ISSN 2234-4438
중소기업융합학회 논문지 제6권 제4호 pp. 151-156, 2016 DOI : http://dx.doi.org/10.22156/CS4SMB.2016.6.4.151

아키텍처 자산의 메타데이터 구성

최한용*
신한대학교 IT융합공학부

MetaData Configuration of Architecture Asset

Han-Yong Choi*
Division of IT Convergence Engineering, Shinhan University

요 약 소프트웨어 생산성을 향상시키고 자동화하기 위한 효과적인 개발방법이 지속적으로 요구되어 왔다. 본 논문에서는 DMI를 기반으로 설계단계에서 도메인 설계정보를 재사용하기 위한 부품자산을 구성하려고 한다. 그리고 설계단계에서 설계정보를 재사용하기 위해서는 개발환경에 독립적인 플랫폼의 추상화된 아키텍처정보가 필요하다. 또한 잘 설계된 아키텍처를 기반으로 응용영역의 설계를 지원할 수 있어야 한다. 그러므로 본 연구에서는 아키텍처 레벨의 플랫폼과 응용 도메인영역의 설계정보를 정형화하여 표현할 수 있는 DMI 구조를 이용하였다. DMI에서 아키텍처 자산은 추상화 레벨이 높은 설계단계에서 설계정보를 자산부품으로 분해하거나 합성이 가능하다. 따라서 아키텍처 자산의 메타 데이터 구성은 도메인 영역의 기반설계구조를 재사용할 수 있는 구조를 지원하게 된다.

키워드 : Asset, Architecture, Reuse, DMI

Abstract It has been constantly demanding for effective way to improve software development productivity and automation. In this paper, we did research to configure the component assets to reuse design information from the design phase domains based on the DMI. It is necessary abstracted information architecture of an independent platform development environment to reuse design information on the design stage. Also, It should be based on a well-designed to support the design architecture of the application domain. Therefore, in this paper, I want to use the DMI architecture that can be represented by formal level of architectural design information platform and application domain area. It is able to decomposition architecture asset with the part design information from the design phase composition or a high level of abstraction on DMI. Therefore, the metadata structure of the asset architecture will support a structure which can reuse the structure-based design of the domain areas.

Key Words : Asset, Architecture, Reuse, DMI

1. 서론

소프트웨어 생산의 효율성을 위해 많은 연구가 오랜 기간 연구되어 왔으며, 생산성을 향상시키고 자동화하기 위한 다양한 연구가 이루어지고 있다[1]. 이와 같이 생산성을 높이기 위해 효율적인 개발방법이 요구되고 있다[2]. 그 중심에는 설계정보의 재사용과 자동화 생산에 대한 다양한 문제점을 갖고 있다[3-6]. 또한 다양한 플랫폼의 개발과 이를 기반으로 파생되는 제품의 수요가 급증하고 있다. 본 연구에서는 이를 해결하기 위해 개발 환경에 독립적이고 설계단계에서 도메인 설계정보를 재사용하기 위한 부품자산을 생성하려고 한다. 설계단계에서 설계정보를 재사용하기 위해서는 개발환경에 독립적인 플랫폼의 추상화된 아키텍처정보가 필요하다. 또한 정형

Received 2016-10-31 Revised 2016-12-05 Accepted 2016-12-07 Published 2016-12-31
*Corresponding author : Han-Yong Choi (hychoi@shinhan.ac.kr)

화된 아키텍처 설계정보를 기반으로 각 응용도메인에 적용하기 위한 설계를 지원할 수 있어야 한다[7]. 그러므로 본 연구에서는 아키텍처 레벨의 플랫폼과 응용 도메인영역의 설계정보를 정형화하여 표현할 수 있는 DMI 구조를 이용하였다. DMI는 추상화 레벨이 높은 설계단계에서 설계정보를 자산부품으로 분해하여 보관하고 이를 합성할 수 있도록 XMI를 이용하여 메타모델을 표현하였다. 따라서 설계정보는 XMI를 이용하여 정형화된 메타모델로 표현하여 부품자산으로 보관되며, 이를 합성하여 설계정보를 재사용하여 설계할 수 있도록 하였다. 또한 DMI 구조는 다양한 플랫폼 기반의 아키텍처 레벨에서 재사용 가능한 설계가 이루어 질 수 있다. 이와 같이 DMI 구조의 효율성을 위해서 하부 계층에서는 모델의 정형화와 이를 변환하기 위한 기술적 지원이 필요하다.

