

한국정보과학회
Korean Society of Information Science and Technology

제 24 권 제 1 호
Vol. 24 No. 1



2022

제 24 회 한국 소프트웨어공학 학술대회 논문집

Proceedings of the 24th Korea Conference on
Software Engineering (KCSE 2022)

- 일시: 2022년 1월 19일(수) ~ 1월 21일(금)
- 장소: 강원도 평창 한화리조트(휘닉스파크점)

주최: 한국정보과학회, 한국정보처리학회

주관: 한국정보과학회 소프트웨어공학 소사이어티
한국정보처리학회 소프트웨어공학연구회

후원: **ETRI** 한국전자통신연구원 Electronics and Telecommunications Research Institute, **SOLUTIONLINK** (주)비트컴퓨터,
한국소프트웨어기술진흥협회(KOSTA), T3Q(주),
(주)유정인베스트먼트, (주)다한테크, (주)브이웨이,
브이플러스랩(주), 슈어소프트테크(주), (주)이에스지,
(주)코어다트랩, STA 테스트컨설팅(주),
TTA 소프트웨어시험인증연구소,
서강대 지능형블록체인연구센터,
KAIST 빅데이터 엣지 클라우드서비스 연구센터

1 월 21 일 (금)				
시 간	행 사 내 용			
	논문 발표			
	E1: 요구사항 및 아키텍처 I 좌장: 고인영 (KAIST) 장소: 그랜드홀 2	E2: 요구사항 및 아키텍처 II 좌장: 유준범 (건국대) 장소: 세미나실 1	E3: SW 모델링 및 테스트 좌장: 이주용 (UNIST) 장소: 세미나실 2	워크숍 W2:
9:25-10:50 (85 분)	[우수 일반 논문] 소프트웨어 요구사항 분류 모델의 성능 향상을 위한 불균형 데이터 처리에 관한 연구 최종우, 이영준, 임채균, 최호진 (KAIST) [우수 일반 논문] 효율적인 아키텍처 패턴 적용을 위한 소프트웨어 요구사항 분류체계에 관한 연구 최종우, 민상윤 (KAIST) [단편 논문] 코드가시화의 고도화 연구 김장환, 문소영, 김영철 (홍익대) [단편 논문] 자카드 지수를 이용한 한국어 요구사항 유사도 분석 자동화 장우성, 문소영, 김영철 (홍익대) [단편 논문] Use case 명세서 기반 자동 가이드 코드 생성 정세준, 문소영, 김영철 (홍익대)	[일반 논문] Status Quo of Software Architecture and Technologies for Cloud-Edge Collaborative Intelligence Xiangchi Song, 고인영 (KAIST) [우수 일반 논문] AIAG-VDA FMEA 방법론 기반 SFMEA : 효과적인 자동차 SW 안전분석 김소연, 권기현 (경기대) [일반 논문] 엣지 클라우드 환경에서 다목적 최적화 알고리즘을 활용하여 비용 및 서비스 지연을 최소화하는 마이크로서비스 배포 방법 김민협, 고인영 (KAIST) [산업체 논문] 소프트웨어 수출을 위한 시장주도 요구공학 적용 사례 이종길, 길문주, 오정섭 (엔에스이)	[일반 논문] 다양한 디스플레이 환경을 위한 적응형 UI 레이아웃 프레임워크 윤진, 이은영, 이상현 (삼성전자), 백종문 (KAIST) [단편 논문] 멀티태스크 PLC 프로그램의 테스트를 위한 주기 별 테스트 시퀀스 실행 제어 방법 은형석, Lingjun Liu, 지은경, 배두환 (KAIST) [산업체 논문] 반도체 공정 유지 보수 비용 절감을 위한 공정 레시피 데이터 학습 및 장비 결함 검출 최민호(피에스케이 / 아주대), 유동연, 이정원 (아주대) [학부생 논문] 아동·청소년을 위한 인공지능 스피커: 심리 상담 통합 시스템 개발 김수아, 이에빈, 정수민, 최미소, 최혜빈 (이화여대) [학부생 논문] 스마트 팩토리-가스라이터 제조 공정에서의 스티커 부착과 관련된 불량 검출 프로그램 개선 신관수, 송호준, 조영진, 신연순 (동국대)	서강대 지능형블록체인 연구센터 장소: 세미나실 4 (10:00~11:50)
10:50-11:00	휴식			
	신진 연구자 초청 세미나 N4 좌장: 김미정 (UNIST) 장소: 그랜드홀 2	신진 연구자 초청 세미나 N5 좌장: 유신 (KAIST) 장소: 세미나실 1	신진 연구자 초청 세미나 N6 좌장: 지은경 (KAIST) 장소: 세미나실 2	
11:00 -11:50 (50 분)	오픈소스 소프트웨어에서의 버그 정정 관리를 위한 지능형 프레임워크 양근석 교수 (경남대)	비형식 요구사항 기반 유스케이스 추출을 통한 노력 추정 방법 박보경 교수 (진주교대)	소프트웨어 신뢰도 보증을 위한 효율적인 도달 가능성 분석 이낙원 박사 (KAIST)	
11:50-12:10 (20 분)	폐회식 장소: 그랜드홀 2 사회: 이정원 조직위원장 (아주대)			

