

[이슈] FOSS(Free and Open Source Software)의 현황과 저작권 이슈

정윤재*

FOSS의 현황과 저작권 이슈

Contents

| | |
|-----------------------------------|----|
| 1장 서론----- | 2 |
| 1.1 도입----- | 2 |
| 1.2 배경----- | 2 |
| 2장 본론----- | 5 |
| 2.1 FOSS의 개요----- | 5 |
| 2.2 FOSS의 특성----- | 8 |
| 2.3 FOSS의 성공 사례----- | 11 |
| 2.4 FOSS 라이선스의 분류----- | 13 |
| 2.5 FOSS 라이선스의 탄생과 발전 방향----- | 16 |
| 2.6 AGPL을 통한 프리 소프트웨어 진영의 견제----- | 18 |
| 2.7 FOSS 저작권 관련 기술----- | 20 |
| 3장 결론----- | 22 |
| 첨부 (참고 문헌) ----- | 24 |
| 부록 (용어 정의) ----- | 25 |

* (주) 오픈위즈덤 대표이사, 한국저작권위원회 위원



1장 서론

1.1 도입

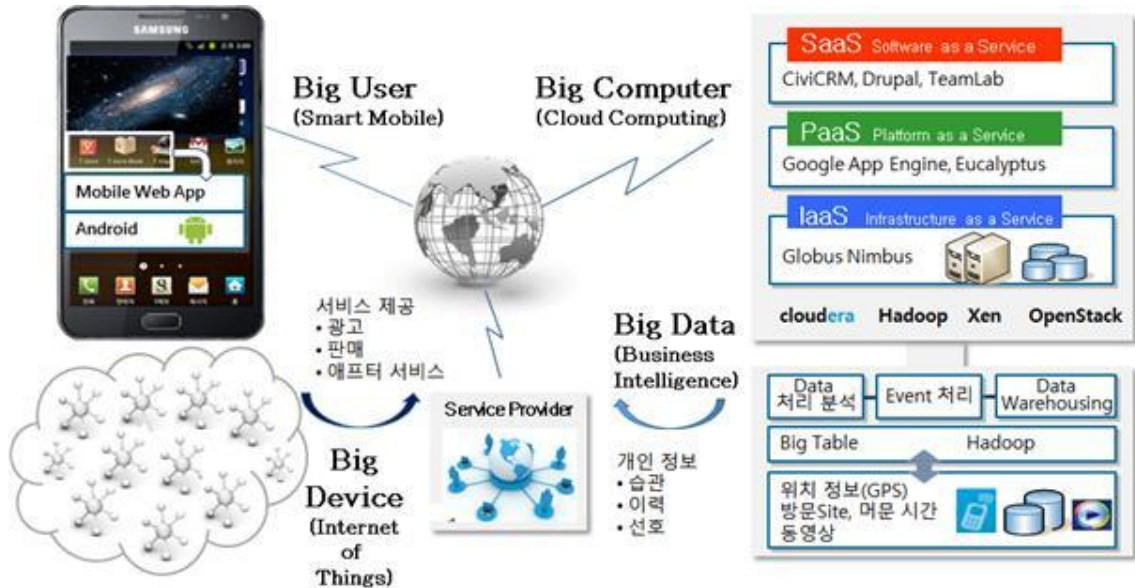
FOSS(Free and Open Source Software)는 프리 소프트웨어와 오픈소스 소프트웨어를 모두 지칭할 때 사용하는 말이다. [1-4] 이 두 소프트웨어는 소스 코드를 공개하는 소프트웨어를 총칭하는 용어로 사용되고 있다¹. 모든 소프트웨어는 사람이 작성하는 프로그램과 컴퓨터가 실행하는 프로그램으로 나누어지는데, 소스 코드를 공개하는 방식 하나만으로 산업계에 엄청난 파장을 일으키고 있다. 상용 소프트웨어는 기계만 알아 볼 수 있는 실행 코드 형태로만 배포되기 때문에 그 안을 들어 있는 아이디어, 로직(logic), 알고리즘(algorithm), 공식(formular), 자료 구조(data structure) 등 인간의 창작 기술을 들여다 볼 수가 없다. 모든 소프트웨어는 사람이 작성하는 프로그램과 컴퓨터가 실행하는 프로그램으로 나누어지는데, 소스 코드를 공개하는 방식 하나만으로 산업계에 엄청난 파장을 일으키고 있다. 상용 소프트웨어는 기계만 알아 볼 수 있는 실행 코드 형태로만 배포되기 때문에 그 안을 들어 있는 아이디어, 로직(logic), 알고리즘(algorithm), 공식(formular), 자료 구조(data structure) 등 인간의 창작 기술을 들여다 볼 수가 없다. 반면에 FOSS는 실행 코드뿐만 아니라 소스 코드도 함께 제공하기 때문에 누구라도 쉽게 공부하고 모방하고 응용할 수가 있다. 따라서 원천 기술이 쉽게 전파되고 다양한 분야에 적용될 수 있어서 최신 핵심 기술의 원천으로 자리매김하고 있는 것이다. FOSS의 소스 코드가 ICT 기술 발전에 큰 영향을 미쳤지만 가격이 무료라는 사실이 저작권자의 발목을 잡는 족쇄 노릇을 하고 있는 것도 사실이다. 상용 소프트웨어에는 대부분 한 회사의 저작권이 붙게 되지만 FOSS의 경우에는 소스 코드의 파일 별로 저작권이 선언되어 있어서 저작권이 보호되지 않으면 수많은 저작권자의 권리가 침해될 수밖에 없다. 따라서 본 문서에서는 FOSS의 전반적인 이해를 돕기 위해서 FOSS의 시장, 규모, 기술분야, 발전 현황 등 기술적인 차원과 컴플라이언스 차원에서 저작권과 라이선스 등 다각적인 방향에서 FOSS를 조명해 보려고 한다.

1.2 배경

FOSS는 광범위한 의미에서 저작권자가 소스 코드를 공개하여 누구나 복제, 개작, 배포할 수 있는 소프트웨어이다. 소스 코드를 무료로 공개하여 자유롭게 사용하고 공유해서 공동으로 개발하자는 FOSS의 정신이 상용 소프트웨어 일변도의 산업계에

¹ 현재 공개소프트웨어라는 용어가 소스 코드를 공개하는 소프트웨어라는 의미로 사용되고 있지만 프리 소프트웨어와 오픈소스 소프트웨어 중 어느 것을 지칭하는 것인지 불분명하다. 다만 언론과 공개된 자료에서 문장과 문맥상 오픈소스 소프트웨어에 더 가까운 의미로 이해될 수 있다.

많은 변화를 일으켜 왔다². 실제로 FOSS가 처음으로 세상에 모습을 드러낸 이래 제법 오랜 세월이 흘렀지만 지금처럼 다양한 산업 분야에서 영향력을 행사한 적은 없었을 것이다. 아래 그림과 같이 현재 FOSS는 첨단 정보통신기술 분야인 스마트 모바일(Smart Mobile), 빅데이터(Big Data), 클라우드 컴퓨팅(Cloud Computing), 사물인터넷(Internet of Things) 등 다양한 기술 영역에서 선두주자로 핵심적인 위상을 점유하고 있다.[5]



〈그림1〉 첨단 기술을 선도하는 4대 공개 소프트웨어의 핵심 영역

더욱이 FOSS가 꽤 오랫동안 사회적 오해와 상용 소프트웨어 업계의 집요한 견제를 뚫고 지금의 핵심 기술력과 사업 경쟁력의 원천으로 자리매김한 과정을 살펴 볼 때 이러한 성장 추세는 쉽게 꺾이지 않을 뿐만 아니라 당분간 지속되고 더욱 가속화될 전망이다.³ 반면에 국내 FOSS 시장은 정부의 호혜적인 정책과 사업 창출에도 불구하고 시장은 얼어붙어 있어서 그 규모가 상대적으로 왜소할 뿐만 아니라 수익구조 측면에서도 매우 열악하다고 볼 수밖에 없다.⁴

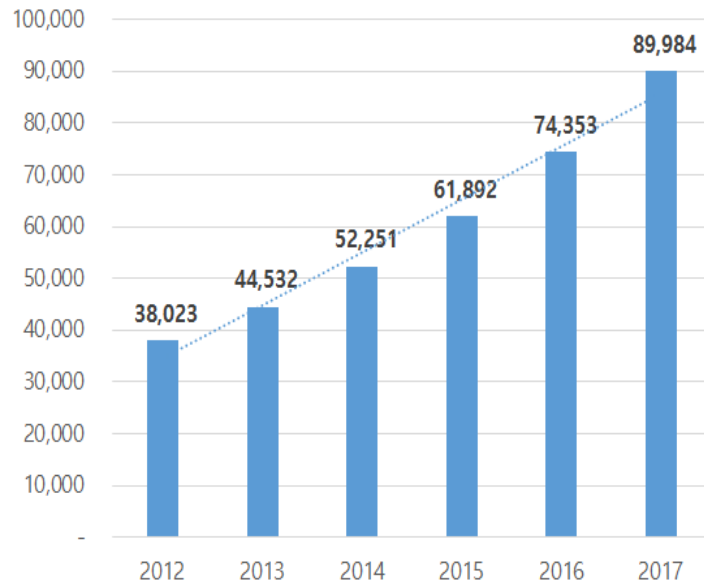
² 본 문서에서는 FOSS라는 용어를 프리 소프트웨어(Free Software)와 오픈소스 소프트웨어(Open Source Software)를 총칭하는 의미로 사용하고자 한다. 많은 사람들이 소스코드를 공개하는 소프트웨어로 오픈소스 소프트웨어만 인지하고 있지만 원래는 프리 소프트웨어가 원조이다. 따라서 소스 코드를 무료로 공유하여 소프트웨어를 자유롭게 공부하고 복제하고 수정하고 배포해서 모든 사람이 혜택을 얻자는 정신은 프리 소프트웨어로부터 시작되었다.

