Journal of The Korea Society of Computer and Information
Vol. 23 No. 4, pp. 97-106, April 2018
www.ksci.re.kr
https://doi.org/10.9708/jksci.2018.23.04.097

A GUI-based Approach to Software Modularization

Dongmin Park*, Yeong-Seok Seo**

Abstract

Software maintenance activities have always been important issues in many domains of the software industry. In order to help to resolve this issue, software modularization approaches have been studied to build adequate modules with high cohesion and low coupling; such modular structures can help the comprehension and maintenance of complex systems. In this paper, we propose a GUI-based automated approach for software modularization based on GUI structure analysis. GUI is a principal manner to allow users to access the overall functionalities of a software system; in particular, GUI is closely related to software functionalities, which makes it a promising tool to identify and understand the entire software system. We also implement a software tool to support our approach and evaluate it with a case study using an open source software.

▸Keyword: GUI, Software, Maintenance, Modularization, Understandability

I. Introduction

소프트웨어 유지보수는 소프트웨어 개발 프로젝트에 있어 반드시 필수적이고 중요한 요소로서 성공적인 소프트웨어 일수록 더욱 중요하게 계획 및 수행되고 있다[1]. 소프트웨어 유지보수는 전체 소프트웨어 개발 주기에서 시간과 비용이 가장 많이 집중되는 단계이기 때문에 그 중요성이 매우 크고, 특히 4차 산업혁명에 따른 소프트웨어 분야에 대한 중요성이 더욱 강화되고 투자가 활발해짐에 따라 유지보수에 대한 비용절감과 효율적인 운영이 주요 이슈가 되고 있다.

개발된 소프트웨어는 배포(deploy)된 이후, 수정 유지보수(corrective maintenance), 적응 유지보수(adaptive maintenance), 완전 유지보수(perfective maintenance), 예방 유지보수(preventive maintenance) 등의 유지보수 활동을 거치게 된다[2]. 이러한 활동들을 통해 낮은 유지보수 서비스로 인한 전체 소프트웨어 서비스 가치가 하락하는 문제를 해결하고, 안정적이고 지속적인 비즈니스(Business)를 수행하기 위한 다양한 방법들이 모색된다.

그런데 개발된 이후 오래된 소프트웨어의 경우, 관련 내용에 대한 문서의 업데이트가 중단되어 최신 내용을 반영하지 못하고 있는 경우도 많고 관련 개발자의 이직 등으로 인한 문제 때문에 원활한 유지보수가 이루어지기 어려운 경우가 자주 발생한다. 즉, 실제 유지보수를 해나가기 위해서는 관련 문서나 해당 소프트웨어에 대한 설계자나 개발자의 도움 없이 배경지식이 없는 상황에서 코드를 분석해야 하는 경우들이 발생한다. 관련 지식 없이 소프트웨어 유지보수를 해야 하는 상황이 발생한다면 해당 소프트웨어를 이해하고 분석하기에는 상당히 많은 시간이 소비될 수밖에 없기 때문에 그 효율성이 급격히 저하된다. 따라서 유지보수 하고자 하는 소프트웨어를 보다 효율적이고 정확하게 이해하고 분석하기 위해 소프트웨어 분석 기법들이 다각도로 연구되고 있다. 기존 연구들의 경우 소프트웨어 구조, 소프트웨어 용어 등등에 따라 분석하게 되는데 이들은 직관적이지 못해 기존 배경지식이 없다면 실제 분석 후 이해하는데 상당한 시간이 소비된다. 분석 방식에 따라 관련 부분을 이해하기 위해 장시간이 소요될 수 있고 소프트웨어 특성별로 적절한 분석 방식이 다를 수도 있다.

본 연구에서는 소프트웨어 유지보수의 이해도와 분석의 효

• First Author: Dongmin Park, Corresponding Author: Yeong-Seok Seo
*Dongmin Park (athleticse@naver.com), Dept. of Computer Engineering, Yeungnam University
**Yeong-Seok Seo (ysseo@yu.ac.kr), Dept. of Computer Engineering, Yeungnam University
• Received: 2018. 02. 07, Revised: 2018. 03. 31, Accepted: 2018. 04. 11.
• This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2017R1C1B5018295).

율성을 높이기 위해 GUI 기반의 소프트웨어 모듈화 기법을 제안한다. 분석하고자 하는 소프트웨어에서 발생하는 GUI 이벤트들을 기반으로 연관 부분들을 분석 및 모듈화 하여 이해도를 높일 수 있도록 하였다 (이벤트는 프로그램에 의해 감지되고 처리될 수 있는 동작으로서 특정 기능을 수행하기 위한 키보드 입력, 마우스 클릭 등의 동작을 의미한다). 일반적으로 GUI는 소프트웨어에서 제공하는 전체적인 기능들과 밀접하게 연결되어 있고, 사용자들이 이들을 쉽게 접근하고 이용할 수 있도록 해줄 뿐만 아니라, 유지보수 단계에서도 이용가능하기 때문에 전체 소프트웨어 시스템을 효과적으로 식별하고 직관적으로 이해하기 위한 중요 매개체가 될 수 있다. 따라서 본 연구에서 제안한 기법은 시각적으로 익숙한 GUI를 활용하여 특정 이벤트에 대한 연결된 동작 결과를 직접적으로 파악 및 식별할 수 있고, 이와 동시에 이벤트들을 구분지어 각 모듈별로 분석할 수 있다. 또한, 실질적으로 소프트웨어 유지보수 단계에서 해당 소프트웨어의 최신 내용을 반영하고 있는 설계 문서 등이 존재하지 않는 등 관련 정보가 매우 부족할 경우 GUI를 이용하여 단시간에 즉각적으로 모듈화 하여 분석할 수 있다.

따라서 본 연구에서 제안한 GUI기반의 소프트웨어 모듈화 기법을 기반으로 유지보수 하고자 하는 소프트웨어 이해도 향상을 통해 소프트웨어 유지보수의 효율성 향상에 기여할 수 있다. 일반적으로 소프트웨어 유지보수는 소프트웨어 이해(software comprehension)로부터 시작된다. 특정 소프트웨어에 대해 유지보수 수행 시, 해당 소프트웨어에 익숙한 개발자가 이를 수행하고 있다면 전체적인 이해의 어려움이 적을 수 있으나 개발한지 오랜 시간이 흘렀거나 혹은 개발자와 유지보수자가 다른 경우에는 이해의 어려움이 발생한다. 이 경우, 현재 문제가 되는 기능이 무엇인지 식별된 상황에서 해당 문제 및 기능과 연결된 모듈이 어떠한 것들인지 파악하는 과정은 실질적으로 유지보수자들의 이해도를 높여 작업 효율성 향상에 큰 도움이 된다. 특정 기능과 관련된 문제가 발생한 경우, 해당 기능과 관련된 진입점(entry point)이 GUI에 존재하므로 GUI를 중심으로 제시된 모듈의 정보는 해당 기능과 연관된 클래스를 파악하는데 매우 유용하게 사용될 수 있다.

