코드 가시화의 서비스 모듈화

이진협[1*], 이근상[2**], 서채연[3*], 김영철[4*]
*홍익대학교 소프트웨어공학 연구실, **전북 TP
e-mail: { ljh[1], yi[2], chyun[3], bob[4] }@selab.hongik.ac.kr

Service Modulization of the Code Visualization

Jin-Hyub Lee[*], Keunsang Yi[**], Chae-Yun Seo[*], R. YoungChul Kim[*]
*Software Engineering Lab. Hongik University, **Jeonbuk TP

요 약

국내 대기업들은 충분한 SW테스팅으로 SW의 품질과 안정성을 점검하고 있다. 반면, 중소기업들은 부족한 인력과 비싼 상용 테스팅 도구 등으로 테스팅 환경이 어려운 실정이다. 이로 인한 테스트 부족 속에서 SW제품을 출시한다. 이 논문에서는 이런 문제의 해결방안 중 하나로 개발자가 코드 내부의 복잡도를 측정하여 잠재적인 오류를 줄이는데 초점을 둔다. 이를 위해 공개 소스프트웨어 기반의 도구 개선 제안 및 가시화 구현을 하였다. 즉, 벤처/중소 기업의 개발자들에게 각각 품질 요소들의 가시화 서비스가 가능하다. 이는 코드 내부의 결합력/응집력/복잡도/재사용 등의 가시적 모듈화로 SW품질 개선이 가능하다.

1. 서론

최근 소프트웨어 테스팅 시장은 연 평균 15%씩 글로벌 성장하고 있다[1]. 또한 세계 소프트웨어 개발 시장에서 테스팅이 차지하는 비중은 50%에 달한다[2]. 이는 소프트웨어의 규모와 복잡도가 높아짐으로 소프트웨어 제품의 안정성과 고품질화 요구의 증가이다. 이에 맞춰 많은 상용 테스팅 도구들이 나오고 있다.

국내 대기업들은 많은 인력과 자금으로 충분히 테스팅에 투자하는 반면에, 중소기업들은 부족한 인력, 시간으로 인해 테스팅에 대한 집중이 부족하다. 결국 제대로 된 테스팅 도구조차 보유하기 어렵다. 기존 연구들은 오픈소스 기반 도구에 결합도&응집도 등의 품질지표를 적용하여 코드 내부 복잡도를 가시화하는 연구 및 개발이 진행되고 있다[3,4,5]. 개발된 툴 체인들의 오픈소스 배포를 위해서는 필수적으로 쉽고 간편한 설치가 가능해야 한다.

이 논문은 기존 연구 및 개발된 프로그램들을 쉽고 간편하게 설치 및 사용하고자 한다. 이 논문은 2장에 서비스 개발에 이용한 오픈소스 기반의 도구들을 소개한다. 3장에서는 기존 서비스와 개선된 서비스 자동화의 설치 절차와 가시화 수행 순서에 대해 기술하고, 마지막으로 결론 및 향후 연구에 대해 언급한다.

2. 관련연구

개발된 프로그램들의 설치를 단일화 하기위해, Eclipse, JSmooth, HM NIS Edit, NSIS 프로그램들이 사용되며, 모두 오픈소스 도구들이다.

- Eclipse

Eclipse는 IBM에 의해 개발된 통합 개발 환경으로 가장 널리 사용되는 자바언어 개발 툴이다. 가시화를 위한 Toolchain은 Eclipse를 이용해 자바언어로 개발되었다. 또한 Eclipse에서 개발된 프로젝트를 배포하기 위해 패키지 파일 포맷인 JAR(자바 아카이브)로 추출한다.

- JSmooth

JSmooth는 Java JAR 파일을 EXE 파일에 래핑하기 위한 도구로, 설치된 JVM을 직접 찾을 수 있기 때문에, Java 배치를 더 부드럽고 사용하기 쉽게 만들어준다. 또한 사용가능한 VM이 없을 때는 래퍼가 자동으로 적합한 JVM을 다운로드, 설치 메시지를 표시하여 웹 사이트로 접속할 수 있다[6].

- HM NIS Edit

HM NIS Eidt은 Nullsoft Scriptable Install System (NSIS)를 위한 OpenSource 스크립트 편집기이다. 인터페이스가 많은 언어로 번역이 가능하고, 색상 및 텍스트 속성을 사용자가 정의 한다. 스크립트 작성 마법사를 지원해 순서대로 자세한 설명이 되고, 예시 정보가 있어 직관적으로 스크립트를 작성한다. 또한 설치에 있어서 세부 옵션들을 디자인 한다.

