버그 정정 관리에 데이터 마이닝 적용

서울시립대학교 ▌양근석 · 장 도 · 이병정*

1. 서 론

소프트웨어 프로젝트와 시스템의 규모 및 복잡성은 지속적으로 증가하고 있는 실정이다. 심지어 소프트웨어 결함이 나아가 결함 저장소(bug repository)의 형성 과정으로까지 이어지고 있다. 규모가 큰 공개 소스 프로젝트(open source project)에 저장된 각각의 소프트웨어 결함들은 결함을 해결하도록 배정된 개발자에 의해서 수정이 되어져야 하며, 이러한 과정을 결함 배정(bug triage)이라고 한다.

매일 수 많은 소프트웨어 결함 리포터들이 보고 되고 있고, 개발자의 업무량도 증가하고 있다. 게다가 만약 소프트웨어 결함 리포트가 적절한 개발자에게 배정되어지지 않는다면, 반드시 그 결함을 해결할 수 있는 적절한 개발자에게 재 배정(re-assigned)되어져야 한다. 많은 소프트웨어 결함 재 배정은 소프트웨어 유지관리를 하기 위한 시간을 증가시키고, 결함을 고칠 수 있는 확률을 감소시킬 수 있기 때문이다.

이러한 문제점들을 해결하기 위해 많은 관련 연구자들이 어떻게 하면 새롭게 보고 되어진 소프트웨어 결함을 고칠 수 있도록 적절한 개발자를 추천하기 위한 결함배정 연구가 진행되고 있는 실정이다.

본고에서는 이러한 연구들을 기반으로 적절한 개발자를 추천할 수 있는 복합 결함 배정 알고리즘을 제안하고자 한다. 새롭게 보고 되어진 소프트웨어 결함 리포트와 유사한 결함 리포트를 찾기 위해 통계적 모델인 유연한 유니그램 모델(smoothed Unigram Model)을 사용하였다.

따라서 본고 연구를 통해 제안하고 싶은 알고리즘은 확률 모델과 경험 모델 결합시킨 복합 결함 배정 알고리즘과, 다중 사용자 모델이다. 확률 모델은 소셜 네트워크(social network)를 기반으로 각각의 개발 후보자들이 주어진 결함을 고칠 수 있는 확률을 분석한 것이며, 나아가 확률 모델의 효율성을 향상시키기 위해 다시 열려진(re-opened) 결함을 새로운 요소로 추가하였다. 경험 모델에서는 새로운 결함에 대해 고칠 수 있는 충분한 능력을 가진 개발자인지를 평가하기 위하여, 결함을 고친 횟수와 결함을 고친 시간을 변수로 보았다.

이렇게 고쳐진 결함들에 대해서는 후보자들의 경험을 확인하기 위해 새로운 요소로 사용한 활동요소(activity factor)를 소개하였다. 다중 사용자 모델에서는 주어진 결함을 수정한 횟수(commit), 개선, 결함에 대한 의견을 남긴 코멘트(comment)를 고려하여 기존 소셜 네트워크를 향상시켰다.

즉 제안한 알고리즘을 사용하면서, 새로운 결함을 고칠 수 있는 가능성에 따라 개발 후보자들의 순위가 정해지기 때문이다.

본고는 다음과 같이 구성된다. 제2장에서는 관련 연구로 최근 진행되고 있는 결함 배정에 대한 동향 및 기술을 다루고, 제3장에서는 결함 정정 관리의 데이터 마이닝 기법에 관한 배경지식을 다루고, 제4장에서는 제안한 복합 결함 배정 알고리즘에 대해 심층적으로 분석하고, 제5장에서는 분석된 연구 내용을 바탕으로 심층토론을 거쳐, 제6장에서는 결론을 맺는다.

2. 관련 연구

주어진 새로운 결함을 고칠 수 있는 적절한 개발자를 어떻게 추천할지에 대해 많은 국내외 연구자들이 관심을 갖고 있다. 2.1절에서는 유사한 결함 리포트 검색에 대해 설명한다. 2.2절에서는 중복된 결함 검색과 분류에 대해 설명하고, 2.3절에서는 자동화 결함 배정, 2.4절에서는 결함 리포트 개발자 추천에 대해 설명한다.

2.1 유사한 결함 리포트 검색

소프트웨어 유지관리에서 유사한 결함 리포트를 찾는 방법은 매우 중요하다.

* 종신회원

† 본 연구는 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No.2012-0007149).

연구 [1-2]는 IR(Information Retrieval) 기반으로 유사한 결함 리포트를 찾는 방법을 제안했다.

