정적/동적 분석 기반의 재사용 메트릭과 가시화 구축

변은영[*1], 손현승[*], 문소영[*], 장우성[*], 박보경[*], 김영철[*2]
[*]홍익대학교 소프트웨어공학연구실
e-mail:{eybyun[*1], son, moon, jang, park, bob}@selab.hongik.ac.kr

Constructing A Visualization & Reusable Metrics based on Static/Dynamic Analysis

Eun-Young Byun[*1], Hyun-Seoung Son[*], So-Young Moon[*],
Woo-Sung Jang[*], Bo-Kyung Park[*], R. Youngchul Kim[*2]
[*1]Software Engineering Lab., Hongik University

요 약

소프트웨어의 적용 분야가 다양화되면서 시장 환경의 변화와 사용자 요구사항의 다양화가 급속도로 진행되고 있다. 하지만 부족한 시간, 예산, 인력 문제로 고품질의 소프트웨어 개발은 더 어려워졌다. 이런 문제의 해결을 위해 레거시 시스템의 모듈을 재사용하여 고품질화하고자 한다. 기존에는 정적 분석 기반의 재사용 모듈/덩어리 식별만 이루어졌지만, 실제 실행 환경에서 적용되는 동적 분석 기반의 재사용 식별이 더욱 중요하다. 이를 위해, 재사용 메트릭을 정의하고 재사용 모듈/덩어리 자동식별 및 가시화를 제안한다. 이는 새로운 프로젝트 개발의 재사용성을 높여, 신뢰성과 생산성 향상시키고 품질 개선에 기여한다.

1. 서론

최근 소프트웨어의 생산성은 시장 환경 변화와 요구사항의 다양화 속도를 따라가지 못하고 있다[1]. 이를 해결하기 위해 재사용성, 유지보수성, 품질, 생산성, 효율성 등의 연구가 진행되고 있다. 이 중에서도 소프트웨어 재사용성은 새로운 프로젝트 개발에 레거시 시스템을 재사용하여 생산성과 품질을 향상 시키고자 한다. 그러나 많은 레거시 시스템들은 재사용을 위해 설계되어 있지 않을 뿐만 아니라 재사용성 분석이 어렵다[2]. 따라서 레거시 시스템에서 재사용에 적합한 모듈/덩어리를 식별해야하고 해당 모듈의 할당 크기, 빈도 등의 요소가 중요한 역할을 한다[3]. 이런 측면에서 모듈의 재사용성은 시스템의 정적 요소인 소스 코드 뿐만 아니라 실제 실행 환경에서 객체의 메모리 할당 크기와 빈도 같은 동적 요소 또한 고려해야한다.

이 논문은 소프트웨어 내의 재사용 모듈을 자동 식별하여 재사용성을 높이고자 한다. 품질 지표인 응집도와 결합도를 측정하여 해당 모듈의 모듈화를 확인하고, 객체 할당 크기와 빈도를 기반으로 재사용에 적합한지 식별 및 가시화한다. 다음 장은 관련 연구로 자동화 시스템에 사용된 오픈 소스 Hprof, CharJS를 설명한다. 3장은 모듈화 메트릭, 재사용 메트릭을 정의한다. 4장은 시스템의 전체 구성도와 자동 추출 가시화 결과를 설명한다. 마지막으로 5장은 결론 및 향후 연구를 언급한다.

2. 관련 연구

2.1 Hprof

Hprof는 IBM에서 제공하는 프로파일러로 heap, cpu 프로파일링 정보를 제공한다. 커맨드 창에서 java 명령어를 통해 간단하게 실행할 수 있다. 프로그램이 실행되면서 종료 시에 분석 결과 파일이 생성된다. 생성된 데이터는 텍스트 또는 바이너리 형식일 수 있다. 이런 텍스트 형식의 결과 파일을 사용한다. 그림 1은 실제 프로파일링 결과 텍스트 파일이다. 동적인 경로를 추적할 수 있고 heap, cpu의 순위를 제공하여 효율성이 떨어지는 코드의 경로를 추적할 수 있다[4].

