소프트웨어 공학 가이드라인 기반 통합개발환경 †

최승용*, 홍찬기**, 김정아***‡
*관동대학교 전자계산공학과
**관동대학교 컴퓨터학과
***관동대학교 컴퓨터교육과

e-mail : {boromi*, chankih**, clara*** }@ kd.ac.kr

A IDE based on Software Engineering Guidelines

Seung-Yong Choi*, Chan-Ki Hong**, Jeong-Ah Kim***
*Dept. of Computer Engineering, Kwan-Dong University
**Dept. of Computer Science, Kwan-Dong University
***Dept. of Computer Education, Kwan-Dong University

요 약

ALM 구현을 위한 툴들 간의 통합 시도는 툴들 간의 통신 복잡성을 증가시키고 툴을 구성하는 컴포넌트 사이의 의존성과 컴포넌트 내부의 복잡성을 증가시킨다. 이는 통합개발환경의 유지보수를 어렵게 하는 결정적 원인이 된다. 따라서 본 논문은 통합개발환경 구성 컴포넌트들 간 통신 및 레이어 구조를 특성(기능) 관리 영역과 지식 관리 영역으로 이원화하고 가이드라인 MVC 모델로 구조화한 통합개발환경 아키텍처를 제시한 소프트웨어 공학 가이드라인 기반 통합개발환경 모델을 제안한다. 제안한 모델은 통합개발환경 구성 컴포넌트의 재사용성, 확장성, 유지보수성 향상과 더불어 통합개발환경 개발/관리자에게 효율적인 개발/관리 지침을 제공한다.

1. 서론

현재의 개발 툴은 애플리케이션 개발 패러다임의 변화에 맞춰 ALM(Application Lifecycle Management) 관점의 협업 기능을 갖추기 시작하고 있으며 무엇보다 요구분석, 디자인, 테스트, 배포 등에 대한 전체 개발 과정을 지원하기 위한 하나의 개발 플랫폼으로 탈바꿈하고 있다. 그러나 이와 동시에 통합개발환경에 ALM 구현이 강화되면서 ALM 관점의 통합개발환경이 해결해야 하는 문제점도 나타나기 시작했다.

ALM 구현을 위한 개발 툴들의 통합으로 발생하는 복잡도의 지수적(exponential) 증가는 통합개발환경의 유지보수를 어렵게 하는 결정적 원인이다. 따라서 통합개발환경의 효율적 유지보수 환경을 구성하기 위해서는 통합개발환경을 구성하는 컴포넌트 내부의 복잡성과 컴포넌트 간 의존성을 낮춰 통합개발환경 구성 컴포넌트를 경량화할 필요가 있다.

이와 같은 통합개발환경 관리 문제의 해결 대안으로는 통합개발환경을 구성하는 컴포넌트에서 기능 요소와 소프트웨어 공학 지식을 분리하여 기능 컴포넌트와 지식 컴포넌트를 독립적으로 관리할 수 있는 통합개발환경 가이드라인을 개발자/관리자에게 제공하는 것이다.

따라서 본 논문에서는 기능 요소와 소프트웨어 공학 지식이 혼재되어 있어 복잡하고 무거운 통합개발환경 구성 컴포넌트를 기능 영역과 지식 영역으로 분리하여 유지보수성을 향상시킨 소프트웨어 공학 가이드라인 기반 통합개발환경 모델을 제안한다.

본 논문의 구성은 다음과 같다. 2 장에서는 배경 이론을 고찰하고 3 장에서는 배경 이론을 적용한 가이드라인 기반 통합개발환경 모델을 제시한다. 4 장에서는 소프트웨어 공학 가이드라인 온톨로지 구현 사례를 살펴보고 5 장에서는 결론 및 향후 연구 방향을 언급한다.

2. 배경 이론

2.1 통합개발환경과 소프트웨어 공학 지식과의 관계

소프트웨어 공학 기술은 소프트웨어 개발 및 유지보수를 지원하는 소프트웨어 개발기술, 소프트웨어 생산성과 품질 향상을 관리 차원에서 지원하는 소프트웨어 관리기술로 분류가 가능하다.

