# 오픈소스 소프트웨어 라이선스 컴플라이언스 검증 도구

윤호영*, 조용준*, 정병옥*, 신동명*
*엘에스웨어(주)
e-mail : {allen, eugene, luca, roland}@lsware.com

# Method for License Compliance of Open Source Software

Hoyeong Yun*, Yongjoon Joe*, Byung-OK Jung*, Dongmyung Shin*
*LSWare Inc., Korea

요 약

오픈소스 소프트웨어를 사용하기 위해서는 개발자가 정의한 라이선스를 반드시 준수해야 한다. 이를 위반할 경우, 소스코드 공개/수정/재작성, 재배포 라이선스 변경, 특허권 포기 등의 피해로 이어질 수 있다. 소프트웨어를 다수의 개발자가 함께 개발하는 경우, 오픈소스 소프트웨어가 혼용되기도 하는데, 이는 서로 다른 라이선스간의 조항이 충돌하는 상황을 유발할 수 있다. 즉, 라이선스 규정 때문에 상황에 따라 혼용할 수 없는 오픈소스 소프트웨어 조합이 존재한다. 이러한 오픈소스 소프트웨어를 사용하는 과정에서 발생할 수 있는 무의적인 라이선스 규정 위반을 미연에 방지하고자 오픈소스 소프트웨어 라이선스 컴플라이언스 검증 도구를 제시하고자 한다.

## 1. 서론

오픈소스 소프트웨어란 소스코드를 누구나 수정 및 개선할 수 있는 소프트웨어를 칭한다. 오픈소스 소프트웨어의 활성화는 임베디드, 모바일, 클라우드 컴퓨팅, 빅 데이터 등 모든 ICT 분야에 기여하고 있다. 상용을 목적으로 하는 소프트웨어에 반하여 누구나 자유롭게 사용하자는 오픈소스 운동은 공유 경제의 대표적 사례로 꼽힌다.

오픈소스 소프트웨어는 “Open”이라는 단어로 인해 무료라고 잘못 인식되는 경우가 종종 발생하는데, 엄연하게는 라이선스 의무 조항을 준수해야 한다는 전제하에 자유롭게 사용 가능한 소프트웨어로 정의할 수 있다. 실제로 오픈소스 소프트웨어의 라이선스 조항을 준수하지 않아 관련 분쟁과 소송 사례가 발생하기도 한다. 사용자뿐만 아니라 오픈소스를 배포하는 개발자도 라이선스에 대한 인식이 미비한데, 가장 규모가 큰 오픈소스 저장소인 GitHub에서 라이선스 파일의 부재를 지적하는 게시글이 5만 건에 달하는 것으로 조사됐다.

현재 오픈소스 소프트웨어의 라이선스는 2,400개 이상 존재하는 것으로 추정된다[1]. 이러한 오픈소스 라이선스는 자유롭게 수정, 배포 가능한 조항부터 소스코드 공개, 라이선스 강제와 같은 강제성을 띄는 조항 등으로 이루어져 있다. 특히 GPL(GNU General Public License)는 강제성이 강한 라이선스인데, GPL로부터 파생된 프로그램도 소스코드가 공개되어야 하고, 파생된 프로그램 또한 강제로 GPL 라이선스 또는 그 호환 라이선스로 배포되어야 한다는 “재귀적 전염성 조항”을 내포하고 있다. 즉, GPL 라이선스를 따르는 오픈소스를 사용하면 소스코드를 공개해야 한다. GPL 라이선스를 따르는 오픈소스를 활용하여 상용화 제품을 개발한 기업들이 이를 인지하지 못하여, 소스코드를 강제로 공개당하거나 리콜, 판매중지 사태가 벌어지기도 하였다.

