학습관리시스템에서 관점지향 소프트웨어 개발방법론의 적용

박종각○ 박옥자 유철중 장옥배
전북대학교 컴퓨터과학과
{canego○, cjyoo, okjang}@chonbuk.ac.kr, ojpark@bsc.ac.kr

Applying Aspect-Oriented Software Development Methodology in Learning Management System

Jong-Kack Park○ Oak-Cha Park Cheol-Jung Yoo Ok-Bae Jang
Dept. of Computer Science, Chonbuk National University

요 약

유스케이스(Use Case)를 기반으로 하는 관점지향 소프트웨어 개발방법론(Aspect-Oriented Software Development, AOSD)은 사용자 중심의 시스템을 설계하는데 있어 매우 중요한 소프트웨어 개발방법론으로 부각되고 있다. 학습관리시스템(Learning Management System, LMS)은 사이버교육을 지원하는 핵심시스템이며, 다양한 요구사항을 제시하는 청소년층을 대상으로 하기 때문에 높은 유지보수성과 확장성을 필요로 한다. 본 논문에서는 관점지향 소프트웨어개발 방법론을 6단계 모델링 방법을 통해 사이버 교육을 위한 학습관리시스템에 적용하였다. 적용 결과 학습관리시스템에서 횡단관심사의 모델링을 통한 관점지향 소프트웨어 개발은 유스 케이스 중심으로 모델링되어 사용자 중심의 시스템 유지보수성과 재사용성을 높일 수 있음을 보였다.

1. 서 론

최근 들어 소프트웨어 개발방법은 관점지향 소프트웨어 개발방법론(Aspect-Oriented Software Development, AOSD) 등 사용자 중심의 소프트웨어개발방법론이 부각되고 있다[1]. 관점지향 소프트웨어 개발방법론의 주 목적은 소프트웨어 개발 초기(early aspect)에 횡단관심사(crosscutting concern)들을 추출한 후 개발기간 동안 정련하고, 확장하고 관리함으로써 모듈성(modularity)과 유지보수성(maintainability) 및 재사용성(reusability)을 향상시키는 것이다[2].

학습관리시스템(Learning Management System, LMS)은 학습자의 시간적·공간적 제약을 해소하기 위해 사이버에서의 교육을 지원하는 시스템이다. 또한 본 논문에서 적용한 LMS는 초등학생과 중학생 및 고등학생 등 청소년층을 대상으로 하므로 매우 많은 요구사항 변경 요청이 수반되고 그에 따른 높은 유지보수성과 확장성을 필요로 하게 되므로 AOSD 방법론을 본 시스템에 적용하는 것은 매우 효과적이라고 볼 수 있다.

따라서 본 논문에서는 AOSD 6단계 모델링 방법을 제안하고 이 방법을 LMS에 적용하였다. 제1단계에서는 요구사항을 분석하고, 제2단계에서는 횡단관심사를 식별한다[4,5]. 제3단계에서는 유스 케이스(Use Case) 행위를 할당하고[1], 제4단계에서는 유스 케이스 슬라이스를 조직하며[1], 제5단계에서는 인터페이스를 설계하고[1], 마지막 단계에서는 추적테이블을 매핑한다.

2. 관련 연구

2.1 Elisa의 횡단관심사 식별에 관한 연구

Elisa의 연구는 애스펙트(aspect)로 부르는 횡단관심사의 식별에 관련된 연구이며, 이 연구에서는 하나 이상의 관심사에 의해서 분리되는 횡단관심사의 식별방법을 정의하였다[4].

2.2 Isabel의 횡단관심사 명세에 관한 연구

Isable의 연구는 횡단관심사 명세에 관련된 연구이며, 이 연구에서는 4개의 작업단계에 걸쳐 횡단관심사를 식별하고 명세하는 방법을 제시하였다[5].

2.3 Jacobson의 AOSD 모델링에 관한 연구

Jacobson은 유스케이스를 사용한 AOSD 모델링에 관련된 연구이며, 이 연구에서는 유스케이스를 분석하고 설계하는 4단계 모델링 방법을 제시하였다[1].

