ISSN 2586-1816
Journal of Convergence for Information Technology
e-ISSN 2586-4440
Vol. 8. No. 5, pp. 101-106, 2018
DOI : https://doi.org/10.22156/CS4SMB.2018.8.5.101

아키텍처 자산의 평가 방법

최한용
신한대학교 IT융합공학부 교수

Evaluation Method of Architecture Asset

Han-yong Choi
Professor, Division of IT Convergence Engineering, Shinhan University

요 약 자산을 등록 관리하기 위한 다양한 소프트웨어가 연구되고 있으며 이와 같은 소프트웨어 시스템을 평가하기 위한 방법은 주관적인 평가기준을 대상으로 이루어져왔다. 본 연구에서는 선행된 자산관리 시스템의 복잡도 측정으로부터 얻어진 복합자산을 평가하기 위한 평가모델을 제안하고자 한다. 아키텍처 자산의 복잡성을 측정하기 위해 논리적 복잡도를 측정하여 제공하는 척도를 이용하였으며, 아키텍처 자산의 속성 값을 표현하고 있는지를 평가하기 위한 방법을 사용 하였다. 소프트웨어 평가 표준인 SQuaRE Series의 ISO/IEC 25010 품질 모델 특성을 기반으로 자산데이터의 사용성을 평가하기 위한 부특성의 평가모델 기준을 구축하였다. 자산은 복합자산으로 구성되어 설계되었을 때 각 자산의 특성에 따라 가중치를 부여한 부특성의 선택적 평가를 적용하여 평가모델의 유연성을 확보하도록 하였다.

주제어 : 복잡도, 자산, 품질평가, 사용성, 아키텍처

Abstract Software are being studied to register and manage assets. And Methods for evaluating software systems have been based on subjective evaluation criteria. We propose an evaluation model for evaluating complex assets obtained from the complexity measurement of the preceding asset management system. We used scales to measure and provide logical complexity to measure the complexity of our architectural assets. And we used a method to evaluate whether it expresses attribute value of architecture asset. We have also built an evaluation model criterion for evaluating the usability of the asset data based on the ISO/IEC 25010 quality model characteristics of the SQuaRE Series. When the designers design the asset as a composite asset, the optional evaluation of the negative property that weights are assigned according to the characteristics of each asset is applied to secure the flexibility of the evaluation model.

Key Words : Complexity, Asset, Quality Evaluation, Usability, Architecture

1. 서론

복잡하고 거대해지는 소프트웨어 산업에서 품질의 우수성을 확보하기 위한 모델개발과 표준화를 위해 다양한 연구가 진행되고 있다[1]. 또한 요구사항의 다양한 변화는 기술적으로 수용되어야 하며, 소프트웨어 생산성의 효율을 증가시키기 위한 방법론과 표준에 관한 연구와 소프트웨어를 정형화하여 생산하기 위한 방법에 어려움을 겪고 있다. 본 논문에서는 소프트웨어의 기본 구성요소인 아키텍처 컴포넌트를 기본자산으로 구성하였으며, 이를 기반으로 정형화하거나 표준화된 설계모델을 확보하였다[2]. 그리고 정형화되거나 표준화된 재사용 가능한 확장 자산을 설계하기 위해 구성된 각 자산의 사용성에 대한 평가 모델을 구축하고자 한다.

이와 같은 소프트웨어 시스템을 평가하기 위한 방법은 주관적인 평가기준을 대상으로 이루어져왔다[3,4]. 그

*This work was supported by the Shinhan University Research Fund, 2018
*Corresponding author : Hanyong Choi (hychoi@shinhan.ac.kr)
Received September 11, 2018 Revised September 27, 2018
Accepted October 20, 2018 Published October 31, 2018

리고 아키텍처 자산의 설계에서는 재사용성에 대한 고려와 조립성을 고려하여 아키텍처 자산을 정제하여야 하도록 하였다. 따라서 제품 라인을 위한 컴포넌트는 재사용할 수 있는 자산을 기반으로 정형화하여 설계된 자산 컴포넌트로부터 합성이 가능하게 자산을 사용하기 때문에 두 영역에 대한 자산을 이원화하여 평가하기 위한 모델이 필요하다[5,6].