2015년 2월 기준, 라이선스 점유율은 GPL 2.0이 25%였으며, 자유도가 높은 라이선스인 MIT 라이선스와 Apache 2.0 라이선스가 각각 19%, 16%의 점유율을 보였다. 2009년부터 2015년까지의 라이선스 점유율 추세는 GPL 라이선스가 21.4%의 점유율을 잃은 반면, MIT 라이선스와 Apache 2.0 라이선스는 각각 15.7%와 12.4%의 성장세를 보였다. 2017년 기준으로는 MIT 라이선스가 29%로 점유율이 가장 높으며, GPL 2.0이 19%, Apache 2.0 라이선스가 15%의 점유율을 보였다. 이는 까다롭고 강제성이 높은 GPL 보다 사용하기 자유로운 라이선스를 선호하는 것으로 볼 수 있다[2].

소프트웨어를 개발할 때, 여러 오픈소스 소프트웨어가 활용되는 경우는 빈번하다. 이러한 경우 여러 라이선스가 혼용되기도 하는데, 각 라이선스 별로 준수해야할 조항이 상충하는 경우가 발생한다. 해당 경우를 라이선스 충돌이라고 하는데, 강제성이 높은 라이선스가 있을 경우 호환되지 않는 “충돌”이 발생하기 쉽다. GNU Operating System은 GPL을 기준으로 호환되지 않는 라이선스에 대해 명시해두고 있다[3].

GPL의 경우는 논쟁이 된 사례가 많아 라이선스 호환 여부에 대한 정보가 많지만 앞서 말했듯이 현존하는 라이선스는 2,400개 이상으로 매우 방대하기 때문에 특허 사용권 허락, 보복 조항, 소스코드 공개 조항, 라이선스 강제

조항 등 다양한 이유로 호환되지 않는 경우가 발생한다.

라이선스 호환 문제로 인한 잘못된 오픈소스 소프트웨어 사용은 법적 분쟁의 원인이 될 수 있다. 그렇기 때문에 소프트웨어 개발 과정에서 오픈소스 소프트웨어 라이선스 간의 호환 여부를 검사하는 것은 필수적으로 이루어져야 한다. 본 논문에서는 소프트웨어를 개발하는 과정에 있어 오픈소스 소프트웨어의 라이선스 호환 여부를 검증하는 일련의 프로세스를 제시하고자 한다.

## 2. 라이선스 컴플라이언스 검증 도구 요구사항

오픈소스 소프트웨어 라이선스간 호환 여부를 검사하기 위해서는 해당 오픈소스 소프트웨어들의 라이선스를 식별해야 한다. 본 논문에서 식별은 프로젝트 파일 내에서 라이선스 관련 내용을 추출하고, 해당 내용을 기반으로 어떤 라이선스인지를 판별하는 것으로 정의한다. 안타깝게도 오픈소스 소프트웨어 라이선스는 프로젝트 파일 내에서 다양한 형태로 존재한다. 리눅스 재단의 워킹그룹인 FOSSBazaar에서는 SPDX(Software Package Data Exchange)라는 소프트웨어 패키지의 구성요소, 라이선스 및 저작권에 관한 표준을 발표하였지만 아직 활용은 미비하다.

### 2.1. 라이선스 추출

오픈소스 소프트웨어 라이선스는 프로젝트 패키지 내에서 다양한 형태로 존재한다. 대표적인 경우가 1) 라이선스에 관련된 내용이 별도의 파일로 존재하는 경우, 2) 라이선스 내용이 소스코드의 주석문에 포함되어 있는 경우이다.

#### 1) 라이선스 관련 내용이 별도의 파일로 존재하는 경우

프로젝트 패키지의 최상위 폴더에 “README”, “COPYRIGHT”, “LICENSE” 등의 파일로 존재하는 경우를 의미한다. 해당 파일에는 개발자의 이름, 프로젝트 명, 개발 일자, 사용법 등 프로젝트와 관련된 내용이 기재되어 있으며, 라이선스와 관련된 내용도 표기되어 있는 경우가 있다.

