오픈 소스 소프트웨어 환경에서 재사용 문제 해결을 위한 개념적 프레임워크와 패턴의 분석

한국과학기술원 ▌구형민 · 고인영*

1. 서 론

소프트웨어 재사용은 이미 개발된 소프트웨어 자산들을 새로운 소프트웨어의 개발에 재활용함으로써 생산성을 높이고 고수준의 품질을 보장하고자 하는 개념이다[1,2]. 최근 많은 기업과 조직들, 개별 개발자들은 이러한 소프트웨어 재사용의 효율을 높이고자 오픈 소스 소프트웨어(OSS: Open Source Software)환경을 많이 이용하고 있다[3,4]. OSS는 온라인을 통해 개발자들 간에 자유롭게 재사용 자산을 생성 및 교환하고 협력적으로 자산을 개선해 나가는 환경을 제공하여 그 중요성이 부각되고 있다[3,4]. 그러나 개발자들이 OSS환경에서 유용한 재사용 자산을 찾았다 하더라도 찾은 자산을 본인의 소프트웨어와 통합하는 과정에서 비호환성, 비상호운용성 등의 문제를 겪게 된다. 성공적인 소프트웨어의 재사용은 효과적인 문제의 해결을 통해 달성할 수 있다[5,6].

본 연구에서 조사한 결과 70%의 미해결 재사용 관련 문제(Reuse-related problems)는 이미 존재하고 있는 동일하거나 유사한 해결책을 이용하여 해결 가능한 것으로 확인되었다. 이 조사를 위해 소스포지(Sourceforge.net)[1]의 '소프트웨어 개발(Software development) > 사용자 인터페이스(User interfaces)' 카테고리에 있는 208개의 소프트웨어 자산을 선택하였다. 특별히 이 카테고리를 선택한 이유는 이 카테고리의 토론 포럼(Discussion forum)에서 개발자들이 실질적으로 가장 많은 문제, 코멘트 및 의견을 활발하게 포스팅(Posting)하여 논의하기 때문이다. 포럼 쓰레드(Forum threads)에서 258개의 미해결 문제를 추출하였고, 이 258개의 문제 중에 179개의 문제를 분석하여 보았다. 나머지 79문제는 문제에 대한 설명이 너무 간략하거나 애매해서 분석이 어려웠다.

본 조사를 통해 OSS에서 재사용 관련 문제를 해결함에 있어서 일정한 패턴이 존재한다는 것을 확인하였다. 개발자들은 자신이 겪는 재사용 문제의 증상들(Symptoms)을 포스팅함으로써 문제를 처음으로 등록하고, 일정 시간이 지난 후 다른 개발자들은 자신의 경험을 바탕으로 증상의 원인들(Causes)에 대해 논의하기 시작한다. 마지막으로 원인에 대해 논의하기 위한 충분한 인터렉션(Interactions)을 거치고 나면 잠재적인 해결책들이 도출되어 제안된다. 이 조사를 통해 본 연구에서는 두 가지 가설을 세웠다:

- **가설 1:** OSS의 포럼 쓰레드 내에 포스팅 된 문제의 증상과 원인 간에는 유의한 상관관계(Significant correlations)가 존재한다.
- **가설 2:** 재사용 관련 문제의 원인을 일찍 알아내는 것은 해결책을 찾는데 도움을 준다.

위의 두 가설에 의거하여 본 연구에서는 재사용 관련 문제의 원인을 추천해 줌으로써 결과적으로 해결책을 찾아줄 수 있는 개념적인 프레임워크를 개발하였다. 그림 1은 이 프레임워크의 개요를 보여주며 프레임워크는 크게 세 부분으로 구성 된다: 1) 의미론적인 방법으로 재사용 문제의 증상, 원인, 해결책간의 상관관계를 표현하기 위한 온톨로지 기반의 지식 표현 및 저장 모델, 2) OSS에서 데이터를 추출해 내는 프로세스, 3) 재사용 문제 해결을 위한 지식 베이스(Knowledge base)에 저장하는 부분이 그것이다. 두 가설을 증명하기 위해 FLOSS[2] 오픈 저장소를 이용해 소스포지의 '소프트웨어 개발 > 사용자 인터페이스' 카테고리에서 358 자산을 추출하였다. 가장 많은 논의가 이루어

* 정회원

† 본 연구는 방위사업청과 국방과학연구소의 지원으로 수행되었음 (2010-SW-12-DM-01).

