평가하기 위한 클라우드 모델은 Fig. 4와 같이 모든 클라우드 환경에 따라 각 서비스 자산을 아키텍처 표준자산으로 제공하도록 구성할 수 있다[15-17]. 이 클라우드의 각 서비스는 제안하는 모델의 구조를 이용하여 자산을 제공하기 위해 몇 가지 고려 사항을 갖고 있다[18].

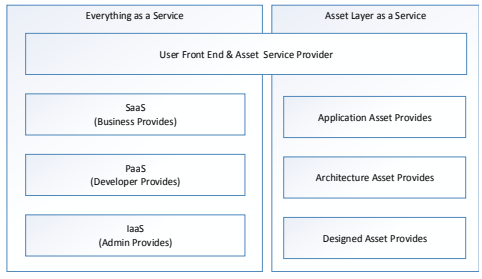

Fig. 4. Basic Model of Cloud Architecture

첫 번째는 자산 서비스 서버의 자산 서비스를 위한 일반 SLA의 개설방법, 두 번째는 자산 서비스 클라이언트는 설계자의 요구 사항에 따른 서비스 브로커 전송방법, 세 번째는 서비스 브로커는 선택 가능한 서비스를 찾기 위한 SLA 수집기의 요청방법, 네 번째는 SLA 수집기는 클라우드 서비스 목록의 제공방법이고 마지막으로 각 자산 서비스에 대해 실시간 평가 수행방법이다.

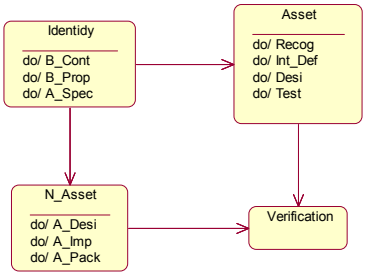

Fig. 5. Cloud based Asset Verification

각 아키텍처 자산은 클라우드 서비스로 제공되기 위해 Fig. 5와 같이 표준화된 자산 검증단계를 거쳐 등록 관리한다. 이때 각 자산은 Fig. 6과 같이 ISO/IEC 25010를 기반으로 자산의 효율성 평가를 위한 특성 항목을 구성하였다.

3.3 클라우드 기반 자산의 평가 모델

자산의 유형에 따라 자산 중 아키텍처 정보를 표준화하여 제공하는 아키텍처 자산은 각각의 개별 항목에 대하여 모두 평가해야 하며, 도메인 특성을 반영한 파생 자산은 설계자의 의도를 반영하기 위해 적용 도메인에 따라 평가 항목을 선택적으로 구성하도록 하였다. Fig. 6은 복합자산의 품질특성에서는 평가 대상에 따라 부 특성의 평가지표를 구성하기위한 방법을 제시하고 있다.

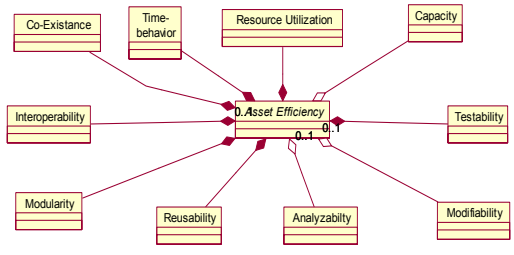

Fig. 6. Asset Efficiency Configuration

클라우드 기반의 표준화된 각 자산을 대상으로 설계 도메인의 특성이 반영된 복합자산은 자산의 설계정보 복잡도에 따라 Fig. 6에 제시한 효율성을 필수적 요소와 선택적 요소로 가져갈 수 있다.

Table 1. Cloud based Efficiency Evaluation

Criteria	Item Review
Crt1	The degree to which the response, processing time and throughput meet the requirements when performing the function
Crt2	Evaluate the extent to which the type and amount of resources used meet the requirements
Crt3	The degree to which the maximum limit of product or system parameters meets the requirements
Crt4	Evaluate the extent to which they perform their required functions effectively while sharing the environment and resources without adversely affecting other software
Crt5	The degree to which a system, product, or component can exchange information or use the information without error.
Crt6	The degree to which individual components with minimal impact
Crt7	Evaluate the extent to which an asset can be used in one or more systems and can build other assets
Crt8	The extent to which reports are available to assess the impact of system changes
Crt9	The degree to which the product or system can be modified effectively and efficiently without failure
Crt10	Verification function required before and during product use

이와 같은 평가를 위해 Table 1과 같이 각 부 특성의 평가항목을 구성하였다. 평가 대상이 되는 자산은 설계 영역의 도메인 특성과 설계자의 능력에 따라 효율성 평가항목을 필수적 요소와 선택적 요소로 선택 적용하여 평가하도록 구성하였다.