최상위 폴더의 파일들 중 라이선스 정보를 담고 있는 파일을 구분하기 위해 SPDX의 라이선스 파일들을 분석하였다. SPDX에서는 사용빈도가 높은 322개(2016년 1월 6일 기준) 라이선스를 공개하고 있다. N-gram은 대표적인 확률적 언어 모델로 n개의 단어의 연쇄를 확률적으로 표현하는 것을 뜻한다. 불용어(stop words)를 제외한 n-gram을 정규식을 통해 수집하였고, 해당 어절들을 토대로 라이선스 정보를 담고 있는 파일을 구분하였다.

#### 2) 라이선스 관련 내용이 소스코드의 주석문에 포함되어 있는 경우

라이선스의 일부 혹은 전문이 소스코드 파일의 주석으로 존재하는 경우를 뜻한다. 주석문은 프로그래밍 언어에 따라 “/* … */”, “<!-- … -->”, “//” 등의 형태를 띄며, 라이선스의 내용은 주로 소스코드의 상단 주석문에 위치한다. 프로젝트 패키지에 존재하는 소스코드 파일의 주석문을 파일 단위로 수집하고, TF-IDF와 코사인 유사도 기법을 활용하여, 라이선스 정보를 담고 있는 주석문을 분류하였다.

### 2.2. 라이선스 판별

추출한 라이선스 정보가 어떤 라이선스의 정보인지를 판별하는 방법은 라이선스의 이름을 비교하는 방법, 라이선스의 핵심 문구를 비교하는 방법, 라이선스의 전문을 비교하는 3가지 방법이 있다.

#### 1) 라이선스의 이름이 명시된 경우

별도의 파일이나 주석문에 라이선스의 이름이 명시된 경우가 있다. 명시된 라이선스의 이름이 SPDX에 등록된 라이선스의 리스트에 존재하면 해당 라이선스라고 판별한다.

#### 2) 라이선스의 핵심 문구가 존재하는 경우

라이선스 전문에는 일반적인 법률 내용뿐만 아니라 라이선스의 고유한 문구들이 존재하는 경우가 있다. SPDX의 라이선스 파일들의 전문을 분석하여, 각 라이선스 별로 핵심 문구를 색인하였다. 프로젝트 패키지 내에서 추출된 라이선스 정보 중에 핵심 문구가 존재하는 경우 해당 라이선스로 판별한다.

#### 3) 라이선스의 내용만 명시된 경우

라이선스 이름이 명시되지 않았고, 핵심 문구도 추출되지 않았다면, 라이선스의 전문 비교를 통해 판별한다. 라이선스의 전문 비교는 기본적으로 코사인 유사도 혹은 리벤슈타인 거리 알고리즘을 사용한다. 코사인 유사도 측정의 경우 연산 속도가 빠른 장점이 있지만, 내용이 조금 변경된 경우 정확도가 떨어지는 단점이 있다. 리벤슈타인 거리 알고리즘은 연산 속도가 느린 단점이 있지만, 정확도가 높은 장점이 있다.

코사인 유사도 기법을 통해 유사도를 측정했을 때, 유사도의 수치가 낮은 경우에는 상위집단을 대상으로 리벤슈타인 거리 알고리즘을 실시한다. 즉 빠른검사를 실시하고 값이 정확하지 않은 경우에는 정밀검사를 실시하는 개념이다.

### 2.3. 라이선스 호환성검사[4]

라이선스 추출과 라이선스 판별을 진행하면, 오픈소스 소프트웨어 패키지 내에 존재하는 라이선스의 항목들이 축출된다. 라이선스가 1개인 경우는 조항에 맞게 사용하면 되고, 2개 혹은 그 이상의 경우에는 호환성 여부를 검사한다.

오픈소스 소프트웨어 라이선스를 자동으로 분석하고 호환 혹은 호환성에 대한 판단 결과를 도출하기 위해서 322개의 SPDX 라이선스를 하나의 일정한 형식으로 맞추었다. 이를 위해서 오픈소스 라이선스가 일반적으로 담고 있는 조항 및 특수한 조항을 추출하였고, 이를 특징점(feature

point)이라고 정의하였다. 라이선스 호환성 관계를 분석하기 위해 18개의 특징점을 구성하였다. 정의한 특징점의 항목은 <표 1>과 같다.

