횡단관심사 추적을 위한 관점지향 슬라이싱 기법
(An Aspect-Oriented Slicing Technique Tracing Crosscutting Concern)

박 종 각 † 박 옥 자 †† 유 철 중 †††
(Jong-Kack Park) (Ok-Cha Park) (Cheol-Jung Yoo)

요 약 관점지향 소프트웨어 개발방법(AOSD)은 시스템을 관심사별로 분류하고 횡단관심사를 식별하여 애스팩트 클래스로 구성함으로써 높은 유지보수성이 요구되는 사용자 중심의 시스템 설계를 위한 중요한 소프트웨어 개발방법으로 부각되고 있다. 그러나 기존의 관점지향 소프트웨어 개발방법에 대한 연구들은 횡단관심사의 식별과 명세에 대한 연구들과 애스팩트 클래스를 중심으로 하는 시스템 구현에 대한 연구들이 대부분이며, 식별된 횡단관심사의 추적 방법에 대한 연구들은 매우 미흡하였다. 따라서 본 연구에서는 AOSD를 기반으로 시스템 모델링 단계에서의 횡단관심사 추적을 위한 관점지향 슬라이싱 기법을 제안하였다. 유스 케이스들 간의 상호작용 분석을 통해 횡단관심사를 식별하여 명세한 후 식별된 횡단관심사 추적을 위한 슬라이싱 기법을 제시하였으며, 사례연구를 위해 학습관리시스템(LMS)에 이 기법을 적용하였다. LMS는 시스템 기능의 잦은 변경 및 확장 요구가 많은 사이버교육 시스템으로 이 시스템의 개발에 유지보수성을 높이는 AOSD를 적용하는 것은 매우 효과적인 방법이다. 그리고 마지막으로 기존 연구와의 비교분석 결과를 제시하였다.

키워드 : 관점지향, 횡단관심사, 슬라이싱, 추적성, 학습관리시스템

Abstract Aspect-Oriented Software Development(AOSD) is the software development methodology that classifies concerns of the system and identifies crosscutting concerns and organizes aspect class, and AOSD has emerged as an important user-oriented software development methodology with high maintainability. However, most of related studies worked on identification and specification and coding of crosscutting concerns. And there have been few studies on slicing technique for tracing method of specified crosscutting concerns. Therefore, this paper proposes slicing technique based on AOSD that identifies and specifies crosscutting concerns through interactive analysis between use cases; also, suggests slicing technique which improves traceability centering on identified crosscutting concerns; applies the aspect-oriented slicing technique to Learning Management System(LMS) for case-study. LMS is a cyber educational system that demands a lot of functional changes and expansion, so it is effective to apply AOSD with high maintainability in developing LMS; and shows the results that compared with related studies by comparing six elements.

Key words : Aspect-Oriented Crosscutting Concern, Slicing, Traceability, LMS

† 정 회 원 : 전라북도교육청 과학정보교육과 장학사
canego@chonbuk.ac.kr
†† 정 회 원 : 숭실대학교 정보미디어기술연구소 전임연구원
ojpark@otlab.ssu.ac.kr
††† 종신회원 : 전북대학교 응용시스템공학부(정보공학) 교수
cjyoo@chonbuk.ac.kr
논문접수 : 2008년 7월 18일
심사완료 : 2008년 10월 16일

정보과학회논문지: 소프트웨어 및 응용 제35권 제12호(2008.12)

1. 서 론

객체지향 소프트웨어 개발방법(Object-Oriented Software Development, OOSD)과 더불어 최근에는 유스 케이스를 활용한 사용자 중심의 관점지향 소프트웨어 개발방법(Aspect-Oriented Software Development, AOSD)이 부각되고 있다[1]. AOSD의 목적은 소프트웨어 개발 초기에 횡단관심사를 식별한 후 소프트웨어 전 생명주기에 걸쳐 정련하고, 확장하고, 관리함으로써 모듈성과 재사용성을 향상시키며, 시스템 이식 후에는 시스템의 기능 변경 및 기능 확장에 따른 유지보수성을

높이는데 있다[2]. 기존의 연구들은 AOSD의 초기 단계에서 횡단관심사의 식별과 명세에 대한 연구들과 명세된 횡단관심사의 구현에 관한 연구정도에 그치고 있다.

추적성은 시스템 요구사항들이 요구사항 분석과 명세 및 구현 과정에서 어떻게 반영되고 있는가를 나타내는 것이며, 횡단관심사는 AOSD에서 가장 핵심적인 요소이다. 따라서 소프트웨어 개발주기에 따른 횡단관심사의 추적은 AOSD에서 매우 중요하다. 그러나 기존의 연구들은 추적다이어그램을 기반으로 하는 횡단관심사 추적에 관한 연구는 이루어지지 않았다.

슬라이싱은 시스템 설계 및 구축을 위해 구현된 모델에서 요구사항의 반영과 질의 및 시스템 분석 등을 통해 유지보수를 지원하는 모델링 기법이며[3]. 관점지향 슬라이싱은 UML(Unified Modeling Language)을 기반으로 패키지 내에서 유스 케이스별로 관심사들을 분류한 후, 공통관심사인 횡단관심사를 식별하고 애스팩트 클래스의 포인트컷과 핵심클래스의 조인포인트를 추출하여 연관관계를 명세하는 모델링 기법이다.

따라서 본 연구에서는 AOSD를 기반으로 횡단관심사 추적을 위한 슬라이싱 기법을 제시하였다. 먼저 유스 케이스들 간의 상호작용 분석을 통해 횡단관심사를 식별하고 명세하는 기법을 제시한 후 슬라이스 템플릿을 제시하였으며, 이를 토대로 횡단관심사를 추적하는 기법을 제시하였다.

본 연구에서는 사례연구를 위해 관점지향 슬라이싱 기법을 학습관리시스템(Learning Management System, LMS)에 적용하였다. LMS는 학습자의 시·공간적 제약을 해소하기 위해 사이버교육을 지원하는 온라인 기반 시스템이며, 이식 후 사용자들의 많은 기능 변경과 확장 요청을 하는 시스템으로 그에 따른 높은 확장성과 유지보수성을 필요로 한다[4]. 따라서 사이버 교육을 위한 LMS에 AOSD를 적용하는 것은 매우 효과적이라고 볼 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 제시하였고, 3장에서는 횡단관심사 추적을 위한 관점지향 슬라이싱 기법을 제시하였으며, 4장에서는 관점지향 슬라이싱 기법을 LMS에 적용한 사례를 제시하였고, 5장에서는 기존 연구와의 비교분석 결과를 제시하였으며, 마지막으로 결론과 향후연구를 제시하였다.

2. 관련 연구

Elisa의 연구[5]는 애스펙트(aspect)로 정의되는 횡단관심사의 식별과 관련된 연구이며, 이 연구에서는 하나 이상의 관심사에 의해서 분리되는 횡단관심사의 식별방법을 유스 케이스를 기반으로 정의하였으나, 슬라이스 명세와 식별된 횡단관심사의 추적에 대한 연구는 미흡하였다.

Jacobson의 연구[1]는 유스 케이스를 사용한 AOSD 모델링에 관련된 연구이며, 이 연구에서는 시스템 관심사들을 유스 케이스를 기반으로 분석하고 설계하는 관점지향 4단계 모델링 방법을 제시하였으나, 횡단관심사의 추적에 대한 연구는 미흡하였다.

