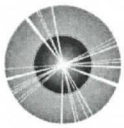

제 18 권 제 1 호
Vol. 18 No. 1

2016

제 18회 한국 소프트웨어공학 학술대회 논문집

Proceedings of the 18th Korea Conference on Software Engineering (KCSE 2016)

- 일시: 2016년 1월 27일(수) ~ 1월 29일(금)
- 장소: 강원도 평창 한화리조트(휘닉스파크점)

주최: 한국정보과학회, 한국정보처리학회

주관: 한국정보과학회 소프트웨어공학 소사이어티
한국정보처리학회 소프트웨어공학 연구회
한국전자통신연구원

후원: ㈜솔루션링크, ㈜코스콤, ㈜모아소프트, ㈜비트컴퓨터, 소프트웨어공학엑스퍼트그룹㈜, ㈜엔에스이, 한국산업기술시험원, 무인자율 및 적응형 SW 연구단, ㈜다한테크, SW 상시모니터링기술연구단, 시스트란 인터내셔널, ㈜티큐엠에스, ITRC 고품질융합소프트웨어연구센터, 슈어소프트테크㈜, ITRC 소프트웨어안전성보증연구센터, STA 테스팅컨설팅㈜, ㈜이에스지, 정보통신산업진흥원 소프트웨어공학센터, ㈜테스트마이다스, TTA 소프트웨어시험인증연구소

1 월 29 일 (금)

시 간	행 사 내 용			
07:00-09:00	**조식**			
	논문 발표 D			
	D1: SW 검증 좌장 유준범 교수(건국대) 장소: 그랜드홀 2	**D2: 요구공학** 좌장 김순태 교수(전북대) 장소: 세미나실 1	**D3: 온톨로지 및 서비스** 좌장 홍장의 교수(충북대) 장소: 세미나실 2	**D4: SW 품질 2 및 SE 교육** 좌장 이병정 교수(서울시립대) 장소: 세미나실 3
09:00-10:40 (100 분)	정형 기법을 이용한 NFV Policy 들의 일치성 검증 [단편논문] 구근회(고려대), 강미영, 최진영(고품질융합소프트웨어센터), 이승익(ETRI) Safety Critical System 에서 요구사항 검증 및 통합 명세화 방법제안 [단편논문] 임혜선, 이석원(아주대) 소프트웨어 연구 문서 산출물의 추적을 위한 연관성 링크 정보 모델 기반 전문가 시스템 설계 백두산(아주대), 이병정(서울시립대), 이정원(아주대) 위기대응 매뉴얼 신뢰성 향상을 위한 검증방안 연구 [단편논문] 이 혁, 최진영(고려대)	국방 전투관리체계 개발을 위한 상호작용 유즈케이스 기반 요구사항 모델링 기법 김두환, 홍장의(충북대), 김동환(LIGNex1) PbD 7 대 기본원칙과 GQM 을 활용한 프라이버시 요구사항 도출 방법론 [단편논문] 조주혜, 이석원(아주대) 지능형 화폐 인식 SW 개발의 Visualization 적용 사례 및 임베디드 SW 개발 조직의 Visualization 적용 로드맵 제안 [산업체논문] 한동준(소프트웨어 안전성 보증 연구센터), 김은비, 한혁수(상명대), 엄영석, 정대식, 조달호(KISAN) 인지적 분석을 통한 임상 의사결정 지원 시스템 인터페이스 설계 [단편논문] 고동균, 김유찬, 윤완철(KAIST)	혈액종합검사에 특화된 온톨로지 구축: 설계와 활용 제현우, 박유경, 홍원의, 이문용(KAIST) 사용자의 서비스 사용 성향을 고려한 클라우드 서비스 추천 최비오, 박준석, 황제승, 김웅수, 윤동규, 염근혁(부산대) 이기종 시맨틱 온톨로지 시스템의 통합 검색 [단편논문] 황상원, 남영광(연세대) 소프트웨어 성능 가시화를 위한 툴 체인 개발 [단편논문] 강건희, 박보경, 장우성, 황준순, 권하은, 이한솔, 이현준, 김영철(홍익대)	PageRank 와 토픽 모델링을 이용한 소프트웨어 재사용 도메인 토픽 추출 시스템 개발 이용석, 남영광, 황상원(연세대) 소프트웨어 개발 프로세스를 대상으로 수행하는 추적성 분석의 세부 관계 정의 [단편논문] 김재엽, 이동아, 유준범(건국대) Legacy 시스템의 안전성 분석기법 및 사례연구 [단편논문] 김선주, 이석원(아주대) 프로젝트 중심 소프트웨어 공학 교육의 성과 분석과 개선 방안 [단편논문] 최은만, 김하영(동국대)
10:40-10:50	**휴식**			