Use case 명세서 기반 자동 가이드 코드 생성

정세준[○] 문소영 김영철

홍익대학교 소프트웨어공학연구실

{bvcx79, whit2, bob}@hongik.ac.kr

Automatic Guide Code Generation based on High-level Use Case Specifications

Se Jun Jung[○], So Young Moon, R. Young Chul Kim

SE Lab, Dept. of Software & Communications Engineering, Hongik University

요 약

소프트웨어의 비가시성으로 인해 요구사항과 설계의 기능들이 제대로 구현되는지 검증은 어렵다. 상위 레벨의 요구사항으로부터 설계 그리고 연속적 구현까지는 점점 더 복잡하므로 추적 및 오류 번식 발생한다. 이런 문제를 해결을 위해, 요구사항에서 상위 레벨 유스 케이스 명세서와 설계 정보인 UML 설계를 통합한 ‘설계 메타 모델’을 제안한다. 이를 통해 가이드 코드 생성과 설계 추적성을 가능하게 한다. 단지 상위 수준의 명세서와 모델로부터 가이드 코드 제공함으로써, 개발자들에게 고품질의 개발이 가능하게 될 것이다.

1. 서 론

소프트웨어는 고객의 요구사항으로부터 설계하고 그 기반으로 구현한다. 이때 소프트웨어의 비가시성으로 인해 특정 설계와 구현된 코드가 어느 요구사항에 관련하는지 추적이 어렵다. 이는 결국 초기에 발생한 문제를 인식하지 못하고 잘못된 산출물을 기반으로 단계가 진행됨을 의미한다. 이는 결국 테스트 단계에 되어야 발견이 되며 오류가 번식하는 문제가 발생한다[1]. 이를 해결하기 위해 ID를 기반으로 산출물들을 추적하고 매트릭스 표를 생성하거나 전반적인 소프트웨어 프로세스를 관리하는 연구가 있다[2][3]. 본 연구는 UML과 Usecase 시나리오를 통합한 메타 모델을 제안한다. 각각의 모델들을 ID와 관계를 통해 추적하고 추적된 클래스 다이어그램을 통해 가이드 코드를 생산한다. 가이드 코드는 추적된 모델의 정보를 바탕으로 스케레톤 코드와 주석 및 JavaDoc을 생성한다. 이를 통해 개발자들이 설계를 기반으로 코드를 작성하고 자동화 코드 생성을 통해 문서화와 코드 작성을 편하게 돕는다.

2. 관련연구

소프트웨어 개발 생명주기에서 요구사항과 발생한 산출물들을 추적하기 위한 방법으로 추적성 매트릭스 기반의 시스템이 존재한다[4]. 이는 프로젝트 관리도구에 저장된 산출물 데이터베이스를 사용하여