Ramesh의 연구[6]는 추적성에 관련된 연구이며, 시스템 요구사항들을 기반으로 하여 시스템을 개발하는 생명주기 동안에 생성된 산출물들과 이에 해당되는 원시코드들을 명확하게 연결하는 방법을 시스템의 특성 중의 하나로 정의하고 있고, 또한 시스템의 주요 기능과 스테이크 홀더 및 프로젝트의 크기와 추적도 간의 관계를 제시하였으나, 객체지향 소프트웨어 개발방법론(Object-Oriented Software Development, OOSD) 기반의 연구이고 횡단관심사와 슬라이싱에 대한 연구는 미흡하였다.

Ishio의 연구[3]는 슬라이싱에 기초한 프로그램 디버깅 툴을 개발하는데 있어 객체지향 프로그래밍(Aspect-Oriented Programing, AOP)을 적용한 연구이며, 4단계로 구성된 프로그램 슬라이싱 기법의 각각의 단계에서는 참조된 변수들을 추출하고, 데이타 및 흐름제어 종속성을 분석하며, 프로그램 종속 그래프를 작성하고, 슬라이스를 추출하는 단계를 제시하였으나, OOSD 기반의 연구이고 횡단관심사 및 추적에 대한 연구는 미흡하였다.

위 관련 연구 결과 기존의 연구들은 OOSD에 기반을 두고 있거나, 또는 각각의 연구들이 독립적으로 진행되어 시스템 개발 실무에서 효과적인 슬라이싱이나 추적이 매우 어렵다. 또한 식별된 횡단관심사를 중심으로 하는 슬라이싱 방법에 대한 연구가 미흡하고, 추적도를 기반으로 하는 명세된 횡단관심사에 대한 추적에 관한 연구가 이루어지지 않았다. 따라서 본 논문에서는 위와 같은 관련 연구들을 바탕으로 AOSD 모델링 단계에서의 횡단관심사에 대한 식별과 슬라이싱 및 추적들이 상호 연관된 체계적인 관점지향 슬라이싱 기법을 제안하였다.

3. 관점지향 슬라이싱 기법

3.1 프로세스 및 산출물

관점지향 프로그래밍(Aspect-Oriented Programing, AOP)에서 하나의 관심사는 단 하나의 애스팩트 클래스로 작성되며[7], AOP 접근 방법은 커스트 마이징된 자바가상머신(Java Virtual Machine, JVM)과 비교하면 상대적으로 비용이 절감되고, 성능이 향상된다[3]. AspectJ는 자바를 지원하는 관점지향 프로그래밍 언어이며, 원시코드 레벨의 클래스에서 객체와 애스팩트들로 구성된 애스팩트 직조 도구이다[3]. 따라서 본 논문에서는 관련된 클래스들을 직조하기 위해서 AspectJ를 사용

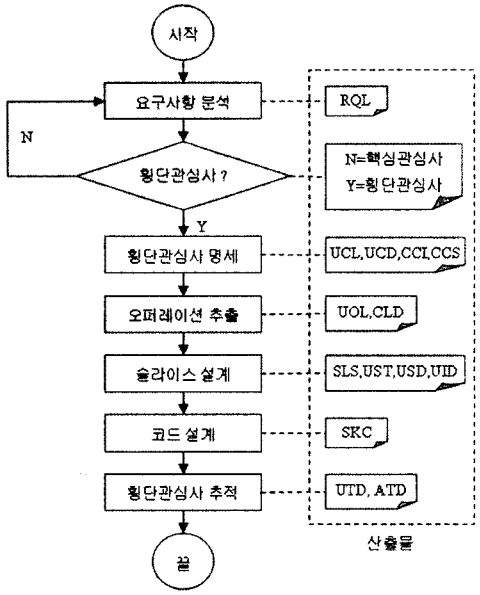

그림 1 관점지향 슬라이싱 프로세스

하였다. 또한 AOSD의 핵심요소인 횡단관심사 명세, 슬라이스 템플릿, 개략코드, 추적도 등을 체계적으로 매핑하였고, 매핑 단계별로 관점지향 슬라이싱을 위한 템플릿을 제시하였다. 관점지향 슬라이싱을 위한 프로세스는 그림 1과 같다.

횡단관심사 추적이란 식별되어 명세된 횡단관심사가 시스템 개발단계 내에서 어떻게 반영되고 있는가를 나타내는 것이며, 본 논문에서는 명세된 횡단관심사의 추적성을 높이기 위해 Phase별로 생성되는 산출물을 표 1에 제시하였다.

표 1 슬라이싱에 따른 Phase별 산출물

이전 산출물	
RQL	요구사항 리스트(ReQuirements List)
Phase 1에서의 산출물	
UCL	유스 케이스 리스트(Use Case List)
UCD	유스 케이스 다이어그램(Use Case Diagram)
CCI	횡단관심사 식별(Crosscutting Concern Identification)
CCS	횡단관심사 명세(Crosscutting Concern Specification)
Phase 2에서의 산출물	
UOL	유스 케이스 오퍼레이션 리스트(Use Case Operation List)
CLD	클래스 다이어그램(CLass Diagram)
Phase 3에서의 산출물	
SLS	슬라이스 명세(SLice Specification)
UST	유스 케이스 슬라이스 템플릿(Use Case Slice Templete)
USD	유스 케이스 순차 다이어그램(Use Case Sequence Diagram)
UID	유스 케이스 인터페이스 다이어그램(Use Case Interface Diagram)
Phase 4에서의 산출물	
SKC	개략 코드(SKeleton Code)
Phase 5에서의 산출물	
UTD	유스 케이스 추적 다이어그램(Use Case Tracing Diagram)
ATD	산출물 추적 다이어그램(Artifact Tracing Diagram)

표 2 명칭 표기법

구분	표기법	표기 예시
패키지	p_패키지명	p_LMS
유스 케이스	u_유스 케이스명	u_Learning
애스팩트 클래스	ac_클래스명	ac_Authorizing
핵심 클래스	c_클래스명	c_Logging

표 3 ATD 표기법

표기	의미	표기	의미
	다이어그램	——▶	매우 강한 연관 관계
	리스트	—·▶	강한 연관 관계
	명세서	·····▶	약한 연관 관계
	개략코드	n...n	연관 관계 비율
	상호작용 분석서	①..⑩	핵심 프로세스

또한 패키지와 유스 케이스 및 애스팩트 클래스와 핵심클래스 명칭간의 혼돈을 피하기 위해 표 2와 같이 표기하였으며, 관점지향 슬라이싱 기법에서 횡단관심사 추적을 위해 표 1의 산출물들을 기준으로 작성하게 되는 산출물 추적 다이어그램(ATD)의 표기법을 표 3에 제시하였다.

3.2 관점지향 슬라이싱

관점지향 슬라이싱은 UML을 기반으로 공통관심사인 횡단관심사를 식별하고 애스팩트 클래스의 포인트컷과 핵심클래스의 조인포인트를 추출하여 연관관계를 명세하는 모델링 기법이고, 슬라이스는 관점지향 슬라이싱을 통해 추출된 포인트컷과 조인포인트 및 어드바이스를 클래스간의 협력다이어그램을 통해 연관관계를 정의한 유스 케이스들의 집합이며, 관점지향 슬라이싱 개념도는 그림 2와 같다.

