소스코드의 그래프 변환 및 그래프 데이터베이스에서의 활용에 대한 연구

장석준[1], 김수현[2], 이임영[2]
[1]순천향대학교 소프트웨어융합학과 석사과정
[2]순천향대학교 컴퓨터소프트웨어공학과 교수
jangsj@sch.ac.kr, kimsh@sch.ac.kr, imylee@sch.ac.kr

A Study on Graph Conversion of Source Code and Its Use in Graph Databases

Seok-Joon Jang[1], Su-Hyun Kim[2], Im-Yeong Lee[2]
[1]Dept. of Software Convergence, Soonchunhyang University
[2]Dept. of Computer Software Engineering, Soonchunhyang University

요 약

최근 수많은 오픈소스로 공개되면서, 대부분의 소프트웨어가 오픈소스를 활용하여 구현되고 있다. 하지만, 오픈소스에 적용되어 있는 라이선스 간의 충돌 문제가 발생하면서, 라이선스 위반 문제가 지속적으로 발생하고 있다. 이러한 문제를 사전에 방지하기 위해 소스코드 분석이 필수적이지만, 다양한 기능이 실행되는 소스코드 특성 상 소스코드만 봤을 경우 직관적으로 분석이 어렵다는 문제점이 있다. 최근 소스코드의 효과적인 분석을 도와주는 다양한 도구들이 개발되었고, 그 중 한 가지 방법은 소스코드를 그래프로 변환하여 시각적인 편의성을 제공하는 방법이다. 그래프로 변환된 소스코드는 해당 시점에는 분석이 가능하지만, 분석이 필요할 때마다 변환을 해야 하는 문제점이 존재한다. 따라서 소스코드를 변환한 그래프 데이터를 저장하는 방법이 요구되었는데, 그래프 데이터베이스의 경우 특정 파일 형식만 지원하기 때문에 그래프 데이터 저장에 어려움이 존재한다. 본 제안방식에서는 소스코드를 변환한 그래프 데이터를 그래프 데이터베이스에 효과적으로 저장하고, 분석이 요구될 때마다 데이터베이스 상에서 즉각적으로 분석이 가능한 방법을 제안한다.

1. 서론

최근 수많은 소스코드가 오픈소스로 공개되면서 오픈소스 활용이 가능한 분야가 확장되고 있다. 하지만, 대부분의 사용자는 오픈소스에 적용되어 있는 라이선스에 대해 정확하게 알 수 없다. 이에 따라 하나의 소프트웨어를 만드는 과정에서 다양한 오픈소스를 분석 및 결합하는 과정이 필수적으로 요구되는데, 이러한 과정에서 라이선스 위반 문제가 지속적으로 발생하고 있다. 따라서 라이선스 위반 문제가 발생하는 것을 사전에 방지하는 것이 중요하다. 하지만, 소스코드는 외부 참조, 함수 호출 등 다양한 기능이 실행되기 때문에, 소스코드만 봤을 경우 직관적으로 분석이 어렵다. 이에 따라, 현재 다양한 소스코드 분석 도구들이 개발되었고, 그 중 한 가지 방법은 소스코드를 그래프로 시각화하여 그래프로 표현하는 방법이다.

그래프는 객체 및 객체 간의 관계를 시각화하여 그림 형태로 표현하는 것을 의미하며, 환경에 따라 다양한 방법으로 표현이 가능하다. 그래프는 시각화를 통해 사용자에게 특정 데이터 분석 및 다른 데이터와의 관계를 이해하는 것에 편의성을 제공한다. 소스코드 역시 직관적으로 분석은 어렵지만, 이를 그래프로 변환하여 시각화했을 때 외부 참조, 함수 호출 등의 관계가 직관적으로 분석이 가능하다.

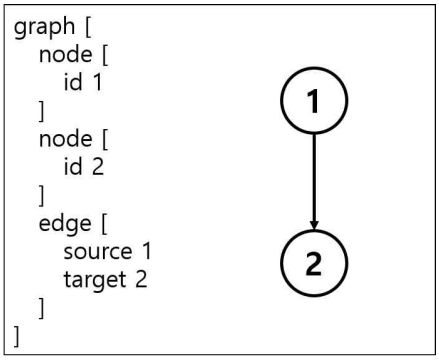
하지만, 그래프로 시각화된 소스코드는 해당 시점에만 분석이 가능하며, 지속적인 분석을 위해서는 데이터베이스로의 저장이 필수적이다.