- NSIS

Nullsoft Scriptable Install System (NSIS)는 스크립트 기반으로 동작하는 윈도우용 설치 프로그램으로 오픈소스이며, 제한 없이 상업적 사용이 가능하다. 압축된 데이터

크기에 비해 오버헤드가 적으며, 설치 제거 파일도 생성할 수 있다. 그 외에도 DLL, ActiveX 컨트롤 등록 및 등록 취소, 바로 가기 파일 생성, 레지스트리 키 읽기, 설정, 삭제 등 많은 기능을 지원한다.

3. 기존 서비스와 개선된 서비스 자동화

3.1 자동화된 서비스 도구 설치

(그림 1) 기존 서비스와 개선된 서비스 설치 프로세스

- 기존 서비스

기존의 서비스는 그림 1의 (A)와 같이 Installer 실행, Source Navigator 설치, Doxygen 설치, GraphViz 설치, Eclipse 설치, 필요 시 Java 설치 순으로 설치된다[7].

우선 설치 파일을 실행시킨다. 실행 시 사용권 계약 인터페이스가 나오고 동의 버튼을 누르면 설치 위치 지정 인터페이스로 넘어간다. 설치 위치를 사용자가 원하는 곳으로 지정하면 해당 위치에 프로그램의 최상위 디렉터리가 생성되며, 내에 도구 별로 하위 디렉터리들이 생성된다. 그 하위 디렉터리에는 각각의 도구들의 설치 파일이 위치한다. 마지막으로 Eclipse 실행을 위해 Java가 설치 되어있지 않으면 설치하도록 유도한다. 이렇게 설치에 필요한 기본 디렉터리와 설치 파일들이 생성 되고나면 자동으로 순차적으로 각각의 설치 파일들을 실행하여 도구들을 설치한다.

- 개선된 서비스 자동화

기존 서비스의 설치 순서보다 조금 더 간략해진 자동화된 서비스의 설치 순서를 그림 1의 (B)에서 보여준다.

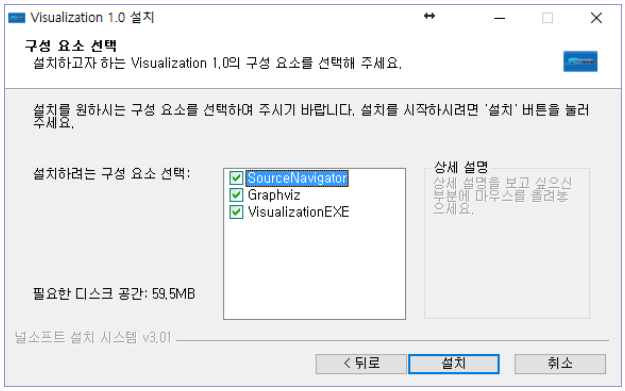
그림 1의 (B) 순서대로 설치를 자동화하기 위해 HM NIS Edit 프로그램의 스크립트 작성 마법사를 사용하여 전반적인 스크립트의 틀을 작성하고, 도구마다 섹션을 구분하고, 다음 버튼을 누르는 것만으로 가시화 도구 설치가 완료 될 수 있도록 구현하였다. 다음 그림 2는 스크립트 코드 일부로 설치할 프로그램들을 설정하는 부분이다.

```
Section "SourceNavigator" SEC01
  SetOutPath "$PROGRAMFILES\Visualization\"
  SetOverwrite ifnewer
  File "C:\Users\jinhy\Desktop\total\VisualizationDir.exe"
  ExecWait "$PROGRAMFILES\Visualization\VisualizationDir.exe"
SectionEnd

Section "Graphviz" SEC02
  SetOutPath "$PROGRAMFILES\Visualization\"
  File "C:\Users\jinhy\Desktop\total\graphviz-2.38.msi"
  ExecWait '"msiexec" /i "$PROGRAMFILES\Visualization\graphviz-2.38.msi"'
SectionEnd

Section "VisualizationEXE" SEC03
  SetOutPath "$PROGRAMFILES\Visualization\"
  File "C:\Users\jinhy\Desktop\total\Visualization.exe"
  CreateShortCut "$DESKTOP\Visualization.lnk" "$PROGRAMFILES\Visualization\
SectionEnd
```

(그림 2) 설치할 서비스를 설정하는 스크립트

SetOutPath는 설치 시에 실행될 각각의 파일들이 저장될 위치를 지정하는 부분이고, ExecWait은 저장된 파일들을 툴 설치 과정 중간에 바로 실행하여 함께 설치할 수 있도록 한다. 이 부분에서 exe 파일과 msi 파일을 설정하는 소스 코드는 조금 다르다. 마지막으로 CreateShortCut은 가시화 서비스 프로그램 실행 파일을 바탕화면에 바로가기로 생성하는 부분이다.