본 논문에서는 관심사들을 유스 케이스별로 분류하고 횡단관심사를 식별한 후 포인트 컷과 조인포인트 및 어드바이스를 각각의 슬라이스 템플릿에 정의하는 슬라이싱 기법을 제시하였으며, 횡단관심사의 추적을 위해 산출물을 기반으로 하는 추적다이어그램을 제시하였다. 프

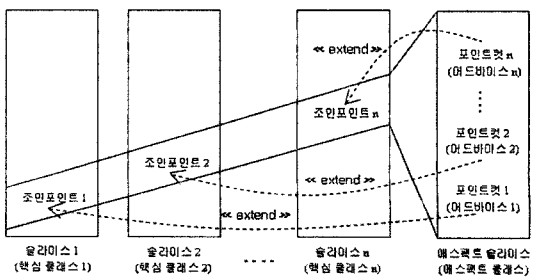

그림 2 관점지향 슬라이싱 개념도

로세스는 횡단관심사의 식별과 명세, 유스 케이스별 오퍼레이션 추출, 슬라이스 명세와 슬라이스 템플릿 작성, 시스템 구현을 위한 개략코드 설계와 횡단관심사 추적 등의 Phase들로 구성되었으며, 각 Phase별 슬라이싱 기법은 다음과 같다.

Phase 1. 횡단관심사 명세

Phase 1에서는 횡단관심사를 식별하고 명세한다. 산출물로는 유스 케이스 리스트(UCL)와 유스 케이스 다이어그램(UCD) 및 횡단관심사 식별(CCI) 템플릿과 횡단관심사 명세(CCS) 템플릿 등이 있다. 또한 생성된 CCS는 Phase 2의 클래스 다이어그램(CLD)의 입력자료로 사용된다.

수요자들의 요구사항을 분석하여 작성해 놓은 요구사항 리스트(RQL)를 기초로 유스 케이스를 추출하여 표 4의 UCL과 그림 3의 UCD를 작성한다.

표 4 UCL 템플릿

패키지	유스 케이스
패키지명	유스 케이스들을 나열

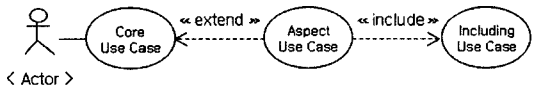

그림 3 UCD 템플릿

다음으로 유스 케이스 간의 상호작용 분석을 통해 상호작용이 매우 강한 관심사를 횡단관심사로 식별하여 표 5와 같이 작성한 후 식별된 횡단관심사를 표 6과 같이 명세한다.

표 6의 횡단관심사 명세는 메인 메소드를 갖는 'Basic Flows'와 예외 처리를 위한 'Alternate Flows' 그리고 초기 프로세스 후에 확장되어 수행될 'Extension Flows'

표 5 CCI 템플릿

패키지명		
	유스 케이스명 1	유스 케이스명 2
유스 케이스명 1	유스 케이스 간의 상호작용 표시	
유스 케이스명 2		

표 6 CCS 템플릿

애스펙트 클래스명
Basic Flows : 기본 프로세스 기술
Alternate Flows : 예외처리 프로세스 기술
Extension Flows EF1. 확장(포함) 프로세스 기술
Extension Points EP1. 확장(포함)을 위한 조인포인트 기술

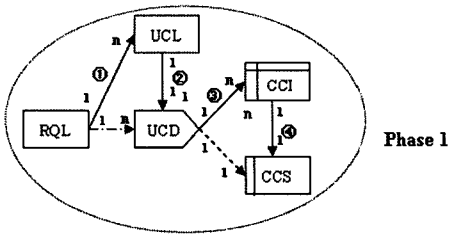

그림 4 Phase 1의 ATD

와 이전 프로세스에 대한 정보를 지닌 'Extension Points'들로 구분되어 명세된다.

Phase 1에서 식별되고 명세된 횡단관심사의 추적을 위해 산출물들을 기준으로 횡단관심사 중심의 산출물 추적 다이어그램(ATD)을 작성하면 그림 4와 같다.

Phase 2. 오퍼레이션 추출

Phase 2에서는 Phase 1의 CCS를 기초로 오퍼레이션을 추출하고, 클래스를 조직한다. 산출물로는 유스 케이스 오퍼레이션 리스트(UOL)와 클래스 다이어그램(CLD) 등이 있다. 또한 생성된 CLD는 Phase 3의 슬라이스 명세(SLS)의 입력자료로 사용된다.

유스 케이스별 오퍼레이션은 Phase 1의 RQL과 UCD에 기초하여 작성한 CCS를 바탕으로 유스 케이스별 행위 분석을 통해 추출하여 표 7의 UOL를 작성하며, UOL 및 CCS 분석을 통해 그림 5의 CLD를 작성한다.

표 7 UOL 템플릿

유스 케이스	주요 오퍼레이션
유스 케이스명	오퍼레이션들을 기술

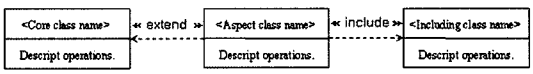

그림 5 CLD 템플릿

Phase 2까지 생산된 산출물들을 기준으로 ATD를 작성하면 그림 6과 같다.

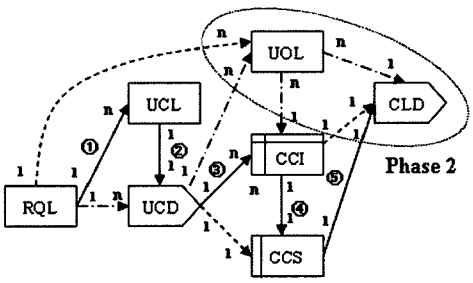

그림 6 Phase 2의 ATD

Phase 3. 슬라이스 설계

Phase 3에서는 Phase 2의 CLD를 기초로 슬라이스를 조직하고 명세한다. 산출물로는 슬라이스 명세(SLS)와 유스 케이스 슬라이스 템플릿(UST) 등이 있다. 또한 생성된 UST는 Phase 4의 개략 코드(SKC)의 입력 자료로 사용된다.

유스 케이스에서 추출된 오퍼레이션들을 중심으로 핵심클래스와 애스팩트 클래스별로 결합점을 선언하는 조인포인트와 교차점을 선언하는 포인트컷 및 충고를 선언하는 어드바이스를 포함하는 슬라이스를 조직하고 명세한다. 핵심 클래스에서는 조인포인트에 해당되는 오퍼레이션을 선언해 주고, 애스팩트 클래스에서는 포인트컷과 어드바이스에 해당하는 오퍼레이션을 선언해 줌으로써 구조적인 슬라이스를 조직하여야 한다. Phase 2의 CLD 및 Phase 1의 CCS를 바탕으로 작성하는 패키지별 SLS는 표 8과 같다.

표 8 SLS 템플릿

핵심 클래스	애스팩트 클래스	
조인포인트	포인트컷	어드바이스
조인포인트 오퍼레이션 기술	포인트컷 오퍼레이션 기술	어드바이스 행위 기술

또한 Phase 1의 CCS 및 Phase 2의 CLD 그리고 SLS를 기초로 UST를 그림 7과 같이 작성한다. 유스 케이스들이 협력관계(collaboration relation)에 있음을 점선으로 된 타원형으로 나타내고, 핵심 클래스와 애스팩트 클래스에서의 포인트컷 및 어드바이스를 표시한다. 또한 슬라이스 템플릿에 대한 노트를 점선으로 된 사각형 안에 표시한다. 각각의 항목에 대한 표기법은 Jacobson이 제안한 표기법을 따랐다[1]. UST는 횡단관심사에 대한 종합적 정보를 제공해 주고, 시스템 실행 프로세스를 내재함으로써 이후 생성될 유스 케이스 시퀀스 다이어그램(USD) 및 유스 케이스 인터페이스 다이어그램(UID)의 기초자료로 사용되며, 추적성을 향상시키는 매우 중요한 자료로 횡단관심사에 대한 추적 및 역추적 자료로 활용된다.