소프트웨어 개발기술은 소프트웨어 생명주기별로 개발 절차, 개발 기법 및 개발 지원도구를 지원하는 요소들로 구성된다. 소프트웨어 관리기술은 소프트웨어 관리 절차, 관리 기법 및 관리 지원도구를 지원하는 세부 요소들로 구성된다.

† 본 과제는 정보통신산업진흥원의 SW 공학 요소기술 개발과 전문인력 양성사업의 결과물임을 밝힙니다.

‡ 교신저자는 관동대학교 컴퓨터교육과 교수 김정아임을 밝힙니다.

2.2 의료정보 분야의 가이드라인 개발 연구 동향

임상 가이드라인은 의료인과 환자가 특정 환경에서 적절한 의학적 진료에 대하여 의사 결정하도록 도와주는 체계적으로 개발된 진술문이며 부당한 진료 변이와 오류, 자원 낭비를 감소시키고 비용을 줄여 진료의 질을 향상시키는 목적을 가진다.

그러나 서술식 가이드라인에서 표현된 지식은 종종 불명확하고, 모호하고, 불완전하며, 다의적이며, 심지어 모순적이기까지 하여 그것을 전산화하기 위해 해석하는데 있어 해결해야 하는 여러 문제점도 나타났다.

이에, 의료정보 분야에서는 이러한 문제점을 해결하고자 하는 가이드라인 표현 모델 연구가 활발이 진행되어 오고 있다. 가이드라인 표현 모델에서 컴퓨터로 해석 가능한 가이드라인 형식화 기술을 Computer-Interpretable Guideline(CIG) 모델링 또는 CIG Formalism 이라 말한다[1].

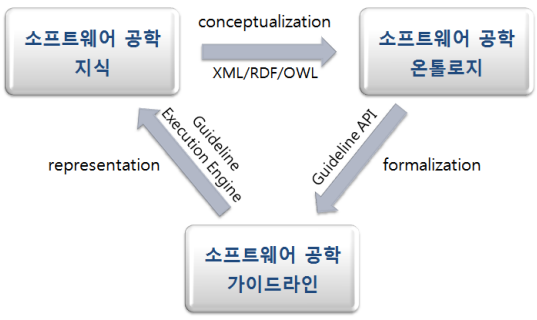
가이드라인 전산화와 관련된 모델링/도구개발 연구는 크게 컴퓨터에서 가이드라인을 실행시키는 방법에 대한 연구와 XML 기반의 방법론에 관한 연구로 나눠볼 수 있다. 컴퓨터에서 가이드라인을 실행시키는 방법에 대한 연구로는 1989 년 Arden Syntax 를 시작으로 영국의 PRODIGY, PROforma, 미국의 EON, GLIF, SAGE 등이 있다. XML 기반의 방법론 연구로는 CPG-RA, GEM, HGML, Stepper 가 대표적이다[2].

본 논문에서는 컴퓨터에서 실행할 수 있는 소프트웨어 공학 가이드라인의 형식화 기술로 의료정보 분야의 가이드라인 표현 모델링 기술을 적용한다.

가이드라인 형식화 기술은 상호 운용 가능한 소프트웨어 공학 가이드라인 컴포넌트 개발을 가능하게 한다. 그 결과, 개발자는 통합개발환경에 새로운 개발 지원 모듈을 쉽게 추가할 수 있고 한번 개발한 지원 모듈을 다양한 통합 환경에 재사용할 수 있는 유연한(flexible) 구조(architecture)의 통합 제작 환경을 확보할 수 있게 된다.