위 관련연구 결과 기존의 연구들은 객체지향 소프트웨어 개발방법론(Object-Oriented Software Development, OOSD)에 의존하고 있거나, 또는 AOSD 기반 연구에서도 횡단관심사를 식별하고 이를 기초로 시스템을 모델링하는 일련의 작업들이 유기적인 연관관계를 가지고 있지 않아 실무에 그대로 적용하기에 미흡한 점이 있다. 따라서 본 연구에서는 실무 적용을 위한 통합된 AOSD 6단계 모델링 방법을 제안한다. 위 관련 연구의 모델링 방법을 본 연구에 적용한 단계는 표 1과 같다.

표 1 관련 연구 적용 단계

저자	관련 연구	연구 방법	적용 단계
Elisa[4]	Crosscutting Concern Identification	with UML	Step 2
Isabel[5]	Crosscutting Concern Specification	with Template	Step 2
Jacobson[1]	AOSD Modeling	with Use Case	Step 3-5

3. 모델링 방법론 설계

본 연구는 Jacobson[1]의 방법을 기반으로 하였으며, Step 2의 횡단관심사 식별은 Elisa[4]의 방법을 참고하였으며, Step 2의 횡단관심사 명세는 Isabel[5]의 방법을 추가하였고, Step 3~Step 5에서는 Jacobson[1]의 방법을 기반으로 하였다. Step 1 및 Step 6은 새롭게 추가하여 6단계 모델링 방법을 제안한다.

3.1 AOSD 6단계 모델링 방법론

본 장에서 제안하는 AOSD 6단계 모델링 방법은 요구사항 분석에서 추적테이블 매핑까지 일련의 모델링 단계를 제안하고 있으며, 그 단계별 내용은 다음과 같다.

(Step 1) 요구사항 분석

요구사항 분석 단계에서는 요구사항 리스트에 근거하여 유스케이스를 식별하며, 산출물로 요구사항 리스트(Requirements List, RL)와 유스케이스 다이어그램(Use Case Diagram, UCD)을 생성한다.

요구사항 분석과 유스 케이스 식별은 소프트웨어 개발 초기에 가장 중요하므로 본 연구에서는 Jacobson의 방법[1]에서 생략되어 있는 요구사항 분석 단계를 추가하였다.

(Step 2) 횡단관심사 식별

Step 1의 결과 분석을 통해 횡단관심사를 식별(Crosscutting Concerns Identification, CCI)하며, 산출물로 횡단관심사 명세(Crosscutting Concerns Specification, CCS)와 유스케이스 명세(Use Case Specification, UCS)를 생성한다.

횡단관심사의 식별은 관점지향 소프트웨어 개발방법에서 핵심 유스 케이스를 추출하는 것이므로, 본 연구에서는 Elisa의 방법을 참고로 하여[4], 유스 케이스간의 상호작용 분석을 통해 Jacobson의 방법[1]에서 생략되어 있는 횡단관심사 식별단계를 추가하였다.

(Step 3) 유스케이스 행위 할당

Step 2의 결과 분석을 통해 클래스(Class)를 식별하고 행위를 할당하며, 산출물로 클래스 식별표(Class Identification, CI)과 순차다이어그램(Sequence Diagram, SD)을 생성한다.

클래스 식별과 순차다이어그램 생성은 유스 케이스의 역할과 행위를 파악하는 중요한 단계로 Jacobson의 방법을 적용하였다[1].

(Step 4) 유스케이스 슬라이스 조직

Step 2와 Step 3의 결과 분석을 통해 유스케이스 슬라이스를 조직하며, 산출물로 유스케이스 슬라이스 템플릿(Use Case Slice Template, UCST)을 생성한다.

유스 케이스 슬라이스를 조직하는 것은 횡단관심사를 추적하고 인터페이스 설계를 위한 기초단계로 Jacobson의 방법을 적용하였다[1].

(Step 5) 인터페이스 설계

Step 2와 Step 3 및 Step 4의 결과 분석을 통해 인터페이스를 설계하며, 산출물로 인터페이스 디자인(Interface Design, ID)을 생성한다.