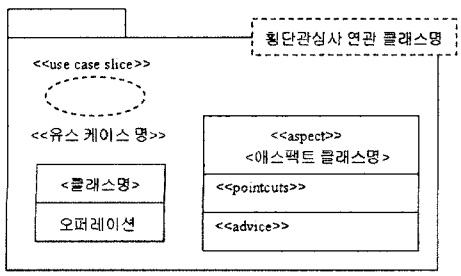

그림 7 UST 템플릿

Phase 3까지 생산된 산출물들을 기준으로 ATD를 작성하면 그림 8과 같다.

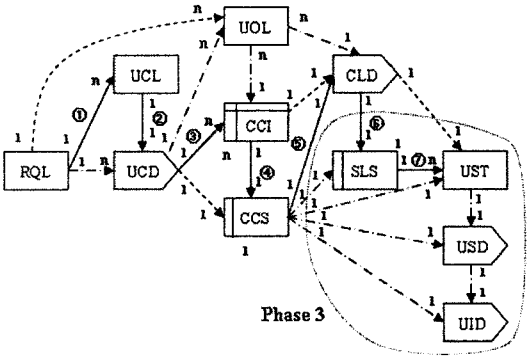

그림 8 Phase 3의 ATD

Phase 4. 슬라이스 코드 설계

Phase 4에서는 슬라이스를 코드로 구현하는 방법을 설계한다. 산출물로는 애스팩트 클래스의 개략 코드(SKC) 템플릿이 있다. 또한 생성된 SKC는 Phase 5의 유스 케이스 추적 다이어그램(UTD)의 입력자료로 사용된다. Phase 3의 UST와 UID를 기준으로 코드를 작성하되, Phase 2의 CLD와 Phase 3의 USD를 참고로 하여 시스템 설계 윤곽이 드러날 수 있도록 개략 코드로 작성한다. 애스팩트 클래스 안에는 핵심 클래스와의 조인포인트를 지정해주는 포인트컷을 선언해 주어야 하며[8], 또한 포인터컷이 지정한 조인포인트에서 실행해야 하는 어드바이스를 포함하여야 한다. 그리고 핵심 클래스들 안에는 액스팩트 클래스에서 지정한 조인포인트가 선언되어 있어야 한다. 애스팩트 클래스의 개략 코드(SKeleton Code, SKC) 템플릿은 표 9와 같다.

표 9 애스팩트 클래스 SKC 템플릿

```
package name ;

import org.aspectj.lang.*;
import org.aspectj.lang.Joinpoint;
...
public aspect name { ...
   pointcut
   ...
   advice { ...
   }
   ...
   joinpoint
}
```

Phase 4까지 생산된 산출물들을 기준으로 ATD를 작성하면 그림 9와 같다.

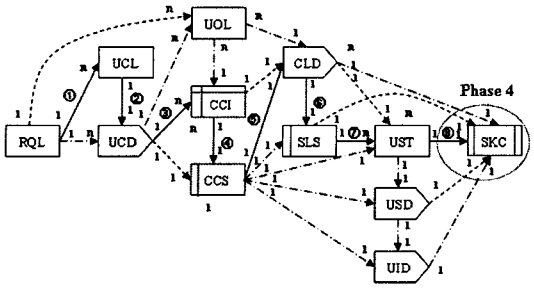

그림 9 Phase 4의 ATD

Phase 5. 횡단관심사 추적

Phase 5에서는 추적 다이어그램을 기반으로 횡단관심사를 추적한다. 산출물로는 유스 케이스 추적 다이어그램(UTD)과 산출물 추적 다이어그램(ATD) 등이 있다.

Phase 3에서 생성된 UST와 Phase 2에서 생성된 CLD 및 Phase 4에서 생성된 SKC를 참고로 하여 그림 10과 같이 애스팩트 클래스를 중심으로 오퍼레이션이 기술된 UTD를 작성한다. UTD는 CLD를 확장한 형태의 추적 다이어그램이며, 핵심 로직의 흐름과 스테레오 타입으로 정의한 어드바이스들의 세부 행위(behavior)를 제시한다.

횡단관심사의 추적을 위해 표 1에 제시한 Phase별

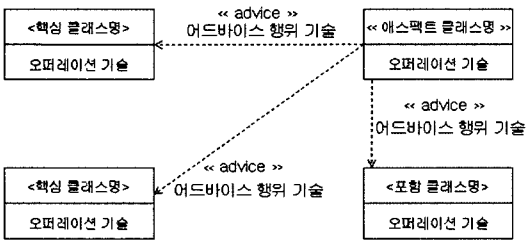

그림 10 UTD 템플릿

산출물들을 중심으로 그림 11과 같이 최종 ATD를 작성한다. 횡단관심사 추적을 위한 ATD에서는 각종 다이어그램, 리스트, 명세서, 개략코드, 상호작용 분석서들의 연관 관계를 화살표로 표시하되, 상호 의존도가 높은 핵심 프로세스일 경우에는 매우 강한 연관(실선 화살표), 요소 추출을 위한 참조 프로세스일 경우에는 강한 연관(일점쇄선 화살표) 및 흐름 파악을 위한 단순 참조 프로세스일 경우에는 약한 연관(점선 화살표) 관계로 표시하며, 화살표의 시작 지점과 끝 지점에 상호 간의 연관비율을 n:n의 비율로 표시하고, 핵심 프로세스는 Phase별 주요 산출물을 중심으로 연관관계 화살표 위에 ①에서 ⑩까지 제시한다.

3.3 횡단관심사 변경

관점지향 소프트웨어 개발방법은 사용자 중심의 시스템 을 구축함으로써 시스템 이식 후 기능 변경 및 유지보수를 사용자 입장에서 쉽게 보장하려는데 의의가 있다[2]. 따라서 관점지향 슬라이싱에서도 시스템 구현 후 횡단관심사를 중심으로 기능변경의 용이성과 유지보수성이 보장된다.

프로그램 개발자 입장에서 횡단관심사의 변경 필요성이 제기되면, 먼저 CCS에서 명세된 횡단관심사에 대한 분석 및 변경처리를 하고, 이를 근거로 SLS에서 오퍼레이션에 대한 분석 및 변경처리를 하며, 핵심 산출물인 UST를 중심으로 포인트컷과 어드바이스 및 조인포인트에 대한 분석과 변경작업을 거친 후 코드를 변경하고 직조(weaving)시킴으로써 횡단관심사에 대한 변경처리가 이루어지며, 그 내용은 그림 12와 같다.

사용자 입장에서 시스템 테스트 또는 이식 후 요구사항 변경 요구가 제기되면, 먼저 사용자 요구사항리스트를 분석하여 변경처리를 하고, 횡단관심사 명세와 슬라이스 명세서를 변경하며, 핵심 산출물인 UST를 중심으로 포인트컷과 어드바이스 및 조인포인트에 대한 분석