각 평가 부 특성은 목적 시스템의 설계 도메인에 부합하기 위해각 부 특성 평가항목의 가중치를 적용하여 효율성을 평가할 수 있도록 하였다. 효율성 평가를 위한 각 자산의 평가 가중치 설정 값은 평가의 도메인 특성과 설계자의 의도에 해당하는 환경 변수 값을 활용하여 평가 결과 값을 계산하기 위해 식 (1)을 정의하였다.

$$Effic = \sum_{i=1}^{n} Spec1_i + \sum_{i=1}^{n} Spec2_i \quad (1)$$

$$Spec1_i = \sum_{j=1}^{7} weight_j * Crt_j$$

$$Spec2_i = \sum_{j=1}^{3} weight_j * Crt_j$$

각 자산에 대한 평가 값은 ISO/IEC 25010의 전체 특성 에서 평가 도메인으로 설정하고자 하는 부 특성을 선정한 후 필수 평가항목과 선택 평가항목의 가중치를 이용하여 계산하도록 하였다. 가중치의 계수 값은 효율성을 평가하기 위해 표준화된 아키텍처 자산에 대하여 필수항목은 모두 동일한 값을 부여한다. 기존의 표준화된 자산 평가방법은 획일적인 평가로 설계특성을 고려하여 자산의 도메인 특성반영하지 못하였다. 그러나 본 연구의 평가방법은 설계자의 도메인 특성을 유연성 있게 반영하도록 하였다. 그리고 설계자의 설계 의도를 수용하기 위해 도메인 자산으로부터 재구성한 자산의 평가를 위해 평가 도메인의 특성 값에 해당하는 환경 변수 값을 재구성 하여 설정할 수 있도록 하였다.

4. 결론

클라우드 기반의 아키텍처 표준 자산을 정의하고 이를 이용하여 설계 도메인의 특성 값을 반영한 자산을 평가하기 위한 방법은 도메인 특성과 설계자의 의도를 반영한 사용성 중심의 평가가 이루어져왔다. 그리고 소프트웨어 자산 시스템을 평가하기 위해 모든 소프트웨어가 구성하고 있는 환경 정보를 반영하고 설계자의 설계 품질 특성을 구축하고 측정하기 위한 척도 값을 설정할 수 있어야 한다. 컴포넌트를 기반으로 하는 소프트웨어에서 설계 자산 정보는 클라우드를 기반으로 표준화한 도메인의 속성 값 또는 응용 플랫폼에 독립적인 정보를 제공하게 한다. 클라우드에서 제공하는 자산을 부품으로써 사용하기 위해 재사용성과 유지보수성에 대한 기본적인 요구 특성을 충족시켜야 한다. 선행된 자산기반의 설계 시스템의 복잡도 측정으로부터 복합자산을 평가하기 위한 평가 모델을 위해 ISO/IEC 25010 품질 모델 특성을 기반으로 자산데이터의 사용성을 평가하였고, 본 논문에서는 이를 기반으로 효율성을 평가하기 위한 부 특성의 평가모델 기준을 구축하였다.

제안된 효율성 평가모델은 클라우드 자산의 주요특성에 해당하는 기본자산과 복합자산으로 구성되어 있으며 재사용성과 유지보수성을 함께 평가할 수 있는 모델로 정형화 하였다. 아키텍처 표준에 부합하는 기본 자산의 특성 값은 표준모델의 값을 그대로 적용하여 필수 평가변수 값을 적용하여 평가할 수 있다. 또한 아키텍처 자산이 복합자산으로 구성되어 설계되었을 때 각 자산의 도메인 특성과 설계자의 설계의도를 반영하기 위한 특성에 따라 가중치를 부여한 환경 변수 값에 해당하는 부 특성의 선택적 평가를 적용하여 평가모델의 유연성을 확보하도록 하였다. 선택적 평가모형을 가져갈 수 있도록 구성한 사용성의 평가모델은 클라우드 자산을 응용 도메인에 적용하고자 하는 가중치 값을 찾기 위해 설계자산에 대한 기댓값을 학습하여 적정 값을 제시할 필요가 있다. 이에 해당하는 기댓값을 학습하기 위한 인공지능 적용이 필요하고 클라우드 자산의 증가에 따른 학습 값 적용이 필요하다.