개선된 서비스에서는 기존 서비스와는 달리 Doxygen을 사용 안해 설치하지 않는다. 또한 Eclipse도 사용하지 않는다. 대신 VisualizationEXE 파일을 설치한다. 처음부터 순서대로 설명하면, 먼저 Installer를 실행 시킨다. 사용권 계약 인터페이스가 나오고 동의 버튼을 누르면, 그림 3과 같이 설치할 도구 목록을 보여준다.



(그림 3) 개선된 자동화 서비스 중 설치할 도구 목록

그림 3의 인터페이스에서 목록을 모두 체크한 상태로 설치 버튼을 누르면, 순서대로 설치가 된다. 다만 후에 서비스 사용 시 편의를 위해 SourceNavigator는 C:\Visualization 위치에 설치되도록 고정시켰다. VisualizationEXE는 Eclipse 프로젝트로 만들어진 서비스를 jar 파일로 추출하여 JSmooth를 이용하여 실행파일화 한 것이다. 따라서 기존 서비스에서 프로젝트를 Import하고 직접 Eclipse를 실행시킬 필요가 없어져 사용자의 편리성이 높아졌다. 따라서 사용자는 설치파일을 실행시킨 후 계속하서 다음 버튼만 누르면 자동으로 설치가 완료된다.

3.2 서비스 수행 절차

(그림 4) 기존 서비스와 개선된 서비스 수행 절차

- 기존 서비스

기존 서비스에서 소스코드를 가시화 할 때, 사용자 입장에서는 다소 귀찮고 번거로운 과정이 있다. 그림 4의 (A)는 기존 서비스를 이용해 가시화하는 수행 절차를 간략히 나타낸다.

▪ SoftwareVisualization Import

: SoftwareVisualization은 Eclipse 프로젝트 파일이다. 기존에 연구한 가시화를 위한 코드가 오픈되어 있다. 장점으로는 소스코드를 이해 가능하면, 사용자가 원하는 대로 수정해서 새로운 가시화를 할 수 있다. 하지만 소스코드를 이해하기 어려운 사용자라면 전혀 불필요한 과정이다.

▪ 소스코드 상에서 경로 수정

: Import 한 소스코드 내에서 그림 5와 같은 부분을 찾아 경로를 수정해 주어야 한다.

```
public void Extract() {
    String args[] = new String[6];
    args[0] = "E:/swvisualization/1.SN/SN-6.0/SN-6.0/bin";
    args[1] = "C:/Program Files (x86)/Graphviz2.24/bin";
    args[2] = "E:/swvisualization/Visualization/filelist.dat";
    args[3] = "E:/swvisualization/1.SN/SN-6.0/SN-6.0/bin/SNDB4";
    args[4] = "E:/swvisualization/1.SN/SN-6.0/SN-6.0/bin/SNDB4";
    args[5] = "Test";
```

(그림 5) 소스코드 내 경로 및 파일 이름 설정 부분

그림 5에서 args[0]은 소스네비게이터를 실행하는 파일인 snavigator.exe이 위치한 경로이고, args[1]은 그래프를 그리는 도구인 Graphviz의 실행 파일 dot.exe이 위치한 경로이다. args[2]는 분석할 타깃 소스코드의 경로 정보를 담고 있는 filelist.dat 파일의 경로이고, args[3]은 소스네비게이터의 출력물인 SNDB가 저장될 경로이다. args[4]는 또 다른 출력물인 *.proj 파일과 db 파일이 생성될 경로이다. args[5]는 SNDB와 proj 파일, db 파일 그리고 최종 결과물인 그래프의 이름을 설정하는 부분이다.

▪ Eclipse Run

: Import하고 소스코드 내의 경로를 수정한 프로젝트를 Eclipse에서 수행한다.

▪ 중간, 최종 결과물 저장

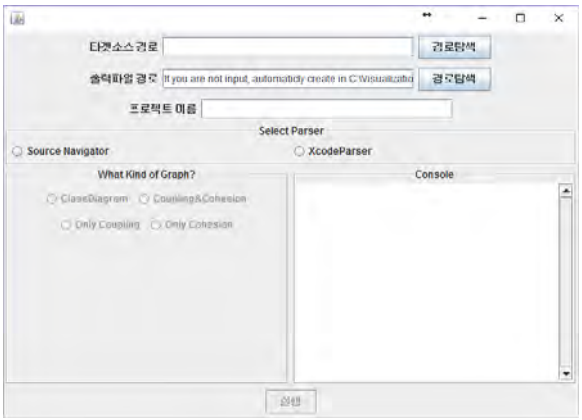
: 서비스 전체 관점으로 봤을 때, 기본적으로 소스네비게이터를 수행하여 얻는 중간 결과물을 하위 디렉터리들 중 SNDB4에 저장되게 된다. 서비스 수행 중에 생성되는 결과물들을 여기저기 저장하게 되면 차후에 필요한 데이터를 찾는데 헷갈리기 마련이다.