인터페이스 설계는 관점지향 소프트웨어 개발방법으로 모델링된 'LMS' 를 구현하는 이전 단계로 Jacobson의 방법을 적용하였다[1].

(Step 6) 추적성 매핑

Step 1에서 Step 5까지의 결과 분석을 통해 상호 강한 연관 관계를 추적하며, 산출물로 추적테이블(Tracing Table, TT)과 추적성 맵(Traceability Map, TM)을 생성한다.

횡단관심사를 추적하는 것은 관점지향 소프트웨어 개발 기간 중 횡단관심사가 식별되고 설계되는 과정을 추적하여 확장성과 유지보수성을 높이고자 하는 것으로, 본 연구에서는 Jacobson의 방법[1]에서 생략되어 있는 추적성 매핑 단계를 추가하였다.

3.2 결과 고찰

AOSD 6단계 모델링 방법론은 요구사항 분석에서 인터페이스 설계까지 일련의 과정을 유스 케이스를 중심으로 제안하고 있어, 개발자 및 사용자가 시스템을 파악하기 쉬우며, 특히 사용자의 소프트웨어 이해도가 높아져서 개발과정에서의 적극적 개입이 유도되고, 추적매핑에 의한 유지보수성이 신장되어 개발과정 및 완료 후 사용자의 요구사항 변경에 매우 탄력적으로 대처할 수 있을 것이다.

4. 적용 사례

청소년층을 대상으로 하는 사이버 교육을 지원하기 위한 LMS는 개발 후에도 요구사항의 잦은 변경 요청이 있을 것으로 생각되고, 소프트웨어는 유지보수성(maintainability)과 확장성(extensibility)을 갖춘 탄력적인 소프트웨어 개발이 요구된다. 따라서 사이버 교육을 위한 LMS 모델링에 제3장에서 제안한 AOSD 6단계 모델링 방법을 적용한다.

4.1 AOSD 6단계 모델링 방법론 적용

이번 장에서는 제3장에서 제안한 AOSD 6단계 모델링 방법을 LMS에 적용한다. 일반적으로 사이버교육 시스템(Cyber Learning System, CLS)은 4개의 패키지(package)

로 구성되어 있으며, 그 내용은 표 2와 같다.

표 2 'CLS'의 패키지와 유스 케이스

패키지	유스 케이스
LMS	Learning, Testing, Boarding, Authorizing, Logging
LCMS	Content Developing, Managing, Logging, Authorizing
TPMS	Testing, Test Developing, Logging, Authorizing
CMS	Boarding, Chatting, Logging, Authorizing

표 2에 제시된 패키지 중 가장 핵심적인 패키지인 'LMS' 패키지는 4개의 유스 케이스와 요구사항들로 구성되어 있으며, 그 내용은 표 3과 같다.

표 3 'LMS'의 유스 케이스와 요구사항

패키지	유스 케이스	요구사항
LMS	Learning	learning, taking lectures, query, report
	Testing	selection, solving, evaluation
	Boarding	reading, writing, modification, chatting
	Authorizing	log-in, certification, authorizing
	Logging	connection, hitting, statistic

AOSD 6단계 모델링 방법을 LMS에 단계별로 적용한 내용은 다음과 같다.

(Step 1) 요구사항 분석

Step 1에서는 사용자 요구사항을 문서와 면담으로 분석하여 요구사항 리스트(RL)와 유스 케이스 다이어그램(UCD)을 생성하였으며, 'Learning' 유스 케이스의 요구사항 리스트에 기술한 내용을 요약하면 표 4와 같고, 표 4에 기술된 요구사항을 근거로 작성된 유스 케이스 다이어그램은 그림 1과 같다.

표 4 RL of 'Learning' 유스 케이스

RL - <01> Use Case : Learning
1. Learner can study by on-line. 2. Learner can taking lecture. 3. Learner can query something. 4. Learner can submit the report.

