역공학 기반 모델간 변경분석 및 시각화 도구의 설계

권진욱[0] ,이정선 ,카오 티 리 ,이우진
경북대학교 전자전기컴퓨터학부
kjw1217@nate.com ,rubyindark@knu.ac.kr ,caoly1980@yahoo.com ,woojin@knu.ac.kr

(A design for reverse engineering based model change information analysis and visualization framework)

Kwon Jinwook[0] ,Jung Sun Lee ,Cao Thi ly ,Lee Woo Jin
School of Electrical Engineering and Computer Science, Kyungpook National University

요 약

소프트웨어를 유지보수하고 관리하는 과정에서 변경된 정보가 프로그램 구조에서의 어느 부분을 나타내고 그 변경이 전체 구조에서 어떤 영향을 미치는가에 대한 체계적이고 효율적인 관리방법이 필요하게 되었다. 기존의 상용화 프로그램에서 채택하고 있는 코드레벨에서의 라인과 라인상의 비교분석 정보만으로는 프로그램의 변경을 구조적인 관점에서 바라보기가 힘들었다. 이 논문에서는 역공학을 이용해서 변경전후의 프로그램 모델구조 정보를 얻고 그 정보를 서로 비교 분석하는 방법을 제시한다. 또, GMF(graphical modeling framwork)로 프로그램의 모델정보를 클래스 다이어그램 형태로 보여주며 비교분석된 정보를 다이어그램에서 보여줄 수 있는 시각화 규칙을 제시한다. 변경된 정보를 다이어그램에서 쉽게 파악할 수 있게 표현함으로써 프로그램 개발자가 아니라도 유지보수에서의 변경을 구조적으로 알 수 있고 관리할 수 있다.

1. 서 론

소프트웨어를 유지보수하고 관리하는 과정에서 변경된 부분에 대한 구조적인 정보를 파악하고 관리하기란 쉽지 않다. 비록 변경 전후의 코드를 서로 비교해서 변경된 부분을 알려주는 상용화 툴이 있지만 이러한 시스템에서는 두 개의 코드를 시작에서 끝까지 순차적으로 라인과 라인을 비교해서 어떤 라인이 추가되고 삭제되었는지에 대한 정보만을 나타내준다. 프로그램 개발자가 아니라면 소스코드 레벨에서의 텍스트적인 변화가 시스템에 어떤 영향을 주는지 시스템에서의 어떤 부분이 바뀌었는지에 대한 정보를 알 수 없다.

지금까지 역공학을 이용한 프로그램의 분석은 대부분 개발과정에서의 오류를 찾기 위한 소스코드의 분석이 주를 이루었다. 그러나 소프트웨어의 비용적인 측면에서 유지보수에 더 많은 비용이 드는 것을 감안하면 개발과정의 오류검증만큼 유지보수과정에서의 변경사항에 대한 정보를 파악하고 관리할 필요가 있다.

본 연구는 방위사업청과 국방과학연구소의 지원으로 수행되었습니다. (UD060048AD)

변화된 정보의 의미를 파악해서 유지 보수에 드는 비용을 줄이고 효율적인 방법으로 관리하기 위해 산업 전반에 걸쳐 시각화를 많이 사용하고 있다. 시각화는 어떤 정보나 데이터를 인식하기 좋게 표현하는 방법으로 텍스트형태의 정보를 그래프나 다이어그램으로 표현해서 직관적으로 이해하기 쉽게 도와주는 역할을 한다. 데이터베이스관련 분야에서 데이터의 흐름과 분석을 효과적으로 하고자 시각화 기법을 사용하고 있고 소프트웨어를 개발하는 과정에서도 UML을 사용해 시각화함으로써 개발에서의 비용과 효율성을 높이고 있다.

이 논문에서는 소프트웨어의 유지보수 입장에서 변경전후의 프로그램의 변화모습을 시각화 기법을 통해 표현함으로써 시스템이 구조적으로 어떻게 바뀌었는지 어떤 영향을 미치는지를 한눈에 파악할 수 있도록 새로운 모델을 제시하고자 한다. 위에서 언급한 코드레벨에서의 코드비교가 큰 의미를 갖지 못하기 때문에 이러한 단점을 보완하기 위해 이 논문은 변경된 소스코드를 역공학을 통해서 클래스 다이어그램으로 표현하고 시각화함으로써 좀더 직관적이고 구조적인 관점에서 프로그램을 바라볼 수 있는 방법을 제시하고자 한다.