- 개선된 서비스 자동화

개선된 서비스 자동화 프로그램에서는 사용자들이 서비스를 실행시키고, 필요한 정보 입력 후 결과를 받는다. 그림 4의 (B)는 개선된 서비스 자동화 프로그램을 이용해 가시화하는 수행 절차를 간략히 나타낸 것이다.

▪ VisualizationEXE

: VisualizationEXE를 실행시키면 소스코드를 가시화하기 위해 필요한 정보들을 입력 및 실행 인터페이스에서 입력이 가능하도록 그림 6과 같이 구현하였다.



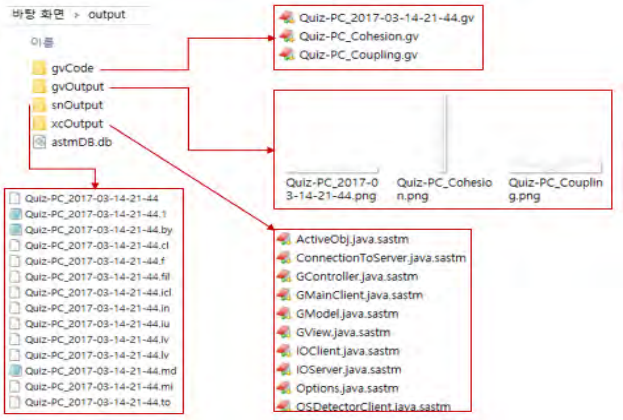
(그림 6) 서비스 자동화를 위한 UI

그림 6에서 타겟소스경로는 경로탐색으로 Java 파일 혹은 코드 최상위 디렉터리를 지정한다. 출력파일 경로는 사용자가 원하는 위치로 지정한다. 프로젝트 이름은 출력 파일의 이름을 입력할 수 있다. 그리고 파서가 기존 서비스에는 소스네비게이터만 사용하고 있었지만, 개선된 서비스에서는 소스네비게이터 외에 XcodeParser도 있다. 사용자는 원하는 파서를 선택하면, 활성화되는 라디오 버튼으로 된 그래프들 중 하나를 선택한 후 실행한다. 실행 버튼 또한 그래프가 선택되면 활성화된다. 원래 이클립스에서 프로젝트를 실행시켰을 때 콘솔 창에 출력되는 결과들을 Console 창에서 볼 수 있다.

▪ 중간, 최종 결과물 저장

만약 출력파일 경로를 바탕화면으로 지정하여 실행하면

바탕화면에 output 폴더가 생성되며, 폴더 내에 결과물들이 각각 snOutput, gvCode, gvOutput, XcOutput에 저장된다. 이는 사용자가 중간 결과물과 최종결과물 등을 쉽게 확인하고 관리할 수 있도록 한다. 그림 7은 오픈소스인 Quiz-PC를 타깃으로 입력하여 생성된 결과물들을 나타낸 것이다.