향후 연구과제로 클라우드 기반의 자산을 재사용하여 시스템을 설계 도메인의 특성에 적용하여 설계할 경우 설계시스템을 국제표준에 부합되는 특성 값을 기반으로 평가할 수 있도록 효율성외 특성에 해당하는 부 특성을 평가하도록 하였으나 설계 적용할 때 환경 변수를 설정하여 학습데이터로 빅데이터를 구축하여 학습망으로 최적 변수를 설계자에게 제시할 필요가 있다.

REFERENCES

[1] H. Y. Choi. (2018). Evaluation Method of Architecture Asset. *Journal of Convergence for Information Technology, 8(5),* 101-106.

[2] Amazon Web Services. (2015). Amazon Web Services (AWS) CloudFormation. Retrieved from http://aws.amazon.com/cloudformation/.

[3] H. T. Mouftah & B. Kantarci. (2014). *Communication Infrastructures for Cloud Computing,* Hershey, PA, USA:UGI Global.

[4] M. Hajibaba & S. Gorgin. (2014). A review on modern distributed computing paradigms: Cloud computing jungle computing and fog computing, *J. Comput. Inf. Tech, 22(2),* 69-84

[5] H. Y. Choi. (2016). MetaData Structure Design of Architecture Asset in DMI. *Journal of Convergence for Information Technology, 6(4),* 151-156.

[6] OMG. (2017). *Reusable Asset Specification OMG Available Specification Version 2.2.* Name of Web Site. http://www.omg.org.

[7] ISO/IEC 25021. (2011). Software engineering - software product quality requirements and evaluation (SQuaRE) - quality measure elements.

[8] Fred B. Bryant & Albert Satorra. (2012). Principles and Practice of Scaled Difference Chi-Square Testing. *Structural Equation Modeling: A Multidisciplinary Journal, 19(3),* 372-398.

[9] Koh, Seokha. (2016). Cause-and-Effect Perspective on Software Quality : Application to ISO/IEC 25000

Series SQuaRE's Product Quality Model, *Journal of Information Technology Applications and Management, 23(3),* 71-86.

[10] Cai, X., Lyu, M.R., Wong, K. F. & Ko, R. (2000). *Component-based software engineering: technologies, development frameworks, and quality assurance, schemes.* Proc. Seventh Asia-Pacific Software Engineering Conf., APSEC 2000.

[11] Antonellis, P. et al. (2007). A data mining methodology for evaluating maintainability according to ISO/IEC-9126 software engineering - product quality standard. Special Session on System Quality and Maintainability - SQM2007.

[12] H. J. Jung, (2017). The Quantity Data Estimation for Software Quality Testing, *Journal of the Korea Convergence Society, 8(10),* 37-43.

[13] Y. S. Jeong, Y. T. Kim & G. C. Park. (2017). A hierarchical property-based multi-level approach method for improves user access control in a cloud environment, *Journal of the Korea Convergence Society , 8(11),* 7-13.

[14] B. K. Lee. (2014), A Study on Security of Virtualization in Cloud Computing Environment for Convergence Services, *Journal of the Korea Convergence Society, 5(4),* 93-99.

[15] Krahn, H., Rumpe, B., Volkel & S. MontiCore. (2010). a framework for compositional development of domain specific languages. *STTT, 12(5),* 353-372.

[16] Simona Bernardi, José Merseguer & Dorina C. Petriu. (2012). Dependability modeling and analysis of software systems specified with UML. *ACM Computing Surveys. 45(1).*

[17] H. Y. Choi & S. H. Sim. (2015). A Study on Software Development method based on DMI. *ICSMB, 2(1),* 359-360.

[18] H. Y. Choi & S. H. Sim. (2019). RAMS architecture based on cloud service, *International Journal of Innovative Technology and Exploring Engineering, 8(4S2),* 431-434.

최 한 용(Han-Yong Choi) [정회원]

· 1993년 2월: 경희대학교 전자계산공학과 학사
· 1998년 2월 : 경희대학교 전자계산공학과 석사
· 2002년 8월 : 경희대학교 전자계산공학과 박사
· 2004년 3월 ~ 현재 : 신한대학교 IT융합공학부 컴퓨터공학전공 교수
· 관심분야 : 재사용, 플랫폼 설계, 품질관리
· E-Mail : hychoi@shinhan.ac.kr