³ 출처 : http://www.oss.kr/oss_intro03 (단위 : 십만 달러)

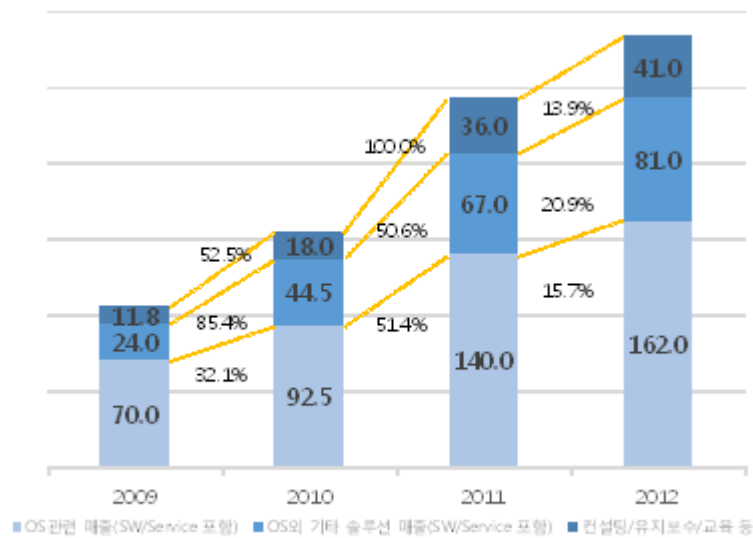
⁴ 출처 : IDC 2014.12

| 연도 | 2013년 | 2014년 | 2015년 | 2016년 | 2017년 | CAGR |
|----|--------|--------|--------|--------|--------|------|
| 세계 | 10,251 | 10,671 | 11,128 | 11,642 | 12,184 | 4.5% |
| 국내 | 104 | 110 | 115 | 120 | 124 | 4.1% |

〈표1〉 국내외 SW 시장 규모



〈그림2〉 전세계 FOSS 매출 성장 추세 (www.oss.kr)

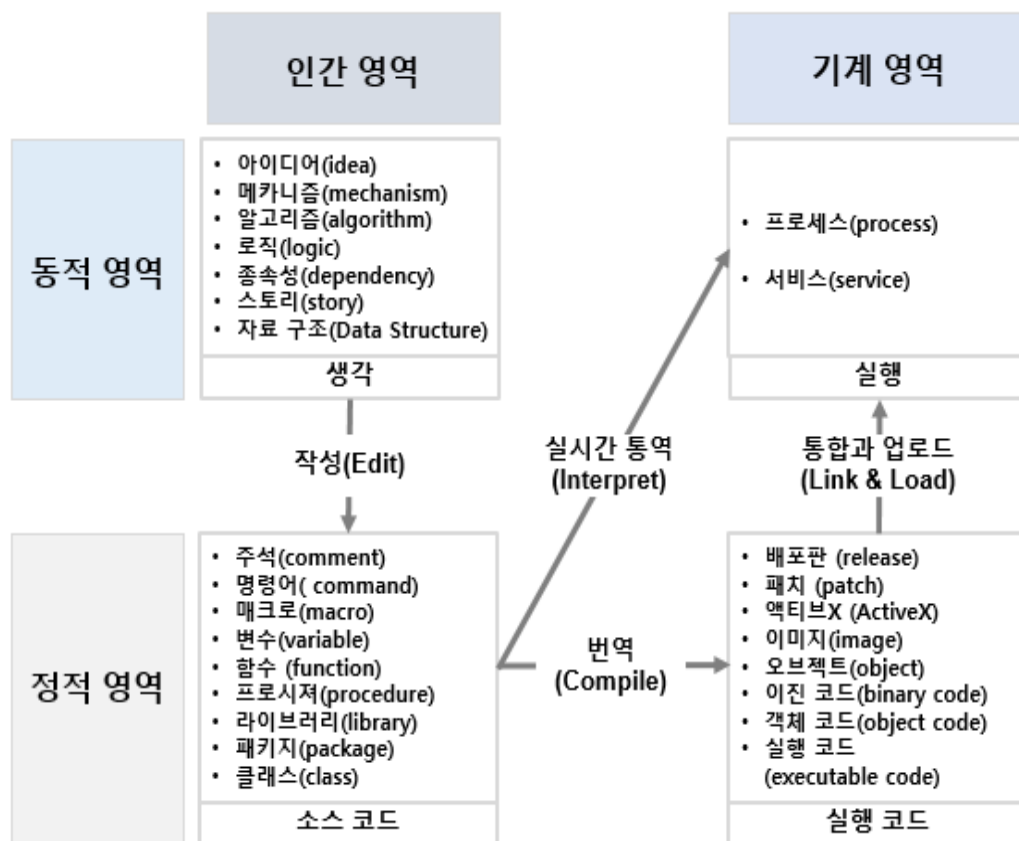


〈그림3〉 국내 FOSS 시장의 성장 추세 (IDC 2012)

2장 본론

2.1 FOSS의 개요

서두에서 설명한 바와 같이 FOSS는 저작권이 있으면서 소스 코드가 공개되어 누구나 공부, 복제, 설치, 사용, 변경, 재배포가 가능한 소프트웨어로 정의를 내릴 수 있다. 이러한 일반적인 정의와는 달리 필자는 FOSS를 “열려 있는 지혜”라는 별명을 주고자 한다. 이는 그 안에 아이디어, 로직(logic), 알고리즘(algorithm), 스토리(story) 등과 같은 지혜의 산물이 녹아 있기 때문이다. 이러한 인간 영역의 생각을 표현하는 방법이 소스 코드이다. 아래 그림과 같이 이 소스 코드를 컴파일과 인터프리터를 통해서 기계 영역으로 변환시키면 실행코드가 만들어 진다.



〈그림4〉 FOSS의 소스 코드와 실행 코드의 동일성

따라서 소프트웨어 그 자체인 FOSS는 두 영역에서 상이하게 표현된 형태를 가지고 있는 것이다. 그러므로 FOSS는 아래 그림에서 보는 바와 같이 사람의 언어로 작성된 텍스트 프로그램 형태의 소스 코드와 컴퓨터가 처리할 수 있는 이진 프로그램 형태의 실행 코드로 나누어진다.



| (1) 소스 코드 | (2) 실행코드 | | | | | | | | | | | | | | | | | | |
|---|--|-----------|-------|-----|----------------|------|-------------------|-------|-------------------|-------|---------|-------|--------------------------|--------|-----------------------------|----------|------------|-------------|-----------------------------|
| <pre> /* Copyright (c) 2007Free Software Foundation, Inc. http://fsf.org/> * Everyone is permitted to copy and distribute verbatim copies * of this license document, but changing it is not allowed. */ Package fr.opensagres.xdocreport.samples.docx.converters.xhtml; import java.io.File; import java.io.FileOutputStream; import java.io.OutputStream; public class ConvertDocxBigToXHTML{ Public static void main(String[]args) { long startTime = System.currentTimeMillis() try { // 1) Load docx with POI XWPFDocument XWPFDocument document = new XWPFDocument(Data.class.getResourceAsStream("DocxBig.docx")); } } } </pre> | <table border="1"> <tr> <td>운영 체제(OS)</td><td>Linux</td></tr> <tr> <td>웹서버</td><td>Apache, Tomcat</td></tr> <tr> <td>DBMS</td><td>MySQL, PostgreSQL</td></tr> <tr> <td>개발 언어</td><td>PHP, Perl, Python</td></tr> <tr> <td>개발 도구</td><td>Eclipse</td></tr> <tr> <td>프레임워크</td><td>Anyframe, Spring, Struts</td></tr> <tr> <td>WAS 서버</td><td>JBFOSS, Glassfish, Geronimo</td></tr> <tr> <td>데스크탑 오피스</td><td>OpenOffice</td></tr> <tr> <td>비즈니스 애플리케이션</td><td>sugarCRM, Alfresco, uEngine</td></tr> </table> | 운영 체제(OS) | Linux | 웹서버 | Apache, Tomcat | DBMS | MySQL, PostgreSQL | 개발 언어 | PHP, Perl, Python | 개발 도구 | Eclipse | 프레임워크 | Anyframe, Spring, Struts | WAS 서버 | JBFOSS, Glassfish, Geronimo | 데스크탑 오피스 | OpenOffice | 비즈니스 애플리케이션 | sugarCRM, Alfresco, uEngine |
| 운영 체제(OS) | Linux | | | | | | | | | | | | | | | | | | |
| 웹서버 | Apache, Tomcat | | | | | | | | | | | | | | | | | | |
| DBMS | MySQL, PostgreSQL | | | | | | | | | | | | | | | | | | |
| 개발 언어 | PHP, Perl, Python | | | | | | | | | | | | | | | | | | |
| 개발 도구 | Eclipse | | | | | | | | | | | | | | | | | | |
| 프레임워크 | Anyframe, Spring, Struts | | | | | | | | | | | | | | | | | | |
| WAS 서버 | JBFOSS, Glassfish, Geronimo | | | | | | | | | | | | | | | | | | |
| 데스크탑 오피스 | OpenOffice | | | | | | | | | | | | | | | | | | |
| 비즈니스 애플리케이션 | sugarCRM, Alfresco, uEngine | | | | | | | | | | | | | | | | | | |

〈그림〉 FOSS의 소스 코드와 실행 코드의 예시

결국 실행 코드는 소스 코드만 있으면 기계적인 방법으로 언제든지 즉각적으로 만들어 낼 수 있지만, 아이디어, 로직(logic), 알고리즘(algorithm), 공식(formular), 자료 구조(data structure) 등의 개념을 구현하고 있는 소스 코드는 인간이 각고의 노력을 기울여야만이 만들어 낼 수 있는 창작물이기 때문에 그 만큼 시간이 오래 걸릴 수밖에 없다. 이렇듯 하나의 소프트웨어라 할지라도 두 가지 다른 형태를 가지고 있으므로 각각의 형태에 대한 접근 방법도 상이할 수밖에 없다. 더욱이 상용의 목적으로 일반인에게 배포되지 않았던 소스 코드가 FOSS를 통해 나누어짐으로써 지금까지는 불가능 했던 지식의 교류와 협업이 가능하게 된 것이다.

그러면 FOSS와 상용 소프트웨어는 소스 코드의 공개 이외에 어떤 차이점이 있는 것일까? 대표적인 차이점은 아래 그림과 같이 비교를 통해서 설명되고 있다. 여기서 FOSS와 상용 소프트웨어의 차이점을 비교하면서 무료(free)와 유료 조건이 두 가지 소프트웨어를 구분하는 유일한 지표가 아니라는 점에 유의하여야 한다. 이는 상용 소프트웨어라 할지라도 애드웨어(Adware), 쉐어웨어(Shareware), 프리웨어(Freeware) 등은 무료이지만 소스 코드는 공개하고 있지 않기 때문이다.

| FOSS - 상용 소프트웨어 | 상용 소프트웨어 - FOSS |
|---|---|
| <ul style="list-style-type: none"> • 소스 코드 • 커뮤니티 • 카피레프트(Copyleft) • 자유(Freedom) | <ul style="list-style-type: none"> • 보증(Warranty) • 마케팅(Marketing) • 영업(Sales) • 제품 가격(Product Price) • 고용인 & 피고용인 |