각각의 산출물에 대한 id 값을 통해 관련된 추적 표를 생성한다. 본연구는 이러한 데이터베이스를 사용한 추적관계를 메타 모델을 사용하여 추적한다. 메타 모델을 사용하면 모델의 통합과 모델 변화 기술을 사용할 수 있다. 예를 들어 다양한 소스 코드 언어를 파서를 통해 AST를 생성하면 서로 다른 AST의 구조가 발생한다. 이러한 서로 다른 언어의 AST를 하나의 ASTM으로 메타 모델링하여 소스 코드를 가시화할 수 있다[5]. 즉 서로 다른 AST를 하나의 ASTM 모델로 정의한 것처럼 메타모델을 사용하여 두개의 모델들을 통합할 수 있다. 또한 스마트폰 앱의 UI는 플랫폼에 따라 다른 구조를 가진다. 이를 하나의 UI Model로 통합하고 모델 변형기술을 사용하여 다중 플랫폼의 UI 파일들을 생성한다[6]. 이처럼 Model To Model, Text To Model 기술을 사용하여 메타 모델을 사용할 수 있다. 또한 UML을 확장한 모델로 시스템의 설계로써 SysML 메타 모델이 존재한다. 해당 메타 모델을 사용하여 안전이 중요한 시스템의 설계에 초점을 맞추어 요구사항들을 추적하고 검증한다[7]. 이는 시스템 프로세스에 효율적인 설계 구성과 추적을 가능하게 해준다. 이와 유사하게 본 논문은 UML 설계와 Usecase 시나리오를 통합 및 추적을 위해 메타모델 RDOM(Requirement Design Object Model)을 정의한다. 또한 최종적으로 자동으로 설계를 반영한 가이드 코드를 생성하여 설계의 이해와 구현을 돕는다. 이를 EMF(Eclipse Modeling Framework)를 사용하여 구현한다.

3. Use case 명세서 기반의 가이드 코드 생성

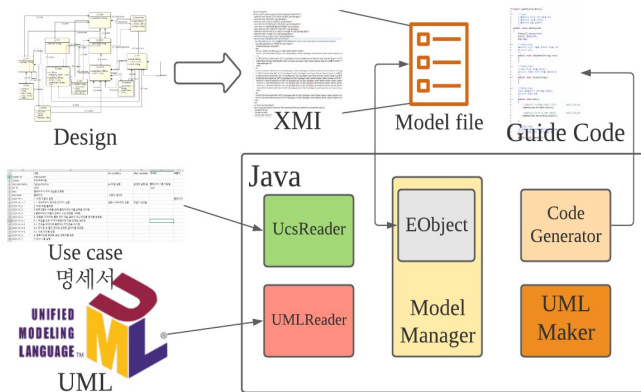


그림 1 시스템 구조

그림 1은 EMF를 통해 구현한 시스템의 구조이다. 입출력인 파일들을 제외하고 모든 과정을 JVM 내에서 처리하여 프로그램의 모듈성을 높이고 간단하게 처리할 수 있다. 먼저 유스 케이스 명세서 파일을 UcsReader 객체가 읽어 파일의 정보를 읽는다. UML의 파일을 입력하면 UMLReader 객체가 파일의 정보를 읽는다. ModelManager 객체는 XMI 형태의 모델 파일을 읽어 EObject 객체들로 변경해준다. 따라서 모델 파일의 내용을 객체로써 변경하고 수정한다. 이로서 자바 프로그램 내에서 해당 객체의 정보를 각각의 Reader가 읽은 정보들로 수정하고 생성하여 하나의 자바 프로세스로서 처리할 수 있다. 그리고 CodeGenerator 객체를 통해 최종적으로 가이드 코드를 생성한다. 또한 UMLMaker 객체를 통해 UML 설계 다이어그램을 생성이 가능하다. 이는 입력을 통해 정상적으로 모델 파일에 저장되었음을 확인하고 해당과정에서 자동으로 생성된 UML 모델들을 검증하기 위함이다. 가이드 코드의 생성은 클래스 다이어그램을 기반으로 생성된다.