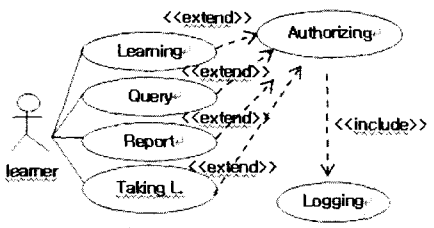

그림 1 UCD of 'Learning' 유스 케이스

(Step 2) 횡단관심사 식별

Step 2에서는 Step 1에서 분석된 요구사항을 근거로 유스 케이스들간의 상호작용 분석을 통해 2개 이상의 유스 케이스와 상호작용을 하고 있는 'Authorizing'과 'Logging'을 횡단관심사로 식별(CCI)하였으며, 그 내용은 표 5와 같다.

표 5 CCI of LMS 패키지

CCI - <01> Package : LMS					
	Learning	Testing	Boarding	Authorizing	Logging
Learning	-	X	X	○	○
Testing	X	-	X	○	○
Chatting	X	X	X	○	○
Boarding	X	X	-	○	○
Authorizing	○	○	○	-	○
Logging	○	○	○	○	-

※ 상호작용 식별(○: interact, X: not interact)

표 6 CCS of 'Authorizing' 유스 케이스

CCS - <01> *Aspect* : Authorizing	
Name	Authorizing
Source	Business process
Stakeholders	Learner, Teacher, Administration
Description	Log-in, certification, authorizing
Classification	Non Functional Concern
Contribution	Positive(+)
Priority	Very important
Required Concerns	Learning, Testing, Chatting, Boarding, Logging

표 7 CCS of 'Logging' 유스 케이스

CCS - <01> *Aspect* : Logging	
Name	Logging
Source	Business process
Stakeholders	Learner, Teacher, Administration
Description	Connection, hitting, statistic
Classification	Non Functional Concern
Contribution	Positive(+)
Priority	Important
Required Concerns	Learning, Testing, Chatting, Boarding, Authorizing

또한 식별된 횡단관심사에 대한 명세(CCS)는 표 6 및 표 7과 같으며, 또한 횡단관심사 명세에 따른 Authorizing 유스 케이스에 대한 명세(UCS)는 표 8과 같다.

표 8 UCS of 'Authorizing' 유스 케이스

UC - <No> Use Case : Authorizing
Basic Flows 1. Learner log in LMS. 2. Learner is authorized for taking lecture. 3. Learner can take a certificate for achievement.
Alternate Flows 1. If learner isn't authorized, go to the initial page.
Extension Flows *EF1. Logging Authorize*
Extension Points *EP1. Learning Authorize*

(Step 3) 유스 케이스 행위 할당

Step 3에서는 Step 2에서 명세된 유스 케이스 명세를 근거로 'Learning' 유스 케이스에서 클래스를 식별(CI)하고 클래스별 역할(role)을 UML 표기법에 의해 부여하였는데[2], 경계 클래스(boundary class)는 'LearningForm' 클래스, 제어 클래스(controller class)는 'LearningHandler' 클래스, 그리고 개체 클래스(entity class)는 'Learning' 클래스와 'Authorizing' 클래스 및 'Logging' 클래스이며, 그 내용은 그림 2와 같다. 또한 클래스별 역할을 바탕으로 클래스간의 상호작용을 시퀀스 다이어그램(SD)으로 순차적으로 나타냈는데, 그 내용은 그림 3과 같다.

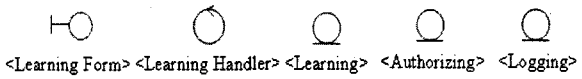

그림 2 CI of 'Learning' 유스 케이스

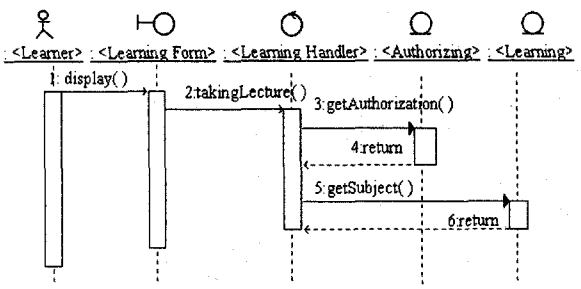

그림 3 SD of 'Learning' 유스 케이스

(Step 4) 유스 케이스 슬라이스 조직

Step 4에서는 Step 2와 Step 3을 근거로 클래스와 패키지를 조직화한 유스 케이스 슬라이스 템플릿(UCST)을 작성하였는데, 이 슬라이스에서는 'Learning' 유스 케이스에서 'Authorizing' 애스펙트 클래스(Aspect Class)와 'LearningHandler' 클래스를 포함한 협력 관계(collaboration relation)를 확장 포인트(expension points)를 통해 보여주고 있으며, 그 내용은 그림 4와 같다.

