교육용 개발 방법론 OOPT에서의 추적성 분석을 위한 프레임워크

김재현 하헌규 정세진 유준범
건국대학교 컴퓨터공학과
{jaehyun739, henry1993, jsjj0728, jbyoo}@konkuk.ac.kr

A Framework for Traceability analysis in OOPT, a Development process for education

Jaehyun Kim Hungu Ha Sejin Jung Junbeom Yoo
Department of Computer Science and Engineering, Konkuk University

요 약

소프트웨어 개발 프로세스는 소프트웨어 공학 교육에서 가장 기초적인 역할을 한다. OOPT는 소프트웨어 공학 교육을 위해 래셔널 통합 프로세스 (RUP: Rational Unified Process)를 간결화한 객체지향 소프트웨어 개발 방법론으로써, 단계별 요구사항, 산출물 및 테스팅, 추적성 분석 등의 내용으로 구성되어 있다. 하지만 OOPT를 통해 소프트웨어 공학 교육을 진행하면서 단계마다 개별적으로 문서 작성 및 UML (Unified Modelling Language) 작성을 하고 있어 산출물 버전 관리 및 추적성 분석에 제약사항이 많다. 이에 따라 본 논문에서는 OOPT를 통한 소프트웨어 공학 교육 시 개발 산출물 관리 및 추적성 분석을 효율적으로 할 수 있는 UML diagram, 문서 작성 및 추적성 분석을 위한 도구가 포함된 프레임워크를 제안한다.

1. 서 론

소프트웨어 개발 방법론은 개발 프로세스를 기반으로 소프트웨어 개발에 필요한 구체적인 사항들과 도구, 표기법 등을 추가한 것을 통칭하는 것으로, 크게 구조적 방법론(SASD: Structured Analysis and Structured Design)[1]과 객체지향방법론(OOAD: Object-Oriented Analysis and Design)[2] 으로 분류할 수 있다. 객체지향 방법론은 객체(Object Classes) 사이의 통신을 통해 시스템의 행위를 구현하는 방법론으로서, C++, Java와 같은 객체지향언어에 적합한 방법론이며, 분석, 디자인에 주로 UML(Unified Modeling Language)을 사용한다. 가장 많이 사용되는 객체지향 소프트웨어 개발 방법론으로는 래셔널 통합 프로세스 (RUP: Rational Unified Process) [3,4]가 있다.

래셔널 통합 프로세스를 교육과정에 그대로 적용하기에는 다소 불필요한 내용들이 포함될 수 있으므로 이전 연구에서는 소프트웨어 공학 수업에서 사용할 수 있는 수준으로 수정(Tailoring)한 교육용 객체지향 소프트웨어 개발 방법론 OOPT (Object Oriented Process with Traceability)[5]를 제안하였다. OOPT는 크게 'Plan and elaboration', 'Build (analysis and design)', 'Deployment' 3 단계 (Stage) 로 구성되어 있으며 각 단계별로 요구사항, 개발 산출물, 테스팅 및 추적성 분석 등의 내용으로 구성된다. 하지만 OOPT는 개발 방법론으로써 각 단계마다 관련 문서 및 UML 산출물을 개별적으로 작성하기 때문에 산출물 관리 및 추적성 분석(Traceability Analysis)에 MS Excel, 워드프로세서를 그대로 이용하는 등의 제약사항이 많다.

본 논문에서는 문서 및 UML을 활용한 분석, 디자인의 효율적인 작성과 추적성 분석 관리를 지원하기 위해 작성 및 추적성 분석 프레임워크를 제안하고, 도구를 개발하였다. 또한, OOPT에서 나타나는 추적성을 모델링 하기 위해 메타모델을 개발하였다.

2. 배경 지식

2.1 Object Oriented Process with Traceability (OOPT)

OOPT는 래셔널 통합 프로세스를 대학 소프트웨어공학 수업에 활용할 수 있도록 수정한 객체지향 소프트웨어 개발 방법론이다[5]. OOPT는 개발 도중 요구사항 변경, 프로젝트 환경의 변경 등에 대처하고 피드백을 반영하기 위해 반복적(Iterative)이고 점진적(Incremental)으로 소프트웨어를 개발할 수 있도록 구성되어 있으며, 'stage 1000: Plan and elaboration', 'stage 2000: Build (analysis and design)', 'stage 3000: Deployment' 3개의 단계로 진행된다. 이 중 실제 교육에 적용되는 단계는 plan and elaboration과 build가 된다.