1) http://sourceforge.net/

2) http://en.wikipedia.org/wiki/Free_and_open_source_software

그림 1 재사용 관련 문제 해결을 위한 개념적 프레임워크

지고 있는 호환성 문제에 관한 10,378개의 문제 포스팅과 그에 따른 29,859개의 답변 포스팅을 포럼 쓰레드로부터 추출하였다.

개발한 프레임워크를 기반으로 재사용 증상과 원인의 타입들을 분석하고 각 타입 간 상관관계의 유의성 수준(Significance level)을 확인하기 위하여 통계학적 분석을 수행하였고 분석 결과 60%의 재사용 증상과 원인간의 상관관계가 통계학적으로 유의함을 확인하였다. 또한, 재사용 문제 해결책 추천을 위한 효과적인 시간을 알아내기 위해 포럼 쓰레드들에서 원인의 재현율(Recall rate)을 계산하는 실험을 수행하였다. 이 실험을 통해 본 연구에서 제안하는 프레임워크를 활용한다면 재사용 관련 문제 해결의 효율성이 1.5배 정도 증가할 수 있음을 확인하였다.

본 논문의 2절에서는 지식 기반 재사용 문제 해결과 관련된 기술동향에 대해 논하고 3절에서는 본 연구를 통해 개발된 개념적 프레임워크에 대해 설명한다. 4절에서는 본 연구의 실현가능성(Feasibility) 및 실용성(Practicality)을 증명하기 위해 수행한 실험들에 대해 설명한다. 마지막으로 결론 및 향후 연구에 대해 5절에서 논한다.

2. 지식 기반 재사용 문제 해결 기술동향

소프트웨어 재사용과 관련된 지식 베이스를 구축하고 이를 통해 개발자들을 돕고자 하는 몇몇 연구들이 진행되어 왔다. 대표적으로 SeCold(Source code ECOsystem Linked Data)는 개발자들이 필요한 소스코드를 다양한 수준에서 관련 지식들과 함께 제공해 주는 환경을 제공하고자 하는 연구이다[7]. SeCold는 시맨틱 웹 데이터를 서로 연결하고 공유하는 웹 기반 구조(Infrastructure)를 정의하고 그림 2에서 보는 바와 같이 소스코드 관련 정보를 의미론적으로 표현하고 추출하는 온톨로지 모델을 제공한다. 이 모델을 기반으로 개발자들에게 소스코드 및 관련 지식을 제공하여 준다.

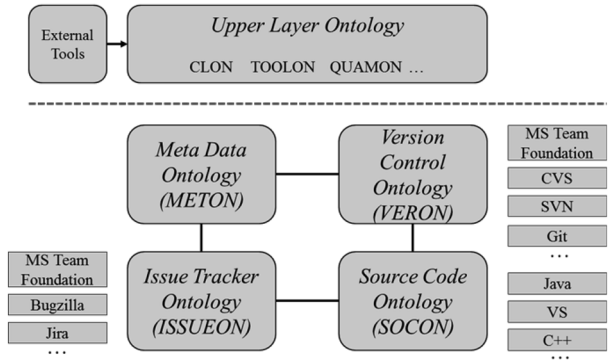

그림 2 SeCold에서 지식 관리를 위한 온톨로지 구조[7]

LD2SD(Linked Data Driven Software Development)는 링크드 데이터(Linked data)[3] 형식으로 구축된 지식 베이스로부터 개발자들이 오픈소스 코드를 쉽게 검색하고 관련 산출물들까지 추천해 줄 수 있는 환경 제공을 그 목표로 하는 연구이다[8]. 그림 3은 LD2SD의 개념적인 아키텍처를 보여준다. 그림에서 보는 바와 같이 LD2SD에서는 Sindice 검색 엔진[4]을 기반으로 지식

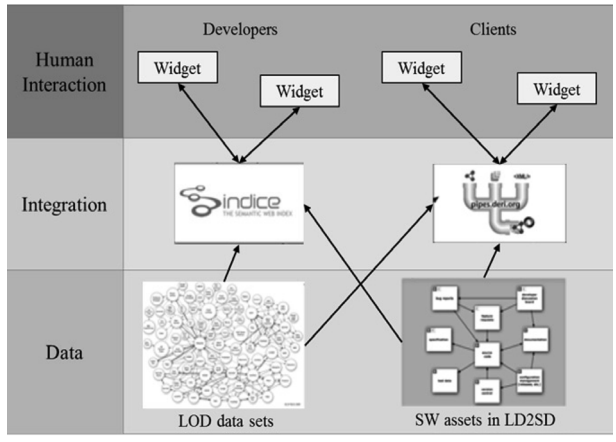

그림 3 LD2SD의 개념적 아키텍처[8]

3) http://www.w3.org/standards/semanticweb/data
4) http://sindice.com/

을 검색하고 위젯의 형태로 개발자들의 통합 개발환경(IDE: Integrated Development Environment)과 연동되어 지식을 제공한다. 개발자들이 IDE에서 소스코드를 컴파일하거나 실행 한 결과들이 지식 베이스에 저장되고 타 개발자들에게 제공된다.