본 논문에서는 이를 지원하기 위해 DMI의 하부 레이에에서 지원하기 위한 아키텍처 모델의 메타데이터를 정의하고자 한다. 여기에서 아키텍처 설계정보를 설계자산으로 구성하고 이를 정형화하고자 한다. 효율성을 위해서 XMI 메타모델을 기반으로 추출하고 클래스 메타데이터의 마크업언어를 생성하여 정형화하였다. 논문의 구성은 2장에서 연구동향과 기반연구, 3장에서 자산의 메타데이터 구성, 4장에서 결론으로 구성하였다.

2. 연구동향 및 기반연구

2.1 연구동향

본 논문에서 해결하고자 하는 목표와 같이 소프트웨어를 단위부품화하고 자동화된 생산라인을 위한 해결 방법이 PLE(Product Line Engineering) 방법론이다[8]. PLE는 도메인공학과 응용공학으로 구성되어있다. 도메인공학은 공통점과 차이점을 분석해 제품자산(Product Asset)을 만들고, 응용공학은 PLA를 이용하여 응용 영역에 필요한 제품을 생산한다. 부품을 설계하기 위해서는 재사용과 조립을 고려하여 공용자산영역에 해당하는 아키텍처를 정제함으로써 재사용 가능한 자산으로부터 조립이 가능하다[9]. 그리고 DMI는 Design Layer, Mapping Layer, Infra Layer의 3계층 구조로 설계되었다[10]. 설계 계층(Design Layer)은 UML을 기반으로 설계정보를 표현하고 저장된 정보를 표현하는 계층이며, Infra Layer는 architecture 정보와 컴포넌트 정보를 XML로 컴포넌트화 하여 저장하는 계층이다. 설계정보와 디자인 계층 간의 정보를 교환해주는 중간 계층인 매핑 계층(Mapping Layer)은 UML의 표현과 XML 정보를 변환하는 계층이고, 컴포넌트를 검색, 관리하고 코드를 생성하는 역할을 수행한다[11]. 매핑 계층에서는 UML로 표현된 설계 구조를 모델링하는 모듈과 인프라 계층의 설계 구조를 구성하는 메타모델들을 저장하는 서비스를 제공한다.

2.2 PLE

소프트웨어 생산성을 증대하기 위해 컴포넌트 기반의 부품을 생산하고 이를 조립하고 있다[12]. 그러나 다양한 플랫폼의 변화에 적응하기 어렵고 생산성을 향상시키기 위한 유지보수가 필요한 경우에는 어려움이 있다[13]. 이를 해결하기 위한 효과적인 개발방법이 PLE(Product Line Engineering) 방법론이다. PLE는 도메인공학과 응용공학으로 구성되어있다. 도메인 공학은 공용자산과 응용자산을 분석해 제품자산(Product Asset)을 만든다[7]. 그리고 응용공학은 PLA를 이용하여 특정한 Product를 생산한다[14,15]. 도메인 공학은 애플리케이션 영역에 포함된 제품들을 분석한 후 분석된 정보를 제품라인의 자산으로 만들게 된다. 여기에는 두 가지 설계가 필요하다. 공용자산에 해당하는 아키텍처 설계영역에서는 도메인 독립적인 설계결정(Design Decision)을 한다. 그리고 응용영역에서는 응용영역에 종속적인 설계 결정을 하게 된다. 자산 설계에서는 부품 조립을 위해 재사용을 목적으로 하므로 아키텍처 자산을 정제하게 되고 자산은 제품라인을 통해 재사용 가능한 자산으로부터 조립을 할 수 있다.