따라서, 본 논문에서는 소프트웨어 또는 소스코드를 그래프를 활용하여 시각화하고, 이를 그래프 데이터베이스에 적합한 파일 형식으로 변환, 그래프 데이터베이스에 저장하는 방법을 제안한다.

2. 관련 연구

2.1 GML(Graph Modelling Language)

GML은 1996년 M. Himsolt에 의해 개발된 그래프 파일 형식으로, 초기 그래프 형식 표준화에 목적을 두고 있다[1]. 정점, 간선의 매우 간단한 구조로



(그림 1) 간단한 GML 예시

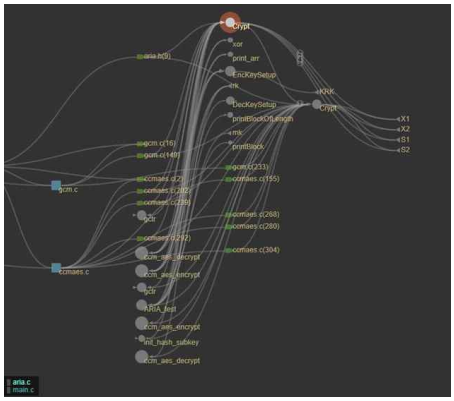
이루어져 있으며, JSON(JavaScript Object Notation) 파일 형식을 사용한다. GML의 간단한 예시는 (그림 1)과 같이 정점 정보(식별자 등)와 간선 정보(시작 정점, 끝 정점)로 이루어져 있다.

2.2 Code Graph

Code Graph는 소프트웨어 통합 개발 환경(IDE, Integrated Development Environment)인 Visual Studio에서 소스코드를 그래프로 분석해주는 도구로, 소프트웨어 문서 생성기인 Doxygen을 기반으로 개발되었다. Code Graph를 통해 Visual Studio 상에서 소스코드를 그래프 형태로 표현할 수 있으며, 그래프로 표현된 결과는 JSON, XML(Extensible Markup Language) 형태로 저장된다[2].

2.3 Neo4j

Neo4j는 Neo4j사가 개발한 그래프 데이터베이스 관리 시스템이다. 현재 상용화 되어 있는 그래프 데이터베이스 시스템 중 가장 대중적으로 사용되고 있으며, 단순하지만 그래프를 효과적으로 표현할 수 있는 Cypher 언어를 사용하여 데이터베이스 내의 데이터들을 그래프 형태로 표현한다[3].



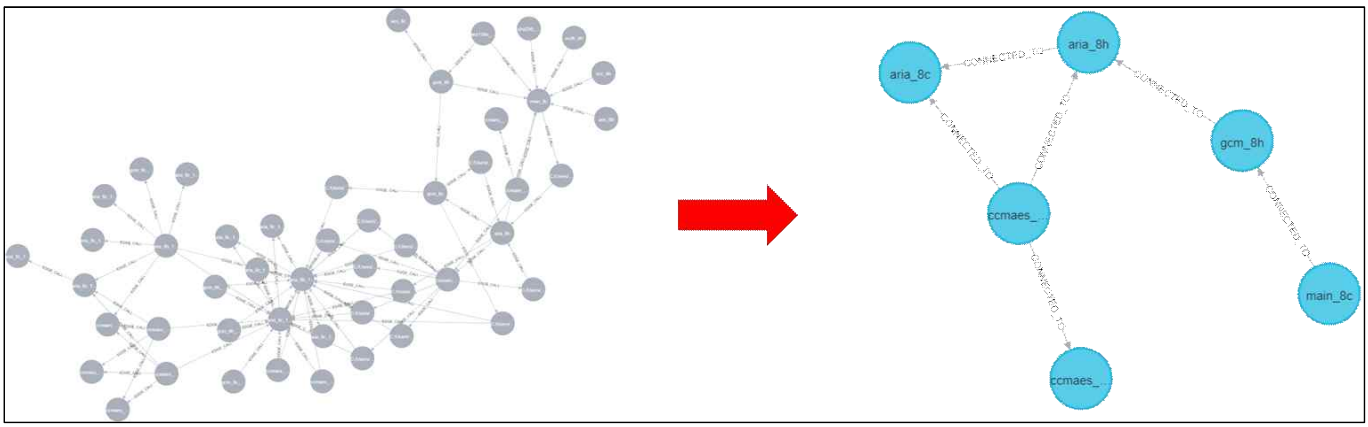
(그림 2) Code Graph 결과

3. 제안방식

본 제안방식은 Visual Studio에서 구현한 소스코드를 Code Graph 도구를 이용하여 그래프로 분석, 저장한 결과를 그래프 데이터베이스인 Neo4j에 효과적으로 적용하는 방법에 대해 설명한다.