본 논문의 구성을 다음과 같다. 2장에서는 관련 연구에 대해 소개하고, 3장에서는 본 논문에서 제안한 기법에 대해 단계적으로 상세하게 기술한다. 4장에서는 제안된 기법에 대해 오픈소스 소프트웨어를 활용하여 실험을 수행하고 결과를 분석한다. 마지막 5장에서는 본 연구의 결론 및 향후 연구에 대해 논의한다.

II. Preliminaries

1. Related work

소프트웨어 모듈화를 위해서는 일반적으로 설계 문서를 통한 모듈간의 기능적 연관성을 유추할 수 있다(Design-based approach). 그러나 이는 완전히 관리된 설계 문서가 존재할 경우를 가정하고 있다. 현실적으로 이러한 문서들이 존재하지 않을 경우 소스코드로부터 모듈들 간의 관계를 파악하고자 하는 다양한 기법들이 연구되었는데 이들은 크게 구조적 기반 접근법(Structure-based Approaches)과 용어 기반 접근법(Term-based Approaches)으로 분류되어질 수 있다. 각각의 접근법 내에서도 세부 기준과 모듈화를 위한 관점에 따라 구체적인 기법들이 연구되고 있다.

구조적 기반 접근법은 소프트웨어 구조 자체를 분석하여 모듈화를 수행하고자 하는 기법이다. 소프트웨어를 구성하고 있는 각각의 요소들(예 : 파일, 클래스 등)과 그 관계(예 : 함수 호출, 상속 관계 등), 그리고 관계들 사이의 방향성 등을 분석하여 모듈화를 수행한다. 현대의 소프트웨어 시스템은 기본적으로 적절한 구조의 추상화가 필요하므로 이러한 구조적 기반 접근법을 활용하기 용이할 수 있다. 본 접근법은 소프트웨어 유지보수 시 전체 소프트웨어의 구조가 소스코드로부터 쉽게 분석되어질 수 있다는 장점과 함께 구조적인 속성(e.g., 결합도(cohesion), 응집도(coupling) 등)이 나타나있지 않거나 고려되고 있지 않다면 효과적이지 못할 수 있다. 또한 소프트웨어와 이의 동작 측면(behavioral aspect)에서 복잡도가 증가한다면 효율성이 매우 떨어질 수 있다[3-8].

용어 기반 접근법에서는 소스코드에 사용된 언어정보를 추출하여 핵심적인 용어들(예 : 식별자, 이름, 주석 등)을 분석하고 이들의 의미를 판단하여 비슷한 정보를 공유하고 있는 요소들끼리 모듈화를 수행하고자 하는 기법이다. 구조적 기반 접근법에 비해 보다 이해하기 쉬운 언어적인 의미 정보를 고려하기 때문에 분석하고자 하는 소프트웨어 개발자의 의도를 보다 정확하게 파악하여 모듈화 할 수 있다. 특히, 주석에는 소프트웨어 개발자의 의도와 구체적인 설명이 포함되어 있기 때문에 이를 이용해 모듈들 사이의 숨겨진 구조나 관계까지도 식별해낼 수 있다. 그러나 분석하고자 하는 소프트웨어에서 사용되고 있는 용어들이 소프트웨어의 전체 구조나 관계 및 의미 등을 함축하지 않을 경우 정확한 모듈화에 상당한 어려움을 겪을 수 있다[9-15]. 따라서 소프트웨어 배포 후 운영 및 유지보수 기간 동안 개발 환경이 변화하더라도 매우 신중하게 용어들을 계속 관리해 나가야 한다.

본 연구에서 제안한 기법은 기존 제안된 접근법들과는 달리 사용자에게 친숙하고 이해도가 높은 GUI를 바탕으로 소프트웨어 구조 변경 없이 보다 직관적이고 효율적으로 모듈화 할 수 있는 새로운 관점의 분석 기법을 제안한다. 각 GUI에 대해 실제 링크되어 활용되고 있는 모든 연관된 클래스들을 하나씩 분석하면서 모듈화를 한다면 쉬운 이해와 함께 유지보수를 필요로 하는 특정 GUI 부분에 대한 분석 데이터를 즉시 제공해 줄 수 있다.

기존 연구 기법들과 본 연구에서 제안한 기법을 비교하면 Table 1과 같다. Table 1의 첫 번째 열은 기법들끼리의 비교 요소들을 기술하였고, 두 번째 열에서 마지막 열에는 기존 기법들과 GUI는 본 연구에서 제안한 GUI기반의 기법을 나열하였다.

소프트웨어 구조 변경 없이 모듈화가 진행되기 때문에 본 연구에

Table 1. Comparison of the proposed approach with the existing approaches

	Design-based approach	Structure-based approach	Term-based approach	GUI-based approach
Characteristic for modularization	Modularization by inferring functional relationships between modules through the analysis of design documents	Modularization by inferring functional relationships through structural dependency between modules	Modularization by inferring functional relationships through the similarity of semantic terms in modules	Modularization by inferring functional relationships through the relationships between GUI and module
Well-managed design documents	required	not required	not required	not required
Well-managed terms in source codes	not required	not required	required (continuous management)	not required
A Level of understandable for "functional relationships" highlighted in modularization	Developer level	Developer level	Developer level	User level (easily understandable through GUI)
The best performance condition	In case that there is perfectly managed design documents which is completely reflected the source code in the maintenance stage	In case that there is a perfect presentation of structural characteristics which is used to infer functional relationships	In case that there is a perfect presentation of terms in class names, function names etc within predetermined vocabularies, as an aspect of functional relationships	In case that there is a perfect presentation of functional relationships in GUI
Limitation	It is difficult to manage and update consistently design documents according to changes across the whole software development lifecycle	It is difficult to identify the functional relationships that is not expressed by structural characteristics	It is difficult to apply the software with terms that is not consistently managed across the whole software development lifecycle	It is difficult to apply the softwares which are not based on GUI and to identify hidden functions which are not related to GUI

서 제안한 기법은 이벤트에 대한 cross-reference 리스트를 사용하는 것과 비교할 수 있다. cross-reference 리스트의 경우, 잘 관리된 설계 문서 및 설계 문서-코드 간의 연결정보가 필요하다. 그러나 이러한 정보는 유지보수 단계까지 완전한 형태로 문서화되어 잘 유지되기가 쉽지 않다. 따라서 유지보수 단계에서의 모듈화(modularization) 연구들에서는 일반적으로 이러한 설계 문서와 cross-reference 리스트가 없는 경우를 가정하고 있다. 본 연구에서도 기존 구조적 기반 접근법 및 용어 기반 접근법처럼 유지보수 시 완전한 형태의 설계문서가 없는 상태에서 소프트웨어의 소스코드(source code)가 가장 신뢰성 있는 정보(source of information)이고 이를 활용해 모듈화 한다는 상황을 함께 따르고 있다.