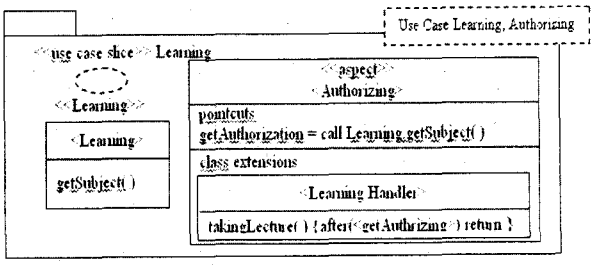

그림 4 UCST of 'Learning' 유스 케이스

(Step 5) 인터페이스 설계

Step 5에서는 Step 2, 3, 4를 근거로 'Learning' 유스 케이스 관련 컴포넌트(component)를 구성하고, 컴포넌트간의 인터페이스를 설계했는데, 'Learning' 유스 케이스에서 컴포넌트간의 인터페이스 설계(ID)는 그림 5와 같다.

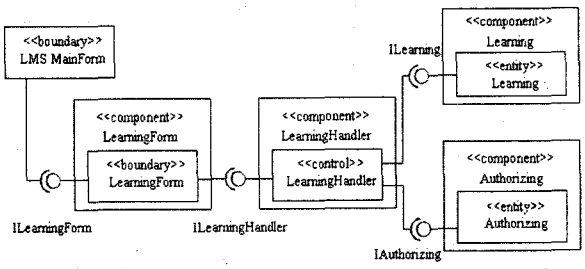

그림 5 ID of 'Learning' 유스 케이스

(Step 6) 추적성 매핑

Step 6에서는 요구사항 추적성을 Step간의 상호 연관 테이블로 나타냈는데, 그 내용은 표 9와 같다. 또한 Step간의 추적관계를 추적성 맵(traceability map)으로 나타냈는데, 그 내용은 그림 6과 같다.

표 9 단계별 추적 테이블

	모델링 단계					
	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
Step 1	-	●	△	△	△	X
Step 2	●	-	●	○	○	△
Step 3	△	●	-	●	○	△
Step 4	△	○	●	-	●	△
Step 5	△	○	○	●	-	△
Step 6	X	△	△	△	△	-

※ 연관성 (●: very strong, ○: strong, △: weak, X: none)

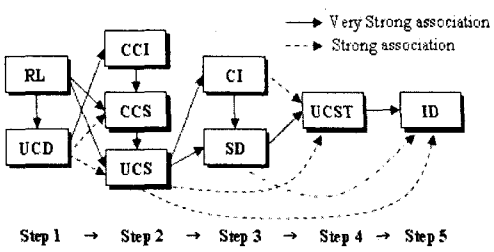

그림 6 추적성 맵

4.4 적용결과 분석

사이버 교육을 위한 LMS에 AOSD 6단계 모델링 방법을 적용한 결과는 다음과 같다. 첫째 관점지향 소프트웨어 개발을 통해 사용자 편의성과 이해성 신장이 가능하다. 둘째 횡단관심사 관련 결합도가 낮아지고 응집도가 높아져 유지보수성 신장이 가능하다. 셋째 추적성 매핑을 통해 단계별 추적이 가능해 유지보수성과 확장성이 높아진다. 마지막으로 AOSD 절차가 일원화되고 간소화되어 효율적인 소프트웨어 개발이 가능하다.