Plan and elaboration 단계는 10개의 Activity로 구성되어 있으며 프로젝트의 요구사항 정의와 같은 전반적인 계획에 대하여 진행한다. Stage 2000 Build 단계는 6개의 페이즈로 구성되며 Analyze부터 시작하여 Design, Construct 및 Test 단계로 순차적으로 분석, 디자인, 구현, 및 테스팅이 진행된다. OOPT에서는 구성된 각 단계, 페이즈 별로 입력과 출력이 정해져 있으며, 이에 맞추어 <표 1>과 같이 요구사항 문서, UML 다이어그램 등의 개발 산출물들이 각 활동의 결과로 도출된다.

2.2 요구사항 추적성(Requirement Traceability)

소프트웨어 요구사항(Software Requirement)은 개발 과정 중 지속해서 변화한다. 이러한 변화를 관리하는 것을 요구사항 관리(Requirement management)라 한다. 요구사항 관리를 성공적으로 수행하기 위해서는 요구사항 추적성을 잘 정의해야 한다. 요구사항 추적성(Requirement Traceability)이란 요구사항과 다른 명세 간의 관계를 정의하는 것을 의미한다. 요구사항 추적성을 활용한다면 요구사항의 변화로 야기되는 영향 분석(Change Impact Analysis)을 가능하게 하고, 불필요한 기능을 구현하는 것을 방지하여 개발을 용이하게 한다. 하지만 요구사항 추적성을 활용하기 위해서는 필요한 명세와 명세 간 관계의 의미에 대한 정의가 있어야 한다[6].

3. OOPT 프레임워크

본 논문에서는 OOPT 산출물 관리 및 추적성 분석을 위한 프레임워크를 제안하고 이를 위해 추적성 메타 모델 및 각 산출물 들을 작성하고 관리할 수 있는 도구를 개발하였다.

3.1 OOPT 추적성 메타 모델

OOPT를 이용한 교육 진행 시 도출되는 개발 산출물들은 요구사항부터 시작해 여러 UML diagram, 및 테스트 케이스 등이 있다. 이들의 추적성 분석을 효과적으로 수행하기 위해서는 각 산출물들의 type 및 관계에 대한 정의가 필요하다. <표 1>은 추적성 분석에 사용되는 OOPT 산출물들에 대한 분류를 나타낸 표이다. OOPT를 통한 프로젝트의 산출물의 type은 요구사항을 대변하는 <<Textual Requirement>>, 중간 분석 및 디자인 결과물인 <<UML diagram>>, 마지막으로 테스팅을 위한 테스트 케이스에 해당하는 <<Test Artifact>>가 있다.

표 1 추적성 분석에 사용되는 OOPT 산출물 분류

OOPT 산출물	Type of Entity
Functional Requirement	<<Textual Requirement>>
Use case	<<UML diagram>>
System test case	<<Test Artifact>>
Unit test case	<<Test Artifact>>
Essential use case specification	<<Textual Requirement>>
Sequence diagram	<<UML diagram>>
Operation contracts	<<Textual Requirement>>
Methods description	<<Textual Requirement>>
Real use case specification	<<Textual Requirement>>
Interaction diagram	<<UML diagram>>
Class diagram	<<UML diagram>>

분류 후에는 추적성 분석을 위해 각 산출물 간의 관계를 <그림 1>와 같이 정의한다. 명시적(Explicit) 관계는 추적성 분석을 위해 사용자가 직접 관계를 정의하는 경우이고 암묵적(Implicit) 관계는 이름이 같은 경우 혹은 추이적 관계가 성립하는 경우처럼 명시적으로 정의하지 않아도 관계가 성립되는 경우이다. 각 Artifact의 관계는 집합(Aggregation) 관계인 'partOf'와 추적(Trace) 관계인 'traceTo', 'verifiedBy'로 정의할 수 있다.