연구 [3-4]는 벡터 공간 모델(Vector Space Model)을 사용하여 유사한 결함 리포트를 찾았다. 이후 많은 연구자들이 벡터 공간 모델을 사용하였지만, 유연한 유니그램 모델(smoothed Unigram Model) 방법이 벡터 공간 모델보다 유사한 결함 리포트를 찾는 데 더 효율적이라고 증명하였다[5].

2.2 중복된 결함 검색과 분류

대부분의 중복 결함 검색 기술들은 중복된 결함 리포트를 찾기 위해 자연어(natural language)의 정보만을 이용하였다.

연구 [5]는 자연어 정보 뿐만 아니라, 실행 정보까지 사용하는 새로운 방법을 도입하였다. 새로운 결함이 도착했을 때, 자연어 정보와 실행 정보를 사용하여 저장소에 있는 유사한 결함 리포트들을 검사하고 그 중 중복된 결함 리포트를 검색하였다.

연구 [6]은 개발자의 시간을 절약할 수 있게 결함 리포트 내에 있는 표면 요소(surface feature), TF-IDF를 통한 단어들의 의미분석, 그래프를 사용하여 중복 결함 리포트를 자동적으로 분류하는 기법을 보여주었다.

2.3 자동화 결함 배정

결함 리포트를 적절한 개발자에게 배정하는 기법은 매우 중요하다. 실제 결함을 고치기 위해 가장 중요한 역할을 하는 것이 배정자의 역할이라고 보여 주기 때문이다[7].

2.4 결함 리포트 개발자 추천

연구 [8]은 공개 결함 저장소로부터 새로운 결함을 고칠 수 있는 적절한 개발자를 추천하기 위해 기계 학습(machine learning) 알고리즘을 이용하여 제안하였다.

연구 [9]는 개발자 소스 코드에서 단어를 추출한 경험 모델을 제안하였으며, 추출 되어진 단어와 결함 리포트에 있는 단어를 비교 하여 새로운 결함을 고칠 수 있는 적절한 개발자를 추천하였다.

비슷한 맥락에서 연구 [10]은 비용인식(Costaware) 배정 알고리즘을 제안하였다. 최근에는 소수의 연구자들이 소셜 네트워크와 같은 자동 결함 배정의 효율성을 향상시킬 수 있는 다양한 기술들을 제안하였다.

연구 [11]은 DREX(Developer Recommendation with K-Nearest-Neighbor Search and Expertise Ranking)를 제안하였는데, 이 알고리즘은 새롭게 제출되어진 결함과 유사한 결함 리포트를 찾아내 그 결함 리포트에서 활동했던 모든 개발자를 추출하였다. 나아가 개발자 추천을 위해 소셜 네트워크 기법을 포함한 다양한 방법들을 비교 하였다.

연구 [12]는 자동 결함 배정 방법을 제안하였는데, 이 방법은 결함 리포트를 고칠 수 있는 개발자를 추천하기 위해 소셜 네트워크 기법을 이용하였고, DRA(Developer Ranking Algorithm)를 통해 개발자 우선순위를 정하였다.

연구 [13]은 개발자의 결함 리포트와 소스 코드에 공헌(contribute)한 것을 비교하여 자동적으로 개발자를 추천해 주는 기법을 보여주었다.

연구 [14]도 소셜 네트워크 기법을 이용하여 개발자 우선순위를 지정하는 방법을 제안하였다. 국내에서도 소프트웨어 결함관련 연구가 진행 중에 있다. 결함 개발자 할당 분야에서 결함 리포트로부터 개발자 재할당 정보를 추출하여 기계 학습을 적용한 개발자 예측 모델을 제시하였다[15].

연구 [16]은 결함을 배정받은 개발자가 다른 개발자에게 넘기는(tossing) 현상이 발생하면 결함을 고칠 수 있는 시간이 늘어나고, 고칠 수 있는 확률이 줄어든다는 것을 보여주었다.

3. 버그 정정 관리에 데이터 마이닝 적용 기법

3.1 결함 리포트와 중복

최근 증가하고 있는 소프트웨어 시스템은 소프트웨어 유지관리 활동을 돕기 위해 결함 리포트를 관리하는 결함 추적 시스템을 사용한다. 결함 리포트는 소스 코드 파일에 존재하는 에러(error)를 다양한 필드로 구조적으로 기록한 것이다. 이러한 필드는 요약, 정의, 중요도, 보고자, 만들어진 시간, 수정된 시간, 상태 등을 포함하고 있다. 실제 결함 리포트는 그림 1과 같다.