3. 가이드라인 기반 통합개발환경 모델

3.1 통합개발환경 아키텍처



(그림 1) 가이드라인 통합개발환경 모델 개념 구조

(그림 1)의 개념 구조를 갖고 있는 가이드라인 기반 통합개발환경 모델은 소프트웨어 공학 개념과 그 개념을 나타내는 용어가 분명히 정의된 소프트웨어 공학 온톨로지를 구현하기 위해 소프트웨어 공학 지식의 개념화(conceptualization) 과정에서 온톨로지 표현 기술(XML, RDF, OWL 등)을 사용한다.

소프트웨어 공학 온톨로지는 소프트웨어 공학 지식의 개념과 용어 정의 그리고 개념과 용어 사이의 논리적 관계를 체계적으로 조직화한 분류체계를 표현하게 된다.

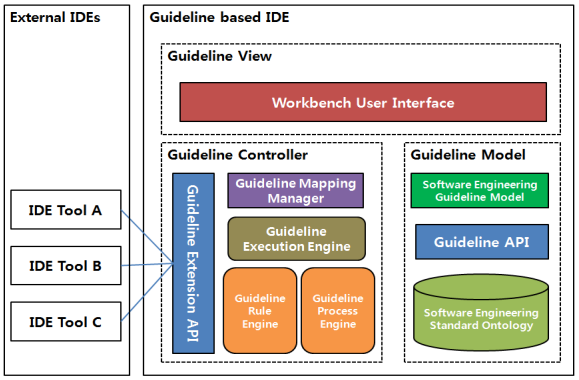
소프트웨어 공학 가이드라인은 소프트웨어 공학 온톨로지를 기반으로 형식화(formalization) 과정을 통해 소프트웨어 공학 가이드라인 모델에 부합하는 인코딩(encoding)된 소프트웨어 공학 가이드라인을 생성하게 된다. 소프트웨어 공학 가이드라인의 형식화 과정에서는 컴퓨터로 해석 가능한 가이드라인을 구현하기 위해 인코딩 처리를 지원하는 Guideline API 를 활용한다.

인코딩한 소프트웨어 공학 가이드라인은 소프트웨어 공학 가이드라인 실행을 위해 가이드라인 실행 엔진(Guideline Execution Engine)을 사용한다.

위와 같은 일련의 과정을 통해 제안한 모델은 소프트웨어 공학 가이드라인을 지식으로 저작하고 컴퓨터가 이해할 수 있도록 인코딩한 가이드라인을 통합개발환경에 포함시켜 통합개발환경 사용자에게 제공 가능하게 한다.

가이드라인 기반 통합개발환경에서 (그림 1)을 실현하기 위한 단계적 과정을 요약하면 다음과 같다.

① 표준이며 공유가능하고 재사용 가능한 소프트웨어 공학 지식을 온톨로지로 모델링한다.
② 온톨로지로 표현한 소프트웨어 공학 지식을 통합개발환경에서 활용 가능한 소프트웨어 공학 지식의 가이드라인으로 인코딩한다.
③ 가이드라인 실행 엔진은 실행 가능한 가이드라인 인스턴스를 생성한다.
④ 가이드라인 인스턴스는 가이드라인 실행 엔진을 통해 해석되어 통합개발환경을 통해 가이드라인 인스턴스 적용 결과를 보게 된다.



(그림 2) 가이드라인 기반 통합개발환경 구조

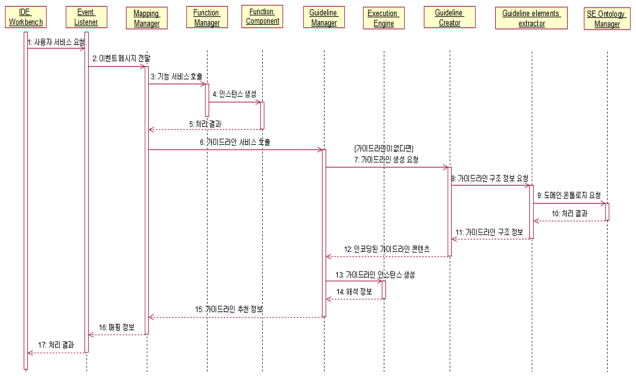
(그림 1)의 구성요소 간 유기적 관계를 고려하여 본 논문에서 제안하는 소프트웨어 공학 가이드라인을 생성하는 통합개발환경 모델의 전체 구성은 (그림 2)와 같다.