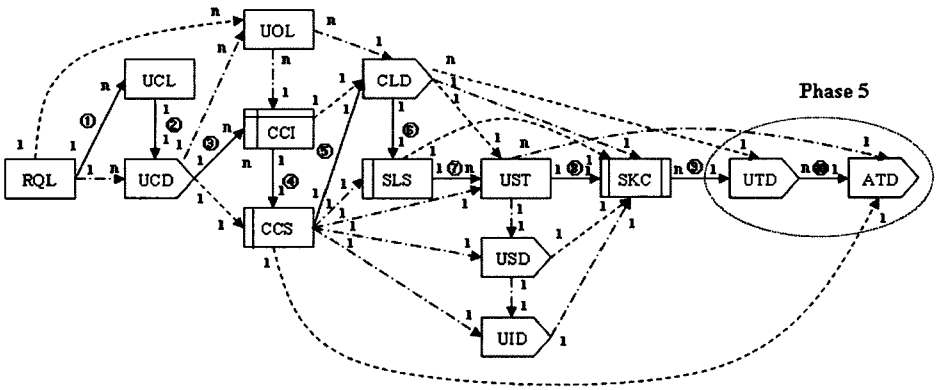

그림 11 최종 ATD

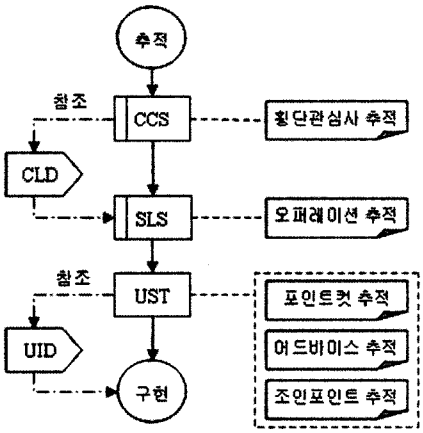

그림 12 횡단관심사 추적 프로세스

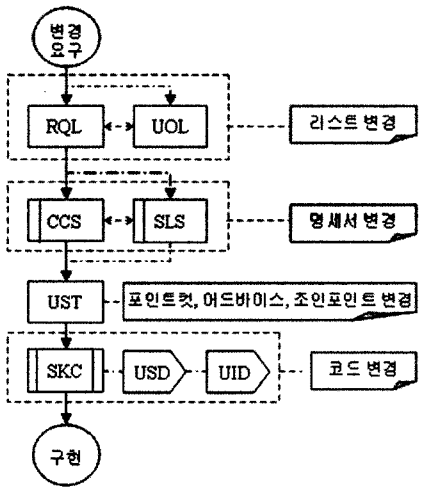

그림 13 횡단관심사 변경 프로세스

과 변경작업을 거친 후 코드를 변경하고 직조(weaving) 시킴으로써 횡단관심사에 대한 변경처리가 이루어진다. 또한 시스템 형상관리를 위해 리스트 변경 시에는 UOL을 변경하고, 코드 변경 시에는 USD 및 UID에 대한 변경처리를 하며, 그 내용은 그림 13과 같다.

4. 적용사례

본 논문에서는 관점지향 슬라이싱 기법의 사례연구를 위해 학습자의 시·공간적 제약을 해소하기 위한 사이버교육을 지원하는 온라인 기반 시스템인 LMS에 적용하였다. LMS는 이식 후 사용자들의 많은 기능 변경과 확장 요청이 발생하는 가변적 시스템으로 높은 확장성과 유지보수성을 필요로 한다[4]. 따라서 LMS에 AOSD 방법론을 적용하는 것은 매우 효과적이라고 볼 수 있으며, AOSD 기반의 관점지향 슬라이싱 기법을 LMS에 적용한 사례는 다음과 같다.

Phase 1. 횡단관심사 명세

Phase 1에서는 횡단관심사를 식별하여 명세하였다. 요구사항 분석서를 토대로 핵심 패키지를 분류하면, 실제 학습이 이루어지는 학습 관리시스템(Learning Management System, p_LMS), 콘텐츠를 관리하는 학습콘텐츠 관리시스템(Learning Contents Management System, p_LCMS), 진단평가와 형성평가 및 총괄평가를 수행하는 평가 관리시스템(Testing Management System, p_TMS), 그리고 메신저와 미니홈피 및 블로그 등의 기능을 수행하는 커뮤니티 관리시스템(Community Management System, p_CMS) 등으로 이루어졌으며, 패키지별 유스 케이스를 표시한 UCL은 표 10과 같다[4].

또한 각각의 패키지들은 유스 케이스 분석 및 유스 케이스 다이어그램 작성을 통해 시스템을 개략적으로 모델링 할 수 있으며, 그 중 핵심패키지인 'p_LMS'의 UCD를 그림 14에 제시하였다.

표 10 사이버 교육 패키지별 UCL

패키지	유스 케이스
p_LMS	u_Learning, u_Testing, u_Boarding, u_Authorizing
p_LCMS	u_ContentDeveloping, u_Managing, u_Authorizing
p_TMS	u_Testing, u_TestDeveloping, u_Authorizing
p_CMS	u_Boarding, u_Chatting, u_Authorizing

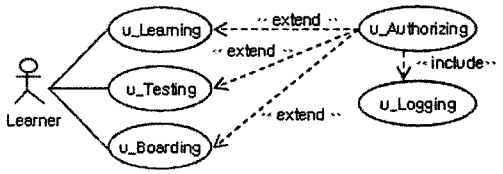

그림 14 'p_LMS'의 UCD

횡단관심사는 도메인이 갖는 독특한 특성을 갖게 되며, 복수의 다른 관심사들과 연관 관계를 갖고 있다[7]. 따라서 하나의 특수한 관심사가 만약에 하나 또는 그 이상의 관심사들에 의해 요구되어진다면 그 관심사를 횡단관심사로 식별해 낼 수 있다[8,9]. 본 사례에서 횡단관심사 추출은 핵심 패키지인 'p_LMS'에서 하였으며, 슬라이싱과 횡단관심사 추적도 'p_LMS'를 중심으로 하였다. 'p_LMS'를 분석하는 동안 'u_Learning'과 'u_Testing' 및 'u_Boarding'과 긴밀한 상호작용이 이루어지고 있는 'u_Authorizing'를 횡단관심사로 식별하였으며, 상호작용이 나타난 CCI는 표 11과 같다.

또한 각각의 유스 케이스들을 'Basic Flows'와 'Alternate Flows' 그리고 'Extension Flows'와 'Extension Points' 들로 구분하여 명세하였는데, 횡단관심사로 식

표 11 상호작용 분석에 의한 CCI

p_LMS				
유스 케이스	u_Learning	u_Testing	u_Boarding	u_Authorizing
u_Learning	-	none	none	interaction
u_Testing	none	-	none	interaction
u_Boarding	none	none	-	interaction
u_Authorizing	interaction	interaction	interaction	-

표 12 'u_Authorizing'의 CCS

u_Authorizing
Basic Flows 1. 학습자는 LMS에 로그인한다. 2. 학습자는 수강승인을 받는다.
Alternate Flows 1. 만약 학습자가 인증을 받지 못하면 초기화면으로 돌아간다. 2. 만약 학습자가 수강승인을 받지 못하면 이전화면으로 돌아간다.
Extension Flows EF1. 인증 후에는 로그 파일을 관리한다. EF2. 수강승인 후에는 학습이력을 관리한다.
Extension Points EP1. 초기화면으로 복귀하는 프로세스 정보를 기술한다. EP2. 'u_Learning'으로 복귀하는 프로세스 정보를 기술한다.

별된 'u_Authorizing'의 'Basic Flows'에서는 학습을 위한 로그인과 인증 과정을 명세하였으며, 'Alternate Flows'에서는 회원 인증이 되지 않을 경우 이전화면으로 복귀하는 과정을 명세하였고, 'Extension Flows'에서는 인증 후의 학습이력관리 과정을 명세하였으며, 'Extension Points'에서는 'u_Authorizing'이 실행되기 이전의 유스 케이스인 'u_Learning'으로 복귀하는 프로세스 정보를 명세하였다. 표 12에서는 횡단관심사인 'u_Authorizing'의 CCS를 제시하였으며, Phase 1의 ATD는 그림 4와 같다.

Phase 2. 오퍼레이션 추출

Phase 2에서는 'p_LMS'의 유스 케이스별 오퍼레이션을 추출하고, 횡단관심사인 'u_Authorizing' 중심의 CLD를 작성하였다. 'p_LMS'의 UCD를 바탕으로 주요 오퍼레이션을 추출하여 작성한 UOL은 표 13과 같다.