2.3 DMI

본 과제에서는 DMI 아키텍처에서 3계층의 하부구조에 해당하는 설계정보 저장단계에서 다양한 플랫폼의 변화에 적응을 위해 설계정보의 효율적 저장과 재사용을 위한 아키텍처 자산의 메타데이터 정의와 추출을 하고자 한다[16]. 그리고 상위레벨에서는 DMI 아키텍처를 정의하여 UML로 모델링하고, 모델링한 설계정보를 XMI의 메타모델로 정의하고 다시 여기서 메타데이터를 추출하는 방법을 사용하고자 한다.

DMI 아키텍저 구조는 Fig 1에 표현된 것처럼 Design Layer, Mapping Layer, Infra Layer의 3계층 구조로 설계

되었다[8]. 설계 계층(Design Layer)은 UML을 기반으로 설계정보를 표현하고 저장된 정보를 표현하는 계층이며, Infra Layer는 architecture 정보와 컴포넌트 정보를 XML로 컴포넌트화하여 저장하는 계층이다. 그리고 저장된 설계정보와 디자인 계층간의 정보를 교환해주는 중간 계층인 매핑 계층(Mapping Layer)은 UML의 표현과 XML 정보를 변환하는 계층이고, 컴포넌트를 검색, 관리하고 코드를 생성하는 역할을 수행한다.

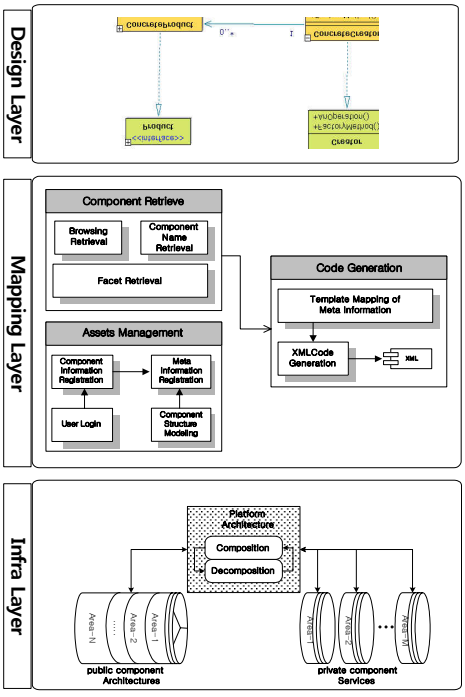

Fig. 1. DMI Architecture

매핑 계층에서는 UML로 표현된 설계 구조를 모델링하는 모듈과 인프라레이어의 설계 구조를 구성하는 메타모델들을 저장하는 서비스를 제공한다. 그리고 저장된 메타모델들을 코드 생성 모듈을 이용하여 XML 코드로 구성한다. XML에서 마크업언어 설계와 적절한 클래스의 생성에 대한 기능을 필요로 한다. 일반적으로 마크업언어로 DTD와 XML 스키마를 사용하고 있다. DTD는 플랫폼에 독립적으로 많이 사용되고 있으며 다양한 도구의 지원을 받고 있다[9]. 그리고 XML 스키마는 비선형 구조의 문법을 사용하고 있으며 다큐먼트와 여러 가지 데이터 종류를 나나낼 수 있다.