또한, 역공학(reverse engineering)을 통한 설계 복구(design recovery)로 소스코드의 디자인 측면을 추출하여 활용하는 방식과도 비교할 수 있다. 역공학을 통해 설계 복구하는 방법과 본 연구에서 제안한 기법들을 포함하여 소스코드를 기반으로 모듈화하는 기존 모든 기법들은 소스코드를 활용하고 설계의 관점을 추출한다는 점에서 동일하지만, 모듈화를 위한 응집성(cohesiveness)을 어떠한 방식으로 정의하는지 그리고 그에 따른 가정에 따라서 모듈화를 위해 효과적으로 적용할 수 있는 기법이 다르다. 구조적 기반 접근법은 소스코드에서 주로 모듈 간의 의존 관계를 추출하여 사용하는데, 이러한 관계 정보를 활용하여 구조적으로 의존 관계가 긴밀한 모듈들을 같은 기능에 기여한다(contribution)고 가정한다. 용어 기반 접근법은 소스 코드에서 식별자(identifier)나 주석(comment)에 존재하는 용어를 추출하여 사용하는데, 용어 중에 각 모듈의 기능적 특성을 나타내는 용어들이 있다고 가정하고 해당 용어들을 기준으로 같은 기능에 기여하는 모듈을 파악한다. 이런 관점에서 GUI 기반 접근법은 GUI에 유지보수자가 관심을 두는 기능 정보가 포함되어 있고 특정 GUI에 기여하는 모듈들이 실질적으로 같은 기능에 기여한다고 가정하고 모듈화를 진행한다.

모듈화 이후 소프트웨어 유지보수를 위해 소프트웨어를 실제 이해하는 과정에서도 차이가 발생한다. 구조적 기반 접근법은 모듈화 결과로서, 연관성 있는 모듈들의 집합(set)과 그들의 구조적 의존 관계만 주어진다. 반면 용어 기반 접근법은 모듈화의 중심이 된 용어들이 주어지기 때문에 용어들을 읽고 의미를 파악할 수 있어 구조적 기반 접근법보다 모듈화 결과에 대한 이해도가 높을 수 있다. GUI 기반 접근법은 연관된 GUI에 연관된 요소들이 함께 주어지며 사용자 이해 가능한 수준에서 모듈화 결과가 제시되므로 유지보수자의 보다 쉬운 이해에 도움이 될 수 있다.

III. The Proposed Scheme

본 논문에서 제안하고 있는 GUI 기반의 모듈화 기법은 다음 Fig. 1. 에 전체적으로 도식화하였다. Step 1.에서는 유지보수 하고자 하는 소프트웨어 대하여 분석에 필요한 4가지 데이터를 수집한다. 이를 위해 어떤 종류의 GUI 이벤트(GUI event)들이 활용되고 있는지 우선 식별한다. 수집된 GUI 이벤트 데이터를 통해 클래스 이름(Class name), 이벤트 메소드(Event method), 부모-자식 메소드 관계(Parent-Child method relationship) 데이터를 함께 수집한다. Step 2. 에서는 이전 단계에서 확보한 정보를 기반으로 GUI 이벤트들의 연관관계를 분석한다. GUI의 경우 계층구조를 이루는 형태로 구성되기 때문에 연관관계 분석 결과는 계층구조 표현에 특화되어 있고 사용자가 이해하기 쉬운 트리 형태로 나타낸

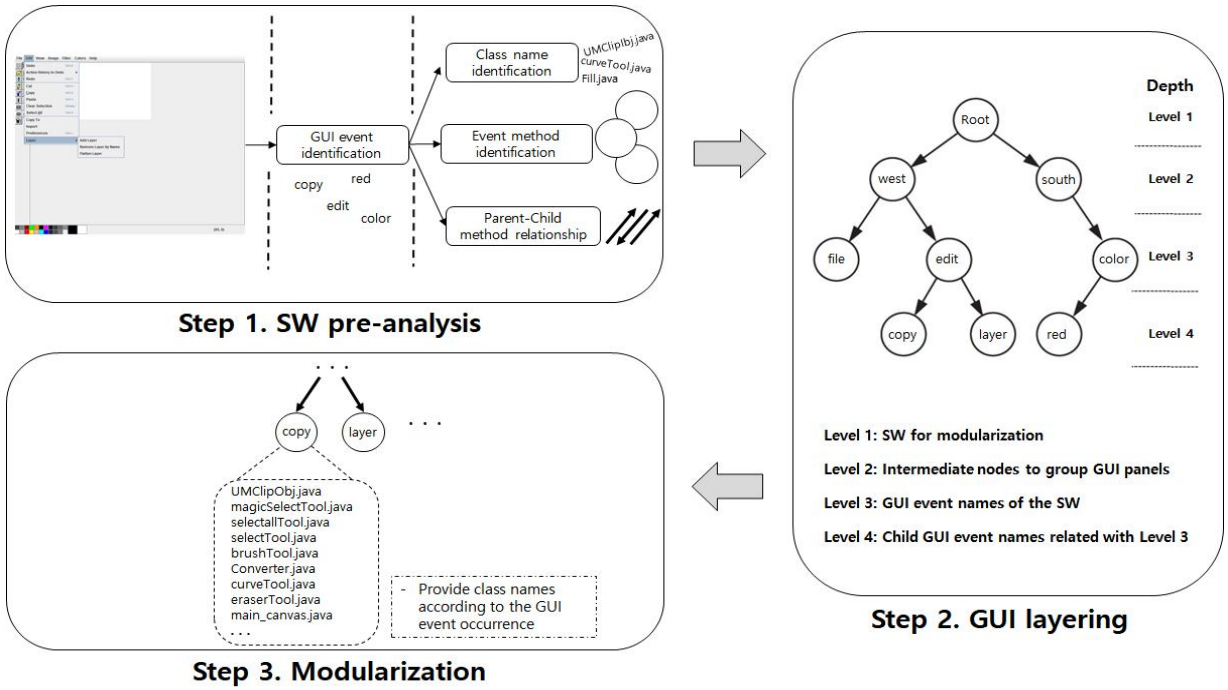


Fig. 1. Overall approach

다. Step 3. 에서는 이전 단계에서의 GUI 연관구조를 기반으로 특정 GUI 이벤트 발생할 경우 관련된 모든 클래스들을 식별하고 결집하여 모듈화를 수행한다. 다음 절부터는 각각의 Step에 대해 보다 구체적으로 소개한다.

1. SW pre-analysis

Step 1. 에서는 분석하고자 하는 소프트웨어에 대해 GUI 이벤트 기반으로 사전 분석하여 모듈화에 필요한 기본 데이터들을 식별하고 수집하는 단계이다.