많은 결함 리포트의 오분류, 중복으로 인한 불필요한 노력이 증가하고 있다. 중복 결함 리포트가 증가하면 결함 분석 시간과 노력도 증가시키는 결과를 초래한다. 또 최근에 결함을 포함하고 있지 않은 리포트가 결함 리포트로 분류가 되는 사례가 증가하고 있다. 이러한 오분류로 인한 결함 예측 결과에 부정적인 영향을 미친다. 따라서 결함 중복 리포트와 오분류를 정확하게 탐지할 수 있는 연구가 필요하다고 하겠다.

3.2 결함 생명 주기

소프트웨어를 정정하기 위한 유지보수 활동은 결함 리포트를 활용하고, 결함 생명 주기는 그림 2와 같다.

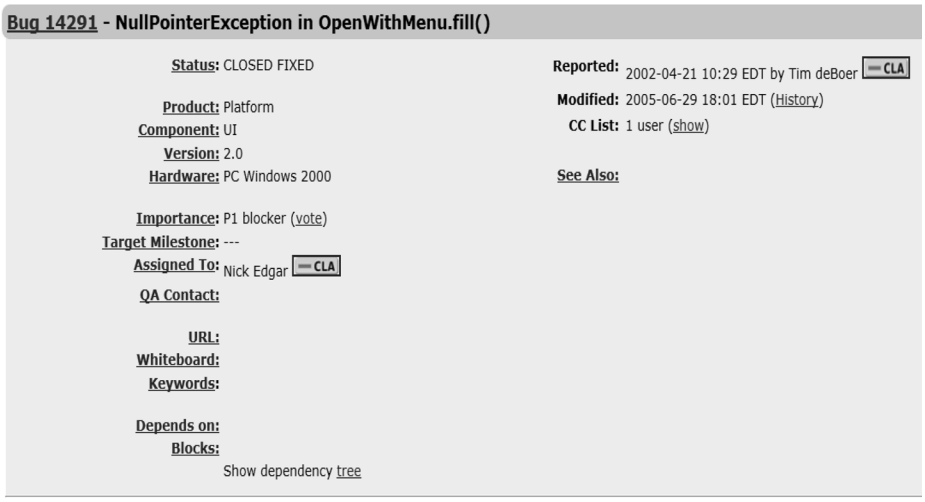

그림 1 실제 결함 리포트(Eclipse #14291)

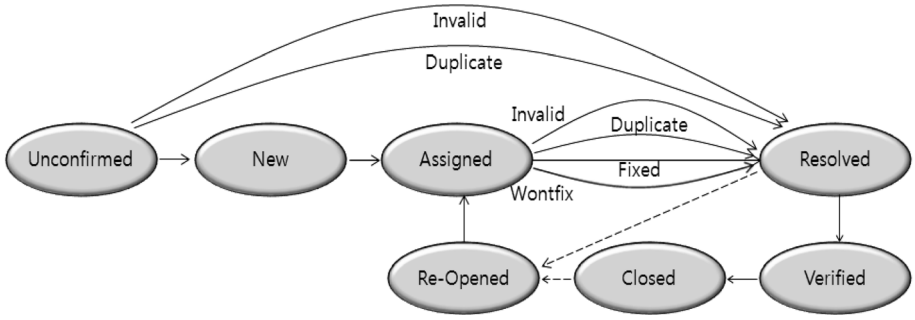

그림 2 결함 생명 주기

대규모 소프트웨어 결함 추적 시스템에 하루에 수 많은 결함 리포트가 매일 제출되는데, 결함 리포트를 수 작업으로 관리하기에는 개발자들의 노력이 많이 필요하다. 이를 위해 자동화를 지원하는 효과적인 소프트웨어 결함 관리가 필요하다.

새로운 결함이 결함 저장소에 도착했을 때 상태는 “Unconfirmed”로 변환이 된다. 결함 저장소 시스템의 배정자가 해당 결함을 개발자에게 배정하기 위해 결함을 열었을 때 상태가 “New”로 변경이 되고, 이어 해당 결함은 개발자에게 정확하게 전달이 되면 상태가 “Assigned”로 변경이 된다. 개발자가 해당 결함을 잘 해결하였다면 상태가 “Resolved”로 변경이 된다. 해당 결함 시스템이 상태가 “Resolved”인 결함들에 대해 확인을 하고 상태를 “Verified”에서 “Closed”로 변환시킨다.