〈그림6〉 FOSS와 상용 소프트웨어와의 차이점

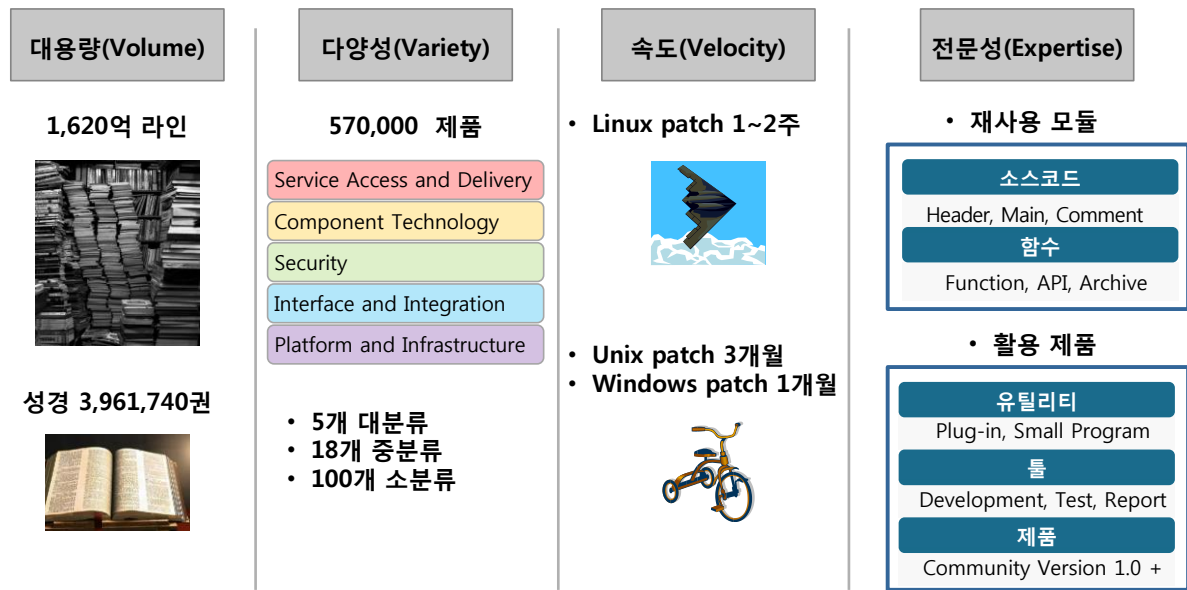
예를 들어 애드웨어는 광고의 대가로 무료로 보급된다. 셰어웨어는 마케팅을 목적으로 일정 기간 동안만 평가판 형식으로 무료로 보급된다. 프리웨어는 아무런 대가도 없이 무료로 사용할 수가 있다. 그러나 이들 세 가지 소프트웨어는 FOSS와는 달리 소스 코드를 제공하지 않는다. 또한 FOSS의 무보증 조건은 품질이나 성능 등 제품을 평가하는 지표와 상관이 없다. 만일 무보증이 아니라 보증으로 FOSS를 배포한다면 지적 재산인 소스코드를 무료로 배포하는 저작자가 그 소프트웨어의 하자나 오류까지 책임져야 하는 문제가 생기게 된다. 따라서 무보증 조건은 저작권자를 보호하기 위한 기본적인 보호 장치이지 제품의 수준을 측정하는 척도로 사용될 수 없다. 한편, FOSS 진영에만 있는 카피레프트(copyleft)는 독점적인 의미의 저작권(copyright)에 반대되는 개념이다. 저작권에 기반을 두고 사용을 제한하려는 의도가 아니라 저작권을 기반으로 소스 코드의 공유를 위한 조치이다. 카피레프트를 주장하는 사람들은 보통, 지식과 정보는 소수에게 독점되어서는 안 되며, 모든 사람에게 열려 있어야 한다고 주장한다.⁵ 그리고 상용 소프트웨어와 FOSS와의 극명한 차이점은 마케팅과 영업에 있다. 상용 소프트웨어는 제품을 출시하는 회사가 사활을 걸고 홍보 활동을 진행한다. 반면에 FOSS는 홍보 보다는 개발자 개인적인 취향과 재무적인 척도 이상의 가치에 의미를 두기 때문에 마케팅과 영업에 관심이 없을 수가 있으며, 이에 관련된 활동이 전무하거나 빈약한 상황이다. 따라서 FOSS의 사용자 입장에서는 업데이트와 패치 서비스가 필수적이며 이러한 서비스가 FOSS 기업에게는 이윤 창출의 방편이 될 수 있다.

2.1 FOSS의 특성

FOSS는 빅데이터의 특성을 그대로 보유하고 있다. 메타그룹(현재 가트너)의 애널리스트 더그 레이니(Doug Laney)는 2001년 그의 연구 보고서[6]와 관련 강의에서 데이터의 급성장에 따른 이슈와 기회를 데이터의 양(volume), 데이터 입출력의 속도(velocity), 데이터 종류의 다양성(variety)이라는 세 개의 차원으로 정의하였다. 이 “3V” 모델은 이후 가장 널리 사용되는 빅데이터의 정의가 되었다.[7]⁶ 그리고 빅데이터의 특성은 아래 그림에서 보는 바와 같이 FOSS의 특성과 그대로 연결되어 있다.

⁵ <https://ko.wikipedia.org/wiki/카피레프트>

⁶ 2012년 가트너는 기존의 정의를 다음과 같이 개정하였다: “빅데이터는 큰 용량, 빠른 속도, 그리고(또는) 높은 다양성을 갖는 정보 자산으로서 이를 통해 의사 결정 및 통찰 발견, 프로세스 최적화를 향상시키기 위해서는 새로운 형태의 처리 방식이 필요하다.”[8] 이에 더해, IBM은 진실성(Veracity)이라는 요소를 더해 4V를 정의하였고,[9] 브라이언 홉킨스(Brian Hopkins) 등은 가변성(Variability)을 추가하여 4V를 정의하였다.[10]



※ 출처 : '12.4.20 Blackduck Software

※ 행안부 FOSS TRM 분류

※ <http://kernel.org>

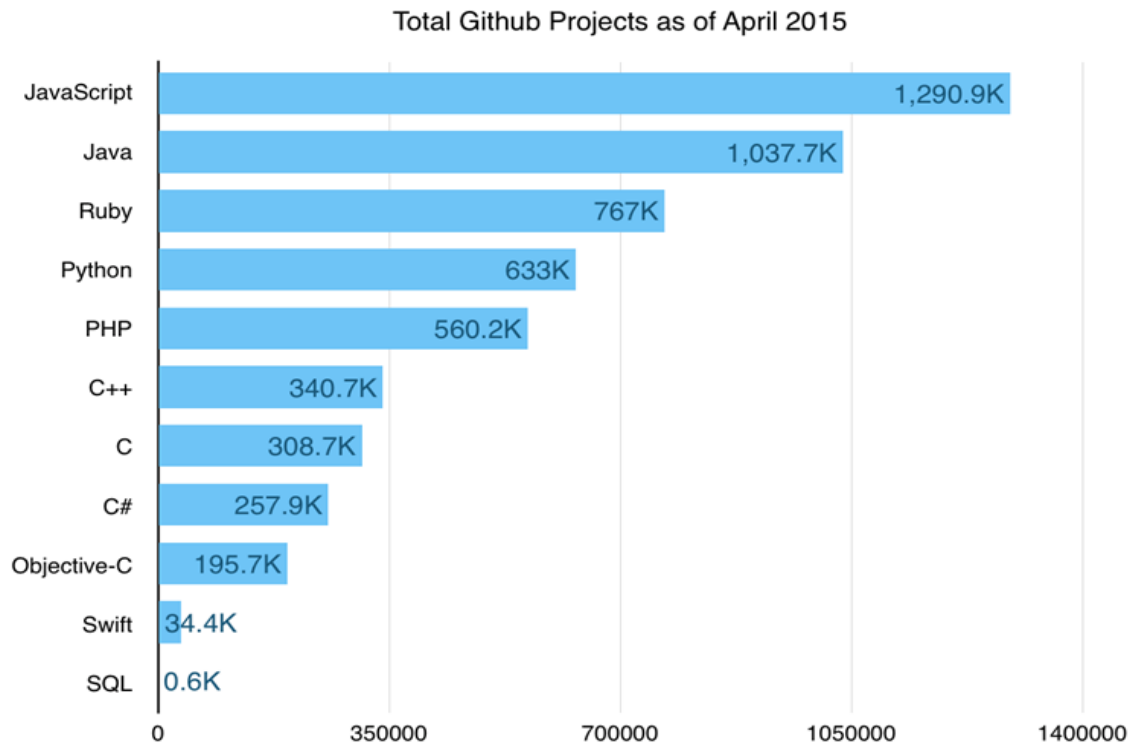
〈그림 7〉 빅데이터(Big Data)의 특성을 가지고 있는 FOSS

그러나 빅데이터와 FOSS의 큰 차이점은 전문성에 있다. 즉 FOSS는 단순히 수치로 표현되는 측정치 또는 결과 값으로 표현되는 데이터가 아니라 지적 창작물이다.

2.2.1 대용량의 특성

용량을 측정하는 단위가 대상에 따라 다양하게 존재하지만 FOSS의 용량은 소스 코드의 바이트 수, 파일의 개수, 프로젝트의 개수 등으로 구분할 수가 있다. 그 중에서도 바이트 수는 가장 정확한 용량을 측정할 수가 있지만 FOSS의 특성상 너무 오래 전에 생성되어 현재는 사용되지 않는 부분도 있고 정확한 합계가 불가능한 경우도 있으므로 프로젝트의 개수를 사용하는 방법이 현실적이다. 또한 최근에 생성되고 발전하는 프로젝트에 대한 통계가 유용하므로 다음 그림과 같이 Github에서 운영되고 있는 프로젝트의 규모를 살펴보는 작업도 의미가 있다.⁷

⁷ 출처 : <https://www.codementor.io/learn-programming/beginner-programming-language-job-salary-community>

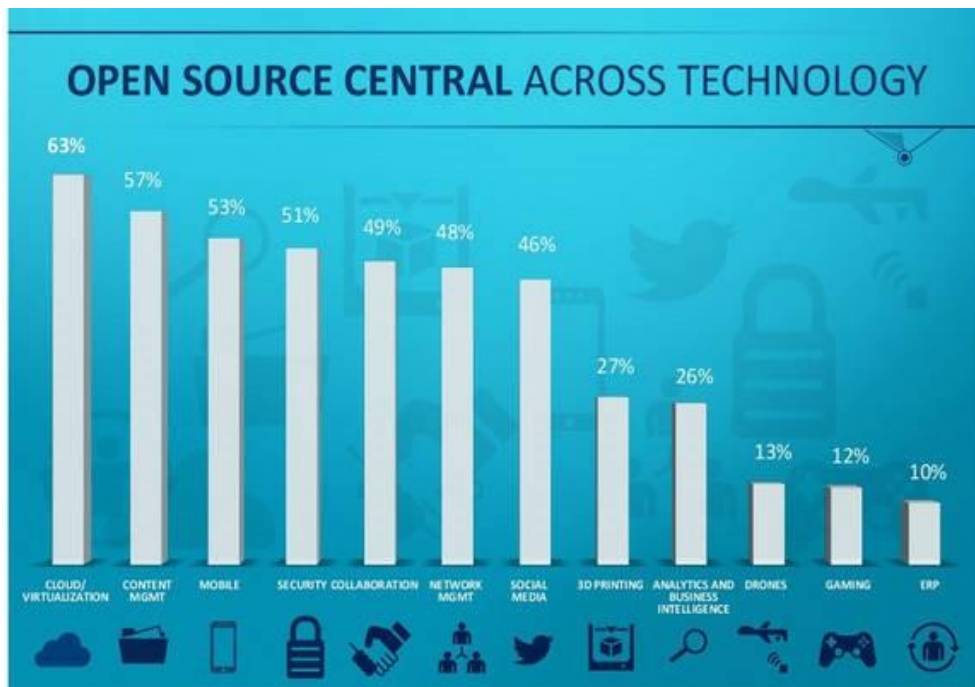


〈그림8〉 프로그래밍 언어로 살펴 본 FOSS의 규모

2.2.2 다양성의 특성

OSS의 큰 특징 중의 하나가 다양성이다. FOSS는 상용 소프트웨어와는 달리 개발자의 직장이나 소속에 제한을 두고 있지 않기 때문에 어떠한 분야에서도 커뮤니티를 통해 생산될 수 있다. 이러한 다양성은 분류체계와 각 분류 항목별 FOSS 프로젝트를 통해서 확인할 수 있지만 다음 그림과 같이 FOSS가 선도하고 있는 산업 분야를 참조하면 실무적인 현황을 파악하는데 도움이 된다.⁸