그리고 'u_Authorizing'의 오퍼레이션들을 다른 유스 케이스들과의 연관관계를 통해 추출하였다. 'u_Learning'와 연관된 'loginUserid()'와 'certifyUser()'을 추출하였으며, 'u_Testing'과 연관된 'authorizeGrade()'을 추출하였고, 'u_Logging'와 연관된 'takeLog()' 오퍼레이션을 추출하였으며, 관련 CLD를 그림 15에 제시하였다.

표 13 'p_LMS'의 UOL

유스 케이스	주요 오퍼레이션
u_Learning	takeLecture(), getLearning() executeQuery(), submitReport()
u_Testing	selectTest(), sloveTest(), evaluateTest()
u_Boarding	readBoard(), writeBoard(), modifyBoard() chattRoom()
u_Authorizing	loginUserid(), certifyUser() authorizeGrade()

'u_Learning'에서는 로그인 및 등급 인증을 위한 오퍼레이션인 'loginUserid()'와 'certifyUser()'를 수행하기 위해 애스팩트 클래스인 'u_Authorizing'로 확장(<<extend>>)되며, 'u_Testing'에서는 응시자 확인을 위한 오퍼레이션인 'authorizeGrade()'을 수행하기 위해 애스팩트 클래스인 'u_Authorizing'로 확장된다. 또한 'u_Logging'은 학습 및 응시 이력 관리를 위한 오퍼레이션인 'takeLog()'을 수행하기 위해 애스팩트 클래스인 'u_Authorizing'에 포함(<<include>>)되어 인증 후 이루어지는 이력을 관리하며, Phase 2의 ATD는 그림 6과 같다.

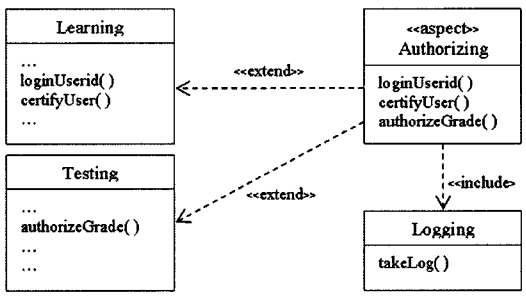

그림 15 오퍼레이션 추출을 위한 CLD

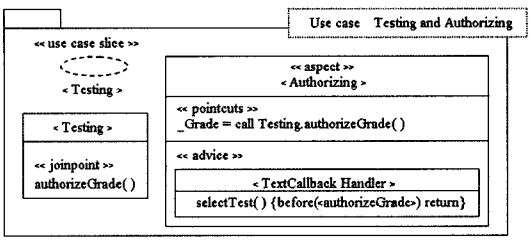

그림 16 'u_Testing'의 UST

Phase 3. 슬라이스 설계

Phase 3에서는 LMS 관련 슬라이스를 조직하고 명세하였다. Phase 2에서 추출된 오퍼레이션들을 중심으로

슬라이스를 조직하여 명세하였는데 'u_Learning'과 'u_Testing' 및 'u_Boarding' 등에서는 조인포인트를 표시하였으며, 횡단관심사인 'u_Authorizing'에서는 포인트컷과 어드바이스를 표시함으로써 하나의 구조적인 슬라이스를 조직하였다. 'u_Authorizing'과 '확장' 관계에 있는 'u_Learning'과 'u_Testing' 및 'u_Boarding'과의 포인트컷으로는 'loginUserid()', 'certifyUser()', 'authorizeGrade()' 등의 오퍼레이션을 선언하고, '포함' 관계에 있는 'u_Logging'과의 포인트컷으로는 'takeLog()'를 선언하였다.

또한 확장 관계에 있는 'u_Learning'과 'u_Testing' 및 'u_Boarding'과의 어드바이스로는 'Before'를 선언하고, 포함 관계에 있는 'u_Logging'과의 어드바이스로는 'After'를 선언하였다. 핵심 패키지인 'p_LMS'의 SLS는 표 14와 같다.

'u_Testing'와 'u_Authorizing'간의 UST를 그림 16에 제시하였다. 'p_LMS'의 'u_Testing'는 학습자가 평가를 위해 조인포인트에서 'AuthorizeGrade()' 오퍼레이션을 통해 사용자 인증을 요구하며, 횡단관심사인 'u_Authorizing'에서는 포인트컷과 어드바이스를 통해 인증을 실시하고, 인증결과를 'u_Testing'에 돌려주게 된다.

시스템의 순차적인 흐름을 파악하기 위해 UST를 기초로 USD를 작성하였다. 그림 17에서는 핵심 유스 케이스인 'u_Learning'에 대한 USD를 제시하였으며, 세부내용은 UML 표기법에 의해 작성하였다. 학습자는 'c_Learning Form'에서 'takeLecture()' 오퍼레이션을 통해 수강할 강의를 선택하고, 핸들러 클래스인 'c_Learninging Handdler' 에서 'loginUserid()' 오퍼레이션에 의해 사용자 인증이 제어되어, 애스팩트 클래스인 'ac_Authorizing'에서 인증절차를 거친 후 인증결과를 'return' 받아 'c_Learning'에서 'getLearning()' 오퍼레이션 등을 통해 학습을 하게 되며, 그렇게 학습한 내용은 'c_Logging'에서 'takeLog()' 오퍼레이션을 통해 학습이력 등의 로그파일을 남기게 된다.

시스템의 코드 설계를 위해 USD와 UST를 기초로 하여 'u_Learning'에 대한 UID를 그림 18에 제시하였으며, 표기는 Jacobson이 제안한 인터페이스 모델링 관련 표기법을 인용하였다[1]. 또한 Phase 3의 ATD는 그림 8

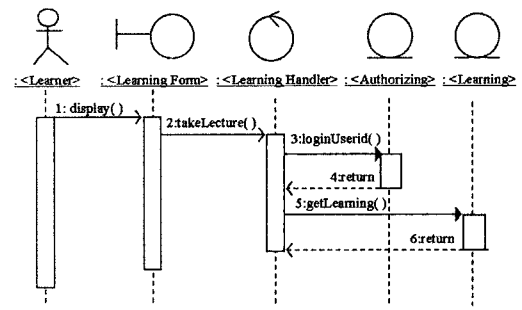

그림 17 'u_Learning'의 USD

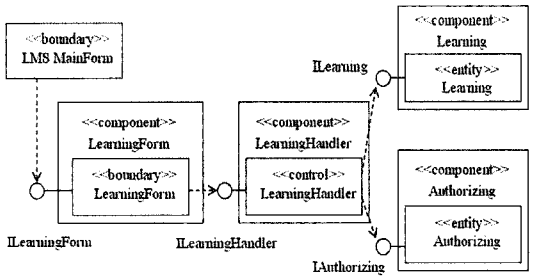

그림 18 'u_Learning'의 UID

과 같다.

Phase 4. 슬라이스 코드 설계

Phase 4에서는 'ac_Authorizing', 'c_Learning', 'c_Testing', 'c_Boarding'의 슬라이스 코드를 설계하였다. CLD를 참고하여 각각의 클래스 들을 소스 코드로 구현하는데, 특히 횡단관심사인 'ac_Authorizing'은 슬라이스 명세를 참조하여 구현하였으며, 핵심 클래스와의 조인포인트를 지정해주는 포인트컷과 포인트컷이 지정한 조인포인트에서 실행해야 하는 어드바이스를 포함하였다. 'p_LMS' 의 횡단관심사인 'ac_Authorizing'를 구현한 개략코드를 표 15에 제시하였다. 애스팩트 클래스의 포인트컷은 'c_Learning'과 교차될 'loginUserid()'와 'certifyUser()'를 선언하였으며, 'c_Testing'과 교차될 'authorizeGrade()'를 선언하였고, 어드바이스로는 'c_Learning' 및 'c_Testing'과 연관된 'before'와 'c_Logging'과 연관된 'after'를 선언하였다. 또한 Phase 4의 ATD는 그림 9와 같다.