<표 1> 오픈소스 소프트웨어 라이선스 특징점 항목

| 항목 |
|---|
| 카피레프트(Copyleft) |
| 추가제한 금지조항(No further restrictions) |
| 소스코드 공개 |
| 소스코드 제공 방법의 유무 |
| 소스코드 공개 범위 |
| 재 라이선싱(Relicensing) |
| 실행파일 재 라이선싱 |
| 법적 공지(Legal notice) 작성의무 |
| 특허보복조항 |
| 특허실시허락 |
| 차별적 면책조항 |
| 상호, 상표 등 사용금지 |
| 라이센서(Licensor)와 저자(Author)의 이름 사용 금지 |
| 수정된 버전이 원본과 다름 표시 의무 |
| 특이 보증·면책 부인 조항 |
| 저작자 혹은 기여자 정보 의무 작성 |
| 출처의 그릇된 표시·표현 금지 |
| 준거법의 유무 |

오픈소스 소프트웨어의 라이선스 호환성에 있어 카피레프트와 추가제한 금지조항은 주요 특징점으로 간주한다. 카피레프트조항이 있는 오픈소스 소프트웨어는 동일한 라이선스 혹은 동일한 조건으로 배포해야 하는 의무가 발생한다. 이 조항에 따라서 카피레프트 조항을 보유하고 있는 라이선스는 보유하고 있지 않은 라이선스와 호환성이 발생하게 된다. 라이선스에 추가제한 금지조항이 있다면 해당 라이선스에 원라이선스에 추가적인 의무사항을 부여하는 것이 불가능해진다. 예를 들면, 동일한 라이선스로 배포 및 추가제한 금지조항이 있는 라이선스(이하 A)와 동일한 조건으로 배포해야 하는 라이선스(이하 B)의 오픈소스 소프트웨어를 결합하여 사용하고자 할 때, B를 A로 변환하여 함께 결합하여 사용 가능하다. 다만, B의 준수 조건이 A와 불일치 할 경우 추가제한 금지조항으로 인하여 호환성이 발생하게 된다.

## 3. 결론

시스템 구축이나 소프트웨어 개발 프로젝트가 발주되면 대형업체가 수주를 하게 된다. 이 업체는 다시 자신보다 규모가 작은 업체들에 하청을 준다. 하청 받은 업체들은 또다시 군소업체들에 재하청을 주는 방식의 수직적 구조를 보인다.

소프트웨어 개발을 수행할 때, 서로 다른 그룹에서 인력이 투입될수록 프로젝트 매니저의 역할이 중요해진다. 라이선스에 대한 인식이 향상된 요즘 프로젝트 발주 단계에서 라이선스 사용에 제한을 두고 있는 경우도 있지만, 아직은 라이선스에 대한 인식은 부족한 상태이다. 다수의 업체에서 오픈소스 소프트웨어를 활용하여 개발할 때, 오픈소스 소프트웨어의 라이선스를 크게 고려하지 않는다. 이러한 상황에서 해당 업체들로부터 개발된 소프트웨어를 병합할 때, 라이선스 호환성 검사는 반드시 필요하다. 또한 병합 단계에서 호환성 검사를 했을 때, 문제가 발생하게 되면 프로젝트의 진행에 차질을 빚기 때문에 프로젝트 계획, 진행단계에서부터 선제적인 라이선스 인식 고양이 필요하다.

## 참고문헌

[1] Pehr Burenuius, “Is Open Source Safe and Secure? The Intelligent Management of Open Source”, 2014

[2] Black Duck Knowledge Base, “Top Open Source Licenses”, https://www.blackducksoftware.com/top-open-source-licenses

[3] GNU Operating System, https://www.gnu.org/licenses/license-list.html

[4] Y.J, Joe, H.K, Bahng, D.M, Shin, “Method for FOSS License Compatibility Verification using Feature Point Comparison”, The 5th International Conference on Ubiquitous Computing Application and Wireless Sensor Network, 2016