분석하려는 소프트웨어의 모듈화를 위해 GUI 이벤트(GUI event), 클래스 이름(Class name), 이벤트 메소드(Event method), 그리고 부모-자식 메소드 관계(Parent-Child method relationship)의 4가지 데이터를 수집한다. GUI 이벤트는 분석하고자 하는 소프트웨어의 모든 GUI 이벤트 이름을 식별한 데이터이다. 클래스 이름은 분석하고자 하는 소프트웨어를 구성하고 있는 클래스들의 이름을 식별한 데이터이다. 이벤트 메소드는 특정 GUI 이벤트가 발생했을 때 실제 실행되는 이벤트 처리 메소드를 의미한다. 특정 GUI 이벤트 발생 시 연관적으로 실행되는 모든 메소드 뿐만 아니라 각각의 메소드 내부에서 다시 추가로 수행될 수 있는 메소드들을 수집한다. 또한, 특정 GUI 이벤트가 발생했을 때 어떠한 클래스들이 실행되었는지 파악할 수 있는 용도로 활용된다. 마지막으로 부모-자식 메소드 관계는 특정 GUI 이벤트 발생 시 앞서 수집한 모든 메소드들 사이의 부모-자식 관계를 식별한다. 이는 이벤트에 이벤트를 추가하는 메소드를 검색하여 구분하였다. 이러한 이벤트 데이터의 경우는 해당 소프트웨어에서 사용되고 있는 GUI들 사이의 계층 구조를 구성하기 위한 기초 데이터로 활용된다. GUI 이벤트들끼리 어떠한 연관관계를 가지고 있는지 분석하여 이벤트들끼리의 상-하위 관계를 분명하게 식별할 수 있도록 하였다.

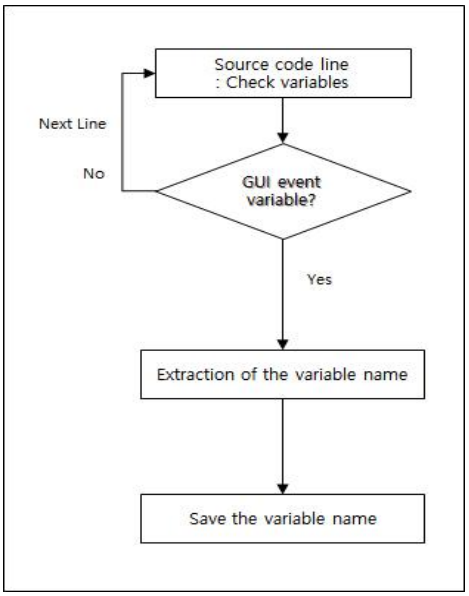


Fig. 2. Identification of the GUI event name

GUI 이벤트를 수집하기 위해서는 Fig. 2.에서처럼 GUI 이벤트 변수들을 검색하여 GUI 이벤트 이름을 쉽게 판단할 수 있도록 하였다. 변수 이름을 살펴보면 오픈소스 소프트웨어의 특성상 실제 활용되지 않은 변수들도 많이 존재하기 때문에 이를 식별하여 제외하고 GUI 이벤트 변수가 아닌 일반 변수들도 많이 존재하기 때문에 이 또한 식별하여 제외한다.

이벤트 메소드의 경우 실제 GUI 이벤트에 대한 명령을 처리하는 메소드를 검색하여 식별하였다. 다음 Fig. 3.과 같이 전체 소프트웨

어에서 특정 GUI 이벤트 발생 시 실행될 수 있는 이벤트 메소드를 식별하고 식별된 메소드 내부에서 다시 연관적으로 수행될 수 있는 메소드들을 계속 찾아 들어가며 모든 관련 메소드들을 추출하였다.

마지막으로 부모-자식 메소드 관계에서는 수집된 이벤트 메소드들 간의 부모-자식 관계를 판단하기 위해 이벤트끼리 연관관계를 가지도록 하는 소스코드 명령어들을 식별하였다. Fig. 4.에서처럼 이벤트들 간의 연관관계를 나타내는 부분을 추출하여 이를 통해 그들 간의 관계를 확립하였다. 예를 들어, A.add(B)와 같은 부분이 있다면 왼쪽 이벤트인 A에서 오른쪽 이벤트 B를 추가하는 것이므로 A가 부모, B가 자식인 관계로 식별한다.

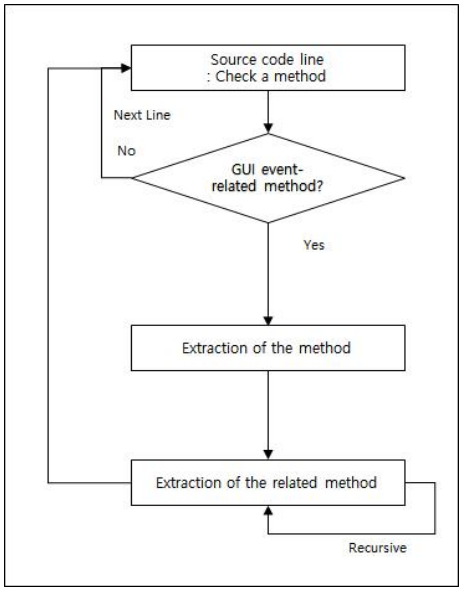


Fig. 3. Identification of the GUI event method

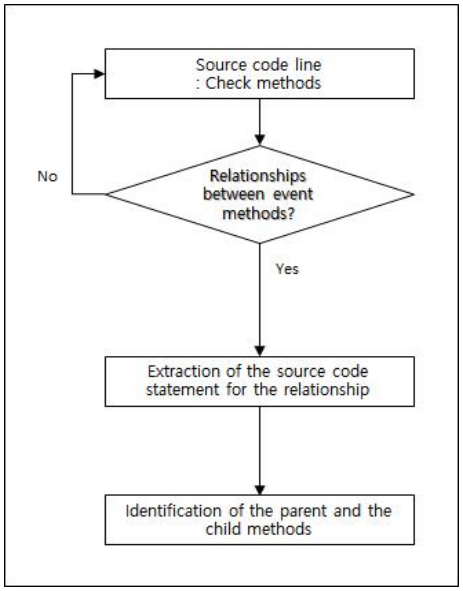


Fig. 4. Identification of the Parent-Child method relationship

2. GUI layering

Step 2. 에서는 Step 1. 에서 식별된 4가지 형태의 정보를 활용하여 GUI의 계층들을 구조화하여 추출한다. Fig. 5.에 나타나 있는 것처럼 GUI 이벤트들을 트리 형태로 표현하여 사용자가 보다 쉽게 인지하고 식별할 수 있도록 한다. 트리에서 각각의 노드는 각각의 GUI 이벤트와 이와 함께 수행되어질 수 있는 연관된 GUI 이벤트들을 나타낸다. 분석하고자 하는 소프트웨어의 GUI에서 제공하는 기능들을 그대로 식별하고 분석하였기 때문에 이질감 없이 이해해 나갈 수 있다.