GATE(General Architecture for Text Engineering)는 OSS에서 소스 코드 및 사용자 매뉴얼 관련 문서로부터 데이터를 추출하여 지식 베이스를 구축하는 것을 그 목표로 한다[9]. 이 연구에서는 크라울러(Crawler)를 활용하여 소프트웨어 관련 문서로부터 핵심 데이터들을 추출하고 수집된 데이터들을 식별하여 의미를 부여한다. 도메인 온톨로지와 의미론적 연결(Annotation)기법을 이용하여 내용의 개념(Concepts) 및 실제 인스턴스들(Instances)을 연결하여 관리하고 지식 검색을 위하여 SPARQL (SPARQL Protocol and RDF Query Language)[5]이 이용된다.

EvoOnt (Evolutionary Ontology)는 OSS 소스코드 관련 버그 및 에러 관련 정보들을 수집하여 관리하기 위한 저장소 구축을 그 목표로 한다[10]. OSS에서 발생하는 에러 및 버그들을 수집하고 저장하기 위해 소프트웨어 온톨로지, 버전 온톨로지, 버그 온톨로지를 정의하였고, iSPARQL(imprecise SPARQL)[6] 검색 엔진을 기반으로 관련 지식들을 검색한다. 개발자들은 재사용 과정에서 버그 및 에러가 발생하였을 때 유사하거나 동일한 문제를 겪은 다른 개발자들의 경험을 검색하여 활용할 수 있다.

본 연구에서 개발된 프레임워크와 유사하게 이 연구들 또한 오픈 소스 자원을 기반으로 지식을 구축하고 개발자들에게 제공하여 주는 것을 목표로 한다. 그러나 이 연구들은 대부분 소스 코드 수준에서의 지식을 다루며 일반적인 소프트웨어 재사용 관련 문제해결에 대한 지식을 다루고 있지 않다.

3. OSS 환경에서 재사용 문제 해결을 위한 개념적 프레임워크

3.1 온톨로지 기반의 재사용 문제 해결 지식 표현 모델

재사용 관련 문제는 개발자들이 그 증상을 토론 포럼에 기술함으로써 보고된다. 재사용 관련 문제의 증상은 'OSS 자산을 재사용하는 과정에서 개발자 들이 겪는 예상치 못한 상황이나 문제'로 정의된다. 증상은 여러 가지 이유가 원인이 될 수 있고 본 연구에서는 증상의 원인을 '재사용 자산에 대한 불확실하거나 잘못된 정보로 인해 자산을 선택하고 통합하는 과정에서 발생하는 개발자들의 잘못된 행동, 혹은 자산 자체에 존재하는 결함'이라고 정의하였다. 마지막으로 문제의 해결책은 '재사용 관련 문제를 발생하는 원인을 제거하거나 회피하는 활동'이라고 정의되었다.

그림 4는 OSS에서 재사용 관련 문제 해결 지식을 표현하기 위한 온톨로지 기반의 메타 모델을 보여준다. 이 모델은 의미론적인 방법으로 문제 해결 지식의 주요 개념(Concepts), 관계(Relations) 그리고 속성(Properties)을 나타내는 몇 가지 온톨로지들로 구성되어 있고, 서론에서 설명한 사전 조사의 결과를 기반으로 구축되었다.