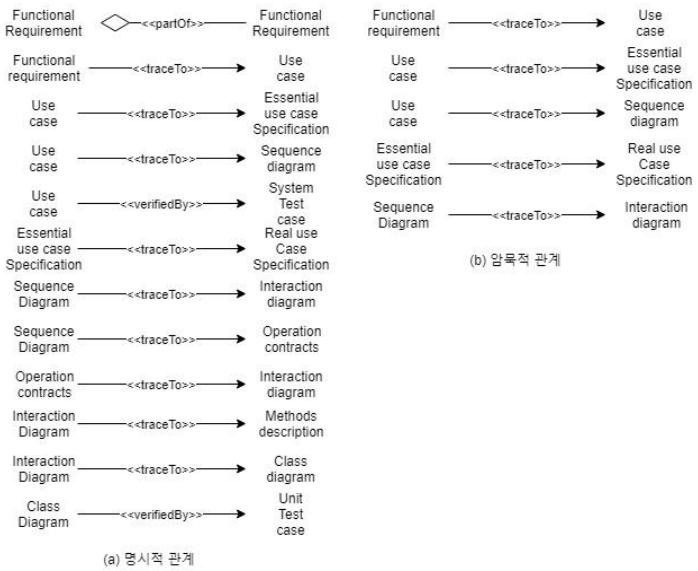

그림 1 OOPT 산출물의 명시적, 암묵적 관계

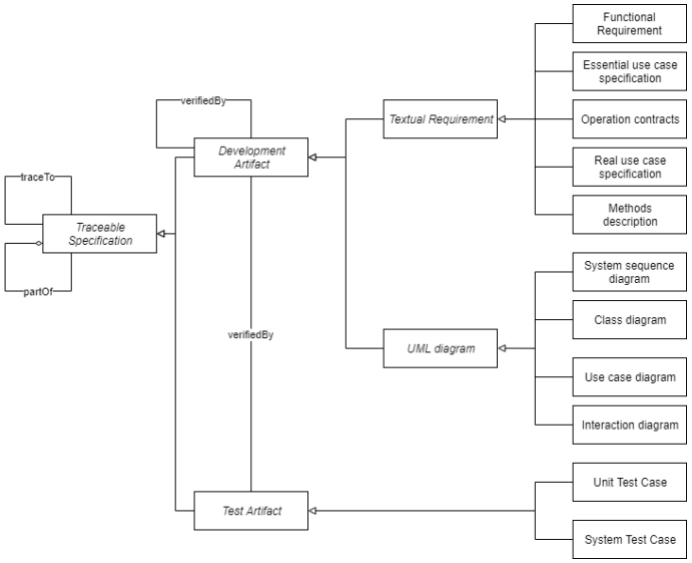

그림 2 OOPT 추적성 메타모델

<표 1>, <그림 1>를 통해 정의한 산출물 분류 및 관계를 바탕으로 OOPT의 추적성을 메타모델로 표현하면 다음 그림 <그림 2> 과 같다. 본 논문에서는 메타모델로 표현한 추적성 관계를 프레임워크의 문서 작성 자동화, 추적성 분석에 활용한다.

3.2 OOPT 산출물 관리 도구

정의한 요구사항 추적성 메타모델을 활용하여 개발한 OOPT 프레임워크는 요구사항 정의, UML 다이어그램 작성, 테스트 케이스 작성 등 OOPT의 각 stage 별 활동들을 수행할 수 있도록 개발되었다. <그림 3>은 본 논문에서 제안하는 OOPT 프레임워크 도구의 작성 화면으로, OOPT의 각 stage 별 산출물을 한번에 확인하고, 관리할 수 있도록 하였다.

본 논문에서 제안하는 프레임워크에서 암묵적 관계로 인한

추적성과, 사용자가 직접 입력해야 하는 추적성 관계를 두 가지로 분류해 관리한다. 암묵적 관계로 인해 자동으로 추적성이 성립되는 경우는 같은 이름을 갖는 경우이기 때문에 문서 작성의 용이성, 산출물 관리의 효율성을 위해 자동으로 산출물 객체를 생성해 입력하도록 한다. 예를 들어 <그림 3>에서 Sequence 다이어그램을 작성하면 <그림 2>의 암묵적 관계에 따라 자동으로 <그림 4>의 Operation contracts 객체가 생성되고 추적성이 성립된다.