GUI 계층 구조 추출을 위해서는 분석하고자 하는 소프트웨어를 Root로 우선 지정한다. 이전단계에서 수집된 정보에서 GUI 이벤트와 이벤트들 간의 부모-자식 관계 데이터를 식별하였기 때문에 이들을 통해 트리를 구성하기 위한 모든 이벤트들 간의 깊이(Depth)를 구해낼 수 있다. 이렇게 구한 깊이 데이터를 이용해 트리의 상위 노드부터부터 차곡차곡 트리형태로 구조화해나간다. 다음 단계에서 실질적인 모듈화를 진행하기 위해 각 GUI 이벤트별로 이벤트 메소드들을 미리 저장하고 있도록 구성하였다.

Fig. 5.에서 깊이 수준(Depth Level) 1은 분석하고자 하는 소프트웨어(Root)를 나타내고 깊이 수준 2에는 Root 노드에서 가장 첫 번째로 하위 레이어로 위치하는 GUI panel들을 표현한다. 깊이 수준 3부터는 각 GUI panel별로 상위 레이어에 위치한 GUI 이벤트들을 단계적으로 나타낸다. Fig. 6.에 표현하였듯이 부모-자식 관계를 식별한 데이터를 이용하여 어떠한 GUI 이벤트 노드에 어떠한 GUI 노드를 하위에 추가하고 할당할지 설정한다.

3. Modularization

Step 3에서는 모듈화 된 GUI 이벤트들에 대하여 각 이벤트를 실제 수행하는데 필수적으로 연관되어 있는 클래스들을 추출하여 실질적인 모듈화를 수행한다. Fig. 7.에서 구체적으로 표현한 것처럼, 각각의 GUI 이벤트 실행 시 직접적으로 실행되는 클래스들을 모두 식별하고 추출하여 연속적으로 보여줄 수 있도록 한다. GUI 이벤트 발생 시 실행되는 클래스를 추출한 이후 서로 GUI layering 구조를 바탕으로 추가적으로 호출하는 클래스들을 추출하여 모든 연관된 클래스들을 모듈화 하여 보여줄 수 있도록 하였다.

GUI layering한 트리 구조에서 GUI 이벤트들로 노드가 구성되어 있는데 분석하고 싶은 노드를 선택하면 연관된 노드들과 이를 수행하기 위한 클래스들을 GUI 이벤트단위로 모듈화하여 추출해 올 수 있도록 하였다. 즉, 실제 분석하고자 하는 소프트웨어에서 특정 GUI 이벤트를 클릭할 경우 해당 이벤트를 수행하는데 직접적으로 사용되는 클래스들을 모두 식별하고 모듈화 하여 제공하게 된다. 특정 GUI 이벤트 실행 시 연관된 모든 클래스들을 모듈화 하는 전체 과정을 Fig. 8.에 나타내었다. 트리화 되어 있는 노드들 중 분석하고자 하는 노드(특정 GUI 이벤트) 선택 시 소프트웨어의 전체 클래스 집합에서 해당 GUI

이벤트 이름을 가진 이벤트 메소드가 있는 클래스를 찾아낸다. 그리고 이벤트 메소드 내부에 또 다른 메소드가 호출되는지 체크하여 해당 메소드를 가진 클래스들을 마찬가지 방법으로 재귀를 통해 모두 검색해낸다. 이를 통해 특정 GUI 이벤트에 대한 클래스들을 식별하여 모듈화 하여 제공한다.