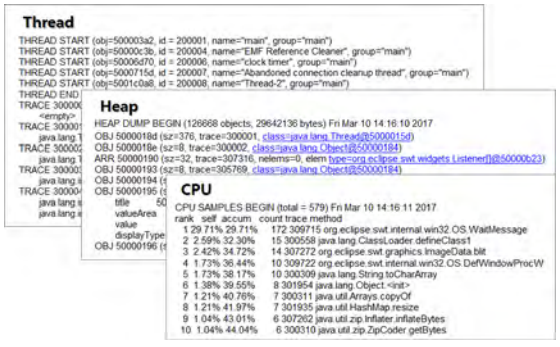

그림 1 hprof 결과 데이터

2.2 ChartJs

ChartJs는 자바 스크립트를 이용해 수치 데이터를 차트로 나타내는 오픈 소스이다. 8개 유형의 차트를 제공하고, 모든 브라우저에서 실행이 가능하다. 기타 차트

라이브러리 D3, HighCharts, Chartist와 비교해보면 Radar, Bubble 등의 다양한 차트 유형을 제공한다. 사용자가 차트 유형을 직접 정의하여 사용할 수 있고 추가 기능을 수행하는 플러그인들이 많은 장점이 있다[5].

3. 메트릭

3.1 모듈화 메트릭

재사용성에서 모듈화는 가장 중요한 토픽 중 하나이다. 모든 모듈이 재사용에 적합한 것이 아니라, 하나의 기능에 대해 정의되고 외부 모듈에 적은 영향을 받는 모듈이 재사용에 적합하고 이를 식별하기 위해 모듈화를 측정해야 한다[6]. 이를 위한 두 가지 핵심 개념은 응집도와 결합도이다. 응집도는 하나의 모듈의 내부 컴포넌트들이 하나의 기능과 관련된 정도를 의미한다. 여러 기능을 수행하는 경우 기능이 제대로 분산되지 않아서 재사용하기 어렵다. 결합도는 모듈 내의 컴포넌트들이 외부 모듈과의 상호 의존된 정도를 의미한다. 상호 의존성이 높으면 작은 변경이 외부 모듈에 큰 영향을 미친다[7].

응집도가 높을수록 결합도가 낮을수록 재사용에 적합함을 기반으로 재사용 점수를 정의한다. 표 1은 응집도와 결합도의 재사용 점수이다. 점수는 재사용에 적합할수록 1씩 증가하고, 가중치는 점수의 편차를 주기위해서 1을 기준으로 각 항목의 수로 나누어 정의하였다[8].

응집도	ce	점수	가중치
Functional Cohesion	ce_f	7	1
Sequencial Cohesion	ce_s	6	0.86
Communicational Cohesion	ce_m	5	0.71
Procedural Cohesion	ce_p	4	0.57
Temporal Cohesion	ce_t	3	0.43
Logical Cohesion	ce_l	2	0.29
Coincidental Cohesion	ce_c	1	0.14

결합도	co	점수	가중치
Data Coupling	co_d	6	1
Stamp Coupling	co_s	5	0.83
Control Coupling	co_c	4	0.67
External Coupling	co_e	3	0.5
Common Coupling	co_m	2	0.33
Content Coupling	co_n	1	0.17

표 1 응집도&결합도의 재사용 점수&가중치

그림 2는 재사용 점수와 가중치를 이용한 모듈화 메트릭이다. 각 응집도, 결합도의 평균을 구하고 평균값을 0~1사이로 표준화시킨다. 중간수치인 0.5보다 점수가 높을 경우 재사용에 적합하다고 가정하여 연구를 진행한다[8].

$$AM(Arithmetic\ mean) = \frac{1}{n}\sum_{i=0}^{n} ce = \frac{ce_f + ce_s + ce_m + ce_p + ce_t + ce_l + ce_c}{n} \quad \cdots ① \ Cohesion\ Average$$

$$AM(Arithmetic\ mean) = \frac{1}{m}\sum_{j=0}^{m} co = \frac{co_d + co_s + co_c + co_e + co_m + co_n}{m} \quad \cdots ② \ Coupling\ Average$$

Normalization

$$NO\_ce = \frac{ce - ce_{min}}{ce_{max} - ce_{min}}\ (0\sim1) \qquad NO\_co = \frac{co - co_{min}}{co_{max} - co_{min}}\ (0\sim1) \quad \cdots ③ \ Nomalization$$

그림 2 모듈화 메트릭

3.2 재사용 메트릭

정적 분석을 통한 모듈화 메트릭 뿐만 아니라 실제 실행 환경에서의 동적 분석도 고려해야한다[9]. 이 때, 객체 메모리 할당 시 크기와 빈도가 중요하다. 객체의 크기가 큰 경우 여러 가지 기능과 관련이 있을 수 있고, 상속과 같은 객체 지향적인 특성으로 인해 재사용하기 위해서 상속 클래스 또한 사용해야 한다. 할당 빈도가 높은 객체는 해당 소프트웨어에서의 재사용성이 높고, 기존 소프트웨어와 유사한 프로그램에 재사용할 가능성이 높다. 따라서 객체의 크기가 작을수록 빈도는 높을수록 재사용에 적합하다. 모듈화와 마찬가지로 각 수치를 표준화하여 사용한다.