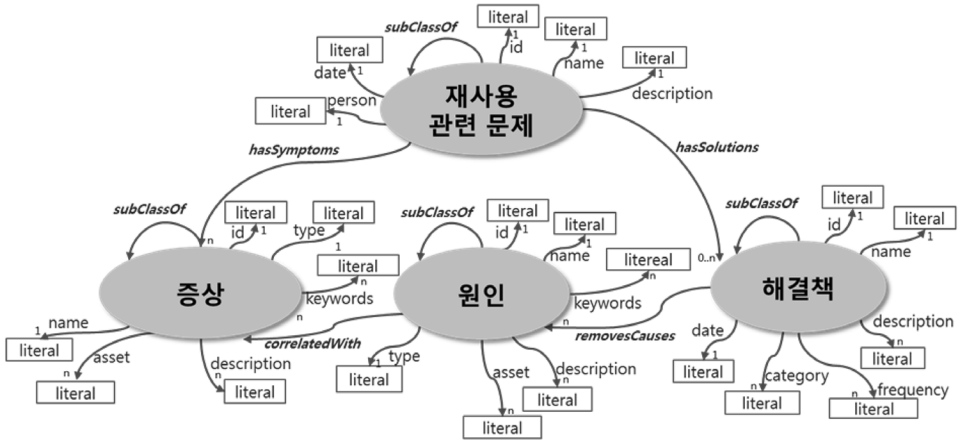

그림 4 온톨로지 기반 재사용 문제 해결 지식 표현 모델

5) http://en.wikipedia.org/wiki/SPARQL

6) https://files.ifi.uzh.ch/ddis/oldweb/ddis/research/completed-projects/semweb/isparql/

3.2 문제 해결 지식 베이스 구축을 위한 데이터 추출 및 분석 프로세스

재사용 관련 문제 해결 지식 베이스 구축을 위한 첫 번째 과정은 문제의 증상과 원인에 대한 기술을 포함하고 있는 메시지 포스팅(Message posting)을 추출하는 것이다. 'Bag of words'[11] 기법을 활용하여 증상과 원인 기술을 위해 가장 많이 나타나는 키워드 패턴들을 추출하였고 온톨로지에는 'keywords' 속성으로 표현된다. 재사용 관련 문제의 증상 혹은 원인을 포함하고 있는 메시지 포스팅들을 추출하고 키워드 패턴을 이용하여 각 증상 및 원인에 대한 타입으로 분류 한 뒤 증상과 원인을 상호 연결한다. 증상과 원인에 대한 타입은 4절에서 보다 상세히 설명한다.

두 번째 과정은 메시지 포스팅에서 판별된 증상 및 원인들의 상관관계를 분석하는 것이다. 상관관계 분석을 위해 PLSA(Probabilistic Latent Semantic Analysis) 방법을 이용하였다[12]. 즉, 특정 원인이 발견된 토론 쓰레드에서 특정 증상이 함께 발견되면 이는 둘 간의 상관관계가 있는 것으로 가정하고, 이 방법으로 모든 토론 쓰레드에서 증상과 원인 타입이 동시에 존재하는 수를 카운트한다. 증상과 원인의 상관관계가 결정되면 '원인' 온톨로지의 'correlatedWith' 관계를 통해 서로 연결되어 표현된다.

4. 가설 검증을 위한 실험

본 절에서는 개발된 프레임워크를 기반으로 앞서 설명한 가설들을 증명하기 위한 실험에 대해 설명한다. 호환성 문제에서 발생할 수 있는 증상과 원인의 핵심 타입을 추출하기 위해 소스포지의 '소프트웨어 개발>사용자 인터페이스'에 등록된 250개 자산에서 3,229 메시지를 분석하였다. 앞서 간략히 설명하였듯이 소스포지에서 가장 활발한 논의가 이루어지고 있는 재사용 문제가 '호환성' 문제이고 개발자들이 가장 많은 코멘트와 의견을 교류하는 카테고리가 '사용자 인터페이스'이다. 표 1은 3.2절에서 설명한 키워드 패턴들을 이용하여 도출된 호환성 문제의 핵심적 증상과 원인 타입을 보여준다.

키워드 패턴 분석과 포럼 데이터에서의 키워드 매칭(Matching)을 위해 본 연구에서는 자연어 처리(Natural language processing) 기반의 데이터 검색 및 추출 기능을 제공하는 Apache Lucene[7] 라이브러리를 사용하였다. 추출된 키워드 패턴이 증상과 원인 타입을 얼마나

표 1 호환성 문제의 핵심 증상 및 원인 타입

증상(Symptoms)	원인(Causes)
s0: 예상치 못하거나 부정확한 결과를 보임	c0: 비호환적 운영체제
s1: 사용자 인터페이스의 영역을 벗어남	c1: 비호환적 하드웨어
s2: 다른 버전들과 작동하지 않음	c2: 비호환적 라이브러리 및 플러그 인
s3: 시스템이 멈추거나 실패함	c3: 비호환적 데이터 및 제어
s4: 특정 기능들을 사용할 수 없음	c4: 비호환적 버전
s5: 네트워크와 연결할 수 없음	c5: 비호환적 표준
s6: 너무 느린 응답 속도를 보임	c6: 잘못 사용되어진 표준
s7: 특정 디바이스를 사용할 수 없음	c7: 비호환적 사용자 인터페이스 디자인 및 모델