하지만 결함이 제대로 고쳐지지 않았을 때 해당 결함을 다시 열어야 하므로 상태가 “Re-opened”로 변환이 된다. 상기 언급하였지만 새로 주어진 결함에 대해 적절한 개발자에게 배정이 되어지지 않는다면 소프트웨어의 성공적인 유지관리로 이어지기는 어렵다. 또 처음 결함이 도착하였을 때 기존 저장소에 있는 결함들과 비교하여 중복된 결함을 정확하게 찾는 연구도 필요하다.

3.3 결함 배정

공개 소스 프로젝트에는 하루에 많은 수의 결함 리포트들이 보고되고 있다. Eclipse 프로젝트에서 사용하는 결함 관리 시스템인 Mozilla의 예를 들면, 하루에 약 300개의 결함 리포트들이 전송 되고 있다[8].

이러한 결함 리포트들을 개발자가 수작업을 통해 유지보수하기에는 많은 노력과 비용, 시간이 든다. 따라서 결함 리포트가 적절한 개발자에게 전달이 되어 가능한 빨리 결함이 수정이 된다면 성공적인 소프트웨어 지름길이 될 수 있을 것이다.

3.4 소셜 네트워크

공개 소프트웨어 프로젝트에서 결함 배정과 결함을 고치는 작업은 개발자들간의 상호협력이 필요하다. 다시 말해 비슷한 관심사를 기반으로 개발자들이 소셜 커뮤니티에서 활동하기 때문이다.

소셜 네트워크는 결함 해결을 논의 하고 있는 개발자들간의 관계를 분석하기 위해 사용된다.

본고에서는 소셜 네트워크 기반으로 확률 모델을 만들었고 더 나아가 소셜 네트워크를 향상시킨 다중 개발자 모델을 제안하였다.

4. 자동 결함 배정을 위한 방법

새로운 결함 리포트가 도착하면 쿼리(query)로 사용하며, 이 쿼리를 유연한 유니그램 모델에 적용하여 결함 저장소에 있는 결함 리포트 내 단어들에 대해 확률 벡터를 구하고, KL-divergence를 통해 결함 저장소에 있는 유사한 결함 리포트를 찾는다. 찾아진 유사한 결함 리포트의 목록으로부터 개발자, 배정자, 코멘터(commenter), 고치기 위해 걸린 시간, 다시 열려진 결함의 수 등을 추출해 내며, 추출 되어진 개발자들을 후보자로 간주한다.

확률 모델은 소셜 네트워크 기법을 사용하여 새로운 결함을 위한 개발자 후보들의 고칠 수 있는 확률을 구할 수 있도록 만들어진 모델이다. 즉 경험 모델로부터 추출 되어진 요소들에 따라 후보 개발자들의 결함을 고친 횟수, 활동요소, 고쳐진 비용 등을 추출하였으며, 마지막으로 이 두 모델을 결합하여 복합 결함 배정 알고리즘을 만들었다. 이러한 모델들을 세부적으로 적용 시킨 절차는 다음과 같다.

4.1 유연한 유니그램을 통한 유사한 결함 리포트 검색

새로운 결함이 나왔을 때 유사한 결함 리포트를 찾기 전에, 미리 전처리(preprocess)를 통하여 결함 리포트 내에 있는 문장들을 단어로 토큰(tokenization)화 하고, 필요 없는 정지단어(stop words)를 제거하며, 과거 동사 등을 일반 동사로 변환하는 스템(stemming)을 통하여, 통계적 모델인 유연한 유니그램 모델에 기반한

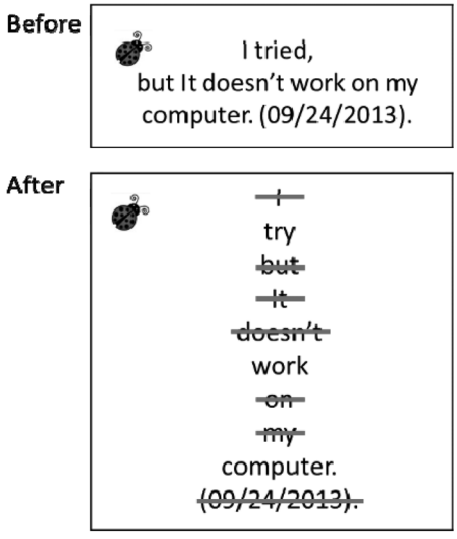

그림 3 결함 리포트에 대한 전처리 과정

유사 결함 리포트들을 찾는다. 전처리 과정은 그림 3과 같다.