3. 자산의 메타데이터 구성

3.1 설계정보의 메타모델

본 논문에서는 도메인 환경에 독립적인 설계구조를 표현하기 위해 메타 모델을 정형화하였다. 그리고 설계정보를 표현하기 위한 메타 모델을 XMI를 이용하여 이식성이 높은 메타데이터로 표현하였다. 설계정보는 아키텍처 메타모델과 확장 메타모델의 두 가지 메타 모델로 설계하였다. 첫 번째 모델은 정형화된 자산으로 관리하기 위한 아키텍처를 표현하기 위한 메타 모델이고 두번째 모델은 설계자의 응용도메인의 특성에 의해 이를 서비스하기 위한 컴포넌트를 표현하기 위해 확장 메타 모델이다. 아키텍처 메타모델은 정형화된 도메인영역의 표준설계정보로 제공하기 위한 것을 의미하며, 이 도메인 아키텍처에 사용자의 서비스 컴포넌트를 조합하여 확장하게 된다. 정형화된 아키텍처의 설계정보를 표현하기 위해 Fig 2와 같이 XMI를 이용하여 메타모델을 정의하였다.

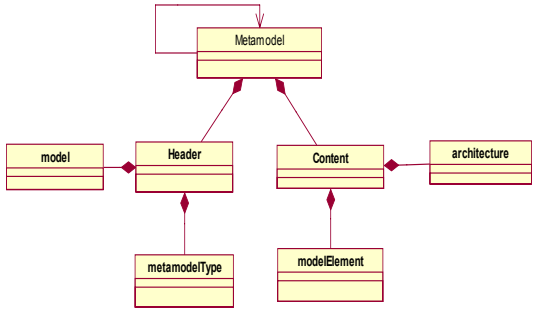

Fig. 2. Meta Model

그리고 Fig 3와 같이 재사용되어 확장되는 아키텍처를 표현하기 위한 메타모델은 미리 정의된 아키텍처를 표현한 메타모델을 이용하여 합성되는 아키텍처 정보와 관계정보를 포함하여 새로운 응용영역의 도메인 아키텍처를 표현할 수 있도록 XMI를 이용하여 표현한 메타 모델을 정의하였다. <XMI.Header> 영역은 메타 모델의 타입과 XMI 정보를 기술한다. <XMI.Content> 영역에는 실제 아키텍처의 메타모델을 표현하기 위한 XMI 정보를 기술한다.

따라서 컴포넌트로 조립된 아키텍처의 메타모델을 구성하기 위한 문법에 따라 아키텍처는 다시 여러 개의 컴

포넌트와 관계의 집합으로 구성되며, 각 컴포넌트는 다시 클래스와 관계의 집합으로 구성되어 있다. 이와 같이 서비스 도메인에 대한 아키텍처를 설계하기 위해 기존의 자산을 활용하여 새로운 도메인 아키텍처를 구성한다.

본 논문에서는 객체지향모델링 방법론을 적용하기 위해 UML을 이용하여 모델링하였다. 그리고 UML로 모델링한 컴포넌트 자산은 XMI를 이용하여 XML형식으로 변환하였다.

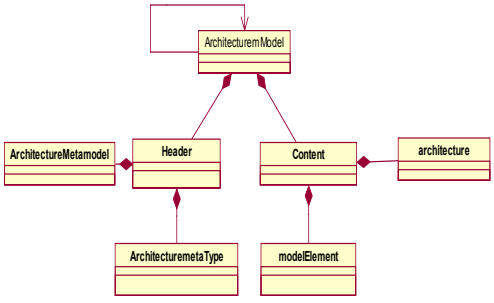

Fig. 3. Architecture Structure

그리고 설계 다이어그램을 이용하여 모델링한 UML 설계정보들은 XMI의 메타모델로 표현할 수 있으고 XML로 변환할 수 있다.

3.2 XMI DB

메타 데이터 부분의 구현영역에 대한부분은 DB를 이용하여 XMI를 표현하기 위해서 DB 정보를 태그로 생성하였다. DB와 테이블(table)은 자신에 종속된 테이블과 컬럼정보를 가질 수 있다. 이로써 UML을 이용하여 모델링한 아키텍처 자산을 XMI를 이용하여 자산화하고 응용영역에서 설계정보를 재사용하기 위한 스키마를 개방하여 재사용이 가능하도록 한 것이다. 그리고 <Column.name>을 이용하여 DB의 항목이나 속성, 함수에 대한 메타데이터를 구성하였다.