본 논문에서는 선행된 자산관리 시스템의 복잡도 측정으로부터 얻어진 복합자산의 평가를 대상으로 하여 기본자산과 복합자산의 평가하기 위한 이원화된 평가모델을 구성하였다. 소프트웨어 평가 표준인 SQuaRE Series의 ISO/IEC 25010 품질 모델 특성을 기반으로 자산데이터의 사용성을 평가하기 위한 부특성의 평가모델 기준을 구축하였다[7,8]. 자산은 복합자산으로 구성되어 설계되었을 때 각 자산의 특성에 따라 가중치를 부여한 부특성의 선택적 평가를 적용하여 평가모델의 유연성을 확보하도록 하였다. 본 논문의 구성은 2절에서 자산의 모델과 구성에 대한 기반연구, 3장에서는 평가구성 항목과 평가방법에 대한 아키텍처 자산의 평가방법과 4장에서는 결론에 대하여 기술하였다.

2. 기반연구

2.1 자산의 메타 모델

선행연구에서는 자산을 기반으로 하는 설계 시스템에서 설계 도메인에 독립적인 설계구조를 표현하기 위해 자산을 메타 모델로써 정형화하였다.

그리고 XMI로 표현된 메타모델을 이용하여 메타데이터의 높은 이식성을 유지하도록 구성하였다. 아키텍처 설계정보는 기본 자산의 메타모델과 확장 메타모델의 두 형식으로 모델을 설계하였다[9,10]. 첫 번째 모델 형식은 정형화되고 표준화된 자산을 시스템에 관리하기 위한 아키텍처 자산을 구성하기 위한 메타 데이터 모델이다. 그리고 두 번째 모델 형식은 자산 설계자의 도메인 특성에 의존성을 부여하기 위고 이를 서비스하기 위한 복합 자산 컴포넌트를 구성하기 위한 확장 메타 데이터 모델이다.

아키텍처 설계 자산의 정보를 정형화하기 위한 표현 방법으로 Fig. 1과 같이 XMI를 이용하여 메타모델을 사용하였다[11]. 따라서 컴포넌트 기반으로 재구성된 아키텍처 자산의 메타 데이터 모델을 구성하기 위한 문법에 따라 아키텍처 자산은 컴포넌트 자산의 속성 값과 자산의 관계 집합으로 구성된다[12-14]. 그리고 각 컴포넌트 자산은 세부 구성 항목인 클래스 집합과 관계 집합으로 구성되어 있다. 이와 같이 설계자의 응용 서비스 도메인에 필요한 아키텍처 자산을 설계하기 위해 기본 아키텍처 자산을 기반으로 합성되어 구성된 도메인 복합 아키텍처 자산을 구성한다.

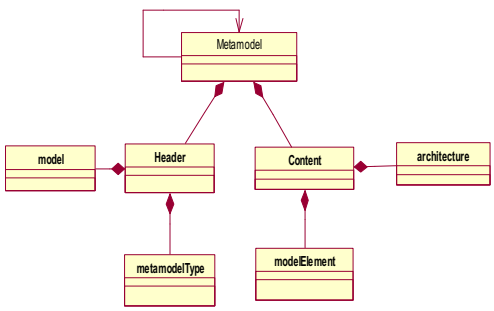

Fig. 1. Asset Meta Model

2.2 자산 구성

본 논문에서는 Fig. 2와 같이 표준 명세방법인 RAS를 채택하여 자산을 명세하기 위한 방법을 사용하였다. Fig 2.의 명세 방법은 설계 자산의 재사용을 위해 절차적인 일관성을 유지한다.