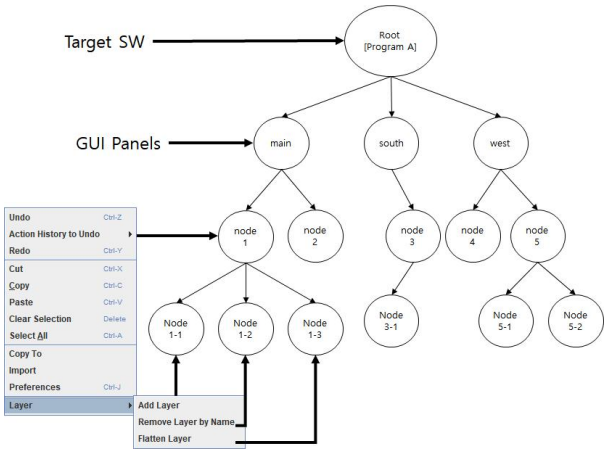


Fig. 5. An example of modularization

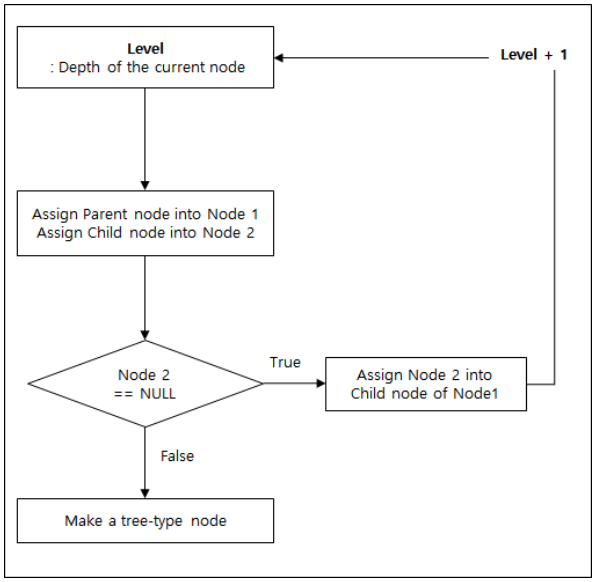


Fig. 6. An algorithm for tree-type GUI layering

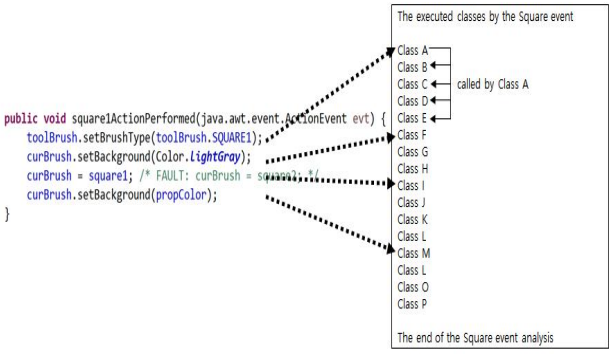


Fig. 7. Class identification for modularization

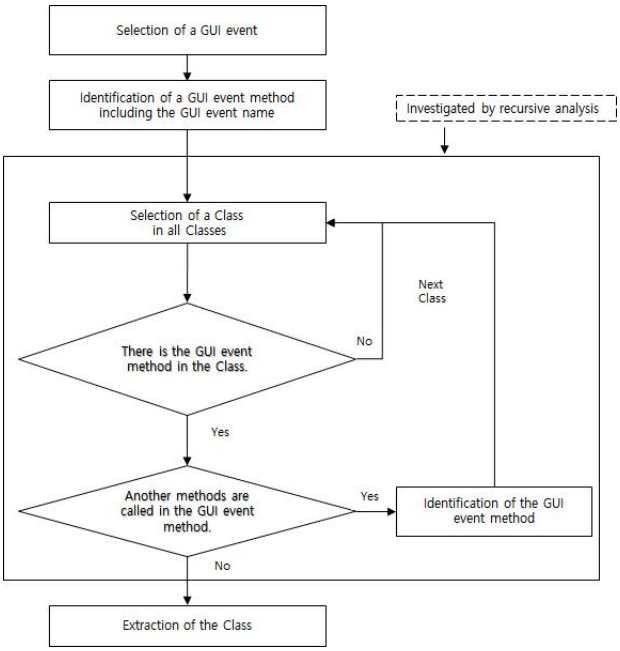


Fig. 8. GUI-based modularization

IV. Experiment

1. Experimental design

본 연구에서 제안한 기법의 실험을 위해 오픈소스 소프트웨어 중 하나인 TerpPaint를 활용하였다[16]. TerpPaint는 University of Maryland에서 개발한 TerpOffice 제품군의 일부로서 Java Swing 기반의 대표적인 그림 작성 편집 도구이다. 이러한 TerpPaint는 윈도우 운영체제에서 기본적으로 제공되는 "그림판"과 유사한 프로그램이다. GUI 버튼 이벤트의 수가 많고 오픈소소로서 관리되고 있는 도구이기 때문에 많은 연구들에서 분석을 위해 활용되고 있다. 본 연구에서 활용한 TerpPaint의 버전은 4.0이고 구성하고 있는 전체 클래스 개수는 57개 이다. Fig 9. 는 TerpPaint를 실행했을 시 초기 화면을 보여준다. TerpPaint 초기 화면을 살펴보면 상위메뉴 위치한 File, Edit, View, Image, Filter, Colors, Help들과 좌측메뉴에 위치한 그림 버튼들, 그리고 하단메뉴에 위치한 색상들이 모두 GUI 이벤트들을 나타낸다.

본 연구에서는 제안한 기법의 효과적인 적용을 위해 기법을 자동화 할 수 있는 도구를 직접 개발하여 실험을 수행하였다.

2. Experimental results and discussion

본 연구에서 제안한 모듈화 기법을 TerpPaint에 적용해보고 그 결과에 대해 분석한다. Fig. 9.에 나타나있듯이 TerpPaint에서 지원하는 그림판 기능들은 크게 세 부분으로 분류할 수 있다. 상위 메뉴에 존재하는 이벤트들(File, Edit, Image 등)은 파일을 불러오거나 옵션을 수정하는 등 여러 가지 부가적인 간접

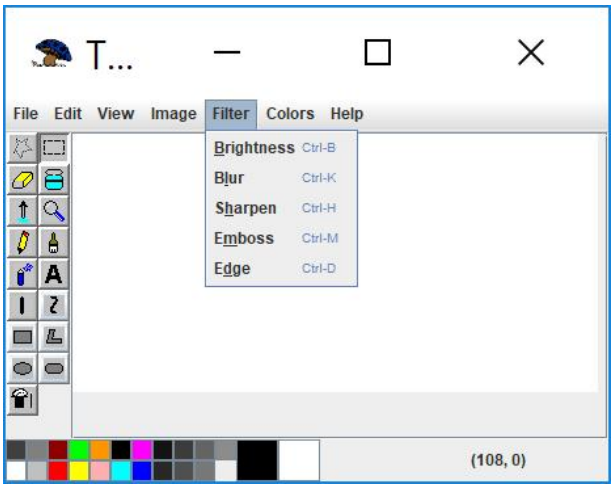


Fig. 9. A snapshot of TerpPaint

기능을 제공한다. 좌측 메뉴에 존재하는 이벤트들은 가운데 흰색상의 메인화면에 직접적으로 그림을 그릴 수 있는 기능을 지원하고 있다. 하단부에 위치한 이벤트들은 그림 작업을 수행 시 선이나 면을 위한 다양한 색상을 선택할 수 있게 해주고 그림상의 좌표정보를 제공한다.

Fig. 10.은 본 연구에서 개발한 도구에 TerpPaint를 적용하여 도출한 결과를 보여준다. 결과화면의 좌측부분은 GUI layering을 통해 모든 GUI 이벤트를 보기 쉽게 계층구조로 표현하였다. 이러한 계층구조를 통해 GUI 이벤트들끼리의 관계를 상위레벨에서 모든 하위레벨까지 직접적으로 파악할 수 있다. 결과화면의 우측부분은 특정 GUI 이벤트를 수행 시 해당 이벤트를 위해 실제 수행되는 클래스들을 모듈화 하여 모두 보여준다. 출력되는 클래스의 순서는 해당 이벤트 수행 시 실제 실행되는 순서이다. 즉, 결과화면의 좌측부분에서 각각의 GUI 이벤트를 선택해보면 하위 GUI 이벤트들을 다시 선택할 수 있게 되고 최하위레벨까지 선택할 수 있으며 관련되는 클래스들이 즉각적으로 결과화면의 우측부분에 나타난다. 이를 통해 특정 기능의 추가 작업을 위해 반드시 분석해야할 클래스들을 쉽게 확인하고 파악할 수 있도록 한다.

Fig. 10.은 사용자가 TerpPaint의 여러 기능들 중 상위 메뉴에 위치한 [File]-[New] 메뉴를 선택하여 관련 클래스들을 도출한 결과이다. [New] 이벤트는 TerpPaint의 가운데 화면에 흰색 배경을 제공하여 그 위에 새로운 그림을 그릴 수 있도록 하는 기능이다. [New] 이벤트를 분석한 결과 총 39개의 클래스가 이용됨을 알 수 있고, 실제 클래스 명칭은 Fig. 10.의 결과화면 우측부분에 상세히 나타나 있다.

Fig. 11.는 TerpPaint의 여러 기능들 중 좌측 메뉴에 위치한 [Brush] 메뉴를 선택하여 분석한 결과이다. TerpPaint상에서 [Brush]의 경우 글씨로 표현되어 있지 않고 그림 아이콘으로 표현되어 있지만, 내부적으로 해당 그림 아이콘에 Brush라는 이름으로 구현되어 있어 도구에서도 이와 같은 이름으로 나타내었다. [Brush] 이벤트는 그림 화면에 색을 칠하는 기능이다.