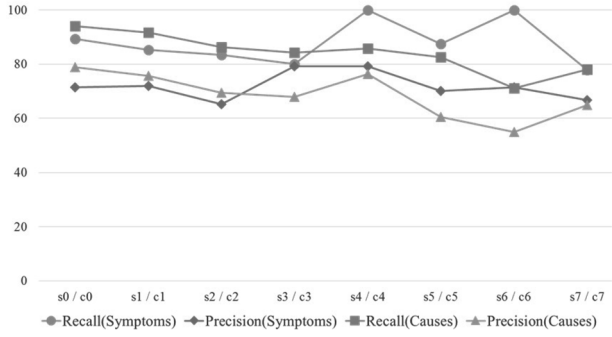

그림 5 키워드 패턴의 정확도 및 재현율

잘 식별하는지를 검증하기 위해 정확도(Precision)와 재현율을 측정하였다. 그림 5에서 보는 바와 같이 평균 재현율은 약 85%이고 평균 정확도는 약 70%를 보여준다. 이는 타 연구에서 진행된 재사용 지식 추천을 위한 실험 및 구글 등의 일반적인 검색 엔진들과 비교하여 충분한 수준이라고 판단한다[5,13].

이러한 기본적 실험을 기반으로 두 가지 가설의 실제 검증을 위해 FLOSS를 이용하여 새로운 소스포지 데이터 셋을 추출하였다. 10,378개의 문제 포스팅과 29,859개의 답변 포스팅을 추출하였고, 개발된 프레임워크를 기반으로 증상과 원인을 포함하고 있는 메시지 쓰레드들을 추출하여 상관관계를 분석하였다. 그림 6은 상관관계 분석의 전체적인 결과를 보여준다.

이렇게 분석된 상관관계들의 유의성 판단을 위해 통계학적 방법인 이원 분산 분석(Two-way Anova)을 수행하였다. 이원 분산 분석을 위해서는 SAS[8] 도구를 활용하였다. 각 증상과 원인의 조합에 대한 유의 값(P-value)을 계산하고 '증상과 원인의 수(독립변수들)는 상관관계의 수(종속변수)에 영향을 끼치지 않는다'는 귀무가설(Null hypothesis)을 세우고 유의 수준을 가장

7) http://lucene.apache.org/core/

8) http://www.sas.com/

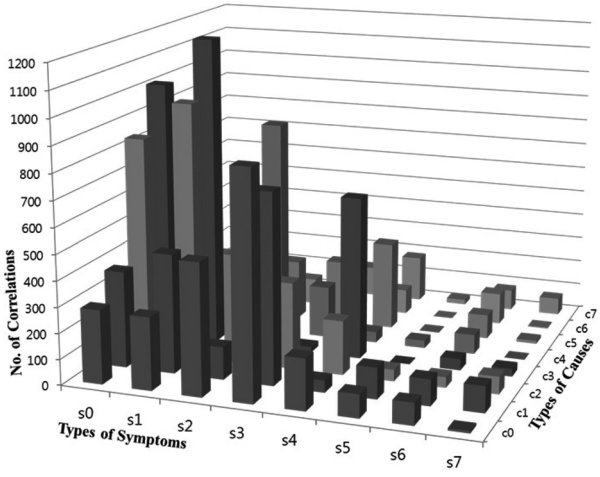

그림 6 증상과 원인의 상관관계 분석

표 2 이원 분산 분석의 부분적 결과

종속변수	독립변수	독립변수	F-value	P-value
c0과 증상들 간의 상관관계	c0	s0	15.13	0.006
		s1	7.35	0.032
		s2	12.18	0.010
		s3	66.95	0.0001
		s4	8.19	0.015
		s5	10.24	0.047
		s6	5.78	0.024
		s7	6.11	0.089
⋮	⋮	⋮	⋮	⋮
c4와 증상들 간의 상관관계	c4	s0	13.62	0.023
		s1	8.79	0.038
		s2	11.89	0.009
		s3	9.91	0.041
		s4	2.94	0.129
		s5	3.01	0.133
		s6	15.69	0.039
		s7	3.19	0.134
⋮	⋮	⋮	⋮	⋮