과거에는 많은 연구자들이 유사한 결함 리포트를 찾기 위해 벡터 공간 모델(VSM)을 사용했지만, 과거 연구진들이 유연한 유니그램 모델이 벡터 공간 모델보다 더 좋다는 것을 증명하였고[17], 이를 근거로 본고에서도 유연한 유니그램 모델을 사용하였다.

각 문서를 대표를 하는 유니그램 모델 [18]은 | V | 차원 확률 벡터이다. 만약 해당 문서에 특정단어가 발생하지 않으면, 확률 벡터가 0이 되는 것을 방지하기 위해 오른쪽 부분 컬렉션(collection) 모델을 사용하여 유연하게 만들었고, 수식 (1)과 같이 정의가 된다.

$$P_{suni}(\omega|\overrightarrow{\omega_k}) = (1-\mu)\frac{\omega_k(n)}{\sum_{n=1}^{|R_k|}\omega_k(n)} + \mu\frac{\sum_{l=1}^{K}\omega_l(n)}{\sum_{l=1(l\neq k)}^{K}\sum_{n=1}^{|R_l|}\omega_l(n)} \quad (1)$$

$\omega_k(n)$: 문서 k에 있는 n^{th} 단어 발생 주기
K : 전체 문서의 수
μ : 가중치

유연한 유니그램을 통하여 결함 리포트를 확률 벡터로 변환을 하였으며, KL-divergence를 이용하여 유사한 결함 리포트를 찾는다. KL-divergence를 이용한 유사도 측정 수식은 (2)와 같다.

$$\begin{aligned}\mathrm{sim}(\overrightarrow{\omega_q},\overrightarrow{\omega_k}) &= -\mathrm{KL}\left(P_{suni}(\omega|\overrightarrow{\omega_q}), P_{suni}(\omega|\overrightarrow{\omega_k})\right) \\ &= -\sum_{i}^{i=|\omega_i|} P_{suni}(\omega_i|\overrightarrow{\omega_q}) * \log\frac{P_{suni}(\omega_i|\overrightarrow{\omega_q})}{P_{suni}(\omega_i|\overrightarrow{\omega_k})} \quad (2)\end{aligned}$$

$P_{suni}(\omega|\overrightarrow{\omega_q})$: 저장소에 있는 결함 리포트의 확률 벡터
$P_{suni}(\omega|\overrightarrow{\omega_k})$: 새로운 결함 리포트의 확률 벡터

계산 되어진 모든 결함 리포트에 대해서는 유사도를 이용하지 않고, 본고에서 정한 임계치(θ = -0.4) 이상의 유사도를 가진 결함 리포트만 이용하여 개발자와 결함을 고치는데 걸린 시간 등의 요소들을 추출하여 확률 모델과 경험 모델을 만들었다.

4.2 확률 모델

확률 모델을 사용하기 위해 소셜 네트워크 기법을 이용하였다. 이 기법은 네트워크 내에 개발자들 사이에서의 코멘트 활동(comment activity)을 조사하여, 주어진 결함을 누가 고칠 수 있는지에 대한 개발자의 확률을 알아 낼 수가 있다. 본고에서 사용된 소셜 네트워크 기법은 그림 4와 같다.

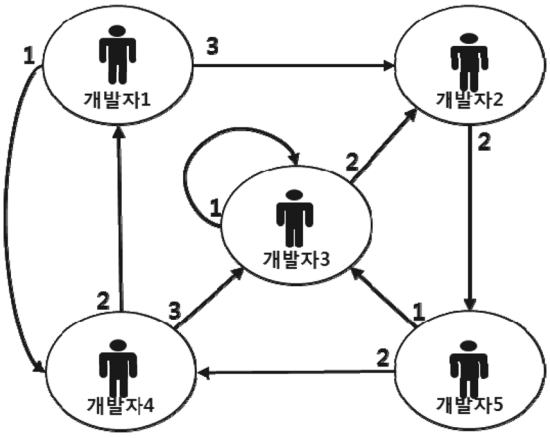

그림 4 소셜 네트워크 기법

위에서 적용된 사례는 상호간에 코멘트 활동을 하는 5명의 개발자 노드(node)가 있고, 코멘트 횟수를 알리는 엣지(edge)가 있다. 예를 들면 개발자 4가 개발자 3에게 3번의 코멘트를 하였다고 가정하여 분석을 해보니 자기 자신에게 코멘트를 하는 셀프 링크(self-link) 현상이 있었다. 즉 개발자 3이 자신에게 1번 코멘트 하였다. 하지만 정확한 개발자의 확률을 구하기 위해 이러한 셀프 링크의 상황은 배제하였다. 새로운 결함에 대해 고칠 수 있는 각각의 개발자의 확률을 구하기 위해 아래와 같이 확률 모델 수식은 (3)과 같이 제시한다.