표 14 'p_LMS'의 SLS

핵심 클래스	애스팩트 클래스	
조인포인트	포인트컷	어드바이스
Learning.loginUserid()	Authorizing.loginUserid()	Before(); loginUserid()
Learning.certifyUser()	Authorizing.certifyUser()	Before(); certifyUser()
Testing.authorizeGrade()	Authorizing.authorizeGrade()	Before(); authorizeGrade()
Logging.takeLog()	Authorizing.takeLog()	After(); takeLog()

표 15 'ac_Authorizing'의 SKC

```
package lms;

import org.aspectj.lang.*;
import org.aspectj.lang.Joinpoint;
import java.security.*;
import javax.security.auth.callback.*;
import javax.security.auth.login.LoginContext;
import javax.security.auth.login.*;
...
public aspect Authorizing {
     //certification & authorization of User
   private Userid;
   pointcut loginUserid();
      //pointcut of certified operation
   before():loginUserid() {
      //advice of certification
      if(_Userid !=null) {
          return;
      }
      try {
          certifyUser();
      } catch(LoginException ex) {
          throw new CertificationException(ex);
      }
   }

   private void certifyUser() throws LoginException {
     //core logic of certification
     LoginContext Learning=new LoginContext("Certification",
          new TextCallbackHandler();
     Learning.login();
     _Userid=Learning.loginUserid();
     _User=Learning.certifyUser();
   }

   pointcut authorizeGrade();
      //pointcut of authorized operation
   before(): authorizeGrade() {
      //advice of authorization
      if(_Userid=!null){
          return;
      }
      try {
          authorizeGrade();
      } catch(GradeException ex) {
          throw new AuthorizationException(ex);
      }
}
   public abstract Permission getPermission(
      // core logic of authorization
      JoinPoint.StaticPart joinPointStaticPart);
   private void authorizeGrade() throws LoginException {
      LoginContext Test=new LoginContext("Authorization",
          new TextCallbackHandler(ex));
      Testing.login();
     _Grade = Test.authorizeGrade();
   }
}
```

Phase 5. 횡단관심사 추적

Phase 5에서는 'p_LMS'의 횡단관심사 추적을 위한 UTD와 최종 산출물 추적 다이어그램인 ATD를 제시하였다. 'ac_Authorizing'을 중심으로 핵심 클래스인 'c_Learning'과 'c_Testing' 및 'c_Logging'들 간의 조인포

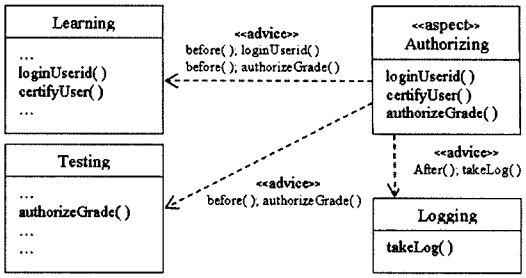

그림 19 'p_LMS'의 UTD

인트와 포인트컷 및 어드바이스들을 나타내는 UTD를 그림 19에 제시하였다.

UTD는 클래스 다이어그램을 확장한 형태의 유스 케이스 추적 다이어그램이며, 이 다이어그램에서는 핵심 로직의 흐름과 스테레오 타입으로 정의한 'advice'의 세부 행위를 볼 수 있다. 핵심 클래스인 'c_Learning'과 횡단관심사인 'ac_Authorizing'과의 관계에서는 'before(); loginUserid' 어드바이스와 'before();authorizeGrade()' 등의 어드바이스가 확장되고 있음을 볼 수 있고, 'c_Testing'과 'ac_Authorizing'과의 관계에서는 'before(); authorizeGrade()'로 정의된 어드바이스가 확장되고 있음을 볼 수 있으며, 'c_Logging'과 'ac_Authorizing'와의 관계에서는 'after();takeLog()'로 정의된 어드바이스가 포함되고 있음을 볼 수 있다. 또한 단계별 주요 산출물간의 횡단관심사 추적을 위해 표 1에 관점지향 슬라이싱 단계별 산출물을 제시하였으며, 횡단관심사에 대한 흐름을 파악하기 위해 모든 산출물간의 추적관계가 나타나 있는 최종 ATD를 그림 11에 제시하였다.

ATD에서는 실선으로 연결된 매우 강한 연관관계 흐름과 일점쇄선으로 연결된 강한 연관관계 흐름 및 점선으로 되어 있는 약한 연관관계 흐름을 제시하였으며, 또한 "①"에서 "⑩"까지 관계선 위에 표시된 번호대로 순차적인 산출물 생성 프로세스-'RQL'→'UCL'→'UCD'→'CCI'→'CCS'→CLD' →'SLS'→'UST'→'SKC'→'UTD'→'ATD'-를 제시함으로써 전체적인 횡단관심사의 흐름과 역할 전이를 추적할 수 있도록 하였다. 그림 11의 ATD를 통해서 전체적인 산출물 중에서 CCS와 UST가 연관관계 흐름을 기준으로 가장 중요한 산출물인 것을 확인할 수 있다.

5. 토의 및 연구의 의의

5.1 토의

본 논문에서 제시하는 관점지향 슬라이싱 기법과 기존 연구들의 횡단관심사와 슬라이스 및 유스 케이스 명세기법[10] 그리고 횡단관심사 추적성 관련 비교자료는

표 16 관련 연구와의 비교

(관련도 ▲매우높음 △높음 ○보통 ▽낮음 ▼매우낮음)

비교 요소	Elisa [5]	Jacobson [1]	Ramesh [6]	Ishio [3]	본 논문
횡단관심사 추출	▲	△	○	△	△
횡단관심사 명세	△	▲	▽	○	▲
유스 케이스 명세	○	▲	▽	▽	▲
슬라이스 명세	▽	○	▼	△	▲
횡단관심사 추적	▼	▽	△	▲	▲
슬라이스 템플릿	▼	▲	▼	▽	▲

표 16과 같이 관련 연구별로 비교 요소별 관련도를 매우 높음(▲), 높음(△), 보통(○), 낮음(▽), 매우 낮음(▼)의 5단계로 표시하였다.

횡단관심사 추출에 관한 연구는 모든 관련 연구에서 이루어졌으나, 횡단관심사 명세에 관한 연구는 Elisa와 Jacobson의 연구에서 이루어졌으며, 유스 케이스 명세에 관한 연구는 Jacobson의 연구에서 이루어졌다. 또한 슬라이스 명세에 관한 연구는 Ishio의 연구에서 이루어졌고, 횡단관심사의 추적성에 관한 연구는 Ramesh 및 Ishio의 연구에서 이루어졌으며, 슬라이스 템플릿에 관한 연구는 Jacobson의 연구에서 이루어졌다. 전반적으로는 Jacobson의 연구가 횡단관심사의 추출과 명세 및 슬라이스 템플릿 제공까지 체계적으로 이루어졌다고 볼 수 있으나 횡단관심사의 추적에까지 이르지는 못하고 있다.