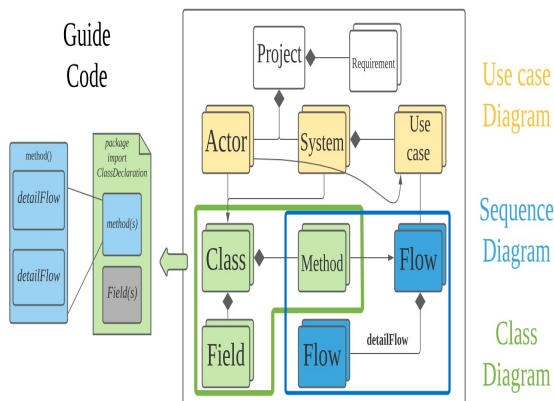


그림 2 추적 및 가이드 코드를 위한 모델 구조

그림 2는 가이드 코드의 생산과 설계와 코드의 추적을 위한 설계 메타모델의 핵심 구조이다. 먼저 Project를 최상위 모델로서 놓고 해당 Project에 대한 정보(남

기 날짜, 규모, 책임자 등등)를 저장한다. 그리고 Requirement에 대한 정보를 Project에 추가한다. 해당 Requirement은 다른 모델들이 참조 및 연결하여 관련된 요구사항을 추적하고 체크할 수 있도록 구조를 정의한다. 그리고 프로젝트에 Actor와 System, Usecase를 생성하여 저장한다. 이들은 추상화 수준이 상대적으로 높은 상위 단계인 Use case 다이어그램의 요소이다. 공통적으로 Id와 이름을 가지며 각각 필요한 세부적인 값들을 가진다. Use case의 경우 System 내부에 정의하며 유스 케이스 명세서의 시나리오의 흐름을 Flow로 정의하며 Use case 내부에 고유한 Id를 주고 정의한다. 이러한 내부 정의를 통해 관계를 정의한다. Flow는 대안이나 예외 혹은 함축적인 흐름의 경우, 상세한 내용을 위해 Flow 내부에 Flow를 정의한다. 또한 pre, post 조건 또한 해당하는 모듈의 내부에 정의한다. Flow의 흐름을 수행 요청하는 client와 수행하는 host와 같은 정보를 입력하여 저장하여 시퀀스 다이어그램을 생성할 수 있게 한다. 이렇게 저장된 상위 단계의 모델들은 최종적으로 클래스 다이어그램의 모듈들과 관계를 갖는다. Flow나 Usecase는 클래스 다이어그램의 메서드로서 구현하며 System과 Actor는 클래스로써 구현한다. 이렇게 단계별로 추적이 가능하도록 하고 상대적으로 추상화가 낮은 클래스 다이어그램을 통해 가이드 코드를 만든다. 클래스 다이어그램을 통해 클래스 파일의 정의 및 구조를 생성하고 각각의 관련된 모델의 정보로 Javadoc을 생성한다. 소스코드의 핵심인 메서드의 내용은 flow를 통해 지원한다.

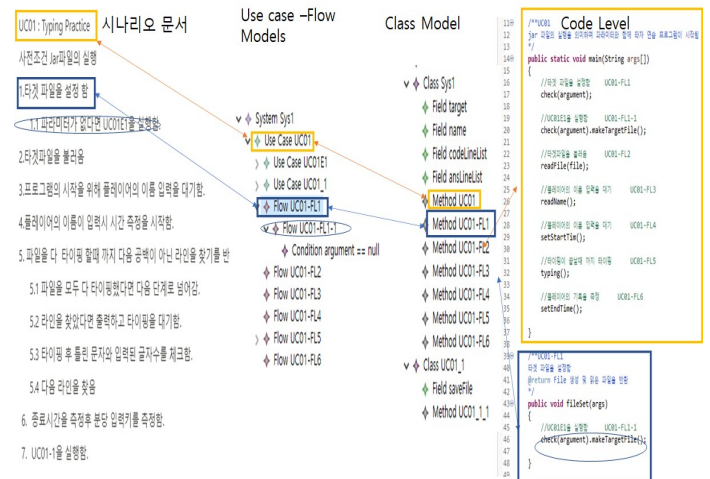


그림 3 Id 기반 추적 사례

그림 3은 Id를 통해 시나리오 문서로부터 가이드 코드가 생성되는 추적을 EMF를 통해 확인한 그림이다. 위쪽의 노란 네모는 Usecase UC01의 추적이며 아래쪽의 파란 네모는 UC01-FL1의 Id를 가진 흐름의 추적이다. 가이드 코드의 기반이 되는 클래스 다이어그램의 모델은 클래스와 필드 메서드 모델로 나뉘며 메서드와 클래스 모델은 Id값을 통해 추적이 가능한 것을 확인할 수 있다. 그림 3의 클래스 모델에서 두 개의 클래스 모델이 Sys1과 Use case의