가이드라인 기반 통합개발환경 모델은 통합개발환경 툴의 유연성, 확장성, 재사용성, 유지보수성을 향상시키기 위해 MVC 모델을 응용한 Guideline MVC 모델 구조를 갖는다. Guideline MVC 모델의 역할은 <표 1>과 같다.

<표 1> Guideline MVC Model 역할

Guideline MVC Model	역 할
Guideline Model	가이드라인 컴포넌트를 직접 호출하지 않으며 가이드라인 구조 정보로 구성되어 있다.
Guideline View	가이드라인 모델로부터 가이드라인 콘텐츠(구성정보, 권고사항 등)를 생성한다.
Guideline Controller	가이드라인 모델 컴포넌트를 호출하고 가이드라인 데이터를 해석하여 Guideline Mapping Manager 를 통해 가이드라인 뷰에 관련 이벤트 메시지를 전달한다. 또한 Guideline Extension API 를 통해 외부의 다른 IDE 와도 통신한다.

가이드라인 기반 통합개발환경 실행 절차는 (그림 3)과 같으며 실행 절차의 내용은 다음과 같다.



(그림 3) 가이드라인 기반 통합개발환경의 순차도

사용자가 가이드라인 기반 통합개발환경의 IDE Workbench 에서 사용하고자 하는 서비스를 요청하면 Event Listener 는 Mapping Manager 에게 이벤트 메시지를 전달한다.

Mapping Manager 는 이벤트 메시지를 분석하여 기능 서비스 메시지는 Function Manager 에게, 지식 서비스 메시지는 Guideline Manager 에게 전달한다.

Function Component 객체는 해당 기능의 모듈을 실행하고 그 결과를 Mapping Manager 에게 반환한다.

Guideline Manager 는 가이드라인의 유무를 판단하고 가이드라인이 존재하면 Execution Engine 에게 가이드라인 인스턴스를 생성하게 한다. 가이드라인 존재하지 않으면 Guideline Creator 에게 가이드라인 생성을 요청하고 Guideline Creator 가 인코딩된 가이드라인 콘텐츠를 반환하면 Execution Engine 에게 가이드라인 인스턴스를 생성하게 한다.

Guideline Creator 가 Guideline elements extractor 에게 가이드라인 생성에 필요한 가이드라인 구조 정보를 요청하면 Guideline elements extractor 는 SE Ontology Manager 로부터 가이드라인 구조 정보를 획득하여 Guideline Creator 에게 가이드라인 구조 정보를 반환한다.

Guideline Manager 는 Execution Engine 으로부터 받은 가이드라인 해석 정보를 분석하여 가이드라인 권고 내용을 Mapping Manager 에게 반환한다.

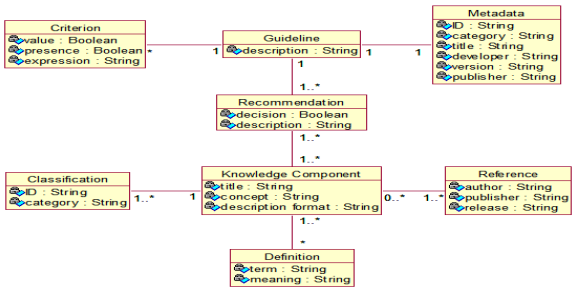
Mapping Manager 는 기능 요소와 가이드라인 요소 간의 매핑 정보를 Event Listener 에게 반환하고 사용자는 IDE Workbench 를 통해 요청 서비스 결과를 확인하게 된다.

3.2 소프트웨어 공학 가이드라인 모델

소프트웨어 공학 가이드라인은 소프트웨어 개발 환경에서 적절한 소프트웨어 공학 지식에 대하여 의사결정 하도록 도와주는 체계적으로 개발된 지침이다.