프로그램을 유지 보수할 때 변경전과 변경후의 변화모습

을 시각화하기 위해 이 논문이 제시하는 작업은 크게 3부분으로 나눌 수 있다. 첫 번째, 역공학을 통해 소스로부터 모델 정보를 추출한다. 변경전후의 두 소스코드를 클래스 단위로 파싱해서 클래스 내의 속성과 메소드의 정보들을 얻는다. 이렇게 얻어진 정보들을 클래스 아웃라인이라 한다. 두 번째, 두 개의 모델 outline 정보를 서로 비교하여 변화된 부분을 추출한다. 얻어진 모델 정보를 2장에서 제시한 규칙을 통해 서로 비교하고 변화된 부분을 테이블에

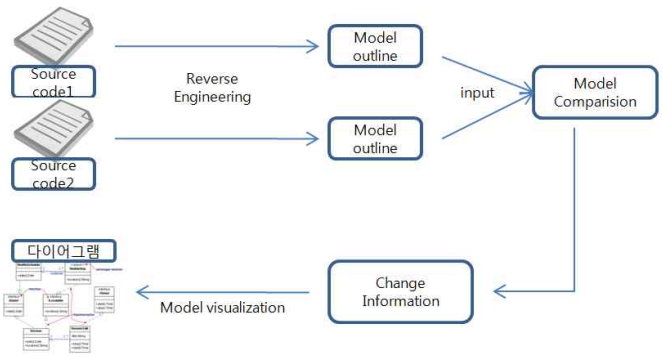

그림 1. 변경 분석 및 시각화

기록해 둔다. 마지막으로, GMF 다이어그램 편집기를 이용해서 위에서 얻어진 변화정보를 다이어그램에 시각화하기 위해 변화부분 테이블을 참고하고 특정 색깔을 이용해 프로그램의 변화된 부분을 시각적으로 한눈에 파악할 수 있도록 한다.

본 논문의 구성은 다음과 같다. 2장에서는 두 모델 사이의 변경된 부분을 분석하기 위해 어떤식으로 접근하고 구분하는지에 대해 다룰 것이다. 3장에서는 두 모델의 비교를 통해 얻어진 정보를 어떻게 다이어그램 상에 표현하고 시각화할지에 대해 다룰 것이다. 4장에서는 프레임워크 구조에 대해 설명할 것이고 마지막으로 5장에서는 결론 및 향후 연구를 기술한다.

2. 모델간 변화정보 분석

프로그램의 변화 정보를 시각화 하기위해 변경전후의 프로그램으로부터 얻은 모델 정보의 비교 분석이 필요하다. 변경된 코드를 단순히 라인과 라인사이의 추가나 삭제정보가 아니라 구조적인 관점에서 접근하기 위해서는 코드의 변경된 부분을 크게 3가지 의미로 나누어 생각해야 한다. 새로이 생성된 부분, 없어진 부분, 수정된 부분으로 나눌수 있다. 변경된 부분은 클래스 내부의 변경과 클래스 간의 관계에서의 변경 두가지가 있다. 클래스 내부의 변경은 속성이나 메소드 등 내부 속성의 변경을 나타내고 클래스 간의 변경은 내부 변경에 의한 연관관계의 변경을 나타낸다. [그림2]은 클래스 내부의 메타모델을 간략히 나타낸 것이다. 각 클래스 자체에서의 생성과 삭제는 프로그램에서 없었던 새로운 클래스가 추가/삭제되었거나 클래스 내부의 속성과 메소드가 이 추가/삭제 되었을 때를 의미한다. 수정은 클래스 내부의 속성과 메소드의 추가/삭제/수정이 이루어 졌을 때 클래스가 변경되었다고 하고, 메소드의 헤더나 몸체부분이 바뀌었을 때 메소드가 변경 되었다고 한다.

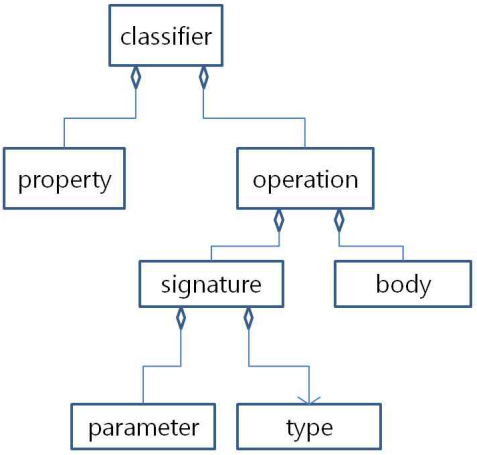

그림 2. Classifier meta-model

변경된 부분을 어떻게 찾고 구분할 것인지에 대해 아래의 규칙을 적용하였다.

