API 변경에 대한 마이닝: 문헌과 실제 조사

박종열○, 이선아
정보과학과, 경상대학교
{bjng237, saleese}@gnu.ac.kr

Literature Survey and Empirical Investigation on Mining API Changes

Jongyeol Park○, Seonah Lee
Department of Informatics, Gyeongsang National University

요 약

API(Application Programming Interface)는 여러 소프트웨어 컴포넌트 사이의 인터페이스로서, 주로 라이브러리나 프레임워크에서 API를 제공하면 응용 소프트웨어에서 제공받은 API를 사용한다. 응용 소프트웨어의 개발자들은 이러한 API를 사용하거나 갱신할 때 적절한 API를 찾고 응용하는데 어려움을 겪는다. 이러한 어려움을 해소하기 위하여 기존 연구들은 API 사용을 위한 추천 시스템이나 혹은 API를 갱신하는데 필요한 정보를 도출하는 마이닝 기법을 제시한다. 우리는 기존 연구들이 제시하는 기법들을 문헌 조사하고, 또한 하나의 소프트웨어 시스템에서 API 호출이 어떤 방식으로 변경되는지를 조사함으로써 문헌과 실제의 차이에 대해서 분석하여 논의한다.

1. 서 론

Application Programming Interface (이하 API)는 소프트웨어 컴포넌트들이 어떻게 상호 작용하는지를 명시한다. 이렇게 명시한 API를 활용하여 소프트웨어 어플리케이션을 구축한다. 객체지향 프로그램에서 API는 공개 클래스, 메소드, 필드로서, 정의된 구현체의 선언부에 해당한다. 개발자들은 이러한 API를 이해하고, 사용하고, 그리고 갱신하는데 어려움을 겪는다 [1].

이러한 어려움을 해소하기 위해 기존 연구자들은 API 사용을 위한 추천 시스템 [2, 3, 4, 5, 6]이나 혹은 API를 갱신하는데 필요한 정보를 도출하는 마이닝 기법 [7, 8]을 제시한다. 그리고 API 관련 연구를 정교화하고 있다 [9, 10]. 그러나, 기존 연구에서 API 추천의 실제적인 효과를 가져오기 위한 논의를 찾아보기 힘들다.

우리는 API 변경에 대한 추천에 초점을 맞추어 다음을 진행한다. 첫째, 기존 연구들이 제시하는 기법들을 문헌 조사한다. 둘째, 실제 하나의 소프트웨어 시스템에서 API 호출이 어떤 방식으로 변경되는지를 조사한다. 마지막으로, 문헌과 실제의 차이에 대해서 분석, 논의한다. 이러한 문헌과 실제의 사이의 괴리를 분석하여 향후 연구 방향에 대해서 논의한다.

본 논문의 구성은 다음과 같다. 2장은 기존 API 관련 연구들이 제시하는 기법에 대해 조사한다. 3장은 실제 하나의 소프트웨어 시스템에서 API 호출이 어떤 방식으로 변경되었는지를 조사한다. 4장은 문헌과 실제 사이의 괴리를 명시하고 이를 통해 향후 연구 방향을 제시하고자 한다. 5장은 논문의 결론을 요약한다.

2. 관련 연구

기존 연구들은 API 사용을 위한 추천 시스템이나 혹은 API를 갱신하는데 필요한 정보를 도출하는 마이닝 기법을 중심으로 발전하여 왔다.

먼저 API 사용을 위한 추천 시스템의 발전을 살펴보자. Bruch와 그 동료들은 2006년도에 프레임워크의 API의 호출 순서가 정해져 있는데 개발자들이 이러한 순서를 잘 발견하지 못한다고 지적하였다. 이를 지원하기 위해 기존에 작성한 API 호출을 기반으로 그 다음에 작성해서 호출해야 하는 API를 추천하는 기법을 제안하였다 [2]. 이와 유사한 연구로는 Zhong의 연구가 있다 [3]. 또한 Bruch와 그 동료들은 좀 더 실제적인 방향으로 발전시켜서 2008년도에는 알파벳 순으로 보여주던 이클립스 IDE에서의 코드 완성 시스템을 여러 개발자들의 활동 히스토리를 바탕으로 많이 사용하는 API를 추천하는 기법을 제안하였다 [4].