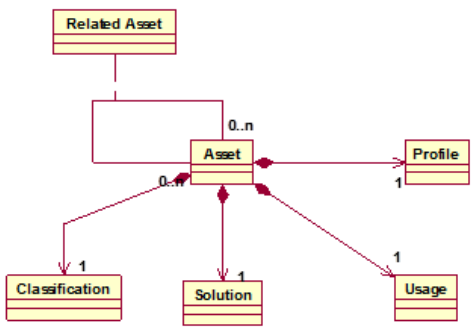

Fig. 2. Asset Configuration

그리고 각 자산은 재사용 가능한 설계 자산들의 아키텍처(Architecture)와 구성 내용 그리고 자산의 설명에 대하여 기술하고 있다. 자산 명세의 내용은 Core RAS와 동일하게 각 자산 명세에 대한 기본적인 구성요소들을

모두 포함하도록 하였다. 그리고 도메인의 특성을 반영한 복합 자산은 특정 형태의 자산들에 대한 도메인 속성 갑에 대한 의미를 나타내기 위해 Core RAS의 확장하여 표현한 프로파일 기능을 갖는다[15].

본 논문에서는 아키텍처 자산의 구성을 계층적으로 구성할 수 있도록 하기 위해 범주를 설정하였다 그리고 각 자산의 행위적 특성을 반영하여 각 자산을 분류하도록 제한하였다.

2.3 ISO/IEC 품질 평가

시스템 및 소프트웨어의 품질을 측정하고 평가하기 위한 표준은 ISO/IEC 25000 Series “Systems and software engineering-systems and software Quality Requirements and Evaluation (SQuaRE)” 이다. QuaRE Series의 구성은 ISO/IEC 9126 Series와 ISO/IEC 14598 Series가 소프트웨어 품질과 품질관리에 대한 국제 표준으로 서로 상호 보완적이었다. 하지만 이 국제 표준은 각기 다른 생명주기를 가지고 있어, 이 두 국제 표준을 적용하는데 불일치를 야기하여, 이러한 문제를 개선하기 위해서 SQuaRE Series를 구성하여 표준으로 사용한다. ISO/IEC 25000:2014를 시작으로 SQuaRE Series 중 품질평가의 기본 모델에 속하는 ISO/IEC 25001, 25010, 25012, 25020, 25021, 25022, 25023, 25024, 25030, 25040, 25041, 25045 총 13개의 국제 표준이 있다. SQuaRE Series의 주요 내용은 Fig 3.과 같다[8,9].

- ISO/IEC 2500n – Quality Management Division.
- ISO/IEC 2501n – Quality Model Division.
- ISO/IEC 2502n – Quality Measurement Division.
- ISO/IEC 2503n – Quality Requirements Division.
- ISO/IEC 2504n – Quality Evaluation Division.
- ISO/IEC 25050 to 25099 – SQuaRE Extension Division.

Fig. 3. SQuaRE Series

여기에서 품질 모델 분야인 ISO/IEC 2501n은 시스템 및 소프트웨어 제품의 사용품질(quality in use) 및 데이터에 대한 상세한 품질모델을 설명한다. 또한 이러한 품질 모델을 적용하는데 필요한 실질적인 지침을 제공한다.

3. 아키텍처 자산의 평가

3.1 자산평가 구성

선행 연구의 자산관리를 기반으로 하는 설계 시스템에서는 아키텍처 자산을 두 영역으로 구분하여 기본 자산과 복합 자산으로 관리하고 있다. 이 시스템내의 자산의 품질을 평가하기 위해 국제 표준 구성항목을 이용하고자 한다. 본 연구에서 자산 관리 시스템의 평가모델로 사용하는 것은 ISO/IEC 25010의 사용품질을 기준으로 적용하였다[7,8]. 자산을 평가하기 위한 방법으로는 독립된 기본적인 아키텍처 자산을 평가 한다. 그리고 독립 자산들 간의 관계성을 평가하여야 한다. 선행연구에서는 독립된 자산의 평가를 위해 복잡성을 평가하고 상관관계를 분석하였다. 그리고 각 자산은 기능성을 비롯한 8가지의 특성을 평가하여야 한다. 기본 아키텍처 자산은 8가지 항목을 그대로 적용한다.