⁸ 출처 : Black Duck Software (2014)



〈그림9〉 FOSS 기술이 선도하고 있는 주요한 기술 산업 분야

2.2.3 변화의 신속성

변하는 속도 관점에서는 FOSS 만큼 빠른 소프트웨어는 없을 것이다. 물론 특정한 제품군이나 제품 하나에 대해서 패치 업데이트 회수나 메이저 버전이 상승하는 속도가 대단히 빠른 상용 소프트웨어도 있겠지만 이들이 FOSS 진영에서 급속히 발전하고 있는 프로젝트를 따라 갈 수는 없다. FOSS가 출현할 당시부터 현재까지 리눅스는 매주에 한번 꼴로 패치가 배포되고 있으며, 전체적으로 보더라도 매년 축적되는 프로젝트는 450개 이상이 되고 있다. 또한 발전 속도 측면에서도 연평균 축적 성장률(CAGR)이 약 75%에 이르고 있을 만큼 급속한 속도로 증가하고 있다. 특히 아래 그림에서 확인 할 수 있는 바와 같이 클라우드 컴퓨팅 분야에서 최근 다양한 기술이 다양한 커뮤니티로부터 속출하고 있다.



| Position | Project | Categorie | Founded |
|----------|---------------|----------------|---------|
| 1 | OpenStack | Infrastructure | 2010 |
| 2 | Cloud Foundry | Platform | 2011 |
| 3 | KVM | Virtualization | 2007 |
| 4 | Docker | Virtualization | 2013 |
| 5 | Apache Mesos | Infrastructure | 2012 |
| 6 | MongoDB | Database | 2009 |
| 7 | Puppet | DevOps | 2005 |
| 8 | Chef | DevOps | 2009 |
| 9 | OpenShift | Platform | 2011 |
| 10 | Jenkins | DevOps | 2011 |
| 11 | Ceph | Storage | 2011 |
| 12 | Salt | DevOps | 2011 |
| 13 | CloudStack | Infrastructure | 2010 |
| 14 | CoreOS | Infrastructure | 2014 |
| 15 | CouchDB | Database | 2005 |

crisp
RESEARCH

〈그림10〉 클라우드 컴퓨팅 분야에서 FOSS의 변화 신속성

2.3 FOSS의 성공 사례

FOSS 기술 중 대표적인 성공사례를 설명하기에 앞서 FOSS에 있어 성공이란 무엇인지 먼저 정하고 들어가야 한다. 상용 소프트웨어에서의 성공은 당연히 매출, 이익, 시장점유율 등의 지표로 평가될 수 있다. 그러나 FOSS 관점에서는 성공한 프로젝트를 구분하는데 여러 가지 차이점이 존재한다. 이러한 차이점은 저작자의 사용조건만 준수해 준다면 가격 지불 없이 무료로 사용할 수 있고 마음대로 고치고 누구에게나 나누어 줄 수 있는 공유 자원이라는 점에서 극명하게 구분된다. 또한, 모든 FOSS는 저작자를 보호하기 위해서 무보증(no warranty) 조건으로 배포됩니다. 따라서 상용 소프트웨어와 동일한 자재로 FOSS의 성공 여부를 판단할 수는 없게 된다. 이러한 차이점으로 인해서 가격과 시장이 아닌 다른 척도를 사용해야만 한다. 즉, FOSS의 평가 척도로는 FOSS 자체의 품질과 성능을 측정할 수 있는 호환성, 신뢰성, 안정성, 확장성, 보안성, 처리 속도 등이 사용되겠지만, FOSS 외적인 수준을 평가할 수 있는 커뮤니티 성숙도, 고객 만족도, 기술지원 서비스, 성장성, 영속성 등이 추가적으로 적용되어야 한다. 지금까지 알려진 FOSS 평가 모델의 평가 방법론에 의해서 평가 점수를 계산한 결과 Linux, Apache, Android, Open Stack, Eclipse, MySQL, MongoDB 정도가 좋은 성적표를 얻고 있으며 대표적인 성공 사례의 반열에 올라와 있다. 먼저 Linux는 프로그램의 실행 환경을 제공하는 운영 체제(operating system)로써 슈퍼컴퓨터에서 스마트 폰 단말기에 이르기 까지 어떠한 제품도 따라올 수 없는 다양한 설치 플랫폼을 자랑하고 있다. 무엇보다도 사용자, 개발자, 플랫

품, 패치, 성숙도, 적용성 등 어떠한 평가 척도에서도 단연 최고봉이라 할 수 있으며 소프트웨어 산업계에서 사실적인 표준으로 자리 잡고 있다. David A. Wheeler는 이 FOSS의 가치가 얼마나 큰지 설명하기 위해서 2008년 현재 25조원이라는 가격적인 수치로 환산해 놓은 바 있다. 따라서 현재 Linux의 가치는 그 이상의 엄청난 수준이라고 말할 수 있을 것이다. Apache는 인터넷 상에서 웹 페이지를 접근할 수 있도록 HTTP(HyperText Transfer Protocol)를 지원하는 웹 서버(web server)로써 1999년부터 Apache Software Foundation에서 개발되어 지금은 이 분야에서 전 세계적으로 가장 큰 점유율을 자랑하고 있다. 또한 이 프로젝트가 속해 있는 Apache Software Foundation은 다양한 하부 프로젝트를 거느리고 있어서 그 규모와 운영 체계가 가장 건강하고 발전적인 커뮤니티 중 하나로 손꼽히고 있다. 한편 Android는 Linux 기반의 모바일 플랫폼으로써 최근 모바일 스마트 폰에 탑재되면서 애플사의 iOS를 능가하는 사실 표준 플랫폼으로 각광받고 있다. 무엇보다도 마이크로소프트웨어사의 Window CE를 대체할 수 있었기 때문에 한 대당 15달러 이상을 상회하는 상용 라이선스 비용을 절감하는데 결정적인 역할을 담당하였다.

| Top 15 Open Source Cloud Computing Technologies 2014 | | | |
|--|---------------|----------------|---------|
| Position | Project | Categorie | Founded |
| 1 | OpenStack | Infrastructure | 2010 |
| 2 | Cloud Foundry | Platform | 2011 |
| 3 | KVM | Virtualization | 2007 |
| 4 | Docker | Virtualization | 2013 |
| 5 | Apache Mesos | Infrastructure | 2012 |
| 6 | MongoDB | Database | 2009 |
| 7 | Puppet | DevOps | 2005 |
| 8 | Chef | DevOps | 2009 |
| 9 | OpenShift | Platform | 2011 |
| 10 | Jenkins | DevOps | 2011 |
| 11 | Ceph | Storage | 2011 |
| 12 | Salt | DevOps | 2011 |
| 13 | CloudStack | Infrastructure | 2010 |
| 14 | CoreOS | Infrastructure | 2014 |
| 15 | CouchDB | Database | 2005 |

〈그림11〉 톱 15 오픈소스 클라우드 컴퓨팅 기술 순위

위의 그림에서 확인할 수 있는 바와 같이 OpenStack의 명성은 다양한 언론 매체를 통해 수 없이 발표되고 있었지만 그 수준을 넘어 참여하고 있는 기업과 재정적 후원 규모 면에서 최근 뚜렷한 강세를 보이고 있으며 소스 코드의 증가 속도는 단연 선두를 달리고 있는 상황이다. Eclipse는 IBM이 소스코드를 공개한 개발 플랫폼으로써 전 세계 개발자들에게는 표준과 같은 개발 환경을 제공하고 있다. MySQL은 Sun

Microsystems가 인수하고 다시 Oracle에 의해서 인수될 정도로 관계형 데이터베이스 분야에서 가장 지명도가 높은 FOSS 중 하나이다. MongoDB는 최근 클라우드 컴퓨팅과 빅데이터가 발전하면서 인기를 얻게 된 NoSQL 데이터베이스이다. 너무나 크고 다양하고 빠르게 변하고 복잡한 FOSS 분야를 몇몇 안 되는 대표주자로 간략하게 살펴보았지만 분명히 예측이 가능한 사실은 앞으로 FOSS는 최첨단 기술만이 살아남는 모바일, 클라우드, 빅데이터, IoT 분야에서 핵심기술의 원천으로 활약하는 미래이다.[10] 더욱이 아래 그림에서 보는 바와 같이 세계적인 ICT 기업들이 FOSS를 기존 사업의 경쟁력 증대와 신규 사업 발굴에 성공적으로 활용함으로써 지금까지는 관망세였던 상용 소프트웨어 진영에서도 FOSS를 전략적으로 사용하기 위해서 커뮤니티에 적극적으로 참여하는 양상을 보이고 있다.