모듈화 메트릭과 동적 분석을 통한 크기와 할당 빈도 수치를 사용하여 재사용 메트릭을 정의한다. 그림 3은 재사용 메트릭이다. 표준화한 응집도와 결합도 수치의 평균에 크기와 할당 빈도의 표준화를 가중치로 사용한다. 크기는 클수록 재사용에 부적합하기 때문에 (1-0.x)를 곱하여 재사용 수치를 절감 시키고, 할당 빈도가 높을수록 재사용에 적합하기 때문에 (1+0.x)를 곱하여 재사용 수치를 증가 시킨다.

$$AVG\_ce = \sum_{i=0}^{n} ce_i\ (n: Method\ Count) \qquad AVG\_co = \sum_{i=0}^{m} co_i\ (m: Call\ Count)$$
$$\cdots ① \ Object\ Cohesion\ \&\ Coupling\ Average$$

$$NO\_size = \frac{size - size_{min}}{size_{max} - size_{min}}\ (0\sim1) \qquad NO\_ALcnt = \frac{ALcnt - ALcnt_{min}}{ALcnt_{max} - ALcnt_{min}}\ (0\sim1)$$
$$\cdots ② \ Object\ size\ \&\ Frequency\ Nomalization$$

Weighted

Reduction Increase

$$R = \left(\frac{1}{2}(AVG\_ce + AVG\_co)\right) \times (1 - NO\_size) \times (1 + NO\_ALcnt)$$
$$(R > 0.5) \ \cdots ③ \ Reusability\ Metrics$$

그림 3 재사용 메트릭

4. 재사용 객체 식별 시스템 구축

4.1. 시스템 구성도

재사용 모듈을 자동 식별하여 가시화하는 Tool-Chain을 구성한다. 오픈 소스를 이용하여 정적/동적 분석을 수행하고 그 산출물을 기반으로 재사용성을 측정한다. 그림 4는 전체 구성도이다. 자바 기반의 타겟 코드를 입력하면 정적 분석과 동적 분석을 통해 ASTM파일과 텍스트 파일을 추출한다. 이 파일들을 구조화하여 SQLite 데이터베이스에 저장한다. 변수와 메소드의 정보는 Component, 메소드 간의 호출과 객체 생성 정보는 Link, 응집도와 결합도는 각각 Cohesion, Coupling, 사이즈와 할당 빈도는 Profiling 테이블에 구조화하여 저장한다. 구조화된 데이터를 기반으로 Graphviz와 ChartJS를 이용하여 전체 프로그램 구조와 차트로 가시화한다.