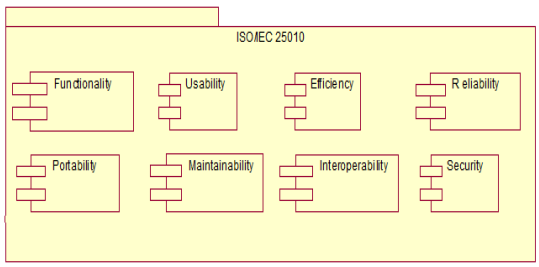

Fig. 4. ISO/IEC 25010 Configuration

본 연구에서는 자산 시스템의 품질평가를 위해 Fig 4.와 같이 ISO/IEC 25010에서 언급하고 있는 명시적 또는 묵시적 필요성을 충족시키는 능력과 관련된 소프트웨어의 특성과 특징으로 제시하고 있기 때문에 이에 대한 평가 방법을 기반으로 자산의 평가모델을 적용하였다. 모든 항목의 평가모델을 측정하여 종합적인 자산의 평가기반을 구축하여야 한다. 본 논문에서는 단계적으로 구성항목을 목적 시스템에 적용하기 위한 방법을 제안하고자 한다. 그리고 자산 관리 시스템의 평가를 위해 우선적으로 Fig 5와 같이 사용성에 대한 평가방법을 위해 ISO/IEC 25010의 사용성 품질 평가항목을 기준으로 적용한다[8,9]. 사용성에 대한 6가지 특성 항목은 그대로 적용하고 각 특성의 측정 방식을 두 가지 방법으로 평가하도록 구성하였다.

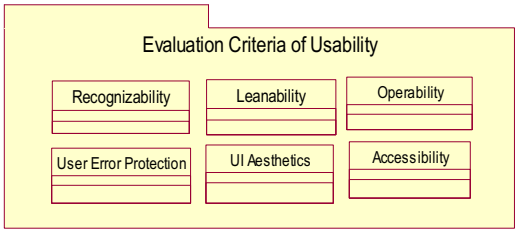

Fig. 5. Evaluation Criteria of Usability

Fig 5는 사용성에 대한 평가항목으로 표준화된 아키텍처 자산과 복합 자산의 종류에 따라 다르게 적용하는 방법을 사용하도록 하였다. 독립자산은 품질 특성 중 6개의 부특성을 모두 사용하도록 하였고, 복합 자산은 사용성 평가를 위해 목적 시스템의 설계 도메인에 따란 선택적 적용을 할 수 있도록 하였다.

3.2 복합자산 평가 모델

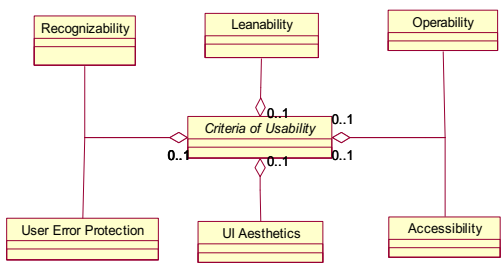

Fig. 6. Evaluation Criteria of Complexity Assets

두 유형의 자산 중 표준화된 아키텍처 정보를 제공하기 위한 자산은 모든 항목에 대한 평가를 대상으로 하나 복합 자산은 적용 모델에 따라 다른 평가를 구성할 수 있도록 하였다. Fig 6과 같이 복합자산의 품질특성에서는 부특성을 선별적 평가지표를 적용하는 방법을 사용하도록 하였다. 각 자산은 구성방법에 따라 초기 설계정보를 보관하기 위해 기초 자산을 설계하여 저장한다. 이와 같은 기초자산은 설계능력을 충분히 확보한 설계자에 의해 검증된 기초자산을 구성하도록 한다.