$$\mathrm{P}(b_{new}|dev_i) = \frac{nc_i * o_i/(nreb_i + 1)}{M_p} \qquad (3)$$

nc_i : 개발자 dev_i에게 코멘트 한 횟수
o_i : 개발자 dev_i를 위한 방출된 선(emitted link)의 수
$nreb_i$: 다시 열려진 결함들의 수
M_p : 공식의 일반화

상기 공식에서 개발자 dev_i를 위해 다시 열려진 결함이 없을 때 전체가 0이 되는 특이한 경우(special case)를 방지하기 위해 1을 추가하였다.

이를 다시 정리하면 코멘트의 수와 관련된 개발자들의 수가 많을수록 주어진 결함을 고칠 수 있는 확률이 높아지고, 다시 열려진 결함이 많을수록 주어진 결함을 고칠 수 있는 확률이 낮아질 수 있다는 것이다

4.3 경험 모델

과거에 많은 연구자들은 결함을 고친 횟수와 고친 비용이 경험 모델을 만드는데 중요한 영향을 줄 것이라고 믿었다. 하지만 본고에서는 새로운 컨셉인 활동 요소(activity factor)를 사용하여 개발자가 결함을 더 잘 고친 경험을 표현하고 있다.

일반적으로 결함 저장소에서 배정자(traiger)는 관련된 결함을 고치기 위해 개발자를 배정할 수 있다. 이러한 개발자를 양수인(assignee), 수정자(fixer)이라고 부른다. 본고에서는 결함 저장소에서 고치기 위해 배정된 시간을 활동요소로 정의하였다. 개발자를 위해 고친 경험을 표현한 수식은 (4)와 같다.

$$\mathrm{E}(dev_i) = \frac{nb_i * \frac{a_i}{c_i}}{M_e} \qquad (4)$$

nb_i : 개발자 dev_i에 의해서 고쳐진 결함 리포트의 수
a_i : 개발자 dev_i를 위한 활동 요소
c_i : 개발자 dev_i가 결함을 고치기 위한 비용
M_e : 경험 모델을 일반화

즉 경험 모델을 한마디로 정리하면, 개발자가 많은 수의 결함을 고쳤고, 결함을 고치기 위해 배정 횟수가 많으며, 결함을 고친 평균 시간이 짧다면, 주어진 결함에 대해 고칠 수 있는 경험이 많다고 할 수 있을 것이다.

4.4 다중 개발자 네트워크 모델

다중 개발자 네트워크 모델은 기존 소셜 네트워크를 향상 시킨 것으로, 네트워크 내에 있는 개발자들 간의 코멘트, 커밋(commit)한 횟수까지 고려하였다. 이 모델은 2차원 공간 모델이며, 개발자 이름과, 엣지 위에 가중 벡터(weight vector)를 가진다. 다중 개발자 네트워크 모델은 아래 그림 5와 같다.

연구 [7]은 결함을 수정하기 위해 가장 영향을 주는 속성은 소프트웨어 컴포넌트(component)라고 증명하였다. 따라서 본고에서도 더 정확한 개발자를 추천하여 순위를 정하기 위해 해당 속성을 사용하였다.

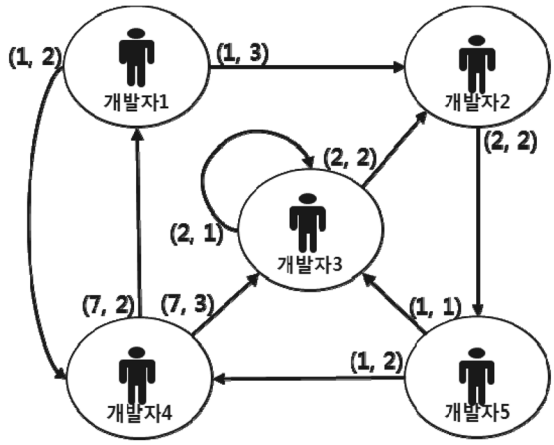

그림 5 다중 개발자 네트워크 모델

유사한 결함 리포트에 활동했던 개발자와 주어진 결함과 같은 컴포넌트에 있는 모든 개발자에 대해 교집합(intersection)을 통해 개발자를 추천하였다.