〈그림12〉 글로벌 ICT 기업의 FOSS를 통한 성공 사례

2.4 FOSS 라이선스의 분류

일반적으로 라이선스는 저작권자가 본인의 작품(Work)를 일정 기간 상업적인 용도로 사용할 수 있도록 허락하는 권리로 해석된다. FOSS의 경우에는 저작자가 만든 소프트웨어를 어떻게 사용할 수 있는지 그 조건과 용도를 라이선스로 규정하고 FOSS 내부에 명시한 후에 배포하고 있다. 이 FOSS 라이선스를 크게 둘로 나누면 FOSS를 인용 또는 연결을 통해 제작한 소프트웨어의 소스 코드를 반환해야 하는 Reciprocal License와 반환할 필요가 없는 Permissive License로 분류된다.⁹ 아래 그

⁹ GPL : General Public License



림에서 보는 바와 같이 반환의 의무가 엄격하게 적용되는 라이선스로는 AGPL, GPL, LGPL 등이 대표적이며 이들 GPL 계열은 모두 Free Software 진영에서 발표되었다. Free Software의 상업적 용도에 관심이 모아면서 OSI(Open Source Initiative)에서는 반환의 의무가 없는 라이선스를 포함할 수 있도록 Open Source Software라는 용어와 조건을 만들어 내었다.¹⁰ 이 단체에서는 약 70개 정도의 인증 라이선스를 선정해 놓고 있으며 이 라이선스 목록에는 다수의 permissive license가 포함되어 있다.¹¹

AGPL : Affero General Public License

LGPL : Lesser General Public License

EPL : Eclipse Public License

QT: Trolltech Q library

MPL : Mozilla Public License

OSL : Open Software License

CPL : Common Public License

IBM: IBM License

BSD : Berkeley Software Distribution

Apache: Apache Public License

MIT : Massachusetts Institute of Technology

W3C : World Wide Web Consortium

¹⁰ (1) 자유 배포(Free Redistribution)

(2) 소스 코드(Source Code)

(3) 2차적 저작물(Derived Works)

(4) 소스 코드 수정의 제한(Integrity of The Author's Source Code)

(5) 개인이나 단체에 대한 차별 금지(No Discrimination Against Persons or Groups)

(6) 사용 분야에 대한 제한 금지(No Discrimination Against Fields of Endeavor)

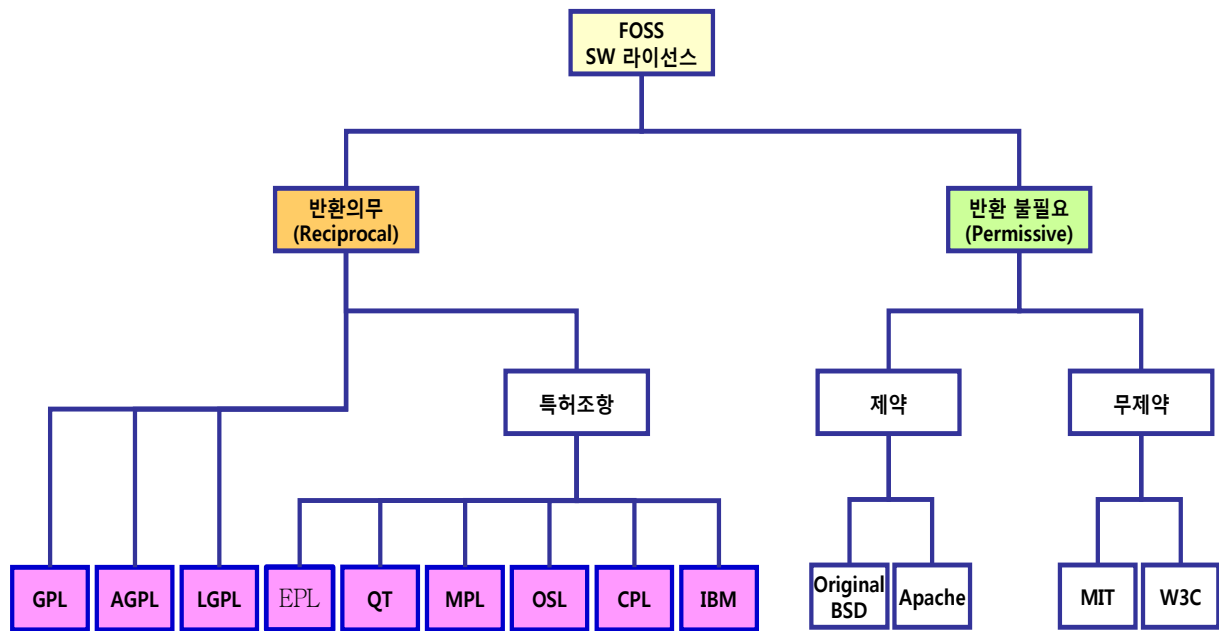
(7) 라이선스 배포(Distribution of License)

(8) 라이선스 적용상의 동일성 유지(License Must Not Be Specific to a Product)

(9) 다른 라이선스의 포괄적 수용(License Must Not Contaminate Other Software)

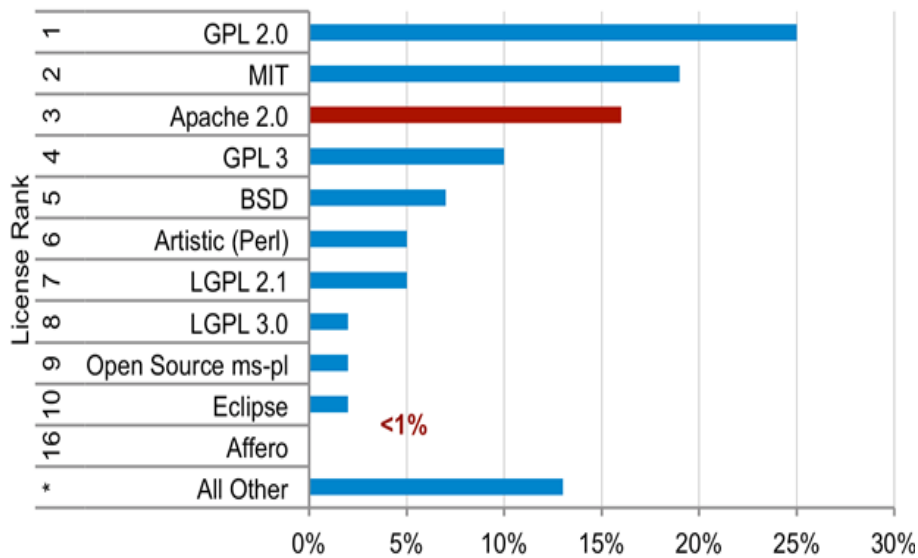
(10) 라이선스의 기술 중립성(License Must be Technology-Neutral)

¹¹ <https://opensource.org/licenses/alphabetical>



〈그림13〉 FOSS 라이선스의 구조

아래 그림에서 보는 바와 같이 FOSS 라이선스 별 프로젝트 현황을 살펴보면 GPL 계열이 약 50%의 비중을 차지하고 있다. 이는 프리 소프트웨어 운동이 시작되면서 모든 프리 소프트웨어 라이선스가 GPL 계열로 배포되었기 때문이다. 이후 오픈소스 소프트웨어가 출현하기 전까지 약 15년의 기간 동안 프리 소프트웨어 라이선스가 축적된 결과이다.



〈그림14〉 라이선스 별 FOSS 적용 현황 (2015년)

2.5 FOSS 라이선스의 탄생과 발전 방향

현재 소스 코드를 제공하는 소프트웨어는 프리 소프트웨어와 오픈소스 소프트웨어 밖에는 없다고 해도 과언은 아니지만 두 소프트웨어는 분명한 차이가 있다. 먼저, 프리 소프트웨어는 소스 코드의 공유와 개방을 통한 협업을 중시한다. 그렇기 때문에 누군가에 의해 개작된 프리 소프트웨어의 소스 코드는 반드시 공유되어야 하며, 개작물의 회수를 위한 반환의 의무가 강력하게 요구되고 있다. 즉, 프리 소프트웨어는 누구나 무료로 소스 코드를 공부하고, 복제하고, 바꾸고, 나누어 쓸 수 있는 자유를 보장해 주는 반면에 변경된 부분이나 연결된 부분에 대해서는 동일한 실행 코드를 만들거나 개작할 수 있도록 필요한 모든 소스 코드를 제공하도록 요구하고 있는 것이다. 이러한 요구 사항에 강제성을 더하기 위해서 프리 소프트웨어 진영은 반환의 의무를 라이선스 규정에 명시할 수밖에 없었으며, 이로 인해 GPL(General Public License)이 최초의 프리 소프트웨어 라이선스로 선포되었다. 그 이후 프리소프트웨어 진영은 라이브러리에 대해서는 공개의 의무가 없는 LGPL(Lesser General Public License)와 통신 인터페이스로 연결만 되어도 소스 코드를 공개해야 하는 AGPL(Affero General Public License) 등을 추가로 발표하면서 프리 소프트웨어 사용자의 요구 변화를 수용하였다.

반면에 오픈소스 소프트웨어는 오픈소스 소프트웨어 협의회 역할을 수행하고 있는 OSI(Open Source Initiative)에 의해 탄생하면서 복제, 수정, 배포 등 기본적인 자유는 프리 소프트웨어와 동일하게 보장하고 있지만, Apache, BSD, MIT와 같이 소스 코드의 반환을 요구하지 않는 라이선스도 포함시킴으로써 반환과 공유보다는 확산과 보급에 치중하는 양상을 보이고 있다. 이러한 변화는 오픈소스 소프트웨어의 확산과 보급에 따른 사용자의 저변 확대에 의하여 기술지원, 유지보수, 교육, 컨설팅 등 다

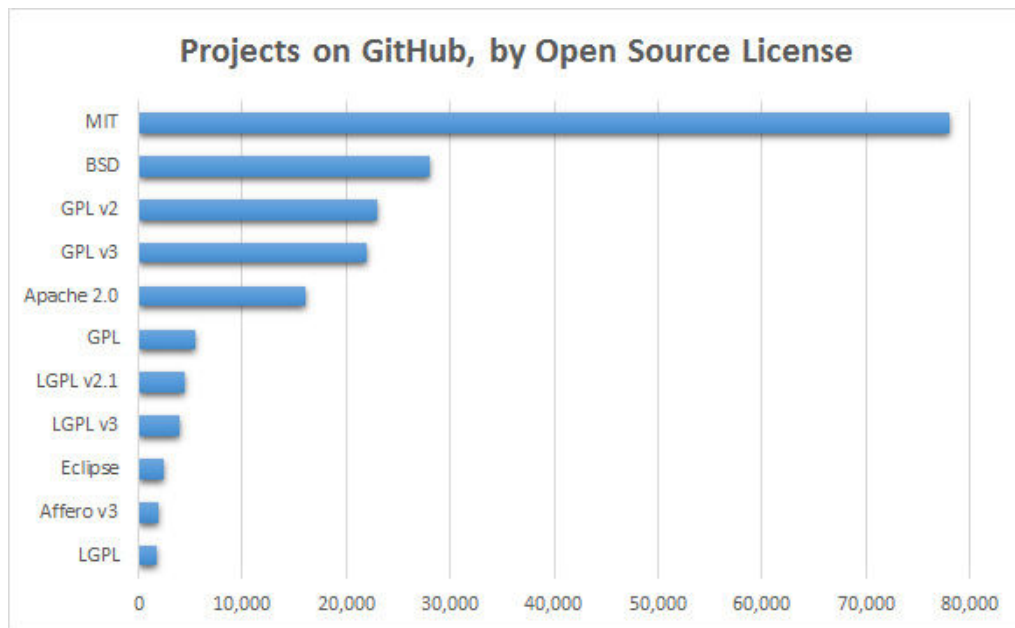
양한 요구 사항의 증대와 이로 인한 사업적 기회 창출이라는 암묵적인 의도에서 비롯되었다고 볼 수 있다.