널리 쓰이는 수준인 '0.05'로 설정하였다. 표 2는 이원 분산 분석의 부분적인 결과를 보여준다. 표에서 '비호환적 운영체제' (c0)와 '예상치 못하거나 부정확한 결과를 보임' (s0)은 유의 값이 '0.006'으로 유의수준 보다 현저히 작으므로 유의하다고 판단할 수 있고, '비호환적 운영체제' (c0)와 '특정 디바이스를 사용할 수 없음' (s7)은 유의 값이 '0.089'로 유의수준 보다 크므로 유의하지 않다고 판단할 수 있다. 그림 7은 이원 분산 분석의 전체적인 결과를 보여준다.

상관관계가 있는 증상과 원인에 대한 메시지를 교환하는 일반적 트렌드를 이해하기 위해 시간적 흐름에 따른 증상과 원인을 위한 평균 인터렉션(메시지 교환)

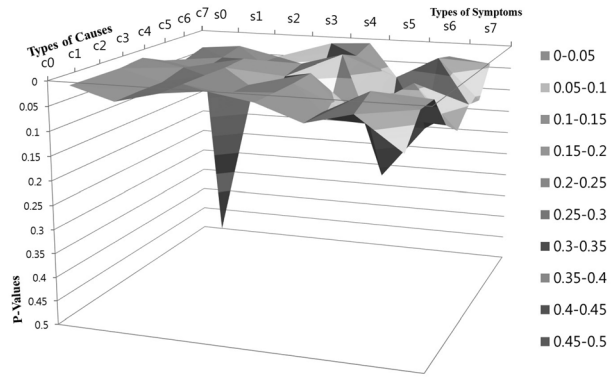

그림 7 재사용 문제 증상과 원인 상관관계의 유의성 분석 결과

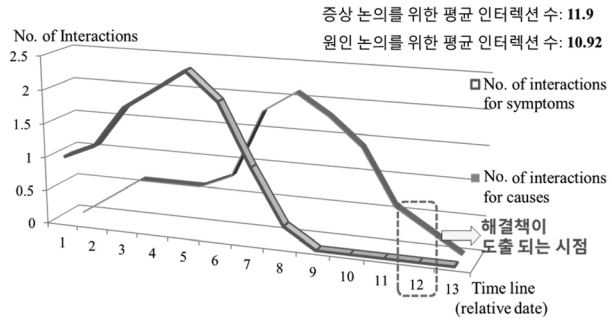

그림 8 증상과 원인 논의를 위한 평균적 인터렉션 패턴

수를 계산하였다. 그림 8에서 x축은 문제가 처음으로 포스팅 된 날짜를 기준으로 계산된 상대적인 날짜를 나타낸다. 그림에서 보는 바와 같이 개발자들은 평균 5일 정도 동안 증상을 논의하기 위한 인터렉션을 수행한다. 그리고 증상의 원인에 대해서는 상대시간 7과 9 사이에서 2~3일 간 활발히 논의되고, 해결책은 가장 활발한 원인의 논의가 끝나고 2~3일 후에 최종적으로 도출된다(상대시간 11~13일). 인터렉션 패턴 분석을 기반으로 본 연구에서는 OSS에서 개발자들 간의 메시지 교환에는 다음과 같은 일반적인 절차가 있음을 확인하였다: 1) 개발자가 토론 포럼에 증상을 포스팅한다, 2) 같거나 유사한 증상을 겪었던 다른 개발자들이 논의에 참여하고 다른 증상들 또한 포스팅 된다, 3) 공통적인 증상들이 식별된다, 4) 가능성 있는 증상의 원인이 논의된다, 5) 잠재적인 해결책들이 제안된다. 이러한 인터랙션 절차는 가설 1이 사실이라는 것을 증명한다. 우리는 이러한 인터렉션 패턴을 '재사용 관련 문제 해결 패턴'이라고 정의한다.