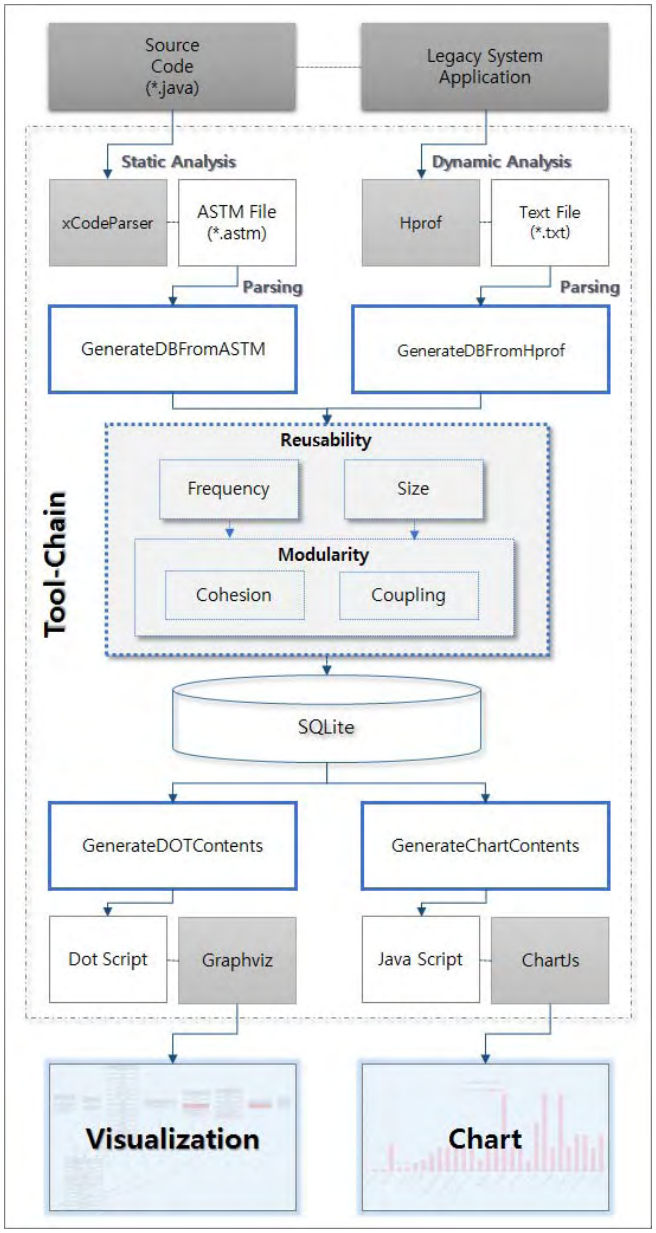

그림 4 시스템 전체 구성도

4.2 자동 추출 가시화

자동화 도구를 통해 가시화한 결과는 전체 프로그램 구조 그림 파일(.png)와 프로파일링 결과 차트(.html)로 추출된다. 가시화하기에 앞서 그림 5는 모든 가시화 노드와 관계의 종류를 정의한다. 클래스는 테이블 형태의 노드로 선언된 변수와 메소드 정보를 나타낸다. 상단 행은 클래스 이름을 나타내고 동적 분석에서의 객체 할당 사이즈와 빈도를 표시한다. 하단의 메소드에는 각 응집도 재사용 점수를 표기한다. 클래스 간의 관계는 상속, 호출, 객체 생성 관계로 구분하고 결합도 재사용 점수와 호출 빈도가 표기된다. 사용된 API는 서브그래프로 묶어서 나타낸다. 재사용에 부적합한 응집도와 결합도는 빨간색으로 가시성을 높였다.