소프트웨어 성능 가시화를 위한 툴 체인 개발

강건희*, 박보경, 장우성, 황준순, 권하은, 이한솔, 이현준, 김영철

*홍익대학교 소프트웨어공학연구실
세종특별자치시 조치원읍 세종로 2639
{kang, park, jang, hwang, kwon, hslee, hjlee, bob}@selab.hongik.ac.kr

요약: 오늘날의 소프트웨어 산업은 점점 커지며 고품질에 대한 이슈가 대두되고 있다. 하지만 커지는 산업에 비해 시장 출하 기간의 단축상황으로 산업현장에서는 빠른 개발을 위한 코드 중심의 개발을 하게 된다. 그 결과 저 품질의 소프트웨어가 양산된다. 경쟁력을 가진 고품질의 소프트웨어 생산을 위한 인력과 비용이 우리나라는 부족하다. 그래서 소프트웨어 가시화가 필요하다. 본 논문에서는 소프트웨어의 성능(반응속도)가시화를 위한 도구의 구성과 가시화 방법을 소개한다. 예를 들면, 프로파일러를 통해 소프트웨어 성능정보, 그리고 기존 소프트웨어 가시화의 도구에서 추출하는 소프트웨어 구조정보를 가지고 소프트웨어의 가시화한다. 제안한 방법은 개발자뿐 아니라 다양한 관계자들이 소프트웨어의 성능에 대한 이해가 쉬울 것이다.

핵심어: 소프트웨어 성능, 소프트웨어 가시화, 역 공학

1. 서론

한국의 소프트웨어 시장규모는 전년 대비 5.8% 성장한 110 억 달러로 파악되며, 2017 년까지도 성장의 지속이 전망된다[1]. 그러나 우리나라 소프트웨어 규모는 세계 소프트웨어 시장에서 차지하는 비중이 약 1%에 불과하다. IT 강국이지만 하드웨어 쪽으로 치우치는 기현상이 보인다. 또한 시가총액 상위 ICT 기업에서 Google, Microsoft, Facebook, Amazon 등 외국 소프트웨어 기업이 주도하며, 반면 국내 소프트웨어 기업의 글로벌 경쟁력은 아직 부족하다. 그리고 국내 대기업의 소프트웨어 개발은 SW 공학적 접근의 개발을 하고 있다. 하지만 중소기업은 SW 공학적용을 위한 전문 인력과 개발시간, 자본 등의 부족으로 구현 중심의 개발을 한다. 이러한 소프트웨어 산업의 대부분인 중소기업의 현실을 개선 위해서 프로젝트의 시작부터 소프트웨어 공학과 다양한 소프트웨어 개발 프로세스의 적용과 이를 위한 인력을 배치가 필수다. 하지만 이미 개발이 시작된 프로젝트 중간에 적용하기란 쉽지 않고, 기 개발된 소프트웨어에 적용은 소용이 없다. 기존연구[2-4]에서 역 공학을 통해 기존의 소프트웨어의 소스코드를 통해 품질지표의 적용과 가시화에 초점을 뒀다. 기존 소프트웨어의 가시화를 통해 소프트웨어의 구조를 쉽게 파악하고 소프트웨어 품질지표도 함께 적용하여 소프트웨어의 품질판단과 어느 부분이 품질상의 문제점이 있는지 파악 가능하다. 그래서 프로파일러를 통해서 소프트웨어 성능정보를 얻고 이를 가시화한다.

본 논문의 구성은 다음과 같다. 2 장 관련연구는 소프트웨어 가시화와 소프트웨어 성능에 대해 설명한다. 3 장은 소프트웨어 성능 측정 위한 도구에 대해 기술한다. 마지막으로 4 장에서는 결론을 언급한다.

2. 관련 연구

2.1 소프트웨어 성능

전통적인 알고리즘에서 성능은 중요한 명령어의 수행 횟수 및 메모리 공간의 점유율로 정의한다. 하지만 컴퓨터 사이언스 중 하드웨어의 성능은 처리속도, 채널 용량, 지연 속도, 대역폭, 처리량, 전력 소모 등이 있고 소프트웨어 성능은 신뢰성, 코드의 크기 및 무게, 반응속도 등도 있다[5,6].