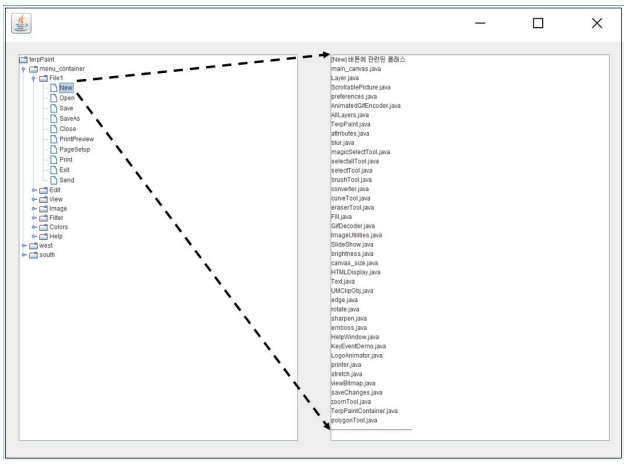


Fig. 10. Experimental results ([New] event)

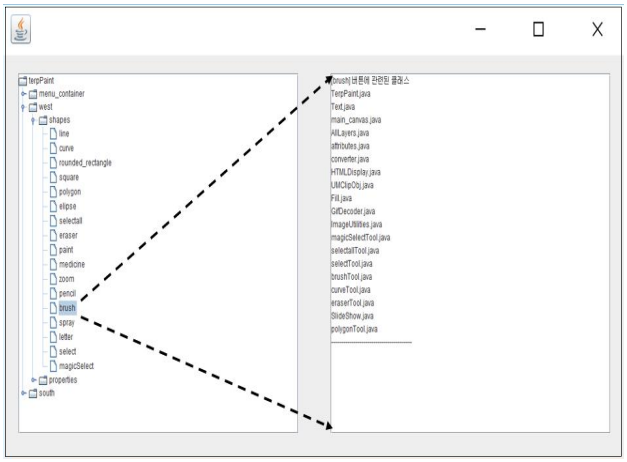


Fig. 11. Experimental results ([Brush] event)

[Brush] 이벤트를 분석한 결과 총 19개의 클래스가 사용됨을 알 수 있고, 실제 사용된 클래스 명칭은 Fig. 11.의 결과화면 우측부분에 표현되어 있다.

TerpPaint의 경우 그림 소프트웨어의 특성상 모든 GUI 이벤트들이 여러 클래스를 활용하지 않는 경우도 존재한다. 예를 들어 검정색 연필을 사용하여 그림을 그리려고 할 때 검정색 외에 다른 색상은 실행이 되지 않도록 설정되어 있다. 즉, 한 가지 색상만을 선택할 수 있고 동시에 2가지 이상의 색상을 선택할 수 없도록 되어있다. 이 경우 내부적으로 검정색이 True값이라면 나머지 색상은 False로 할당된다. 이처럼 GUI 이벤트가 단순히 Boolean기능만을 지원하는 경우도 존재한다. 또한, TerpPaint의 경우 오픈소스 소프트웨어의 특성상 추후 확장을 위해 형태만 있고 실제 기능이 아직 구현되지 않은 버튼들이 존재한다. 색상 선택이나 업데이트 중인 버튼의 경우 다른 연관된 클래스가 존재하지 않기 때문에 Fig. 12. 에서처럼 “단순 boolean 버튼 또는 업데이트 중인 버튼은 검색할 수 없습니다.”라는 메시지가 표시될 수 있도록 하였다.

Fig. 12. An example of the boolean type in the GUI event

3. Detailed analysis and discussion

본 논문에서 제안한 기법을 통해 실제 분석하고자 하는 소프트웨어의 모듈화가 실제로 충실히 이루어졌는지 세부적인 검증을 진행하였다. 검증을 위해서는 TerpPaint의 기능들 중 “line” 기능에 대하여 해당 기능을 수행하기 위해 필요로 하는 클래스들과 이들의 호출관계들을 하나씩 모두 식별해가며 실제 TerpPaint와 같은 형태로 분류되고 구성되어 있는지 분석하였다.

“line” 은 화면에 직선을 그릴 수 있는 기능이다. 매뉴얼하게 분석한 결과 Fig. 13.에서 나타나있듯이 이를 구성하고 있는 클래스는 총 8개로서 각각 TerpPaint.java, selectTool.java, magicSelectTool.java, selectallTool.java, polygonTool.java, converter.java, curveTool.java, main_canvas.java를 사용하고 있었다. 우선 TerpPaint.java에서 line이벤트가 발생하면 line의 기능을 수행하기 이전에 공통적인 기능을 가진 클래스를 종료하기 위하여 selectTool.java, agicSelectTool.java, selectallTool, polygonTool.java 4가지 클래스를 실행한다. 4가지 클래스 중 해당 도구의 기능이 실행되어 있으면 종료하기 위해서 실행되며 line의 기능이 올바르게 수행되도록 돕는 클래스들이다. 4가지 클래스가 실행이 되고 나면 실제 line 기능을 직접적으로 수행할 수 있도록 하는 converter.java, curveTool.java, main_canvas.java 3가지의 클래스가 실행이 된다. 픽셀 등의 모든 변수를 초기화를 하기 위한 converter.java가 실행이 되고, 마우스를 사용하여 선을 곡선 형태로 그릴 수 있으며 마우스를 드래그 할 수도 있는 curveTool.java가 실행이 된다. 마지막으로 화면에 직선의 도형을 그리기 위한 세팅을 도와주는 main_canvas.java가 실행된다.

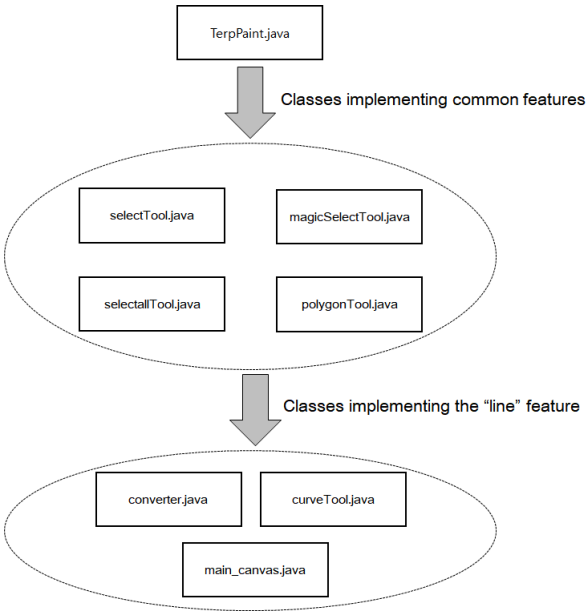


Fig. 13. Manual analysis for the Line feature

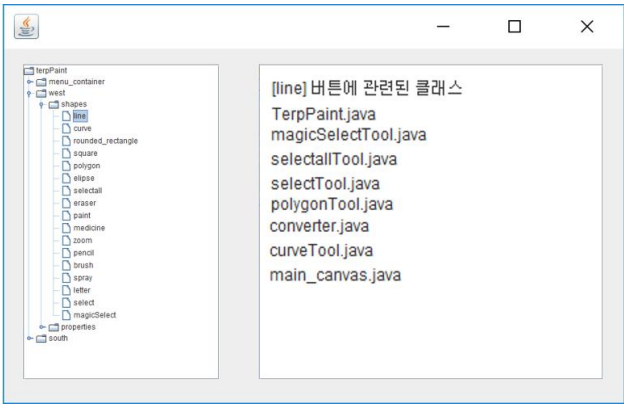


Fig. 14. Automatic analysis for the line feature

“line”기능의 분석결과를 바탕으로 Fig. 14.처럼 본 연구에서 제안한 기법을 적용한 결과와 함께 비교해 보았다. 그 결과 앞서 매뉴얼하게 분석한 모듈화 결과와 같은 결과를 제공해 주었다.