소프트웨어 공학 가이드라인의 목적은 소프트웨어 개발/관리자의 주관적 오류를 제거하고 소프트웨어 공학 지식의 오용으로 인한 자원 낭비를 감소시킬 수 있도록 소프트웨어 개발/관리자에게 효율적인 소프트웨어 개발/관리 환경을 지원하는 것이다.

소프트웨어 공학 가이드라인의 구조와 각 항목에 대한 상세 내용은 (그림 4), <표 2>와 같다.



(그림 4) 소프트웨어 공학 가이드라인 객체 다이어그램

<표 2> 소프트웨어 공학 가이드라인 구성 항목 설명

항목	내용
Guideline	소프트웨어 공학 지식 정보(Knowledge Components)와 분류체계 계층구조(Metadata)에 기반한 추상화를 사용하는 기준(Criterion)으로부터 구체화된 가이드라인
Criterion	가이드라인 추상화 및 해석 기준 정보
Recommendation	가이드라인의 필수 권고사항과 조건별 권고사항을 구별하고 실행하는 기능 담당
Metadata	가이드라인 추상화를 위한 계층적 구조 정보
Knowledge Component	소프트웨어 공학 지식 범주를 분류하는 정보와 소프트웨어 공학 지식 범주 내에서 중요한 정보를 갖고 있는 전문 지식 컴포넌트
Classification	소프트웨어 공학 지식 체계에 대한 식별 정보
Reference	소프트웨어 공학 지식의 이론적 근거를 참조한 정보
Definition	소프트웨어 지식 영역의 중요한 용어와 용어 의미 정의

XML 기반 가이드라인은 트리 형태의 문서로 고정되게 설계되어 노드가 문서에 포함되어 인덱스 되기 때문에 메타데이터를 표현하기 위한 유연성이 부족하다. 반면 온톨로지 기반 가이드라인에서 노드는 인덱스 되지 않고 URI 를 갖는 자원이기 때문에 메타데이터를 표현하기 위한 유연성을 지원한다.

본 논문의 소프트웨어 공학 가이드라인 모델은 가이드라인의 지식 정보를 표현하는데 있어 XML 이 구문적 계층에서 데이터의 구조를 정의하는데 이용되고 RDF/OWL 이 의미적 계층에서 지식표현을 위한 기능을 담당하게 된다.

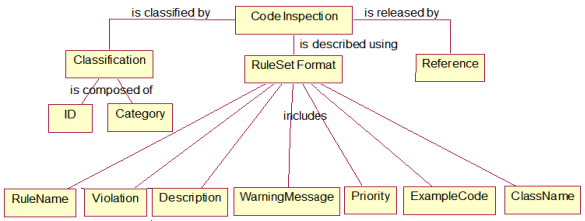
<표 3> 온톨로지 기반 소프트웨어 공학 가이드라인 모델의 장점

구분	XML 기반 가이드라인	온톨로지 기반 가이드라인
데이터 모델	순서화된 트리지향 모델	객체의 관계지향 모델
스키마	문법적 해석	의미적 해석
순서 중요도	노드의 배치가 다르게 되어 있다면, 파서는 문서를 다르게 인식하기 때문에 노드의 순서가 매우 중요함	각각의 트리플들은 독립적으로 존재함으로 트리플들의 발생 및 배치 순서는 중요하지 않음
문서구조 이해도	XML 데이터를 사용하기 위해서 문서의 전체 구조를 이해해야 함	사용될 부분만 이해하면 됨
질의어 처리	복잡한 트리구조로 되어 있어서 쿼리가 복잡함	서로 독립적인 트리플들의 집합이므로 쿼리가 간단함

4. 가이드라인 온톨로지 구현 사례

본 논문에서는 SWBOK 의 소프트웨어 품질 지식 영역에서 언급하는 소프트웨어 인스펙션(Inspection)을 도메인 지식으로 선정하고 소프트웨어 공학 가이드라인 온톨로지 프로토타입을 구현했다.