소프트웨어성능에서 소프트웨어 가용성은 평균고장발생간격(MTBF)에서 평균 수리시간을 뺀 뒤 평균 수리시간으로 나눈 값의 100 을 곱하여 소프트웨어가 구동되는 전체 시간에서 실제 기능을 하는 시간을 백분율로 계산한다. 평균고장발생간격(MTBF)은 오류가 발생할 때까지 응용프로그램이 실행되는 평균시간이다. 그리고 평균복구시간(MTTR)은 오류가 발생한 후 서비스를 복구 및 복원하는데 필요한 평균시간을 의미한다. 그리고 코드의 크기와 무게는 모바일 시스템과 같은 임베디드 시스템에서 자원이 제한되어있기 때문에 소프트웨어의 크기 가 성능에 많은 영향을 끼친다. 소프트웨어 반응속도는 시스템 혹은

† 이 논문은 2015 년 교육부와 한국연구재단의 지역혁신창의인력양성사업(NRF-2015H1C1A1035548)과 2015 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2013R1A1A2011601).

실행단위에 입력이 주어진 뒤 반응까지 걸린 시간을 의미한다. 반응속도가 길어질수록 시스템의 속도는 느려진다.

본 연구에서는 다양한 소프트웨어 프로젝트의 이해관계자가 쉽게 이해되는 소프트웨어 메소드 단위의 소프트웨어 반응속도를 프로파일러를 통해 측정 및 가시화 한다.

2.2 소프트웨어 가시화

성공적인 소프트웨어 개발 및 관리를 위해 소프트웨어 개발, 프로세스, 테스트 자동화, 그리고 품질인증 등은 필수적이다. 하지만 이러한 일을 수행하는 자원과 전문인원이 너무나 부족하다. 소프트웨어 가시화는 유지보수와 품질관리의 효율성을 향상하는 기법이다. 소프트웨어 가시화에는 시각화와 문서화가 있다[6].

첫째, 시각화는 소프트웨어 개발의 가장 어려운 점인 소프트웨어 비가시성을 극복하고 전체 소프트웨어 개발 과정을 파악하여 품질관리를 수행하는 방안이다. 둘째, 문서화는 기업 혹은 단체의 개발 노하우 관리 및 내부 인력간의 업무 이해도 향상과 특정 상황에서 외부와의 의사소통을 위한 방안이다.

본 연구는 기존연구[2-4]와 다르게 코드의 복잡도(결합도, 응집도)뿐만 아니라 소프트웨어의 성능 정보를 가시화 한다.

3. 소프트웨어 성능 측정을 위한 도구

그림 1 은 소프트웨어 성능 가시화를 위한 도구의 구성도이다. 소프트웨어성능 가시화를 위한 툴체인에는 기존 소프트웨어 가시화를 위한 툴체인[2]에 소프트웨어 성능의 좋지 않은 패턴을 추출하기 위한 RuleChecker(PMD[7])와 소프트웨어의 동적분석을 위한 Profiler(Hprof[8]), Profiler 에서 추출된 데이터를 Xml 데이터로 정제하기 위한 HprofDataExtractor 를 추가하였다. 소프트웨어 성능 가시화를 위해서는 소프트웨어의 정보 추출-> DB 저장-> 품질지표정의->가시화 총 4 단계를 거치게 된다.

- 1 단계: 소프트웨어의 정보추출을 위해서 소프트웨어 구조정보, RuleChecker 를 통한 성능저해요소 정보, Profiler 를 통한 성능 추출이 수행 한다.
  - ■소프트웨어 구조정보: 소프트웨어 구조정보를 추출하기 위해서는 SourceNavigator 에 소스코드를 입력하여 SNDB 파일을 추출한 뒤 바이너리로 되어있는 정보를 DBdump.exe 로 볼 수 있는 문자열로 변경, 추출한다.
  - ■소프트웨어 성능저해요소정보: 성능저해요소의 대한 패턴을 XPath 로 정의하고 RuleChecker(PMD)에 적용하여 XML 정보로 추출한다.
  - ■Profiler 를 통한 성능정보: 실제 소프트웨어를 프로파일러와 함께 구동하여 소프트웨어 메소드 단위의 구동시간과 호출횟수정보를 추출한다. 하지만 추출된 정보가 규칙이 없는 문자열 값이기 때문에 HprofDataExtrator 프로그램을 통해서 Xml 형식의 정제화된 정보로 추출한다.
- 2 단계: CreateDB 프로그램을 통해서 DBdump 를 통한 SNDB 의 정보를 해당하는 테이블에 저장한다. 그리고 RuleChecker 와 Profiler 에서 추출된 성능에 대한 정보도 DB 에 입력한다.
- 3 단계: 품질지표 적용단계에서는 소프트웨어 구조정보를 통해서 각 모듈(클래스) 간의 결합도를 측정한다.
- 4 단계: 가시화 단계에서는 GenerateDotContents 프로그램을 사용하여 쿼리문으로 추출된 정보로 코드에 대한 아키텍쳐와 정량화된 품질지표 수치를 DOT 스크립트로 생성하고 graphViz 도구를 통해서 그래프로 그린다.