3.3 메타데이터 저장 시퀀스

XMI 메타모델의 항목을 저장하기 위해서 SMI 구성항목을 각각의 단위 클래스로 구분하였다. Fig 4는 메타모델 DB 시퀀스를 나타내는 다이어그램이다. XMI_header에 해당하는 abstract 클래스와 concrete 클래스에 대한 DB 생성의 모델링을 표현한 것이다. 액터는 DB Open에 대한 Logon 권한 부여받고 모델내의 XMI_header의 테이블생성을 하게 된다. 그리고 XMI_content에 대한 테이블생성을 하게 되고, XMI_difference와 XMI_extensions은 선택적으로 수행된다.

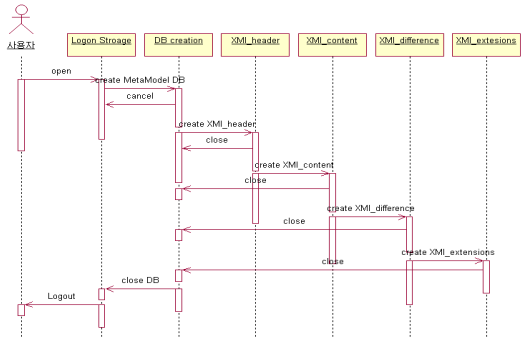

Fig. 4. Meta Model DB Sequence

4. 결 론

DMI는 추상화 레벨이 높은 설계단계에서 설계정보를 자산부품으로 분해하여 보관하고 이를 합성할 수 있도록 하기 위해 XMI를 이용하여 메타모델을 표현하였다. 따라서 설계정보는 XMI를 이용하여 정형화된 메타모델로 표현하여 부품자산으로 보관되며, 이를 합성하여 설계정보를 재사용하여 설계할 수 있도록 하였다. 또한 DMI 구조는 플랫폼 기반의 아키텍처레벨에서 재사용 가능한 설계가 이루어 질 수 있다. 생산방식에서는 기존의 설계정보를 정형화 하여 재사용하는 문제점과 플랫폼을 기반으로 소프트웨어를 설계하기 위한 문제를 해결할 수 있는 Extractive방식의 PLE 구조를 채택하였다. 따라서 본 과제의 결과는 DMI 구조를 기반으로 설계단계에서 아키텍처레벨의 설계를 하게 되고, 컴포넌트 설계에서는 좁은 의미의 설계가 가능하게 하였다. 또한 설계정보는 플랫폼에 독립적인 재사용 가능한 부품자산으로써 재사용성과 조립성을 고려하여 아키텍처 컴포넌트를 정제하여 보관할 수 있으며, 부품자산은 설계자의 설계목적에 따라 유연하며 독립적으로 부품 합성이 가능하다.

DMI 아키텍처는 재사용가능한 자산정보를 활용할 수 있는 효율적인 생산구조를 갖게 된다. IoT를 기반으로 다양한 플랫폼을 요구하고 있으며 또한 새로운 플랫폼이

구성되고 있다. DMI는 이와 같은 시장의 변화에서 다양한 플랫폼의 개발단계에서 정형화되지 못한 설계정보를 통합할 수 있게 된다. 또한 진화되는 플랫폼과 기존의 플랫폼을 기반으로 설계된 설계정보를 자산화하여 재사용할 수 있는 기반을 마련하게 된다. 소프트웨어 시장은 과거와 다르게 다양한 플랫폼을 요구받게 되고 이를 기반으로 다양한 변형제품을 요구하고 있다. 이는 다양한 플랫폼을 구성할 수 있는 배경을 만들게 되며 각 플랫폼으로부터 구성된 설계정보를 자산화 필요로 하게 된다. 그리고 소프트웨어의 생산라인에 대한 변화를 가져오게 되며 설계정보의 자산을 재구성하여 다양한 플랫폼의 진화와 빠른 시장적응력을 갖기고 생산성을 높이는 효과를 얻을 수 있다.