5. 결론

LMS는 청소년 층을 대상으로 하는 사이버교육 시스템으로 다양한 요구사항과 지속적인 요구사항 변경이 수반된다. 관점지향 소프트웨어 개발방법론은 사용자 중심의 방법론으로 부각되고 있다. 따라서 본 논문에서는 Jacobson, Isabel, Elisa의 모델링 방법을 기초로 하여 AOSD 6단계 모델링 방법을 횡단관심사를 중심으로 'LMS' 패키지의 'Learning' 유스 케이스에 적용해 보았다. 적용 결과 관점지향 소프트웨어 개발은 유스 케이스 중심으로 모델링 되어 사용자 중심의 개발 방법론임을 보였으며, 'LMS'에서 횡단관심사 모듈화를 통해 유지보수성과 재사용성을 높일 수 있음을 보였다.

또한 스테이크홀더(stakeholder) 관점에서 관점지향 소프트웨어 개발방법을 객체지향 소프트웨어 개발방법(OOSD)과 비교하면, 개발자 관점에서는 애스펙트 클래스(Aspect Class)의 추가로 클래스 사이의 복잡도가 다소 증가하는 편이나 요구사항 변경에 따른 수정용이성과 개발 절차의 단순화 등에 따른 유지보수성의 증가로 개발 및 유지보수에 따른 소요기간의 단축효과를 얻을 수 있었으며, 사용자 관점에서는 소프트웨어 아키텍처에 대한 이해가 용이하여 개발과정에서의 사용자의 참여기회가 확대되어 요구사항 반영이 용이해 질 수 있었다.

본 연구 결과 관점지향 소프트웨어 개발방법은 사용자 중심의 소프트웨어 개발에 있어 모듈성과 유지보수성 및 재사용성을 높여줄 것으로 기대된다. 향후 연구에서는 횡단관심사 명세기법, 횡단관심사 슬라이싱 기법, 횡단관심사 추적기법 등에 관한 연구를 진행하고자 한다.

6. 참고문헌

[1] Ivar, Jacobson., Pan-Wei, Ng., "Aspect-Oriented Software Development with Use Case", Addison Wesley, 2005.

[2] Georgia, S., Sergio, S., Paulo, B., Jaelson, C., "Separation of Crosscutting Concerns from Requirements to Design", Aspect-Oriented Requirements Engineering and Architecture Design Workshop, pp. 93-102, 2004.

[3] Chethana, K., Armin, E., "Aspect-Oriented Requirements Engineering for Software Product Lines", ECBS'03. Proceedings of the 10thIEEE International Conference pp. 673-676, 2003.

[4] Elisa, B., Siobhan, C., "Finding Aspects In Requirements with Theme/Doc", Aspect-Oriented Requirements Engineering and Architecture Design Workshop, pp. 15-22, 2004.

[5] Isabel, B., Ana, M., "Integrating the NFR framework in a RE model", Aspect-Oriented Requirements Engineering and Architecture Design Workshop, pp. 27-32, 2004.

[6] Ivar, Jacobson., "Use Cases and Aspects-Working Seamlessly Together", In Journal of Object Technology, Vol. 2. No. 4. pp. 7-28, 2003.

[7] Joseph, Schmuller., "Teach Yourself UML in 24 Hours. 3rd Ed", Sams, 2004.

[8] Ramnivas, Laddad., "AspectJ in Action", Manning, 2005.

[9] Bedir, T., Ana, M., "Aspect-Oriented Requirements Engineering and Architecture Design", Aspect-Oriented Requirements Engineering and Architecture Design Workshop, pp. 4-14, 2004.

[10] Tomson, J.M. and Heimdahl, M.P.E., "Structuring Product Family Requirements for n-Dimensional and Hierarchical Product Lines", Requirements Engineering, Vol. 8. pp. 42-54, 2003.

[11] France, R., Ray, I., Georg, G., Ghosh, S., "Aspect-Oriented approach to early design modeling", IEE Proceedings online, Vol. 151. No. 4. pp.173-185, 2004

[12] Craig, Larman, "Applying UML and Patterns. 2nd ED", Prentice Hall PTR, 2002.

[13] Ishio, T., Kusumoto, S., Inoue, K., "Program slicing tool for effective software evolution using aspect-oriented technique", IWPSE'03, Proceedings of the Sixth International Workshop on Principle of Software Evolution, 2003