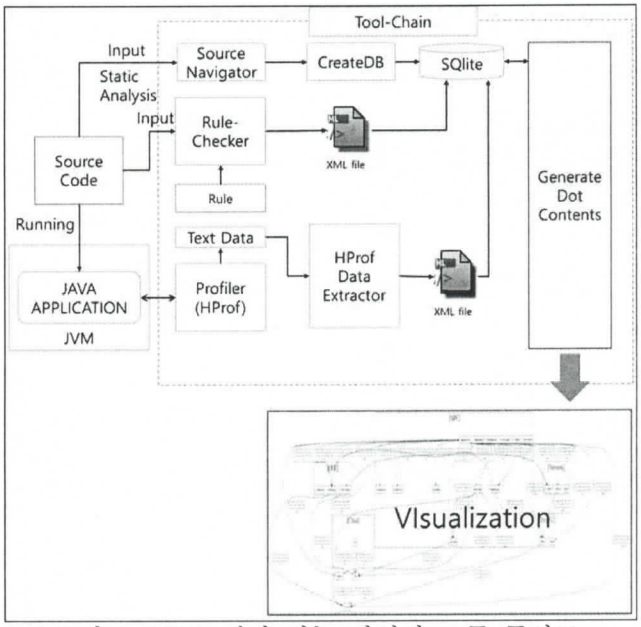

그림 1 소프트웨어 성능 가시화 도구 구성도

그림 2 는 소프트웨어 성능가시화도구에 결합도(자료, 스탬프, 제어, 외부, 공유, 내용)와 성능 저해요소(Loop Unrolling, Loop DownCounter, 반복문 내부의 불필요한 조건문, 다중 조건문)가 포함된 소스코드를 입력 후 구동한 가시화 이미지이다.