그림 9는 서로 상관관계가 있는 증상과 원인을 위한 인터렉션 패턴을 보여주는데, 이는 그림 8에서 보여주는 평균적인 인터렉션 패턴과 유사한 형태를 가진다. 그림 10은 서로 상관관계가 없는 증상과 원인을 위한

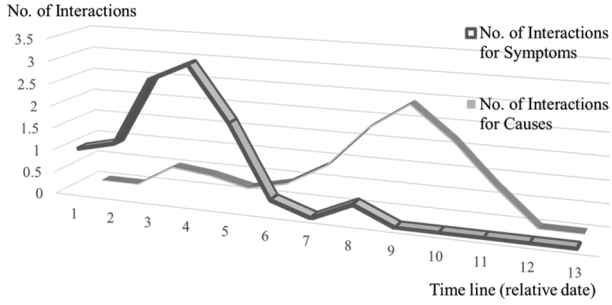

그림 9 상관관계가 있는 증상과 원인의 인터렉션 패턴

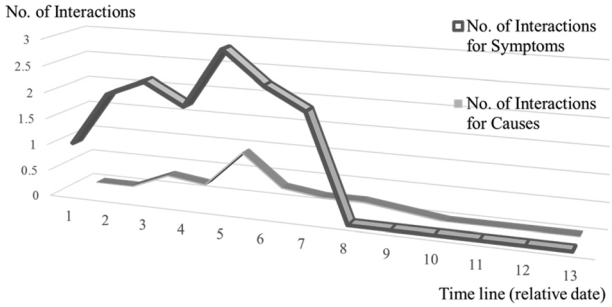

그림 10 상관관계가 없는 증상과 원인의 인터렉션 패턴

인터렉션 패턴을 보여주며 이러한 경우는 대부분이 개발자의 실수로 인한 사소한 문제로서, 쉽게 해결 가능한 문제이므로 인터렉션이 빈번하지 않다는 것을 확인할 수 있었다. 67.9%의 논의가 상관관계가 있는 증상과 원인을 위한 논의로 파악되었고 이는 대부분의 재사용 관련 문제들이 사소하지 않고, 이 문제 해결을 위해서는 개발자간에 충분한 논의가 필요하다는 것을 뜻한다.

본 연구에서는 모든 토론 쓰레드에서 실질적인 원인을 파악할 수 있는 키워드 패턴의 재현율을 계산하여 보았다. 각 토론 쓰레드에서 증상과 원인, 해결책을 가지는 모든 메시지들을 시간의 흐름에 따라 정렬하였고 각 시간의 시점을 기준으로 재현율을 계산하였다. 그림 11에서 삼각형으로 표시된 빨간색 선이 재현율의 결과 값을 보여주는데 상대 시간 6~7일 사이가 재현율이 가장 급격히 증가 되고 그 이후로는 평준화 되는 것을 볼 수 있다. 이는 이 시점에 개발자들이 실질적인 원인을 이해한다는 것을 의미한다(상대 시간 6일에 재현율이 약 80% 임). 상대시간 11.5일은 원인에 대한 논의는 점점 사라지고 포럼에서 해결책이 나타나기 시작하는 시점이다. 따라서 문제의 증상에 대한 핵심적인 원인을 미리 이해할 수 있도록 지원해 준다면 해결책을 알아내는 시간을 대폭 줄일 수 있다. 그림 11의 예제에서는 상대시간 6일의 시점에서 가능성 있는 원인을 추천해 줌으로써 해결책을 찾는 시간을

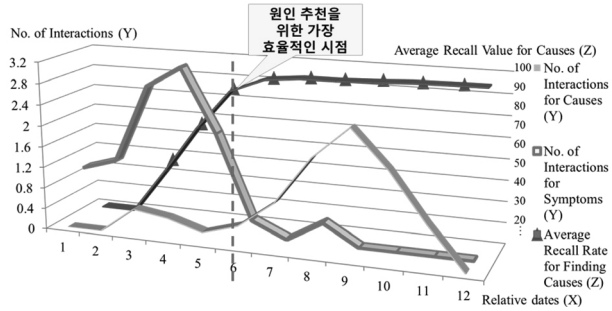

그림 11 문제의 원인 추천을 위한 효율적 시점 분석

최대 5.5일 줄일 수 있을 것이다. 이 실험을 통해 가설 2가 사실이라는 것을 증명할 수 있었다.