임의의 소스코드 A를 Code Graph 도구를 이용하여 그래프로 시각화했을 때, Visual Studio 상에서 출력되는 그래프는 (그림 2)와 같다. 프로젝트 파일 내의 소스코드가 표현되며, 내부 함수와 변수는 각각의 소스코드에 간선으로 연결된다. 또한, 각 소스코드, 함수, 변수가 호출되는 경우 역시 간선으로 연결되어 표현된다.

하지만, Code Graph는 특정 변수가 호출될 때마다 간선으로 연결하여 표현하기 때문에, 같은 소스코드임에도 그래프 상에 여러 번 반복되어 나타난다는 특징이 있다. 이는 JSON, XML 파일으로 저장될 때도 나타나게 되는데, 파일 상에서는 추가적으로 동일한 변수를 구분하기 위해 변수 명 뒤에 특정 문자열을 추가하여 저장한다. 따라서 특정 문자열이 추가되어 여러 번 기록된 코드가 저장된 파일을 전처리 없이 그래프 데이터베이스에서 호출할 경우,



(그림 3) 그래프 전처리 전후 비교

(그림 3)의 왼쪽 그래프와 같이 무수히 많은 정점이 생성된다.

본 제안방식에서는 이러한 문제점을 해결하기 위해, Code Graph 도구의 결과물인 JSON, XML 파일에 대해 그래프 데이터베이스 입력 전 전처리 과정을 구현하였다. JSON 파일과 관련된 라이브러리를 지원하는 Python 언어를 활용하여 구현하였으며, 변수명 뒤에 추가된 특정 문자열 제거, 중복 정점 제거의 2가지 전처리 과정을 구현하였다.

전처리 과정을 구현한 결과, 그래프 데이터베이스에서 호출했을 때, 프로젝트 파일 내의 소스코드 간의 관계가 (그림 3)의 오른쪽 그래프와 같이 소스코드 명 뒤의 특정 문자열과 중복된 정점이 제거된 그래프 형태로 표현되었으며, 각 정점(소스코드)에는 소스코드 명, 포함된 함수 등의 정보가 저장되었다. 또한, 각 간선에는 포함 관계(CONNECTED_TO)가 저장되었다.

4. 결론

최근 수많은 소스코드가 오픈소스로 공개되면서 이를 활용하는 대부분의 사용자는 해당 오픈소스에 적용되어 있는 라이선스에 대해 정확하게 알 수 없다. 소스코드의 외부 참조, 함수 호출 등 다양한 기능이 실행된다는 특징으로 인해 직관적인 분석이 어려웠고, 이를 그래프로 시각화하여 그래프로 표현하는 방법이 고안되었다. 하지만, 해당 시점에만 그래프로 조회가 가능하다는 단점이 존재했고, 별도로 저장함으로써 지속적으로 조회가 가능한 방법이 요구되었다.

본 제안방식에서는 그래프로 표현된 소스코드를 그래프 데이터베이스에 맞는 파일 형식으로 변환하고 전처리 과정을 통해 효과적인 그래프 저장 및 표현이 가능한 방식을 제안한다.

이를 통해 직관적으로 분석이 어려운 소스코드 및 프로젝트 파일을 그래프로 시각화하고, 그래프로 변환된 데이터를 그래프 데이터베이스에 저장함으로써 지속적으로 소스코드 분석, 수정 및 변경에 편의성을 제공한다.

향후 연구로는 소스코드 간 관계를 표현하는 간선에 관련 함수 데이터를 저장하는 방법이 요구되며, 전체적인 그래프 데이터에 대해 기밀성을 보장할 수 있는 보안적인 조치가 필요할 것으로 사료된다.

ACKNOWLEDGEMENT

본 연구는 한국연구재단 4단계 두뇌 한국21사업(4단계 BK21사업) (과제번호:5199990914048)과 문화체육관광부 및 한국콘텐츠진흥원의 2023년도 SW저작권 생태계 조성 기술개발 사업으로 수행되었음 (과제명 : 클라우드 서비스 활용 구축 형태별 대규모 소프트웨어 라이선스 검증 기술개발, 과제번호 : RS-2023-00224818, 기여율: 50%)

참고문헌

[1] Himsolt, M., "GML: A portable graph file format", Technical report, Universitat Passau, 1997.

[2] “Code Graph - Visual Studio Marketplace”, https://marketplace.visualstudio.com/items?itemName=YaobinOuyang.CodeAtlas

[3] “Neo4j Documentation”, https://neo4j.com/docs/