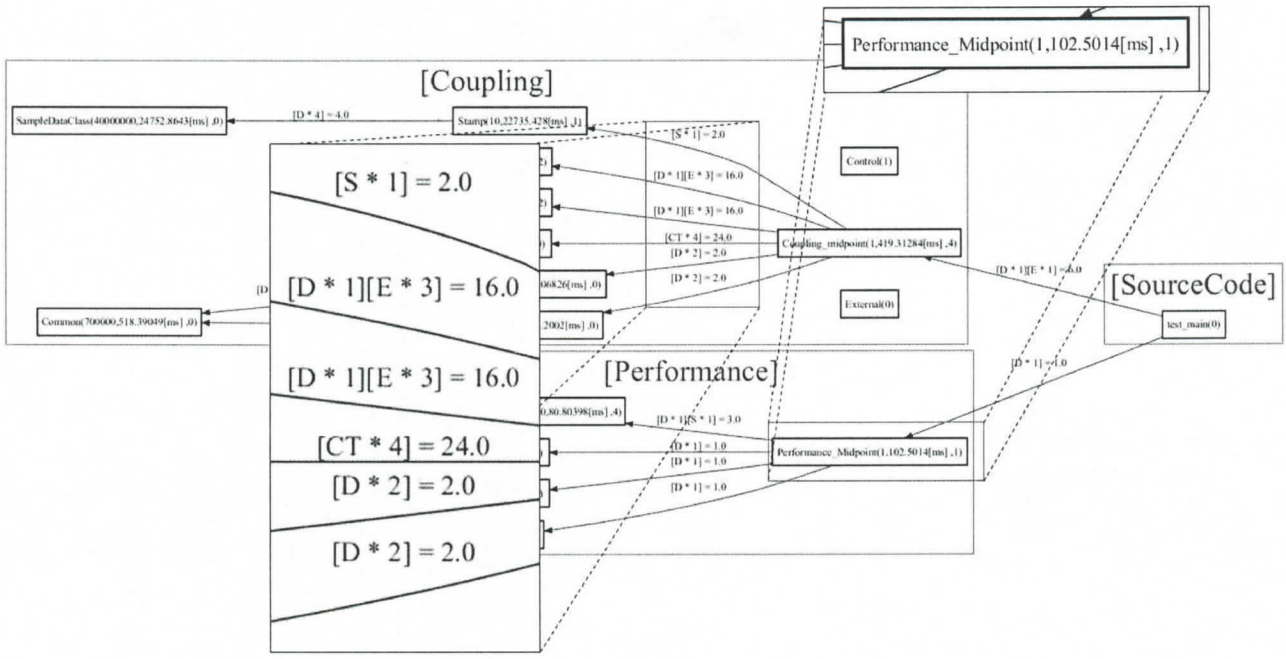

그림 2 추출된 소프트웨어 성능 가시화 이미지

그림 2 를 보면 네모로 표시되는 모듈에서는 클래스의 실제 호출 수, 클래스 전체의 수행시간, 성능저하요소 순으로 데이터를 표기한다. 그리고 모듈 간의 화살표에는 모듈 간의 결합도(D: 자료, S: 스탬프, C: 제어, E: 외부, CM: 공유, CT: 내용)의 개수와 결합도 총합점수를 표기한다. 이렇게 소프트웨어의 구조와 함께 성능지표, 결합도 점수를 같이 표현하여 쉽게 소프트웨어의 성능에 대한 평가가 가능하다.

4. 결 론

현재 우리나라의 거대해지는 소프트웨어 산업에 의해서 소프트웨어 고품질에 대한 이슈가 대두되고 있다. 하지만 그에 비해 단발성의 소프트웨어 개발, 시장 출하기간의 단축 등으로 인한 구현 중심의 개발 프로세스와 잦은 개발 인력의 이동으로 개발 문서의 부재, 잦은 패칭으로 소프트웨어의 품질저하의 악순환이 지속되고 있다. 그래서 본 논문에서는 소프트웨어의 고품질화를 위한 소프트웨어의 성능 측정 및 개선을 위한 역 공학 기반의 자동화 메커니즘 제안과 개발한다. 우리는 역공학과 정적 분석을 통해 소프트웨어의 재 사용성을 높이는 결합도와 소프트웨어성능에 위배되는 성능저하요소를 추출 했다. 그와 동시에 자바 성능측정도구(HProf)를 통한 동적 분석을 수행하여 정적 분석에서 얻을 수 없는 메소드의 수행속도, 실제 수행 횟수의 정보를 얻어 이전과는 다른 새로운 소프트웨어 가시화 했다. 이를 통해 실제 개발자뿐 아니라 많은 이해관계자들도 소프트웨어의 품질 지표와 구조를 쉽게 이해가 가능하다. 그리고 실제 소프트웨어가 실행 될 때 제일 많이 수행시간을 소모하는 부분을 보다 쉽게 찾는 것이다. 그래서 그 부분의 문제를 발견하고 리펙토링 한다면 성능적 품질의 개선이 가능하다.

현재 Java 언어에만 프로파일러가 적용되어, 향후 C, C++의 성능 가시화를 적용할 예정이다. 또한 다양한 언어의 의존된 성능문제를 찾아 패턴화 적용할 예정이다. 그리고 임베디드 시스템의 성능 가시화중 프로파일러를 통한 속도 측정뿐 아니라 실제 디바이스를 통한 사용전력을 측정하여 전력이 많이 소모되는 소스코드의 패턴을 찾아 가시화에 적용할 예정이다.

참고문헌

[1] Gartner "Gartner Market Databook, 2Q15 Update, 2015.06

[2] 강건희, 이근상, 김동호, 황준순, 김영수, 박용범, 김영철, "절차식 언어 기반의 코드 정적 분석을 위한 툴 체인 사례 연구," 한국정보과학회 2014 한국 컴퓨터 종합학술대회 논문집, pp. 559-561, 2014

[3] 강건희, 이근상, 김동호, 황준순, 김영수, 박용범, 김영철, "SW 가시화 기반 리펙토링 기법 적용을 통한 정적 코드 복잡도 개선," 2014 한국정보처리학회 추계학술대회, 제 21 권, 제 2 호, pp. 646-649, 2014

[4] 강건희, 이근상, 김영수, 박용범, 손현승,

김영철, "Open Source 기반 툴 체인화를 통한 코드 정적 분석 연구, 한국 정보과학회 컴퓨팅의 실제 논문지, 제 21 권, 제 2 호, pp. 148-153, 2015

[5] Henry H. Liu, "Software Performance and Scalability, A quantitative Approach," Wiley, 2009

[6] Henry H. Liu, "Java Performance and Scalability, A quantitative Approach," Wiley, 2013

[7] PMD, https://pmd.github.io/

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