| Free Software 진영 | | Open Source Software 진영 |
|--|--------|--|
| <ul style="list-style-type: none"> • 공유 / 협업 / 반환 / 회수 | 차이점 | <ul style="list-style-type: none"> • 보급 / 확산 / 허용 / 활성화 |
| <ul style="list-style-type: none"> • GNU 프로젝트의 소스 코드를 GPL 라이선스로 배포함 | 1983 년 | |
| <ul style="list-style-type: none"> • Linux의 소스코드가 GPL 2.0 라이선스로 배포됨 | 1991 년 | |
| | 1993 년 | <ul style="list-style-type: none"> • Red Hat 창립 • 가입자 사업 모델로 기술지원 서비스를 시작함 |
| | 1998 년 | <ul style="list-style-type: none"> • OSI(Open Source Initiative) 창립 • 허용 라이선스가 포함됨 |
| <ul style="list-style-type: none"> • SFLC가 GPL 3.0 라이선스를 발표함 | 2013 년 | |

Open Source Software 진영은 소스코드 배포라는 공통점에 착안하여 Free Software의 Free 를 Open Source로 대체함

〈그림15〉 프리 소프트웨어와 오픈소스 소프트웨어의 탄생 배경

앞의 그림은 오픈소스 소프트웨어 진영이 기존의 프리 소프트웨어를 대체하는 용어를 만들어 낸 의도와 오픈소스 소프트웨어가 출현하게 된 배경을 설명해 주고 있다. 1998년 OSI(Open Source Initiative)가 출범한 당시에는 소프트웨어 내에 포함된 지적재산 까지도 공개해야 하는 반환의 의무가 상용 제품을 개발하여 판매하는 기업에게는 큰 부담이 아닐 수 없었다. 레드햇(Red Hat)과 같이 프리 소프트웨어를 기반으로 기술지원 서비스를 사업 모델로 개척하고 있는 상황에서는 공짜를 의미하는 프리(free)는 수익 모델을 방해하는 걸림돌이 될 수밖에 없었다. 즉, 프리 소프트웨어가 왕성하게 확산되고 보급되어야 기술 지원 서비스에 대한 니즈가 생길 기회가 있었으나 GPL과 같이 반환 조건이 강력한 Copyleft 라이선스와 프리라는 용어는 기술 지원 서비스를 중심으로 사업을 추진하는데 휘방꾼이 될 수밖에 없었다. 따라서 사업적 관점에서는 반환이 불필요한 허용(permissive) 라이선스가 필요하였고 유료 기술 지원 서비스에 방해가 되는 프리라는 말을 대체할 당위성이 제기되었다. 결국 OSI는 소프트웨어의 소스 코드를 공개한다는 공통점에 착안하여 반환의 의무가 없는 라이선스까지 포함할 수 있도록 오픈소스 소프트웨어라는 용어를 만들게 되었다는 해석이다.



〈그림16〉 최신 인기 FOSS 개발 플랫폼을 통해 살펴 본 FOSS 라이선스의 분포

이러한 오픈소스 소프트웨어의 라이선스 발전 현상은 GitHub의 FOSS 라이선스의 분포에서 확인할 수 있다.¹² 즉, 전체 라이선스에서는 프리 소프트웨어 진영의 라이선스가 우세를 점하고 있지만 최근 MIT와 같이 오픈소스 소프트웨어 진영의 permissive 라이선스가 급격히 성장하고 있어 오픈소스 소프트웨어의 라이선스가 강세를 보이고 있다. 즉, 신규 FOSS 프로젝트가 모이고 있는 GitHub에서 허용 라이선스가 두각을 나타내고 있는 현상은 산업계에서 다분히 비즈니스 관점으로 FOSS를 접근하고 있는 시대적 추세를 반영하고 있다고 판단된다. 그러므로 프리 소프트웨어의 취지가 지나치게 상용화로 물들게 되는 현상을 우려하고 있는 프리 소프트웨어 진영은 이타주의 정신과 금전적 이익 관점에서 오픈소스 소프트웨어 진영과 불편한 관계를 형성하는 일이 종종 발생하고 있는 것이다.

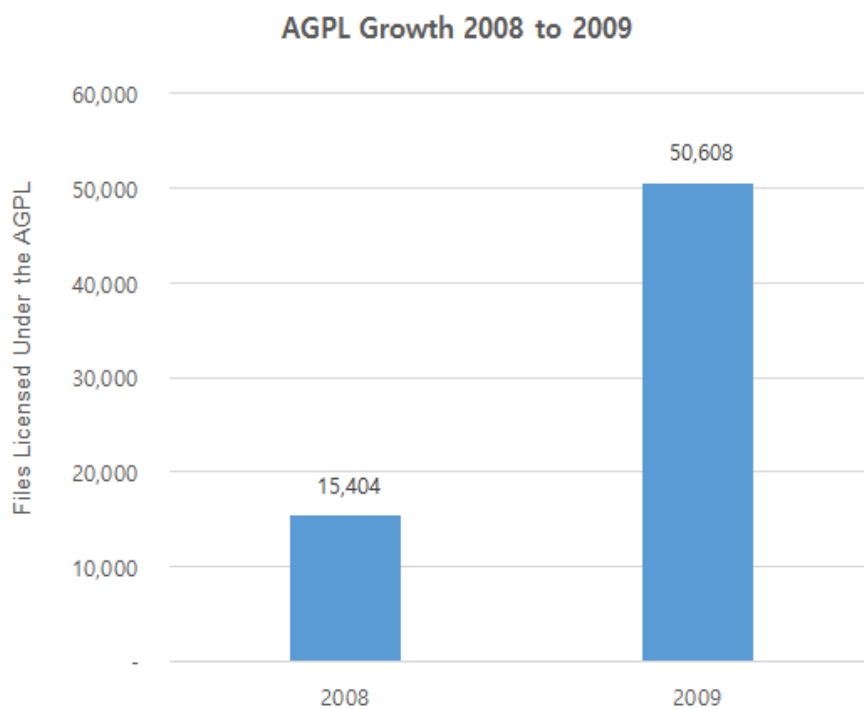
2.6 AGPL을 통한 프리 소프트웨어 진영의 견제

Affero GPL은 GPL 계열 중에서 가장 반환의 의무가 강력한 라이선스이다. LGPL의 경우에는 라이브러리의 형태로 연결되면 본 프로그램의 소스코드를 공개할 의무가 없다. GPL의 공개 범위는 LGPL 보다 넓어서 라이브러리를 포함하여 일체로 실행되는 프로세스의 모든 소스 코드를 공개하여야 한다. 물론 소스코드의 공개를 원하지 않는다면 별도의 프로세스로 분리하는 방법으로 의무 사항을 피할 수 있다. 그러나 AGPL은 별개의 독립된 프로세스라 할지라도 네트워크 인터페이스를 통해서 통신하게 되면 상대방 소스코드도 공개해야 하는 의무가 생긴다. 즉 모바일 앱이 AGPL인

¹² 출처 : http://www.theregister.co.uk/2013/04/18/github_licensing_study/ Aaron Williamson

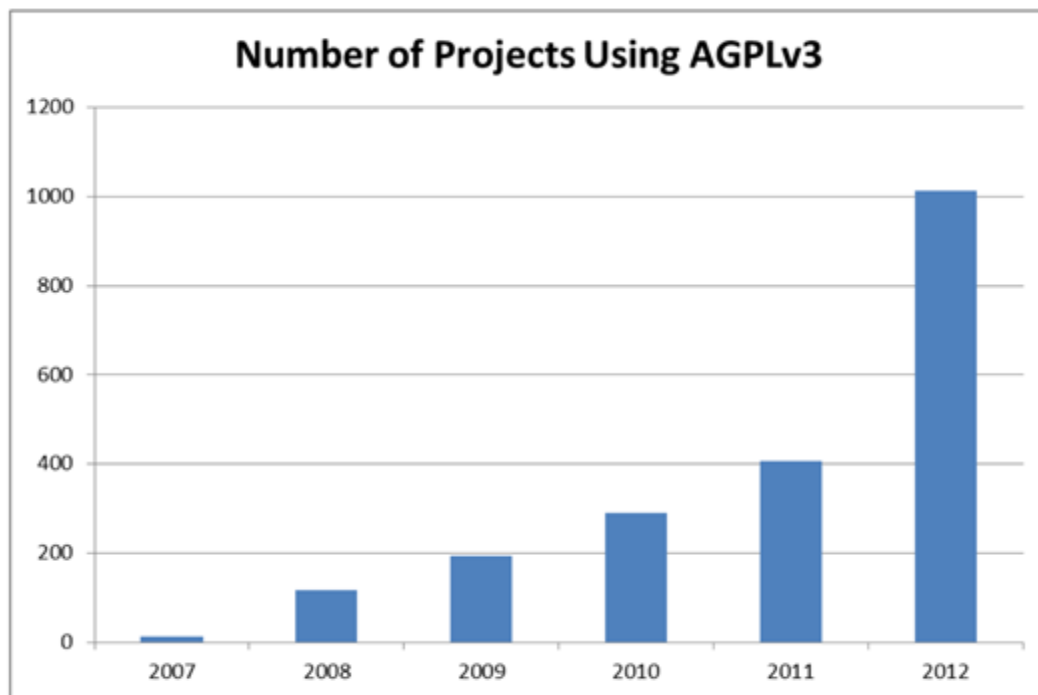
경우 이 앱에게 인터넷을 통해 서비스를 제공하는 서버의 소스코드도 공개해야 하는 것이다. 이러한 라이선스 규약은 아마도 웹 기반의 서비스를 제공하면서 엄청난 수익을 가져가고 있는 기업들이 FOSS를 사용해서 축적한 부를 사회에 환원하지 않고 이기적인 사업 행태를 보이고 있는 공룡 기업에 대한 견제와 규제로 해석될 수 있겠다.

아래 그림에서 확인 할 수 있는 바와 같이 과거 상황을 살펴보면 2008년에는 15,404 개 파일에 불과했던 AGPL 라이선스가 2009년에는 50,608파일로 급성장한 내역을 확인할 수 있다.¹³



〈그림17〉 초기 Affero GPL의 발전 추세

¹³ 출처 : Your Work in Open Source, the Numbers published by OSCON in July, 2009



〈그림18〉 지속적인 Affero GPL의 발전 추이

위 그림에서 확인 할 수 있는 바와 같이 이러한 AGPL의 성장세는 멈추지 않고 지속되어서 2014년 현재 1,266개의 AGPL 프로젝트가 등록되어 있는 상황이다.¹⁴

2.7 FOSS 저작권 관련 기술

FOSS의 저작권은 상용 소프트웨어 비해서 다양하고 복잡하다. 상용 소프트웨어의 경우에는 보통 하나의 소프트웨어 제품에 단일한 저작권이 부여된다. 반면에 FOSS는 소스코드에 저작권을 명시할 수 있으므로 최대 파일 개수만큼 저작권 선언이 가능하다. 이는 단순히 저작권자가 많아진 정도가 아니라 저작권자가 사용한 라이선스에 따라 상이한 사용 규정이 더욱 얹히고 설켜서 엄청나게 복잡해졌다는 의미이다. 더구나 소스코드는 복제가 가능하고 부분적으로 인용할 수 있으므로 하나의 파일에 여러 개의 FOSS 라이선스가 포함되는 경우도 있다. 이렇듯 공유와 협업 환경에서 무작위적으로 유통되고 있는 FOSS의 특성상 저작권 보호와 라이선스 준수는 최초 원본 파일을 찾는 작업이 관건이다. 일반적으로 프로그램 파일의 주석문에 저작권과 라이선스가 선언되어 있지만 이러한 정보가 유실되었거나 훼손된 경우에는 원본 파일과 라이선스를 식별하기가 매우 어려워진다. 따라서 아래 표와 같이 이러한 문제를 해결하는 기술이 등장하게 되었다.