4.5 복합 결함 알고리즘

본고에서는 소셜 네트워크에 의해서 추출 되어진 후보 개발자들을 위해 새롭게 주어진 결함을 고칠 수 있는 확률 모델 뿐만 아니라, 개발자의 경험 모델까지 고려하였으며, 복합 결함 알고리즘을 만들기 위해 수식은 (5)와 같다.

$$\text{RankScore}(dev_i) = \alpha * P(b_{new}|dev_i) + (1-\alpha) * E(dev_i) \quad (5)$$

$P(b_{new}|dev_i)$는 주어진 결함에 대한 고칠 수 있는 확률을 나타내고, $E(dev_i)$는 후보 개발자를 위한 고친 경험을 나타낸다. α는 가중치로서 $0 \leq \alpha \leq 1$의 값을 갖는다.

따라서 새로운 결함이 나왔을 때 제안한 복합 결함 배정 알고리즘과 다중 개발자 네트워크를 이용하여 주어진 결함을 고칠 수 있는 적절한 개발자를 추천할 수 있는 장점이 있다.

5. 토 의

현재 연구 실정을 볼 때, 본고와 가장 관련이 깊은 연구는 연구 [8,11,12]이다. 연구 [8]은 기계 학습 기법인 SVM(Support Vector Machine)을 사용하여 새로운 결함에 대해 적합한 소수의 개발자를 추천하였다.

연구 [11]은 KNN(K-Nearest-Neighbor)을 사용하여 주어진 결함에 대해 개발자 추천과 동시에 전문성을 고려하여 개발자 우선 순위를 정하였다. 또 결함 리포트 내에 있는 단어의 빈도수를 계산하여 벡터 공간 모델(VSM)을 사용하였다. 하지만 벡터 공간 모델보다 유연한 유니그램 모델이 더 좋다는 연구 결과가 나왔다 [17]. 본 연구에서도 유사한 결함 리포트를 더 정확히 찾기 위해 유연한 유니그램 모델을 사용하였다.

연구 [12]는 소셜 네트워크를 이용하여 네트워크 내 개발자들의 상호 코멘트한 횟수를 사용하여 적합한 개발자를 추천하였다. 하지만 소셜 네트워크 기반은 네트워크에 속한 개발자들 간의 코멘트 활동만 고려한 것이므로, 코멘트가 많다고 적합한 개발자라고 단정지을 수는 없다. 본고에서는 이러한 문제점을 해결하기 위해 개발자의 코멘트 활동 뿐만 아니라, 결함을 수정한 횟수까지 고려하였다.

본고에서 JBoss와 Eclipse 같이, 규모가 큰 오픈 소스 프로젝트에 적용해 보았지만 실제 비즈니스 프로젝트에서 적합할지는 확실하지 않았다. 왜냐하면 비즈니스 프로젝트에서는 각 제품 요소들이 특정 그룹에 배정 되어 있고 결함 리포트 또한 해당 그룹에 배정되기 때문이다.

앞으로 제안한 알고리즘을 실제 비즈니스 프로젝트에 실현할 수 있도록 수정하고, 더 많은 요소들을 추가시켜 결함 배정의 성능을 향상시킬 과제가 남아 있다고 본다.

6. 결 론

본고에서는 지금까지의 여러가지 모델 중에서 주어진 결함에 대해 고칠 수 있는 적절한 개발자를 추천하는 복합 결함 배정 알고리즘을 제안하였다. 즉 새로운 결함이 나왔을 때 유연한 유니그램 모델을 통해 유사한 결함 리포트를 찾고, 다시 열려진 결함들과 활동요소들을 사용하여 확률 모델과 경험 모델을 통해 적절한 개발자를 추천과 동시에, 다중 사용자 네트워크를 이용하여 개발자를 추천하여 성공적인 소프트웨어 유지보수의 가능성을 높이는데 기여하였다.

향후 자동 결함 배정 알고리즘을 비즈니스 프로젝트에 적용할 수 있도록 보완해 나갈 예정이며, 더 많은 요소들을 사용하여 자동 결함 배정의 성능을 향상시킬 수 있도록 노력해 나갈 예정이다.

참고문헌

[1] Manning, C. D., Raghavan, P. and Schtze, H., 2008, Introduce to Information Retrieval, Cambridge University Press, New York.

[2] Ponte, J. M. and Croft, W. B., 1998, “A Language Modeling Approach to Information Retrieval”, in Proceeding of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp.275-281.

[3] Castells, P., Fernandez, M. and Vallet, D., 2007, “An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval”, IEEE Trans. Knowledge and Data Engineering, Vol.19(2), pp.261-272.

[4] Erk, K. and Pado, S., 2008, “A Structured Vector Space Model for Word Meaning in Context”, in Proceeding of the Conference on Empirical Methods in Natural Language Processing, pp.897-906.