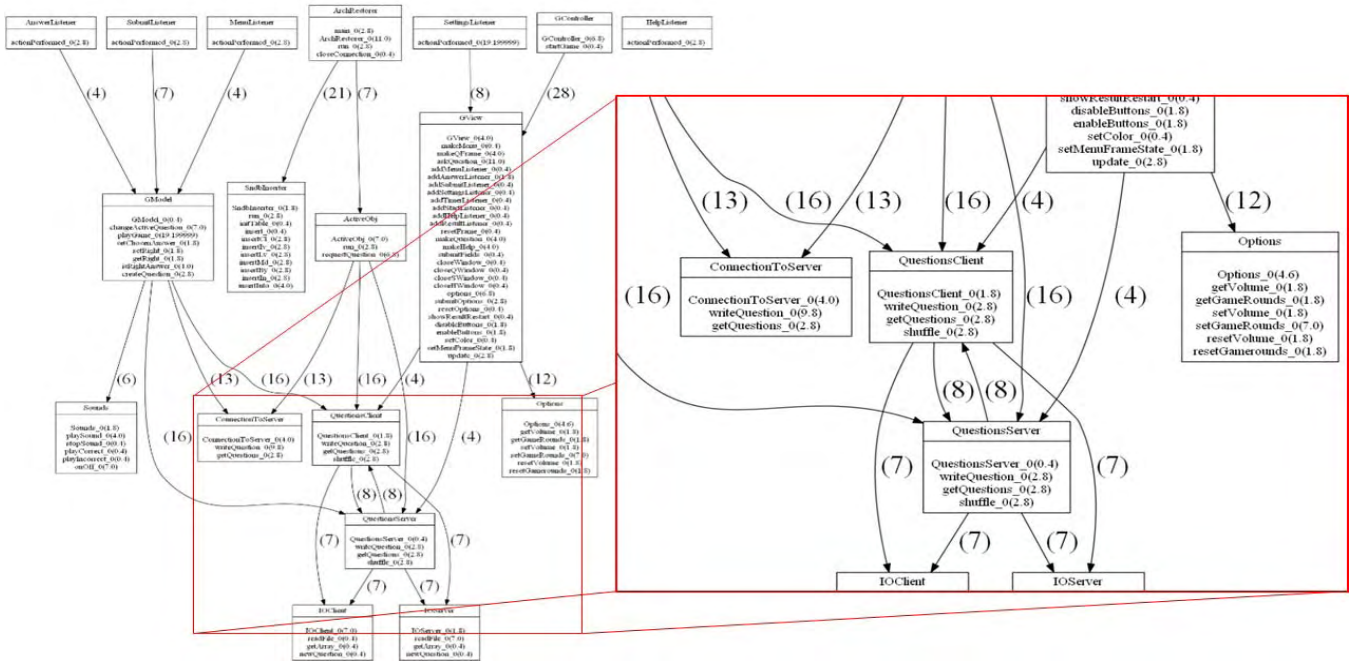
(그림 7) 결과물들이 저장된 output 폴더

4. 적용 사례

그림 8은 코드 가시화 서비스를 이용하여 오픈소스인 Quiz-PC 프로그램을 클래스 다이어그램 형태로 결합도와 응집도를 그려낸 사례이다. 클래스들 사이의 선의 숫자는 결합정도를 의미하며, 클래스 내 메소드 이름 옆에 있는 괄호 내의 숫자는 각 메소드들의 응집정도를 의미한다.

5. 결론 및 향후연구

이 논문에서는 기존 가시화 서비스에서 불필요한 것은 줄이고, 보완하여 가시화 서비스를 자동화 하였다. 소프트웨어 가시화 서비스를 자동화하기 위해 사용자가 별도의 기능 설명문서가 없어도 가시화 서비스를 설치하고 입력, 실행을 직관적으로 수행 할 수 있다. 또한 중간, 최종 결과물들을 한 디렉터리에 저장하여 결과물 확인과 관리를 편리하도록 하였다. 이 연구를 통해 앞으로 많은 사용자들이 오픈소스 기반의 가시화를 위한 툴체인을 사용하여 소프트웨어 내 소스코드를 분석할 때 쉽고 편리하다. 현재 서비스 자동화를 위한 설치 파일은 기존에 연구된 여러 가시화 방법들 중 일부만 적용되어 있다. 차후 다른 파서와 다른 그래프도 가시화할 수 있도록 적용할 예정이다.



(그림 8) 코드 가시화 서비스 적용 사례

참고문헌

[1] 이형근, “SW 호환성 검증 ‘테스팅 사업’ 부각”, 디지털타임스, 2016. 01. 29

[2] 유진상, “소프트웨어 테스팅 시장 뜨는데... 자동화 길은 멀어”, IT조선, 2016. 06. 09

[3] 강건희, 박보경, 장우성, 황준순, 권하은, 이한솔, 이현준, 김영철, “소프트웨어 성능 가시화를 위한 툴 체인 개발”, 소프트웨어공학회 학술대회 논문집, 제 18권, 제 1호, pp.395-398, 2016. 01

[4] 박지훈, 장우성, 이진협, 손현승, 김영철, “확장된 나쁜 코딩 습관 식별 구현”, 한국정보과학회, pp.401-403, 2017.02

[5] 변은영, 박보경, 장우성, 김영철, “객체 지향 패러다임에서의 코드 재사용을 위한 응집도 레벨 식별 모범 사례”, 한국정보처리학회, 제 23권, 제 2호, pp.475-478, 2016. 11

[6] JSmooth, http://jsmooth.sourceforge.net/

[7] 이한솔, “공개 Software 기반의 가시화 도구 개발”, 홍익대학교, 2017. 02