¹⁴ 출처 :

https://www.blackducksoftware.com/files/webmedia/_webinars/2013_06_25_Understanding%20the%20LGPL%20and%20AGPL.pdf



| FOSS 저작권 관련 기술 | 기술 항목 |
|----------------|-------------------------|
| 라이선스 복원 기술 | 저작권 선언문 및 라이선스 정보의 원복 |
| 라이선스 계보 제작 기술 | 생성 일자에 따른 대표 라이선스 추적 |
| 자동 분류 기술 | 프로젝트의 기능과 용도에 따른 자동식별 |
| 자동 추천 기술 | 프로젝트의 성숙도와 적용성 기반 자동 평가 |

〈표2〉 FOSS 저작권 관련 기술의 분류

이들 기술들이 차지하는 비중으로 본다면 FOSS 라이선스 복원기술 분야는 약 29%, FOSS 라이선스 계보 제작 기술분야는 약 18.0%, FOSS 자동 분류 기술분야는 약 27%, FOSS 자동 추천기술 분야는 약 26%에 이르고 있다. 또한 특허 관점에서 이들 기술의 현황을 파악해 보면 다음 표와 같다.

| FOSS 저작권 관련 기술 | 한국 특허 | 미국 특허 | 일본 특허 | 유럽 특허 | 합계 |
|----------------|-------|-------|-------|-------|-----|
| 라이선스 복원 기술 | 7 | 79 | 12 | 8 | 106 |
| 라이선스 계보 제작 기술 | 7 | 53 | 3 | 4 | 67 |
| 자동 분류 기술 | 5 | 87 | 7 | 3 | 102 |
| 자동 추천 기술 | 4 | 81 | 10 | 3 | 98 |

〈표3〉 FOSS 저작권 기술의 특허 등록 현황

3장 결론

FOSS가 무료로 자유롭게 사용되기 때문에 FOSS에 컴플라이언스 위험이 없다고 말할 수 없다. 이는 FOSS에도 엄연히 저작권이 있기 때문이다. 그렇다고 이러한 법적 이슈 때문에 FOSS의 사용을 꺼려한다면 애초에 프리 소프트웨어 진영이나 오픈소스 진영에서 소스 코드를 공유하고 자유롭게 협업하자는 취지를 전혀 이해하지 못하고 거꾸로 가는 꼴이 된다. 운전 중에는 차선과 신호를 지키고 보행 중에는 침을 뱉거나 쓰레기를 버리지 말아야 한다는 기본적인 의무사항이 무서워 차를 몰지 않거나 길거리에 나가지 않는다면 참으로 우스운 일이 아닐 수 없다. FOSS의 컴플라이언스는 저작자를 보호하고 프리 소프트웨어 운동(movement)을 보장하기 위해 만들어 놓은 최소한의 의무사항이다. 따라서 FOSS 진영에서 저작권을 보호하는 사람들을 만나면, 어떻게 해서든 증거를 찾아서 위반자를 소탕하겠다는 의도가 아니라 본연의 이타주의 정신이 훼손되지 않도록 라이선스에 명시된 기본적인 사용 조건 정도는 지켜 주기를 바라는 정도로 기대하는 수준이다. 실제로 OSCON, Linux Conference, OpenStack Conference, Open Source Business Conference 등 세계적인 FOSS 행사를 보아도 법무 세션이 발표 또는 토의 트랙에 잡혀 있는 경우도 드물고, 혹 순서가 잡혀 있더라도 발표 그룹과 청중의 수가 별반 크게 차이가 나지 않을 정도로 참여가 미진하다. 그렇다고 법은 법인데 아무리 단속이 어렵고 발각되기도 쉽지 않더라도 법적 의무 사항을 간과할 수는 없는지라 어느 정도 수준에서 비용도 아끼고 안전한 수준으로 컴플라이언스 체제를 유지할 수 있는지가 가장 큰 관건이다. 불행하게도 최적의 컴플라이언스 예방과 관리 수준은 누구도 알 수 없다. 그러므로 사용자 또는 회사의 정책과 기준에 의해서 자체적인 수준을 결정해야 한다. 필자가 최적의 컴플라이언스 예방과 대응 방안을 제시한다면 기본은 사람이다. 즉, 최소한 1명을 컴플라이언스 담당자로 지정하고 있어야 한다. 물론 전담자일 필요는 없지만 컴플라이언스가 상시 업무로 지정되어 있어야 한다. 또한 이 담당자는 외부적으로는 대표 창구 역할도 수행하여야 한다. 그렇게 하기 위해서는 외부 채널로 사용하는 이메일이 이 담당자에게 바로 전송되도록 계정을 할당해 놓아야 한다. 한편, FOSS 라이선스 식별은 사람도 가능하지만 소스 코드의 양이 클 경우에는 실수 방지와 시각 절약을 위해서 컴플라이언스 툴을 사용해야 한다. 이 툴의 종류에는 무료의 커뮤니티 버전도 있고 유료의 상용 버전도 있지만 가장 중요한 사항 중 하나는 어떤 툴을 사용하든 GPL 계열의 라이선스 식별과 대응에 가장 유의하고 집중해야 하는 사안이다. 먼저, GPL 라이선스 하에서 프로그램을 내부적으로 사용하면 문제가 없겠지만 배포를 염두에 두고 있다면 GPL과 별개의 프로그램으로 분리하여야 한다. LGPL의 경우라면 소스 코드를 사용하지 말고 라이브러리로 분리해야 안전하다. 소스 코드의 반환의무가 가장 강력한 AGPL을 사용하는 경우에는 반드시 소스 코드의 배포를 전제하고 있어야 한다. 이러한 대응 방안은 각기 다른 회사의 규모와 사업의 특성에 따라서



알맞게 컴플라이언스 체제를 구축해 놓아야 한다. 이러한 체제를 구비하고 있다는 사실 하나만으로도 외부 고발이나 클레임(claim)을 대항할 수 있으며 설사 고의적인 소송에 휘말리더라도 악의가 없는 실수로 인정받을 수 있게 된다. 더욱이 프리 소프트웨어와 오픈소스 소프트웨어 진영에서 활동하고 있는 국제적인 전문가 또는 전문기관에게 도움의 손길을 요청하면 효과적인 지원을 받을 수 있으므로 평상시에 우호적인 관계를 유지하는 일이 매우 중요하다. 이렇듯 법적인 문제로부터 자유로워야만 이 더욱 적극적으로 FOSS를 활용해서 사업 경쟁력과 개발 역량을 한껏 끌어 올릴 수 있는 기반을 마련할 수 있을 것이다.

■ 참고문헌

- [1] <https://gnu.org/philosophy/free-sw.html>
- [2] Richard Stallman, “Richard Stallman and the Free Software Movement“, The International University College of Turin, December 2010, pp.6~8
- [3] Bruce Perens, “The Open Source Definition”, <http://perens.com/OSD.html>, 2008
- [4] Gacek ,C.& Arief, “The many meanings of open source”, IEEE Software, vol.21, no.1, pp.34, 2004,
- [5] “What is Big Data?”. Villanova University.
- [6] Laney, Douglas. “3D Data Management: Controlling Data Volume, Velocity and Variety” (PDF). Gartner. 2001년 2월 6일.
- [7] Beyer, Mark. “Gartner Says Solving 'Big Data' Challenge Involves More Than Just Managing Volumes of Data”. Gartner. 2011년 7월 10일
- [8] Laney, Douglas. “The Importance of 'Big Data': A Definition”. Gartner. 21 June 2012에 확인함.
- [9] Brian Hopkins; Boris Evelson (2011). “Expand Your Digital Horizon With Big Data”. Forrester Research Inc. 2013년 12월 17일에 확인함.
- [10] Jeremy Bennett, “Open Source in Embedded System Development”, Presented at the Embedded Masterclass, 2011

■ 용어 정의

개정 (Revision):

모든 개정, 변경, 증보 및 삭제에 포함하여 고쳐 쓰고 원판과 바꾸기 위해 다시 인쇄한 출판물.

거버넌스(governance):

오픈소스의 활용 가치를 극대화시키고 위험 수준을 최소화시키기 위해 정책, 프로세스, 조직, 시스템 등을 관리 통제하는 전사적인 체제

라이선스 검증(Verification):

소스코드 분석, 라이선스 identify, 결과 확인까지의 오픈소스 라이선스를 검증하는 일련의 작업

고지:

소송법에서, 법원이 결정 사항이나 명령을 당사자에게 알리는 일

* 게시나 글을 통하여 알림

공개(대상):

오픈소스 SW의 공개 대상은 라이선스별로 차이가 있으며 일반적인 공개 대상자는 오픈소스 SW를 공급(배포) 받은 자를 의미

오픈소스 SW는 라이선스에 따라서 오픈소스를 사용하여 수정한 부분이 있을 때 해당 부분의 소스코드를 공개하여야 한다고 명시하는 경우가 있으며, 공개 의무는 오픈소스 SW를 배포함으로써 공개의 의무가 발생하고 공개 범위는 각각의 라이선스에서 정하고 있는 범위에 따라 달라질 수 있다.

라이선스 (License) :

권리자가 다른 사람에게 일정한 내용을 조건으로 하여 특정행위를 할 수 있는 권한을 부여하는 행위

라이선서 (Licensor) :

저작권 소유자 혹은 저작권 소유자의 위임을 받아 라이선스를 부여하는 자

라이선시 (Licensee) :

라이선시란 권리를 대여 받는 자를 지칭하는 것이다. 라이선시는 라이선싱 계약을



통해서 특정 제품을 생산, 사용, 양도, 판매, 전시 등을 할 수 있고 그 보상으로 로열티를 지급한다. 라이선싱으로 부여받은 권리는 법적으로 독점성·배타성이 보장되는 법적 권리로 독점적 이익을 얻을 수 있다.

라이선스 충돌:

두 개 이상의 FOSS 라이선스 내에 명시되어 있는 사용 조건이 서로 상반된 상태로써 특히 배포 이전의 상황을 말 한다

라이선스 위반:

라이선스가 충돌된 상태로 배포된 상황을 말 한다

라이선스 양립성:

두 라이선스의 요구 조건이 서로 달라서 배포하는 것이 불가능하게 되는 문제로 함께 사용이 가능하면 compatible, 사용이 불가능하면 incompatible 라고 한다.