- 변경 전/후의 outline을 가지고 프로그램상에 있는 모든 클래스의 이름을 서로 비교 해서 같은 이름의 클래스를 골라낸다.
- 위 규칙을 통해서 얻어진 클래스 내부의 속성과 메소드를 비교해서 같은 이름과 타입의 속성 와 메소드를 골라 낸다.
- 메소드의 헤더를 비교한다. 헤더를 비교할때는 매개변수의 정보를 가지고 비교한다.
- 메소드 몸체의 변화된 부분이 있는지 라인비교를 통해 구분한다.
- 이름만이 다른 같은 클래스인지 아니면 전혀 새로운 클래스인지를 클래스 id를 통해 구분한다.
- 두 번째 규칙을 만족하지 못하는 타입이 같고 이름만이 다른 속성과 메소드의 id를 비교해서 이름이 변경된것인지 새로운 것인지를 구분한다.

이 6가지 규칙을 적용하여 class 내부의 추가/삭제/수정의 정보를 알 수 있고 이 정보는 diagram에서 시각화 하기 위해 추가/삭제/수정 Table(intra_class_table)을 각각 따로 만들어서 관리한다.

클래스 간의 관계정보는 다음의 규칙을 따른다. 첫째, 클래스의 outline 정보를 가지고 속성의 타입과 메소드의 매개변수 타입, 몸체 부분에서의 지역변수 타입들을 분석해서 연관관계를 table에 저장한다. 여기서의 연관관계는 UML 클래스 다이어그램에서 정의하는 각 클래스간의 관계를 그대로 이용한다. 둘째, 클래스 내부의 변경정보를 가지고 있는 table을 참고하여 이름이 변경된 클래스와의 연관관계정보를 찾아 연관관계 table을 업데이트한다.

관계에서의 변경은 변경전의 관계 table과 변경후의 관계 table을 서로 비교함으로써 변경된 관계에 대한 정보를 얻을 수 있다. 얻어진 정보는 inter_class_table에 기록해 둔다.

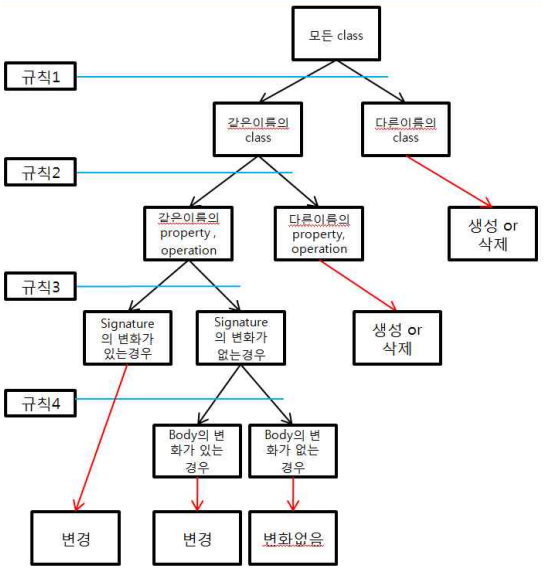

그림 3. 변경된 부분을 구분하는 방법

3. 모델의 변화정보의 시각화

모델의 변화 정보를 다이어그램에서 표현하기 위해 역공학을 이용하고 시각화 규칙을 적용 할 것이다. 역공학은 바이너리 파일로부터 소스코드를 얻거나 소스코드로부터 소프트웨어의 구조적인 정보를 얻고자 할 때 사용하는 소프트웨어 분석기법이다. 이 논문에서는 소스코드의 비교만으로는 얻을 수 없는 시스템의 구조적인 변화를 시각화 하고자 역공학 기법을 통해 프로그램의 모델 정보를 얻고 클래스 다이어그램에서 코드의 변화(추가,삭제,수정)를 몇몇 규칙을 통해 시각화 하고자 한다.

소스코드에서 parser를 통해 클래스의 속성과 메소드의 이름과 타입 그리고 메소드의 매개변수정보를 얻을 수 있는데 얻어진 정보를 hierarchy하게 tree구조로 저장한다. GMF를 통해 구현된 클래스다이어그램 뷰어를 이용하여 프로그램의 outline 정보를 가지고 클래스 다이어그램을 그려나갈 때 변경정보가 저장된 두 table(inter_class_table과 intra_class_table)을 참고하여 그려나간다. 이때 변경된 부분을 다이어그램 상에 시각화를 해주어야 한다. 이 논문에서는 클래스 다이어그램 상에 나타나는 노드들과 간선, 그리고 클래스 내부의 속성과 메소드들의 이름을 특정한 색과 모양으로 구분지음으로써 프로그램의 변경 여부를 시각화해서 나타내고자 한다. 시각화를 위한 규칙을 아래에서 기술한다.