ACKNOWLEDGMENTS

이 논문은 2016학년도 신한대학교 학술연구비 지원에 의해 수행된 연구임.

REFERENCES

[1] Y. J. Kim, “Convergence of Business Information System Process using Knowledge-based Method,” *Journal of the Korea Convergence Society*, Vol. 6, No. 4, pp. 65-71, Aug. 2015.

[2] S. Y Min, S. H Park and N. H Lee, “SW Quality of Convergence Product: Char acter istics, Improvement Str ategies and Alternatives,” *Journal of IT Convergence Society for SMB*, Vol. 1, No. 1, pp. 19-28, Nov. 2011.

[3] J. M. Thompson and M. P. E. Heimdahl, “Structuring Product Family Requirements for N-Dimensional and Hierarchical Product Lines,” *Requirements Eng,* Vol. 8, No. 1, pp. 42-54, Jan. 2002.

[4] N. I. Altintas, S. Cetin, A. H. Dogru and H. Oguztuzun,, “Modeling Product Line Software Assets Using Domain-Specific Kits, Software Engineering,” *Journal of IEEE Transactions on Software Engineering*, Vol. 38, No. 6, pp. 1376-1402, Nov. 2011.

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

[6] H. S. Min, “Deadlock Detection of Softwar e System Using UML State Machine Diagram,” *Journal of IT Convergence Society for SMB*, Vol. 1, No. 1, pp. 75-83, Nov. 2011.

[7] Y. W. Kim, “A study on Convergent & Adaptive Quality Analysis using DQnA model”, *Journal of the Korea Convergence Society*, Vol. 5, No. 4, pp.21-25, Dec. 2014.

[8] N. I. Altintas, S. Cetin, A. H. Dogru, H. Oguztuzun, “Modeling Product Line Software Assets Using Domain-Specific Kits,“ *Journal of IEEE Transactions on Software Engineering*, Vol. 38, No. 6, pp. 1376-1402, Nov. 2011.

[9] R. K. Keller and R. Schauer, “Design Components: Towards Software Composition at the Design Level,” *Proceedings of the 20th International Conference on Software Engineering*, pp. 302-311, 1998.

[10] H. Y. Choi and S. H. Sim, “A Study on Software Development method based on DMI,” *Journal of IT Convergence Society for SMB*, Vol. 2, No. 1, pp. 359-360, Jun. 2015.

[11] A. Ben, Y. Younes, B. Hlaoui, L. Jemni and B. Ayed, "A Meta-Model Transformation from UML Activity Diagrams to Event-B Models," *Proceedings of the 2014 IEEE 38th International Computer Software and Applications Conference Workshops (COMPSACW)*, pp. 740-745, Jul. 2014.

[12] R. Motschnig-Pitrik and J. Kassboll, "Part-Whole Relationship Categories and Their Application in Object-Oriented Analysis," *Journal of IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, Issue 5, pp. 779-797, Sep. 1999.

[13] S. Bernardi, J. Merseguer and D. C. Petriu, "Dependability modeling and analysis of software systems specified with UML," *Journal of ACM Computing Surveys*, Vol. 45, No. 1, pp. 2, Nov. 2012.

[14] S. K. Kim and J. E. Hong, “Application of Safety Analysis and Management in Software Development Process,” *Journal of IT Convergence Society for SMB*, Vol. 6, No. 1, pp. 7-15, Mar. 2016.

[15] J R. Abrial, *Modeling in Event-B: system and software engineering [M]*, Cambridge University Press, 2010.

[16] M. Klettke, H. Meyer, "XML and Object-Relational Data-base Systems-Enhancing Structural Mappings Based on Statistics," *Lecture Notes in Computer Science*, Vol. 1997, pp. 151-170, 2001.

저 자 소 개

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 교수

<관심분야> : 재사용, 아키텍처설계, 프로덕트공학