5. 결 론

본 논문은 OSS환경에서 재사용 관련 문제들을 해결하기 위한 지식들을 추출, 분석 및 저장하는 개념적인 프레임워크에 관한 것이다. 사전 조사를 통해 두 가지 가설을 세웠고 개발된 프레임워크를 기반으로 소스포지 데이터를 활용하여 몇 가지 실험을 수행하였다. 실험을 통하여 재사용 관련 문제의 증상과 원인 간에는 상관관계가 존재하고 이중 60%는 통계학적으로 유의하다는 것과 OSS에서 재사용 관련 문제 해결을 위한 일반적인 패턴이 있다는 것을 확인하였다. 이러한 패턴을 이용하면 문제의 해결책을 알아내는데 소요되는 시간을 크게는 48% 줄일 수 있다는 것을 확인하였다(11.5일에서 6.0일로 감소). 이 실험들을 통해 본 연구에서 세운 두 가설이 사실이라고 증명되었고 본 프레임워크의 확장을 통해 개발자들로 하여금 큰 노력 없이 유용한 재사용 문제 해결 지식을 제공할 수 있을 것으로 예상한다.

현재 본 연구를 기반으로 OSS의 다양한 카테고리들에 대한 실험을 수행하고 있고 지식 추론 기능들을 개발하여 프레임워크를 확장하는 연구를 수행중이다. 또한 베이지언 네트워크(Bayesian network) 기반의 확률 모델을 활용하여 상관관계를 분석하는 실험을 수행중이고 링크드 데이터 기반의 지식 베이스를 설계 중에 있다.

참고문헌

[1] C. W. Kruger, “Software Reuse”, ACM Computing Surveys, vol. 24, no. 2, pp. 131-184, June 1992.

[2] M. Ali and et al., “Toward an Engineering Discipline of Software Reuse”, IEEE Software, vol. 15, no. 5, pp. 22-31, 1999.

[3] A. W. Brown and G. Booch, “Reusing Open-Source Software and Practices: The Impact of Open-Source on Commercial Vendors”, Lecture Notes in Computer Science, vol. 2319, Springer, 2002.
[4] M. W. Godfrey, “Evolution in Open Source Software: A case study”, Proceedings of International Conference on Software Maintenance, pp. 131-142, October 2000.
[5] Y. L. Hing, “Software Engineering Knowledge for Software Reuse”, Proceedings of the 6th International Workshop on Computer-Aided Software Engineering, IEEE, pp. 236-269, July 1993.
[6] K. Timo and et al., “Improving Knowledge Management in Software Reuse Process”, PROFES, LNCS 2188, vol 2188, pp. 141-152, September 2001.
[7] K. Iman and et al., “A Linked Data Platform for Mining Software Repositories”, Proceedings of International Workshop on Mining Software Repositories, pp. 32-35, June 2012.
[8] I. Aftab and et al., “LD2SD: Linked Data Driven Software Development”, Proceedings of the 21st International Conference on Software Engineering and Knowledge Engineering, pp. 240-245, 2009.
[9] D. Danica and B. Kalina, “Enhanced Semantic Access to Software Artefacts”, Proceedings of the 4th International Workshop on Semantic Web Enabled Software Engineering, 2008.
[10] C. Kiefer and et al. “Mining Software Repositories with iSPARQL and Software Evolution Ontology”, Proceedings of International Workshop on Mining Software Repositories, 2007.
[11] H. M. Wallach, “Topic modeling; beyond bag of words”, Proceedings of In International Conference on Machine Learning, pp. 977-984, 2006.
[12] H. Thomas, “Probabilistic latent semantic analysis”, Proceedings of the 15th conference on Uncertainty in artificial intelligence, pp. 289-296, 1999.
[13] S. M. Shafi and Rafiq A. Rather, “Precision and Recall of Five Search Engines for Retrieval of Scholarly Information in the Field of Biotechnology”, Journal of Webology, vol. 2, no. 2, August 2005.

약 력

구 형 민

2004 금오공과대학교 컴퓨터공학과 졸업(학사)
2007 한국과학기술원 졸업(석사)
2007 한국전자통신연구원 위촉연구원
2007~현재 한국과학기술원 전산학과 박사과정
관심분야 : 소프트웨어 재사용, 지식 기반 소프트웨어 공학, 사용자 중심 소프트웨어
E-mail : hmkoo@kaist.ac.kr

고 인 영

1990 서강대학교 전산학과 졸업(학사)
1992 서강대학교 전산학과 졸업(석사)
2003 미국 Univ. of Southern California졸업 (박사)
1993~1996 공군사관학교교관(전임강사)
2003 미국 USC Information Science Institute(ISI) Postdoctoral Research Associate
2004~2009 한국정보통신대학교 공학부 조교수
2004 미국 카네기멜론대학 방문교수
2009~현재 한국과학기술원 전산학과 부교수
관심분야 : 소프트웨어 공학, 웹 공학
E-mail : iko@kaist.ac.kr