- 새로 생성된 클래스와 속성, 메소드, 연관관계는 파란색으로 나타낸다.
- 없어진 클래스와 속성, 메소드, 연관관계는 녹색으로 나타낸다.
- 변경된 클래스와 속성, 메소드, 연관관계는 붉은색으로 나타낸다.

4. 모델 변경의 시각화 프레임워크 구조

[그림4] 는 본 논문에서 제시하는 모델 변경분석 시각화 프레임워크 구조를 보여준다. 이 프레임워크는 모델 비교 컴포넌트와 모델 시각화 컴포넌트로 구성된다. 모델 비교 컴포넌트는 역공학을 이용해서 파싱과 모델정보의 제너레이터로 구성된다. 생성된 모델정보를 가지고 모델 분석 컴포넌트에서는 클래스 내부 분석과 클래스간의 분석이 이루어진다. 모델 시각화 컴포넌트에서는 모델분석 컴포넌트에서 얻은 정보와 GMF를 이용해 클래스 다이어그램을 생성하고 시각화표현 규칙을 적용하여 전체 프로그램의 변경정보를 구조적인 시각에서 바라볼 수 있게 한다.

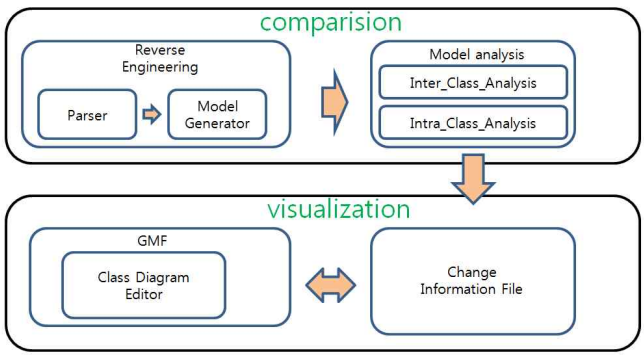

그림 4. 프레임 워크 구조

5. 결론 및 향후 연구

소프트웨어를 유지보수하고 관리하는 과정에서 변화된 부

분을 구조적인 관점에서 바라보기위해 본 논문에서 제시한 모델은 소스코드의 변경부분을 역공학을 통해 클래스다이어그램으로 표현하고 효율성과 인식성을 높이기 위해 변경된 부분을 색깔을 이용해 표현하는 것이다. 변경부분을 다이어그램에서 한눈에 파악하고 구조적인 변화와 소스레벨에서의 작은 변화를 쉽게 알 수 있게 함으로써 프로그램 구조의 빠른 이해와 진행사항의 파악에 효율성을 높일 수 있을 것으로 예상된다. 이 논문에서 제시된 프레임워크를 구현하는 작업을 앞으로의 연구에서 해야할 것이고 특히, 모델간 변경정보를 정확하게 얻기 위한 연구가 필요할 것이다. 또한 제시한 모델은 객체지향언어를 전제하에 이루어졌기 때문에 객체지향 언어를 사용하지 않은 소프트웨어의 관리를 어떻게 할 것인지에 대한 연구가 필요할 것으로 보인다.

6. 참고문헌

[1] The Current Official UML Specification, http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML

[2] Eclipse UML2 Tools, http://www.eclipse.org/modeling/mdt/?project=uml2tools

[3] Eclipse GMF(Graphical Modeling Framwork) http://www.eclipse.org/modeling/gmf/

[4] Ajila, S.A. Alam, S.“Using a Formal Language Constructs for Software Model Evolution”, Semantic Computing, 2009. ICSC '09. IEEE International Conference

[5] Cleland-Huang, J. ; Hayes, J.H. ; Domel, J.M. “Model-based traceability”, Traceability in Emerging Forms of Software Engineering, 2009. TEFSE '09. ICSE Workshop

[6] Antoniol, G., Canfora, G., Casazza, G., and De Lucia, A. " Information retrieval models for recovering traceability links between code and documentation" in proceedings of IEEE international conference on softwqre maintenance(ICSM,00)

[7] Weibgerber,p. and Diehl, s., "Identifying Refactorings from source-code changes" in Proceedings of 21st IEEE/ACM Interantional Conference on Automated Software Engineering (ASE'06)