Id를 가지는 UC01_1의 Id를 가지는 것을 그림에서 확인할 수 있다. 메서드의 경우도 Use case 혹은 Flow의 Id와 동일한 값을 가지고 있다. Usecase의 시나리오 문서는 Usecase와 그 내부의 Flow 모델로 저장되며 파란 원과 같이 1.1과 같은 대체 흐름은 Flow 내부의 Flow로 저장되어 있다. 이러한 내부 구조는 가장 우측의 가이드 코드에서 메서드 내부의 정보로도 활용되며 노란색의 Usecase는 Usecase의 흐름도가 메서드 내부로 활용된다. 이렇게 추적된 설계 정보를 바탕으로 JavaDoc과 주석을 생성하여 가이드 코드에 설계에 대한 정보와 ID 정보를 추가한 것을 확인할 수 있다. 또한 그림 3과 유사하게 각각 최종적으로 생성된 RDOM을 사용하면 동일한 가이드 코드를 생성할 수 있다.

4. 결론

본 연구를 통해 자바를 기반으로 설계 정보를 입력하면 자동으로 추적이 가능한 메타 모델을 생산하고 해당 모델을 기반으로 가이드 코드를 생성하고 UML 다이어그램을 복원하였다. 또한 JavaDoc과 주석의 생성으로 코드의 문서화를 지원한다. 이를 통해 설계를 기반으로 개발자들이 설계에 대한 이해, 구현, 문서화에 대한 시간 소요를 줄이고 설계를 기반으로 구현하도록 도와 보다 더 쉽게 개발이 가능할 것으로 기대한다. 향후 연구로는 설계한 모델을 확장하여 요구사항에 대한 설계 정보를 확장하고, 상태 다이어그램, 객체, 컴포넌트 등 다양한 설계 요소를 확장한 시스템을 만들 예정이다. 그리고 모호하고 정제되지 않은 설계 문서를 정제하여 모델을 생성할 방법과 그를 응용하여 보다 더 구체적인 설계로 확장하여 코드의 자동화를 더욱 더 높이는 연구를 계획하고 있다.

Acknowledgment

이 논문은 교육부 및 한국연구재단의 4단계 두뇌한국21 사업의 지원(F21YY8102068)과 2021년도 정부(교육부)의 재원으로 한국연구재단의 지원(No. 2021R111A305040711, No. 2021R111A1A01044060)을 받아 수행된 연구임.

참고 문헌

- [1] 최규식(Che Gyu-Shik). "테스트 및 운영중 소프트웨어의 오류 원인 분석." 한국콘텐츠학회 종합학술대회 논문집 4.1 (2006): 455-458.
- [2] 박보경(Bo Kyung Park),문소영(So-Young Moon),서채연(Chae Yun Seo), 김영철(R Young Chul Kim). "국방 소프트웨어 자원관리시스템의 소스 코드 품질 향상을 위한 소프트웨어 가시화 적용 구축

사례." 한국정보과학회 학술발표논문집 2017.12 (2017): 161-163.

[3] 권하은, 박보경, 김영철, 김영수, 이상은. "요구사항 스펙과 소스 코드 간 동기화를 위한 자동 프로젝트 문서 도구 개발", 한국정보처리학회 추계학술발표대회 논문집 2015.10: 1035-1037

[4] 이진협. "소프트웨어 프로세스 자동화 기반의 요구사항 추적성 매트릭스 연구." 국내석사학위논문 홍익대학교 대학원, 2018. 서울

[5] Hyun Seung Son, R Young Chul Kim. "xCodeParser based on Abstract Syntax Tree Metamodel (ASTM) for SW Visualization" INFORMATION -YAMAGUCHI: 963-968.

[6] 손현승, 김우열, 김영철. "이종 스마트폰 앱 개발환경 구성을 위한 모델변환 적용 방법." 정보과학회논문지: 컴퓨팅의 실제 및 레터 20.4 (2014): 238-242.

[7] 김영현(Young-Hyun Kim), 이재천(Jae-Chon Lee). "안전중시 시스템의 모델기반 설계에서 메타모델을 활용한 기능 고장의 탐지 및 안전 요구사항 검증." 한국산학기술학회논문지 17.9 (2016): 308-313.