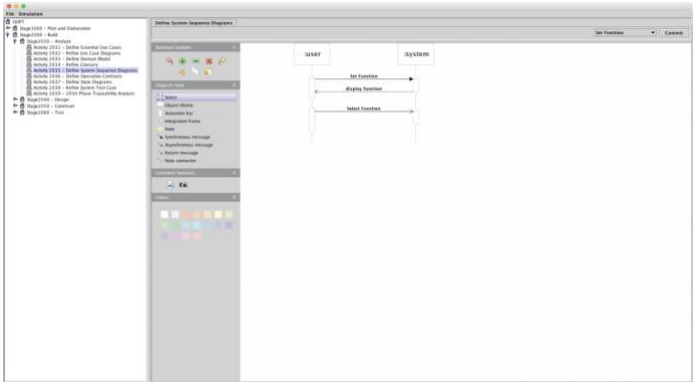

그림 3 Sequence 다이어그램 작성

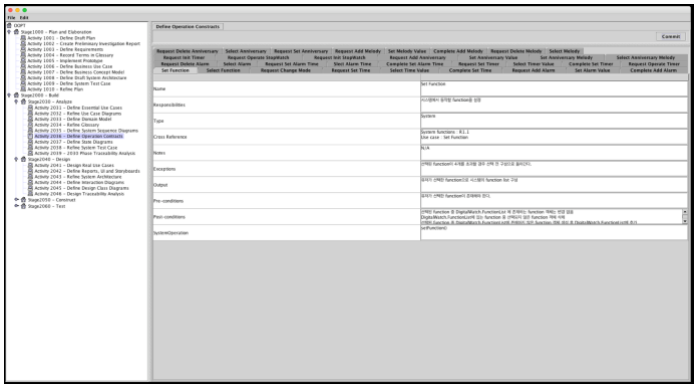

그림 4 Operation contracts 작성

사용자가 직접 입력해야 하는 경우는 추적성 분석 다이어그램을 만드는 데 사용된다. 예를 들어 <그림 4>의 Operation contracts의 'SystemOperation'을 작성하면 그 'SystemOperation'을 포함하는 Interaction diagram과 추적성이 성립된다. 사용자가 직접 입력해야 하는 경우로는 Functional requirement와 Use case, Operation contracts와 Interaction diagram, Use case와 System test case, Class diagram과 Unit test case 간의 관계가 있다. 이를 통해 사용자가 효율적으로 산출물들을 작성하고 관리하며, 추적성 분석을 수행할 수 있다.

4. 사례 연구

본 논문에서 제안하는 OOPT 프레임워크의 활용 가능성을 확인하기 위해 이전에 진행한 손목시계에 대한 예제 프로젝트를 이용해 적용하였다. 해당 예제는 실제 구현만을 제외한 OOPT의 모든 단계를 작성한 예제로써 적용한 결과의 추적성 분석 다이어그램은 다음 <그림 5>와 같다.

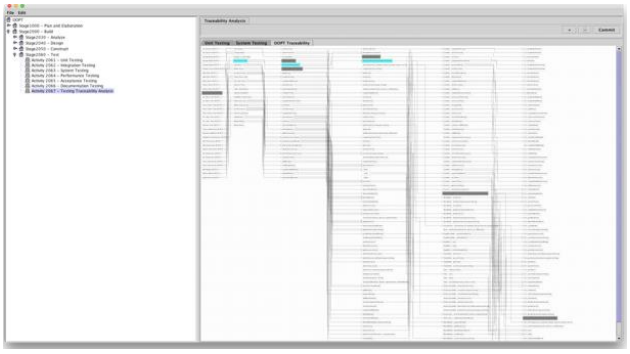

그림 5 OOPT 추적성 분석 다이어그램

OOPT 추적성 분석 다이어그램은 OOPT Stage 2000: Build의 각 페이즈를 수행하면서 점점 확장된다. 최종적으로 <표 1>의 모든 OOPT 산출물의 입력을 완료하면 <그림 5>와 같은 Use case부터 System test case, Unit test case까지 이어지는 추적성 분석 다이어그램을 확인할 수 있다.

5. 결론 및 향후 연구

본 논문에서는 교육용 개발 방법론 OOPT의 추적성 분석을 위한 프레임워크 및 도구를 제안하였다. 제안한 프레임워크를 활용해 OOPT를 통한 개발 산출물 관리 및 추적성 분석을 효율적으로 할 수 있으며, 기존에 진행된 프로젝트를 활용한 사례 연구를 통해 본 논문에서 제안하는 프레임워크의 효율성을 확인할 수 있다. 향후 코드 구현의 부담을 줄이고 개발 방법론 학습에 더 집중하기 위해 시뮬레이션 기반의 방법에 관해 연구할 계획이다.

Acknowledgement

본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 SW중심대학지원사업의 연구결과로 수행되었음(2018-0-00213)

참고 문헌

[1] FA Masoud, MU Shaikh, SH Mustafa, "SASD methodology from a practical point of view problems and suggestions for improvement," WIT Transactions on Information and Communication Technologies, Vol. 17, pp. 93-102, 1997.

[2] Grady Booch, "Object-oriented Analysis and Design with Applications 3rd edition," Addison-Wesley, Boston, 2007.

[3] Per Kroll, Philippe Kruchten, Grady Booch, "The Rational Unified Process Made Easy a Practitioner's Guide to the RUP," Addison-Wesley, Boston, 2003. [4] Philippe Kruchten, "The rational unified process: an introduction," Addison-Wesley Professional, Boston, 2004

[5] 정세진, 이동아, 김의섭, 장천현, 유준범, "OOPT: 소프트웨어공학 교육을 위한 객체지향 소프트웨어 개발 방법론,"정보과학회논문지, 제44권, 제5호, pp.510-521, 2017.

[6] J.-P. Corriveau. "Traceability Process for Large OO Projects". IEEE Computer, pp. 63-68, September 1996.

[7] Letelier, Patricio. "A Framework for Requirements Traceability in UML-based Project" Proc. of 1st International Workshop on Traceability in Emerging Forms of Software Engineering, pp.173-183, 2002.