이러한 API 추천 기법은 API 매개 변수를 추천하는 기술로 발전하고 있다 [5, 6]. 먼저 Zhang과 그 동료들은 API 매개변수를 추천하기 위해 새로운 기법(Precise)를 개발하였다 [5]. 해당 기법은 먼저 매개 변수 사용과 코드 베이스 안의 문맥을 분석하고 매개 변수 사용에 대한 데이터베이스를 구축한다. 그 다음 데이터베이스에 요청의 내용에 대한 답이나, 추상적인 패턴 사용의 그룹을 검색하는 질문을 하여 API 매개변수를 추천한다. 실험 평가에서는 53% 정확도를 보여주었다. 다음으로 Asaduzzaman과 그 동료들은 Precise가 API 매개 변수를 추천하지만 다루고 있는 매개 변수의 타입 수가 적다는 점을 지적하였다[6]. 이를 위해 매개 변수 타입 수를 늘린 Parc라는 기술을 제시하였다. Parc의 특징은 지역성을 고려한 추천을 수행한다는 점이다. 저자들은 Parc이 Precise와의 비교에서 추천의 정확도가 높음을 보였다.

다음 API를 갱신하는데 필요한 정보를 도출하는 기법들의 발전 상황을 살펴보자. 먼저 Dagenais와 Robillard는 두 개의

프레임워크 버전 사이에서 변경된 API 호출을 분석하여, API의 호출 변경에 대한 규칙을 도출하는 기법인 SemDiff를 제안하였다 [7]. 그 다음 Wu와 그 동료들은 두 개의 버전 사이에서 변경된 API 호출 사이에 이름의 유사성까지 분석하여 호출에 대한 변경 규칙을 도출하였다 [8]. 이들은 텍스트 유사성과 호출 의존의 기법을 적용한 하이브리드 형식의 접근 기법인 AURA를 제안한다. AURA에서는 API 호출 변경에서 1:1, 1:m, n:1, n:m에 대한 맵핑에 대해서 고려하였고, 단순하게 삭제한 API 경우에 대해서도 논의하였다.

API 변경 규칙 도출의 후속 연구로 두 개의 버전 사이의 차이점을 분석하는 것을 확장하여 소프트웨어 리비전 전체를 분석하였다. Meng과 동료들은 두 개의 버전 사이에서의 호출 변경 규칙을 분석하던 것을 전체 리비전 히스토리에서의 호출 변경 규칙으로 확장하였다 [9]. 이들은 HiMA라는 도구를 제안하였는데, HiMA는 리비전 히스토리에서 각 버전 사이의 차이를 분석할 뿐만 아니라 리비전 히스토리의 로그 메시지에서도 호출 변경 규칙에 추가적으로 도움을 줄 수 있는지 체크한다 [9]. Hora와 동료들은 리비전 히스토리에서의 변경을 모니터링하여 API 변경 규칙을 생성하는 도구인 Apievolutionminer를 제시하였다 [10]. 마지막으로 Azad와 동료들은 함께 바뀐 API 호출 사이의 연관 관계 분석할 때 스택오버플로우 데이터를 활용하였다. 연관관계 분석을 230개의 안드로이드 앱들을 대상으로 12,000개의 리비전들과, 113,303개의 스택오버플로우 게시물에 대해 적용하였다. 그 결과 유사한 그룹의 프로젝트들의 리비전 히스토리를 마이닝한 결과의 정확도가 가장 높음을 보였다 [11].

덧붙여 API 실제 사례 연구들이 있다. 예를 들어 Hao와 Yao는 API는 설계상의 결함을 제거하기 위해 변경한다는 관점에서, 설계 의도에 따라 API의 변경 사항을 분류, 조사하였다. 대상은 JDK의 필수 구성 요소인 AWT와 Swing의 변경에 대한 분석을 진행하였으며 ,1386개의 새로운 API 메소드 추가 중에서 80%가 리팩토링임을 밝혀 내었다. 이 논문은 Java API 문서에서 추출한 정보를 기반으로 한 API 발전에 대한 첫번째 사례 연구이다 [12].

3. 실제 조사 결과

소프트웨어 시스템에서 API 호출이 어떤 방식으로 변경되는지를 조사하였다. 조사 대상으로는 GitHub에 있는 Android App 중 하나인 IRCCloud[1]를 조사하였다. IRCCloud는 그룹 채팅을 지원하는 앱이다.