[5] X. Wang, L. Zhang, T. Xie, J. Anvik and J. Sun, “An Approach to Detecting Duplicate Bug Reports Using Natural Language and Execution,” in Proceedings of

the 30th ACM/IEEE International Conference on Software Engineering, pp. 461-470, 2008.

[6] N. Jalbert and W. Weimer, “Automated Duplicate Detection for Bug Tracking System,” in Proceedings of the 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp.52-61, 2008.

[7] M. Ohira, A. Hassan, and N. Osawa, “The Impact of Bug Management Patterns on Bug Fixing: A Case Study of Eclipse Projects,” In Proceedings of the 28th IEEE International Conference on Software Maintenance, pp. 264-273, 2012.

[8] J. Anvik, L. Hiew, and G. C. Murphy, “Who Should Fix This Bug?,” In Proceeding of the 28th International Conference on Software Engineering(ICSE'06). IEEE CS Press, pp. 361-370, 2006.

[9] D. Matter, A. Kuhn, and O. Nierstrasz, “Assigning Bug Reports Using a Vocabulary-Based Experience model of Developers,” In Proceeding of the 6th International Working Conference on Mining Software Repositories (MSR'09). IEEE Press, pp. 131-140, 2009.

[10] J. Park, M. Lee, J. Kim, S. Hwang, S. Kim, “CosTRIAGE : A Cost-Aware Triage Algorithm for Bug Reporting System,” In Proceeding of the 25th AAAI Conference on Artificial Intelligence(AAAI'11). AAAI Press, pp. 139-144, 2011.

[11] W. Wu, W. Zhang, Y. Yang, and Q. Wang, “DREX: Developer Recommendation with K-Nearest-Neighbor Search and Expertise Ranking,” In Proceeding of the 18th Asia-Pacific Software Engineering Conference (APSEC'11). IEEE CS Press, pp. 389-396, 2011.

[12] T. Zhang, and B. Lee, “How to Recommend Appropriate Developers for Bug Fixing?,” In Proceeding of the 36th Annual IEEE International Computer Software and Application Conference(COMPSAC'12). IEEE CS Press, pp. 170-175, 2012.

[13] D. Matter, A. Kuhn and O. Nierstrasz, “Assigning Bug Reports Using a Vocabulary-Based Expertise Model of Developers,” in Proceedings of the 6th IEEE Working Conference on Mining Software Repositories, pp. 131-140, 2009.

[14] J. Xuan, H. Jiang, Z. Ren, and W. Zou, “Developer Prioritization in Bug Repositories,” In Proceeding of the 34th International Conference on Software Engineering (ICSE'12). IEEE Press, pp. 25-35, 2012.

[15] 정가을, “버그 관리 시스템에서의 기계학습을 이용한 버그 리포트 프로세스 개선 방안”, 석사학위논문, 서울대학교, 2010.

[16] G. Jeong, S. Kim and T. Zimmermann, “Improving bug triage with bug tossing graph,” Proc. of Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, pp. 111-120, 2009.

[17] S. Rao, and A. Kak, “Retrieval from Software Libraries for Bug Localization: A Comparative Study of Generic and Composite Text Models,” In Proceeding of the 8th International Working Conference on Mining Software Repositories(MSR'11). ACM Press, pp. 43-51, 2011.

[18] A. Hulth, and B. B. Megyesi, “A Study on Automatically Extracted Keywords in Text Categorization,” In Proceeding of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL(ACL'06). ACM Press, pp. 537-544, 2006.

약 력

양 근 석

2013 한국기술교육대학교 컴퓨터공학부 학사
현재 서울시립대 컴퓨터과학과 석사과정 재학 중
관심분야 : 소프트웨어 결함, 결함 리포트 관리
E-mail : ypats87@uos.ac.kr

장 도

2005 동북대학교(중국) 자동화학과 학사
2008 동북대학교(중국) 소프트웨어공학과 석사
2013 서울시립대 컴퓨터과학과 박사
현재 서울시립대 컴퓨터과학부 연구교수
관심분야 : 데이터마이닝, 소프트웨어 유지관리
E-mail : kerryking@ieee.org

이 병 정

1990 서울대 계산통계학과 학사
1998 서울대 전산과학과 석사
2002 서울대 전기·컴퓨터공학부 박사
1990~1998 (주)하이닉스반도체 연구원
2002~현재 서울시립대 컴퓨터과학부 교수
관심분야 : 소프트웨어공학, 웹 서비스 기술, 소프트웨어진화
E-mail : bjlee@uos.ac.kr