V. Conclusion

GUI의 경우 시각적으로 사용자가 보다 쉽게 이해하고 접근할 수 있기 때문에 분석하고자 하는 소프트웨어의 GUI 이벤트별로 모듈화를 수행할 수 있다면 복잡한 소프트웨어에 대해서도 보다 효율적으로 전체 소프트웨어 구조를 이해 및 분석할 수 있다. 따라서 본 연구에서는 소프트웨어 유지보수 시 소프트웨어 분석 이해도 및 효율성 향상을 위해 GUI 기반의 소프트웨어 모듈화 기법을 제안하였다. GUI 이벤트 및 이벤드 메소드

등을 바탕으로 연관된 부분들을 모듈화 함으로써 기존 모듈화 기법들과는 다른 새로운 관점에서 접근하였다. 또한 제안한 기법의 보다 유연한 활용을 위해 자동화 도구를 개발하여 실제 산업체에서 직접 활용할 수 있도록 그 기반을 마련하였다.

본 연구에서 제안한 기법은 오픈소스 소프트웨어 중 하나인 TerpPaint를 활용하여 실험을 수행하여 모듈화 결과를 살펴보았고, 적절한 모듈화가 수행되고 있는지 판단하기 위해 실제 특정 기능에 대해 매뉴얼하게 조사하여 그 결과를 비교 분석해보았다.

본 논문의 연구를 기반으로 향후에는 다양한 오픈소스 소프트웨어들을 대상으로 추가 실험을 진행하고 그 결과에 대해 세밀하게 경험적으로 분석해보고자 한다. 또한, 다양한 언어에 대한 분석을 수행할 수 있도록 도구를 확장할 예정이다. 마지막으로 점차 복잡다양하고 방대해지는 소프트웨어들에 대한 유지보수를 진행할 시 보다 빠른 시간 내에 사용자가 원하는 관점에서 원하는 정보를 선별하여 제공할 수 있도록 모듈화 기법을 더욱 발전시키고자 한다.

REFERENCES

[1] K. H. Bennett, and V. T. Rajlich, “Software maintenance and evolution: a roadmap,” Proceedings of the Conference on The Future of Software Engineering, pp. 73-87, 2000.

[2] Ian Sommerville, “Software Engineering(10th Edition),” Pearson, 2015.

[3] B. S. Mitchell, and S. Mancoridis, “On the automatic modularization of software systems using the bunch tool,” IEEE Transactions on Software Engineering, Vol. 32, No. 3, pp. 193-208, 2006.

[4] U. Erdemir, and F. Buzluca, “A learning-based module extraction method for object-oriented systems,” Journal of Systems and Software, Vol. 97, pp. 156-188, 2014.

[5] K. Praditwong, M. Harman, and X. Yao, “Software module clustering as a multi-objective search problem,” IEEE Transactions on Software Engineering, Vol. 37, No. 2, pp. 264-282, 2011.

[6] G. Bavota, F. Carnevale, A.D. Lucia, M.D. Penta, and R. Oliveto, “Putting the developer in-the-loop: an interactive GA for software remodularization,” Proceedings of the International Symposium on Search Based Software Engineering, pp. 75-89, 2012.

[7] W. Mkaouer, M. Kessentini, A. Shaout, P. Koligheu, S. Bechikh, K. Deb, and A. Ouni, “Many-objective software remodularization using NSGA-III,” ACM Transactions on Software Engineering and Methodology, Vol. 24, No. 3, pp. 1-45, 2015.

[8] A. Prajapati, and J.K. Chhabra, “Improving package structure of object-oriented software using multi-objective optimization and weighted class connections,” Journal of King Saud University - Computer and Information Sciences, Vol. 29, No. 3, pp. 349-364, 2017.

[9] O. Maqbool, and H. A. Babri, “The weighted combined algorithm: a linkage algorithm for software clustering,” Proceedings of the 8th European Conference Software Maintenance and Reengineering, pp. 15-24, 2004.

[10] D. Doval, S. Mancoridis, and B. S. Mitchell, “Automatic clustering of software systems using a genetic algorithm,” Proceedings of the Software Technology and Engineering Practice, pp. 73-81, 1999.

[11] G. Santos, M. T. Valente, and N. Anquetil, “Remodularization analysis using semantic clustering,” Proceedings of the IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering, pp. 224-233, 2014.

[12] G. Scanniello, A. DAmico, C. DAmico, and T. DAmico, “Using the kleinberg algorithm and vector space model for software system clustering,” Proceedings of the international conference on program comprehension, pp. 180-189, 2010.

[13] A. Corazza, S.D. Martino, V. Maggio, and G. Scanniello, “Investigating the use of lexical information for software system clustering,” Proceedings of the 15th European Conference on Software Maintenance and Reengineering, pp. 35-44, 2011.

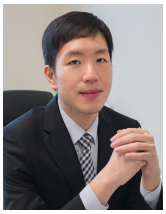
[14] A. Corazza, S.D. Martino, V. Maggio, and G. Scanniello, “Weighing lexical information for software clustering in the context of architecture recovery,” Empirical Software Engineering, Vol. 21, No. 1, pp. 72-103, 2016.

[15] M. Risi, G. Scanniello, and G. Tortora, “Using fold-in and fold-out in the architecture recovery of software systems,” Formal Aspects of Computing, Vol. 24, No. 3, pp. 307-330, 2012.

[16] TerpPaint, *http://terppaint.sourceforge.net/download/*

Authors

Dongmin Park is studying computer science and engineering at Yeungnam University, Korea. He is interested in software modularization, software analysis, mining software repositories, and software testing.



Yeong-Seok Seo received the M.S. and Ph.D. degrees in Computer Science from KAIST, Korea, in 2008 and 2012, respectively. Dr. Seo is currently a professor in the Department of Computer Engineering, Yeungnam University, Gyeongsan, Korea. He is interested in software modularization, software cost estimation, software measurement and analysis, mining software repositories, and software process improvement.