IRCCloud의 변경 기록에서 최신의 시점부터 대략 300여개의 변경이력(commit)들을 조사하였다. 조사 할 때의 기준은 Java에서 API의 호출에 관련된 변경이 발생한 commit으로 한정하였다. 또한 API 추가와 삭제가 동시에 발생하는 변경에 초점을 맞추었다. 그 결과 표 1과 같이 36개의 변경 사례를 찾을 수 있었다.

[1] IRCCloud, https://github.com/irccloud/android

[표 1] IRCCloud에서의 API 호출 변경 분석표

Type		개수	확률
메소드 호출 변경		**15**	**41.5%**
	Method 추가	5	13.8%
	Method 삭제	2	5.5%
	Method 변경	8	22.2%
매개변수 사용 변경		**11**	**30.4%**
	Parameter 추가	5	13.8%
	Parameter 삭제	3	8.3%
	Parameter 변경	3	8.3%
필드 연관 변경		**10**	**27.6%**
	Method ⇒ Field	5	13.8%
	Field ⇒ Method	2	5.5%
	Field 변경	3	8.3%
Total		36	100%

표 1은 API 변경을 크게 3가지로 분류할 수 있음을 보여준다. 첫째, 메소드 호출 변경으로 메소드의 추가, 삭제, 변경이 있다 (41.5%). 둘째, 메소드의 매개변수 사용 변경으로 매개변수의 추가, 삭제, 변경이 존재한다 (30.4%). 마지막으로, 필드 연관 변경이 있다. 필드 연관 변경에서는 메소드에서 필드로의 변경, 필드에서 메소드로의 변경, 필드 내 변경이 있다 (27.6%).

첫째, 메소드 호출 변경은 주로 디스플레이의 개선, 이미지로딩 개선과 같은 상황에서 발생하였다. 메소드 호출 변경의 예를 보면 다음과 같다.

- 메소드 추가: TextUtils.htmlEncode(e.msg)를 TextUtils.htmlEncode(e.msg).replace(" ", " ")로 변경하는데 있어 replace(" ", " ")를 추가하였다. (#b24cb08[2] 참조)
- 메소드 삭제: RemoteInput.Builder("extra_reply").setLabel("Reply to " + title).build()).build()에서 마지막의 build() 메소드를 삭제하고 RemoteInput.Builder("extra_reply").setLabel("Reply to " + title).build()로 변경하였다. (#18e00b7 참조)
- 메소드 변경: url.contains("/wiki/File:")를 url. Matches (".*/wiki/.*/File:.*")로 변경하였다. (#b28ac14 참조)

둘째, 매개변수 사용 변경은 주로 App의 네트워크 연결, 암호 알고리즘의 변경과 같은 상황에서 발생하였다. 매개변수 사용 변경의 예는 다음과 같다.

- 매개변수 추가: conn.connect()에 true라는 매개변수를 추가하여 conn.connect(true)로 변경하였다. (#6673cbe 참조)
- 매개변수 삭제: oobTasks.get(bid).cancel(true) 에서 true를 삭제하고 oobTasks.get(bid).cancel ()로 변경하였다. (#07866fa 참조)
- 매개변수 변경: MessageDigest.getInstance ("SHA-

[2] IRCCloud의 GitHub에서 명시한 해쉬 번호로 참조한 변경 이력(Commit)을 찾을 수 있다.

1")에서 매개변수값을 SHA-256으로 변경 하여 MessageDigest.getInstance("SHA-256")으로 변경하였다. (#2fd4725 참조)

셋째, 필드 연관 변경은 주로 다른 버전 지원, 최적화, 네트워크 변경과 같은 상황에서 발생하였다. 필드 변경의 예는 다음과 같다.

- 메소드를 필드로 변경: isCancelled()를 isCancelled로 변경하였다. (#07866fa 참조)
- 필드를 메소드로 변경: object.eid를 object.eid()로 변경하였다. (#9a5b793 참조)
- 필드 변경: Build.VERSION_CODES.N를 Build.VERSION_CODES.JELLY_BEAN_MR1로 변경하였다 (#51bd390 참조)

조사 결과 IRCCloud의 변경 이력들에서 메소드 호출 변경이 가장 많았으며 (41.5%). 다음 매개변수 변경이며 (30.4%), 마지막으로 필드 연관 변경이 뒤따른다 (27.6%). 이 세가지 분류는 비슷한 비율을 차지한다. 또한 메소드 호출 변경의 경우에는 디스플레이 관련 변경, 매개변수 변경의 경우는 네트워크 연결과 암호알고리즘 변경, 필드 연관 변경은 다른 버전 지원, 최적화 등의 하드웨어 변경과 관련된 상황에서 발생하였다.