그리고 복합자산은 설계의 경험과 자산의 설계정보 복잡도에 따라 Fig 6.에 제시한 사용성을 선택적으로 가져갈 수 있다. 이에 대한 평가를 위해 Table 1.과 같이 각 부특성의 평가항목에 대해 자산의 설계영역의 특성과 설계능력에 따라 평가요소를 선택 적용하여 평가하도록 자산의 사용성 부특성 항목의 내용을 구성하였다.

Table 1. Sub-Criteria of Usability Evaluation

Criteria	Item Review
Crt1	Evaluate the identities of the design assets in the two areas according to the requirements of the asset designer
Crt2	Evaluate design achievement
Crt3	Evaluate ease of operation and control
Crt4	Evaluate designer's protection against errors in asset-based systems
Crt5	Evaluate the designer's satisfaction on the UI of the asset-based system
Crt6	Evaluation of usability in the category of asset designer

세부 구성항목으로는 재사용성과 조립성을 확보하고 이를 평가하기 위한 기준항목을 포함하고 있다. 각 평가 부특성은 평가시스템의 목정에 부합하기 위해각 부특성 평가항목의 중요도에 따른 가중치를 적용하여 평가할 수 있도록 하였다.

$$CompEval = \sum_{i=1}^{n} Eval_i \qquad (1)$$

$$Eval_i = \sum_{j=1}^{k} weight_j * Crt_j$$

가중치 설정 값은 평가의 목적성에 부합하는 계수 값을 적용하여 식(1)로 평가 결과 값을 계산할 수 있다. 각 평가 결과 값은 ISO/IEC 25010의 6가지 특성 값에 중 평가대상으로 설정하고자 하는 부특성을 선정한 후 가중치를 이용하여 계산하도록 하였다. 가중치 값은 표준화된 아키텍처 자산의 특성을 평가할 때 동일한 값을 부여하지만, 설계 도메인의 특성을 반영하여 복합 자산을 평가할 때는 평가 영역에 따른 가중치 값을 다르게 가져 갈 수 있다. 이때 가중치는 설계자의 기댓값으로 대치할 수 있으며 도메인별 특성 값으로 설정할 수 있다.

4. 결론

자산기반의 시스템을 구성하고 평가하기 위한 방법은 주관적인 평가기준을 대상으로 이루어져왔다. 그리고 컴포넌트 기반의 시스템을 평가하기 위해 모든 소프트웨어

가 구성하고 있는 품질 특성 체계를 구축하고 측정하기 위한 척도 값을 설정할 수 있어야 한다. 컴포넌트를 기반으로 하는 소프트웨어에서 설계 자산 정보는 도메인의 속성 값 또는 응용 플랫폼에 독립적이어야 한다. 컴포넌트 자산을 부품으로써 사용하기 위해 재사용성에 대한 요구와 조립성에 대한 기본적인 요구 특성을 충족시켜야 한다. 선행된 자산기반의 설계 시스템의 복잡도 측정으로부터 얻어진 복합자산을 평가하기 위한 평가모델을 위해 ISO/IEC 25010 품질 모델 특성을 기반으로 자산데이터의 사용성을 평가하기 위한 부특성의 평가모델 기준을 구축하였다.

제안된 평가모델은 자산의 주요특성에 해당하는 기본 자산과 복합자산으로 구성되어 있으며 이를 함께 평가할 수 있는 모델로 정형화 하였다. 기본 자산의 특성 값은 표준모델의 값을 그대로 적용하여 평가할 수 있다. 또한 아키텍처 자산이 복합자산으로 구성되어 설계되었을 때 각 자산의 특성에 따라 가중치를 부여한 부특성의 선택적 평가를 적용하여 평가모델의 유연성을 확보하도록 하였다. 선택적 평가모형을 가져갈 수 있도록 구성한 사용성의 평가모델은 응용 도메인에 적용하고자 하는 가중치 값을 찾기 위해 설계자산에 대한 기댓값을 학습하여 적정 값을 제시할 필요가 있다.