5.2 연구의 의의

본 논문에서는 관련 연구들을 바탕으로 횡단관심사의 식별과 명세 기법을 제시하였고, 슬라이스 명세 기법을 제시하였으며, 포인트컷과 조인포인트 및 어드바이스의 연관관계를 명세한 슬라이스 템플릿과 개략 코드를 제시하였고, 산출물 기반의 추적 다이어그램을 제시하는 등 일련의 체계적인 관점지향 슬라이싱 기법을 제시함으로써, AOSD 적용 시 시스템 개발 시간과 비용 등 효율성 측면에서 효과가 있을 것으로 기대된다. 또한 산출물을 기반으로 하는 ATD를 통해 횡단관심사에 대한 추적성을 향상시켜 개발된 시스템을 이식한 후에도 사용자의 요청에 따른 시스템 확장성 및 유지보수성을 신장시킬 수 있을 것으로 기대된다.

6. 결 론

AOSD는 시스템을 관심사별로 분류하고[11], 그 중 상호작용이 강한 횡단관심사를 식별하여 애스팩트 클래스로 구성함으로써 시스템 확장성과 유지보수성을 신장시키는 사용자 중심의 소프트웨어 개발방법으로 부각되고 있으나[12] 기존의 연구들이 횡단관심사의 식별에서 슬라이싱에 이르기까지 체계적인 방법에 대한 연구와 식별된 횡단관심사의 추적에 대한 연구가 매우 미흡한 실정이다. 따라서 본 논문에서는 유스 케이스들 간의 상호작용 분석을 통해 횡단관심사를 식별하고 명세하는 방법과 이를 기반으로 슬라이스를 조직하고 포인트컷과 조인포인트 및 어드바이스를 명세하는 방법 및 개발 과정에서 생성되는 산출물들을 기반으로 횡단관심사를 추적하는 방법 등 횡단관심사 중심의 관점지향 슬라이싱 기법을 제시하였다. 단계별로 살펴보면 Phase 1에서는 횡단관심사를 식별하고 명세하는 방법을 제시하였으며, Phase 2에서는 유스 케이스별 오퍼레이션을 추출하고 애스팩트 클래스 중심의 클래스를 조직하는 방법을 제시하였고, Phase 3에서는 조인포인트와 포인트컷 및 어드바이스를 포함하는 슬라이스를 조직하고 명세하는 방법을 제시하였으며, Phase 4에서는 조직된 슬라이스에 대한 코드 설계를 하는 방법을 제시하였고, Phase 5에서는 오퍼레이션과 산출물들을 중심으로 횡단관심사를 추적하는 방법을 제시하였다.

이러한 일련의 연구를 통해 본 논문에서는 횡단관심사 식별 및 명세기법 부터 횡단관심사의 추적에 이르기까지 체계적인 관점지향 슬라이싱 기법을 제시하고 있어, 다음과 같은 잇점을 제공할 수 있을 것으로 기대된다. 첫째는 유스 케이스간의 상호작용 분석에 의한 횡단관심사 식별 및 명세가 용이할 것이고, 둘째는 슬라이스 명세서에 의해 유스 케이스들 간의 슬라이싱을 체계적으로 진행할 수 있을 것이며, 셋째는 추적 다이어그램을 기반으로 횡단관심사에 대한 추적성을 향상시킬 수 있을 것이고, 마지막으로 일련의 체계적인 관점지향 슬라이싱 기법 적용을 통해 시스템 이식 후에 시스템 확장성과 유지보수성을 향상시킬 수 있을 것으로 기대된다.

향후 연구에서는 포인트컷 중심의 슬라이싱 상세기법과 횡단관심사 명세 자동화 기법, 그리고 횡단관심사 추적성 신장 기법 및 추적성 맵핑 기법 등에 관한 연구를 진행하고자 한다.

참 고 문 헌

[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] Ishio T., Kusumoto S., and Inoue K., "Program Slicing Tool for Effective Software Evolution Using Aspect-Oriented Technique," IWPSE Proceedings of the 6th International Workshop on Principles of Software Evolution, pp. 3-12, 2003.

[4] Jong-Kack P., Dae-Gon K., Cheol-Jung Y., Ok-Bae J., "Applying Aspect-Oriented Software Development Methodology in Learning Management System," APIS 2007, Proceeding of the 6th International Symposium, pp. 55-58, 2007.
[5] Elisa, B., Siobhan, C., "Finding Aspects In Requirements with Theme/Doc," Aspect-Oriented Requirements Engineering and Architecture Design Workshop, pp. 15-22, 2004.
[6] Balasubramaniam Ramesh, and Matthias Jarke., "Toward Reference Models for Requirements Traceability," IEEE Transactions on Software Engineering, Vol. 27, No. 1, pp. 58-93, 2001.
[7] Bedir, T., Ana, M., "Aspect-Oriented Requirements Engineering and Architecture Design," Aspect-Oriented Requirements Engineering and Architecture Design Workshop, pp. 4-14, 2004.
[8] 박옥자, 유철중, 장옥배, "프로그램 개발 및 유지 보수를 지원하는 횡단관심사 명세 기법", 한국정보과학회 논문지 제34권 제9호, pp. 773-784, 2007.
[9] Isabel, B., Ana, M., "Integrating the NFR Framework in a RE Model," Aspect-Oriented Requirements Engineering and Architecture Design Workshop, pp. 27-32, 2004.
[10] Jianjun Zhao, "Slicing Aspect-Oriented Software," IWPC Proceedings of the 10th International Workshop on Program Comprehension, pp. 251-260, 2002.
[11] 정인상, 윤광식, 이완권, 권용래, "명세 기반 프로 그램 슬라이싱 기법과 응용", 한국정보과학회 논문지 제29권 제8호, pp. 529-542, 2002.
[12] Ivar, Jacobson., "Use Cases and Aspects-Working Seamlessly Together," In Journal of Object Technology, Vol.2. No.4. pp. 7-28, 2003.

박 종 각

2003년 전북대학교 대학원 교육학과 졸업(교육학석사, 전자계산교육전공). 2003년~현재 전북대학교 대학원 컴퓨터통계정보학과 박사 수료. 2006년~2007년 전라북도교육정보과학원 교육연구사. 2007년~현재 전라북도교육청 과학정보교육과 장학사. 관심분야는 관점 지향 소프트웨어 개발 방법, 관점 지향 추적기법, 관점 지향 명세기법, 교육공학 등

박 옥 자

1997년 전북대학교 대학원 전산통계학과 졸업(이학석사). 2008년 전북대학교 대학원 전산통계학과 졸업(이학박사). 2004년~2006년 백석대학 IT 프로그래밍 교수요원. 2008년 9월~현재 숭실대학교 정보미디어기술연구소 전임연구원. 관심분야는 컴포넌트 기반 소프트웨어 개발 방법론, 관점 지향 프로그래밍, 소프트웨어 재사용, 소프트웨어 유지보수 및 추적기법. 소프트웨어 품질관리 등

유 철 중

1982년 전북대학교 전산통계학과 졸업(이학사). 1985년 전남대학교 대학원 계산통계학과 졸업(이학석사). 1994년 전북대학교 대학원 전산통계학과 졸업(이학박사). 1982년~1985년 전북대학교 전자계산소 조교. 1985년~1996년 전주기전여자대학 전자계산과 전임강사~부교수. 1997년~2007년 전북대학교 자연과학대학 컴퓨터과학과 전임강사~부교수. 2008년~현재 전북대학교 공과대학 응용시스템공학부(정보공학) 교수. 관심분야는 소프트웨어 개발 프로세스, 소프트웨어 품질, 컴포넌트 소프트웨어, 소프트웨어 메트릭스, 소프트웨어 에이전트, GIS, 교육공학, 인지공학 등