4. 논 의

문헌 조사를 통해 배운 점을 정리하면 다음과 같다. 첫째, API 추천에 있어서 매개변수 분석 연구를 진행하고 있으나, API 변경 분석에 있어서는 매개변수 분석을 진행하고 있지 않다. 둘째, 리비전 히스토리가 아닌 스택오버플로우 데이터는 도리어 정확도가 낮았다. 이러한 두 가지 사항을 볼 때 리비전 히스토리를 대상으로 매개변수 분석을 포함한 여러 변경 타입을 좀 더 체계화하는 연구가 있어야 함을 알 수 있다.

실증적 조사를 통해 배운 주요한 사항은 다음과 같다. 조사를 통해서 필드 연관 변경이 상당한 비율을 차지함을 밝혔다. 즉, 필드가 메소드로, 메소드가 필드로, 필드가 또 다른 필드로 변하는 경우가 있었다. 이 경우는 전체의 27.6%를 차지했다. 그러나 문헌 조사와 대비하였을 때, 문헌 조사에서는 이러한 필드 연관 변경에 대한 마이닝을 수행하고 있지 않음을 파악할 수 있었다. 따라서 필드 연관 변경 사항에 대한 마이닝 연구가 추가로 필요함을 알 수 있다.

5. 결 론

본 논문에서는 API 사용 추천을 위한 마이닝 기법 들을 문헌 조사하였다. 또한 하나의 프로젝트의 300여 개의 API 변경을 조사하였다. 조사 결과 36개의 API 변경을 찾을 수 있었고, 변경을 메소드 변경, 매개변수 변경, 필드 연관 변경으로 분류할 수 있었다. 그 중 메소드와 매개변수에 대한 연구가 있음을 확인할 수 있었다. 반면 필드 연관 변경은 아직 진행되지 않음을 확인하였다. 향후 연구는 이러한 문헌 조사와 실증적 조사의 차이에서 지목된 필드 연관 연구를 하여 API 변경 추천에 기여하고자 한다.

6. 참고 문헌

[1] Robillard, Martin P. "What makes APIs hard to learn? Answers from developers." IEEE software 26.6 (2009).

[2] Bruch, Marcel, Thorsten Schäfer, and Mira Mezini. "FrUiT: IDE support for framework understanding." Proceedings of the 2006 OOPSLA workshop on eclipse technology eXchange. ACM, 2006.

[3] Zhong, Hao, et al. "MAPO: Mining and recommending API usage patterns." European Conference on Object-Oriented Programming. Springer Berlin Heidelberg, 2009.

[4] Bruch, Marcel, Martin Monperrus, and Mira Mezini. "Learning from examples to improve code completion systems." Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering. ACM, 2009.

[5] Zhang, Cheng, et al. "Automatic parameter recommendation for practical API usage." Proceedings of the 34th International Conference on Software Engineering. IEEE Press, 2012.

[6] Asaduzzaman, Muhammad, et al. "Exploring API method parameter recommendations." Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on. IEEE, 2015.

[7] Dagenais, Barthelemy, and Martin P. Robillard. "SemDiff: Analysis and recommendation support for API evolution." Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on. IEEE, 2009.

[8] Wu, Wei, et al. "Aura: a hybrid approach to identify framework evolution." Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1. ACM, 2010.

[9] Meng, Sichen, et al. "A history-based matching approach to identification of framework evolution." Software Engineering (ICSE), 2012 34th International Conference on. IEEE, 2012.

[10] Hora, André, et al. "Apievolutionminer: Keeping api evolution under control." Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on. IEEE, 2014.

[11] Azad, Shams, Peter C. Rigby, and Latifa Guerrouj. "Generating API Call Rules from Version History and Stack Overflow Posts." ACM Transactions on Software Engineering and Methodology (TOSEM) 25.4 (2017): 29.

[12] Hou, Daqing, and Xiaojia Yao. "Exploring the intent behind api evolution: A case study." 2011 18th Working Conference on Reverse Engineering. IEEE, 2011.