코드 인스펙션 가이드라인 온톨로지는 코드 인스펙션 시 기준이 되는 요소들을 룰(rule)로 정의하고 있다.



(그림 5) 코드 인스펙션 지식의 계층 구조

본 논문에서는 (그림 5)와 같이 코드 인스펙션 지식 구조를 코드 인스펙션의 룰 정보를 식별하기 위한 분류(Classification) 항목, 코드 인스펙션을 지정된 형식으로 기술하기 위한 포맷(RuleSet format) 항목, 코드 인스펙션에 관한 참조(Reference) 항목으로 대별한다.

룰의 구성은 룰의 이름(RuleName), 룰의 위반사항 내용(Violation), 룰에 대한 상세 설명(Description), 룰의 위반사항에 대한 경고 메시지(WarningMessage), 룰의 우선순위(Priority), 룰과 관련된 오류코드 보기(ExampleCode), 룰과 관련된 자바 클래스(ClassName)로 구성되어 있다.

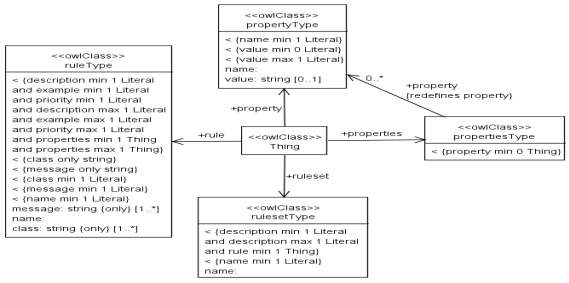
해당 인스펙션 도메인에 대한 가이드라인 온톨로지를 클래스와 그들 간의 관계성으로 표시하면 (그림 6)과 같고 (그림 7)은 인스펙션 가이드라인 OWL 문서의 일부이다.

5. 결론

본 논문에서는 통합개발환경 내의 기능 컴포넌트와 지식 컴포넌트를 독립적으로 개발하고 관리할 수 있는 가이드라인 기반 통합개발환경 모델을 제안했으며 제안한 모델을 통해 통합개발환경 내의 혼재된 기능 요소와 소프트웨어 공학 지식을 분리하기 위한 방법으로 온톨로지 기반 소프트웨어 공학 가이드라인 구조를 제시했다.

또한, 통합개발환경 내 기능 컴포넌트와 지식 컴포넌트를 효율적으로 개발하고 효과적으로 관리할 수 있도록 컴포넌트들 간 통신 및 레이어 구조를 가이드라인 MVC 모델로 구조화한 통합개발환경 아키텍처를 제시했다.

본 논문은 통합개발환경의 개발/유지보수 시 효율적인 컴포넌트 개발/관리 접근 방법으로 소프트웨어 공학 가이드라인 기반 통합개발환경의 개념을 제안한 것으로 향후에는 소프트웨어 공학 가이드라인 형식화 방법을 구체적으로 연구하고 컴퓨터에서 실행 가능한 소프트웨어 공학 가이드라인을 개발해야 한다. 이를 토대로 하여 소프트웨어 공학 가이드라인 기반 통합개발환경 구현도 필요하다.



(그림 6) 인스펙션 가이드라인 온톨로지 클래스 다이어그램

(그림 7) 인스펙션 가이드라인 온톨로지의 일부 예

참고문헌

[1] Robert A. Greenes, “CLINICAL DECISION SUPPORT: The Road Ahead”, Academic Press, Inc., 2006

[2] Kaiser K, Akkaya C, Miksch S, “How can information extraction ease formalizing treatment processes in clinical practice guidelies? A method and its evaluation”, Artif Intell Med., pp.151-163, 2007

[3] Pierre Bourque, Robert Dupuis, “Guide to the Software Engineering Body of Knowledge: 2004 Version”, IEEE., 2005

[4] Tom Gilb, Dorothy Graham, “Software Inspection”, Addison-Wesley, 1993