향후 연구과제로 자산을 기반으로 하는 설계시스템을 국제표준에 부합되는 특성 값을 기반으로 평가할 수 있도록 사용성외 특성에 해당하는 부특성을 평가하기 위한 모델의 단계적 개발과 통합이 필요하다. 본 연구에서 제안한 사용성에 대한 부특성을 평가하기 위한 방법을 모든 특성 값을 평가하기 위한 방법으로 확장할 필요가 있다. 그리고 제안하는 자산의 사용성 평가 방법이 실사례에 적용하였을 때 결과 값을 평가하기 위한 추가 연구가 필요하다.

REFERENCES

[1] N. I. Altintas, S. Cetin, A. H Dogru & H. Oguztuzun. (2011). Modeling Product Line Software Assets Using Domain-Specific Kits. *Software Engineering, IEEE Tr. 38(6),* 1376-1402.

[2] H. Y. Choi. (2016). MetaData Structure Design of Architecture Asset in DMI. *SMB, 6(4),* 151-156.

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

[4] V. B. Misic & S. Moser. (1997). Measuring Class Coupling and Cohesion : A Formal Metamodel Approach. *ASPEC'97.* 31-40.

[5] K. Phol, G. Böckle & F. van der Linden. (2010). *Software Product Line Engineering: Foundations, Principles, and Techniques*: Springer.

[6] H. Y. Choi & S. H. Sim. (2017). A Study on the Optimization of Architecture Assets in DMI. *Journal of Advanced Research in Dynamical and Control Systems,* 143-149.

[7] ISO/IEC 25010. (2011). *System and Software Engineering-System and Software Quality Reauirements and Evaluation(SQuaRE)-System and Software Quality Models.*

[8] ISO/IEC 9126-1. (2011) *Software Engineering-Product Quality- Part1: Quality Model.*

[9] S. Bernardi, J. Merseguer & D. C. Petriu. (2012). Dependability modeling and analysis of software systems specified with UML. A*CM Computing Surveys (CSUR), 45(1),* 2.

[10] A. B. Younes, Y. B. Hlaoui & L. J. Ben Ayed. (2014). A Meta-Model Transformation from UML Activity Diagrams to Event-B Models. *Computer Software and Applications Conference Workshops (COMPSACW) 2014 IEEE 38th International.* 740-745.

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

[12] Y. S. Choi & J. E. Hong. (2017). Designing Software Architecture for Reusing Open Source Software. *Journal of Convergence for Information Technology, 7(2),* 67-76.

[13] H. D. Ryu & W. J. Lee. (2012). A Study on UML based Modeling and Automatic Code Generation for Embedded Software. *Journal of Convergence for Information Technology, 2(1),* 22-40.

[14] J. S. Park, J. J. Kwon, J. E. Hong & Min Choi. (2013). Software Architecture Recovery for Android Application Reuse. *Journal of Convergence for Information Technology, 3(2),* 9-17.

[15] S. Rajesh & A. Chandrasekar. (2016). An Efficient Object Oriented Design Model: By Measuring and Prioritizing the Design Metrics of UML Class Diagram with Preeminent Quality Attributes. *IJST, 9(21),* 1-9.

최 한 용(Han-Yong Choi) [정회원]

- 1993년 2월 : 경희대학교 전자계산공학과(공학사)
- 1998년 2월 : 경희대학교 전자계산공학과(공학석사)
- 2002년 8월 : 경희대학교 전자계산공학과(공학박사)
- 2004년 3월 ~ 현재 : 신한대학교 IT융합공학부 교수
- 관심분야 : 재사용, 플랫폼 디자인, 품질관리
- E-Mail : hychoi@shinhan.ac.kr