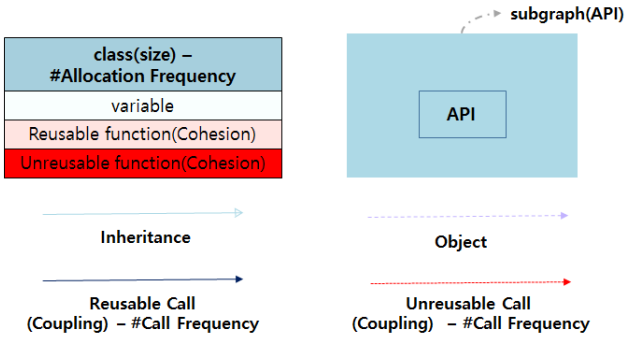

그림 5 가시화 노드 & 관계 종류

가시화 타겟 코드는 M-PVMS(Metamodel based Photovoltaic Monitoring System)이다. 태양광 모니터링 시스템으로 로컬과 서버의 통신을 위해 이종 데이터의 메타모델을 사용하는 시스템이다[10]. 그림 6은 동적 분석 결과 데이터인 객체 할당 크기와 빈도를 가시화한 차트이다. 꺾은선은 객체의 할당 빈도를 막대는 객체의 크기를 의미한다. ImageButton객체는 크기는 보통이지만 할당 빈도가 제일 높기 때문에 재사용을 고려해야 한다. GaugeFigure, StringData 객체는 크기가 작고 빈도가 높은 편이기 때문에 재사용에 적합하다. 그림 7은 [11]의 확장 연구 결과로 동적 분석과 정적 분석 데이터를 기반으로 전체 프로그램 구조를 가시화한 그림이다. 대규모 프로그램이지만 가시화를 통해 어떤 클래스들로 구성되어 있고, 어떤 관계로 이루어져 있는지 쉽게 확인할 수 있다. 모듈화가 가능한 객체를 응집도와 결합도를 통해 식별할 수 있고, 해당 객체의 크기와 빈도로 재사용에 적합한지 확인할 수 있다. 재사용의 단위는 하나의 객체인 단일 모듈과 하나 이상의 객체들의 복합 모듈로 나누어진다. StringData객체의 경우 표준화시킨 응집도와 결합도 재사용 점수의 평균을 구하고, 사이즈와 할당 빈도로 가중치를 계산하면, 그 결과 R수치는 0.5이상으로 재사용에 적합한 단일 모듈로 식별된다. ImageButton객체는 동적 할당에서 제일 할당 빈도가 높은 객체로 재사용 메트릭에 의해 마찬가지로 재사용에 적합하다고 판단된다. 하지만, GaugeFigure객체와 높은 결합도 갖고 있어 단일 모듈로 재사용이 어렵다. 이 때, GaugeFigure 또한 재사용 메트릭에 의해 재사용에 적합다고 판단되므로 두 객체를 복합 모듈로 재사용할 수 있다.

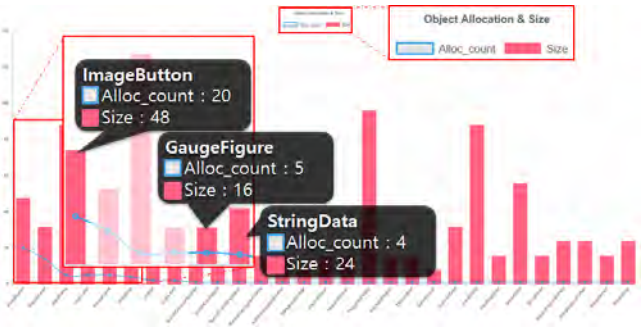

그림 6 객체 할당 크기 & 빈도 차트

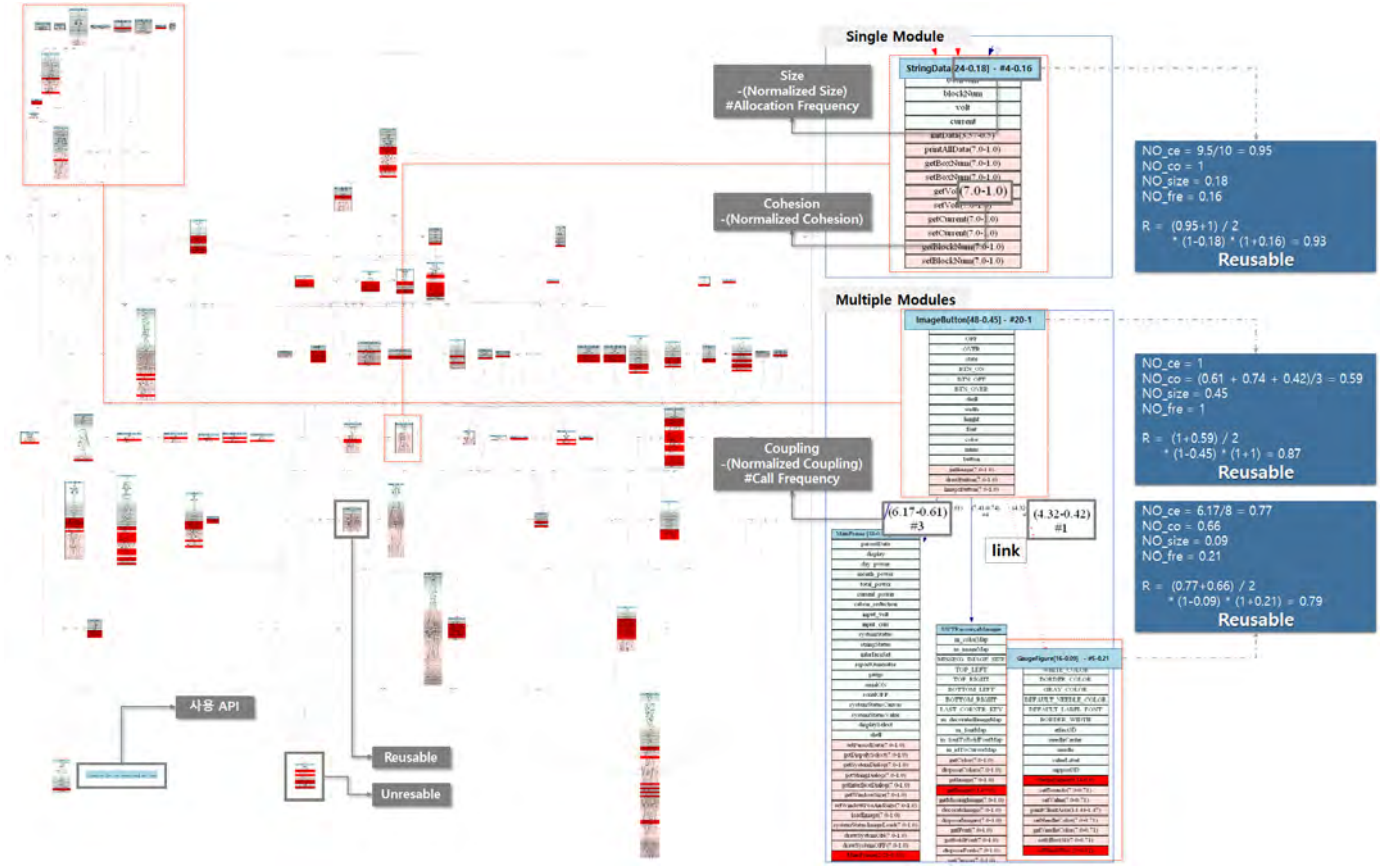

그림 7 전체 프로그램 가시화[11]

5. 결론 및 향후 연구

본 논문은 객체 지향 프로그램에서 재사용 메트릭을 통해 재사용 객체/덩어리 자동식별 및 가시화 시스템 제안한다. 정의된 재사용 메트릭은 응집도와 결합도를 기반으로 하고 추가적으로 동적 분석의 객체 메모리 할당 크기와 빈도를 적용한다. 가시화 결과로 재사용에 적합한 단일/복합 모듈 식별이 가능하다. 식별된 모듈의 재사용을 통해, 새로운 프로젝트의 신뢰성과 생산성을 향상시켜 품질을 개선할 수 있다. 향후에는 재사용 레벨을 코드 레벨이 아닌 상위 레벨로 확장하여 연구할 예정이다.

ACKNOWLEDGE

이 논문은 2015년 교육부와 한국연구재단의 지역혁신창의인력양성사업의 지원을 받아 수행된 연구임 (NRF-2015H1C1A1035548)

참고문헌

[1] Richard N. Taylor “Software Development Using Domain-Specific Software Architectures’ ACM” Vol. 20, No. 5, pp. 27-38, December, 1995

[2] L.H. Etzkorn, W.E. Hughes Jr., C.G. Davis “Automated reusability quality analysis of OO legacy software’ ELSEVIER” Vol. 43, No. 5, pp. 295-308, April, 2001.

[3] Sonia Chawla, “Review of MOOD and QMOOD metrics sets’ Computer Science and Software Engineering” Vol. 3, No. 3, pp. 448-451, March, 2013.

[4] HPROF: A Heap/CPU Profiling Tool, http://docs.oracle.com/javase/7/docs/technotes/samples/hprof.html

[5] Chart.js: API Documentation, http://www.chartjs.org/docs

[6] Chris Luer “Assessing Module Reusability’ IEEE Computer Society” May, 2007.

[7] G.Gui, P.D Scott “Coupling and Cohesion Measures for Evaluation of Component Reusability’ ACM” May, 2006.

[8] 변은영, 박보경, 장우성, 김영철, “가치 있는 모듈 식별을 위한 오픈 소스 기반 소프트웨어 현대화 시스템 구축’ KIISE” pp. 404-406, December, 2016.

[9] 강건희, 이근상, 이진협, 김영철, “프로파일러를 이용한 소프트웨어 메모리 성능 가시화 방법‘ KISM” Vol. 5, No. 1, pp. 296-297, April, 2016.

[10] Hyun Seung Son, R. Young Chul Kim “Modeling a Photovoltaic Monitoring System based on Maintenance Perspective for New&Renewable Energy’ International Joint Conference on Convergence” pp. 144-147, January, 2016.

[11] 변은영, 박보경, 장우성, 김영철, 손현승, “재사용성 추출 메카니즘을 위한 오픈 소스 기반 소프트웨어 시스템 구축‘ KTCP”(게재 예정)