(예를 들어, SW를 작성하고자 할 경우 기존에 만들어진 코드를 재사용하거나 결합하는 경우가 많은데, 결합되는 각 코드의 라이선스가 상호 상충되는 경우가 있다. 실질적으로 MPL 조건의 A코드와 GPL 조건의 B코드를 결합하여 'A+B'라는 프로그램을 만들어 배포하고자 하는 경우, MPL은 'A+B'의 A부분을 MPL로 배포할 것을 요구하는 반면, GPL은 'A+B' 전체를 GPL로 배포할 것을 요구하기 때문에, 'A+B' 프로그램을 배포하는 것은 불가능하게 된다.)

모듈 (Module) :

잘 정의된 한 가지 일을 수행하는 프로그램의 논리적인 일부분. 주 프로그램은 논리적으로 몇 개의 모듈로 나눌 수 있다. 모듈은 여러 프로그램 작성자에 의해 나뉘어 작성되는


성질을 지닌다. 오픈소스 SW 라이선스에서 모듈에 대한 명확한 정의는 현재 없음

메일링 리스트 (Mailing List) :

공통된 관심을 가진 인터넷사용자끼리 서로 메일을 주고 받음으로써 정보를 공유하기 위해 만들어놓은 일종의 전자우편목록이라고도 하며 이러한 서비스를 하는 일종의 인터넷서비스라고도 합니다. 이러한 메일링리스트에 가입을 하게되면 가입한 특정분야의 정보를 메일로 계속 받아볼 수 있으며 가입한 사람 또한 메일링리스트에 가입되어 있는 사람들에게 정보를 보낼 수 있습니다.

바이너리 (binary) :

바이너리란 2진수라는 의미로서 바이너리 파일은 0과 1, 즉 2진수로 이루어진 파일



을 의미한다. 컴퓨터 통신에서는 주로 화상, 음성 등의 대부분 파일들이 바이너리 파일로 처리된다.

반환의 의무:

오픈소스를 사용하여 수정한 소스코드를 공개해야 하는 의무를 말함.
공개의무와 동일한 의미

반환 라이선스:

Reciprocal 오픈소스 SW 라이선스는 오픈소스를 자유롭게 사용했으면 동일하게 본인이 만든 소스 코드도 오픈소스로 내놓으라는 의미로 GPL 라이선스가 대표적인

배포:

저작물의 원작품 또는 그 복제물을 일반공중에게 대가를 받거나 받지 아니하고 양도 또는 대여하는 행위이다. 또한, 저작물을 이용하는 방법이자 저작물을 시장에 유통시키는 방법으로 B2B, B2C, 제품 형태의 유통 등을 포함한다.

- 직접적인 실행파일 또는 소스코드의 제공
- Active-X, CD, 인터넷 다운로드 등의 형태로 실행파일 또는 소스코드 제공
- 전송을 통한 저작물 이용 제공 등

보증 (Warranty):

저작물의 사용 계약에 있어서, 저작권자가 사용자에게, 계약의 대상이 된 저작물이 독창적인 창작물이며 계약에 표시된 사람 이외에는 저작자임을 주장할 수 없고 양도 또는 이용 허락된 권리를 방해하는 권리가 계약으로 다른 사람에게 부여되지 않았다고, 통상 문서로 하는 보증을 말한다.

복제:

원래의 저작물을 재생하여 표현하는 모든 행위. 인쇄, 사진, 복사, 녹화 등이 해당

* 프로그램을 유형물에 고정시켜 새로운 창작성을 더하지 아니하고 다시 제작하는 행위

사용여부 명시(고시):

많은 오픈소스 SW 라이선스들은 소스코드를 자유롭게 열람하고 수정 및 재배포할 수 있는 권리를 부여하는 한편, SW를 사용할 때 해당 오픈소스 SW가 사용되었음을 명시적으로 표기하는 것을 의무사항으로 채택하고 있다. 이것은 마치 논문을 쓸 때 인용을 하는 것과 비슷하여, 이 SW는 오픈소스 SW인 무엇 무엇을 사용하였습니다.'라는 식으로 사용 여부를 명확히 기술하라는 것이다.



상표권 (Trademark right):

상표권(trademark right)이란 상표권자가 지정상품에 관하여 그 등록상표를 사용할 독점적인 권리로서 일정한 절차에 따라 등록하여야 효력이 발생한다. 이러한 상표를 사용하기 위해서는 반드시 상표권자의 허락을 얻어야 하며 허락 받지 않고 상표를 이용할 경우 처벌을 받게 된다. 상표권을 취득한 SW의 경우 상표를 사용하려면 상표권자의 명시적인 허락을 받아야 한다.

서로 다른 라이선스의 조합:

서로 다른 오픈소스 SW간의 라이선스 의무조항이 상충하는 문제에 따라 이를 해결할 목적으로 타 라이선스의 수용 여부를 명시하는 조항

설정 파일 (Configuration file):

응용 프로그램을 설치할 때 사용자가 선택한 시스템 구성 요소의 조건이나 설정한 특성 등을 보관한 파일. 응용 프로그램 측에서 생성되며, 사용자가 응용 프로그램을 시동할 때 사용할 수 있다.

소스코드 (Source code):

컴퓨터 프로그램을 사람의 언어로 표현한 형태임

쉐어웨어 (Shareware):


실행파일의 형태로 무료 배포되지만 일정기간의 시험기간 이후 유상으로 전환되는 SW를 의미한다. 누구든지 임의대로 복제, 배포할 수 있다. 쉐어웨어와 프리웨어는 무료로 자유롭게 이용할 수 있다는 측면에서 오픈소스 SW와 일면 유사한 점이 있지만, 쉐어웨어나 프리웨어는 저작권자가 상업적 또는 개인적 이유에서 해당 SW를 이용할 수 있도록 배려하는데 불과하다는 점에서 오픈소스 SW의 공유정신과는 매우 상이하다.

애드웨어 (Adware):

프로그램의 기능이나 날짜상의 제한 없이 무료로 사용하는 대신 해당 SW로 작업하는 동안 광고 창을 통해 지속적으로 새로운 광고가 노출되도록 한 새로운 마케팅 기법으로 등장한 소프트웨어. 대표적인 예로는 알집, 제트 오디오 등이 있다.

오브젝트 코드 (Object code):

컴파일러 또는 어셈블러에 의해 생성되는 부호로, 어떤 프로그램의 원시 부호로부터 컴퓨터의 중앙 처리 장치(CPU)가 직접 실행할 수 있는 기계어 부호로 번역되어 있거나 실행할 수 있는 기계어 부호로 만드는 데 적합한 형태로 되어 있는 것. 목적



(object)이라고도 한다.

오픈소스 소프트웨어:

저작권이 있으면서 소스코드가 공개되어 있는 SW를 말하며, 일반적으로 자유롭게 사용, 복제, 배포, 수정할 수 있다. 오픈소스 SW의 대표적인 예로는 Linux 커널 및 관련 GNU SW, 아파치 웹서버, FireFox 웹브라우저, MySQL 데이터베이스시스템, Python/PHP/Perl 언어, Eclipse 툴 등이 있다

오픈소스 라이선스 <프리 소프트웨어 라이선스와의 차이점이 있다>:

OSI에서 자격조건으로 제시한 10가지 항목을 만족해야 한다. 프리소프트웨어 라이선스(Copyleft license) 뿐만 아니라 허용 라이선스까지 포함한다.

오픈소스 SW 개발자와 이용자 간에 사용방법 및 조건의 범위를 명시한 계약을 말한다. 오픈소스 SW를 이용하기 위해서는 오픈소스 SW 개발자가 만들어놓은 사용방법 및 조건의 범위에 따라 해당 SW를 사용해야 하며, 이를 위반할 경우에는 라이선스를 위반함과 동시에 저작권 침해로 인해서 이에 대한 처벌을 받게 된다. 대표적인 라이선스로는 GPL, LGPL, BSD, MPL 등의 라이선스가 있으며, 이런 오픈소스 SW 라이선스는 기본적으로 사용자의 자유로운 사용, 수정, 배포를 보장하고 있다.

응용 프로그램:

컴퓨터를 사용하는 본래의 목적을 이루기 위한 프로그램. 이에 대해 본래의 목적에는 직접 관여하지 않고 응용 프로그램을 수행하기 위한 환경을 준비하는 운영체제 등을 기본 프로그램이라고 한다.

이차 저작물 (Derivative works):

원래 소스코드를 수정하여 만든 프로그램.

오픈소스 SW 라이선스에는 프로그램 원저작물의 개작이나 이를 이용한 2차적 프로그램의 창작이 허용 되어야 하며, 이러한 파생적 프로그램들은 최초의 프로그램에 갖고 있던 라이선스의 규정과 동일한 조건하에서 재 배포될 수 있어야 한다.

인터페이스 (Interface):

하나의 시스템을 구성하는 하드웨어와 소프트웨어 또는 2개의 시스템이 상호 작용할 수 있도록 접속되는 경계(boundary)나 이 경계에서 상호 접속하기 위한 하드웨어, 소프트웨어, 조건, 규약 등을 포괄적으로 가리키는 용어.

자유 소프트웨어 (Free Software):

무상으로 자유롭게 사용할 수 있도록 배포하는 소프트웨어를 말한다. 프리웨어는 이



용기간이나 기능의 제약은 없지만 이용 목적이나 사용자를 구분 짓는 경우가 있기 때문에 구체적인 이용 허락의 범위를 확인하여야 한다. 보통의 경우 특별한 제한이 없지만 개작 및 배포 또는 상업용 목적으로 이용 시에는 달리 취급하는 경우가 많다.

저작물 (Works):

저작물은 인간 사상 또는 감정을 표현한 창작물을 말한다.

저작권 (Copyright):

저작권(copyright)은 문학·학술 또는 예술의 범위에 속하는 창작물(저작물)의 창작에 의하여 그 창작물에 대하여 창작자(저작자)가 취득하는 권리로서 창작물의 아이디어를 보호하는 것이 아니라 그 표현(expression)의 결과물을 보호하는 것이다. 저작권은 권리의 발생에 있어 등록과 같은 요건이 필요하지 않고 창작과 동시에 권리가 발생한다(무방식주의). 따라서 어떤 프로그래머가 특정 SW를 개발하게 되면 컴퓨터프로그램저작권이 자동적으로 발생하며, 그 권리는 프로그래머 또는 그가 속한 회사에 부여된다. 저작권이 있는 저작물의 경우 누구도 원 저작자나 저작권자의 허가가 없이는 해당 저작물을 사용복제, 배포, 수정할 수 없다.

저작권 관련 문구 유지:

SW의 경우 소스코드 상단을 보면 프로그램 이름, 개발자 이름, 버전, 연락처, 라이선스명 등이 기록되어 있으며, 이러한 정보는 저작권에 의해 보호받기 때문에 수정 또는 배포하는 사람이 임의로 삭제하거나 수정하는 것을 금지하는 조항 잘 관리되는 오픈소스 SW들의 경우 거의 대부분 소스코드 상단에 개발자 정보와 연락처 등이 기록되어 있다

- ex) Copyright (C) 1991, 1994-1999, 2000, 2001 Free Software Foundation, Inc.

전송:

일반 공중이 개별적으로 선택한 시간과 장소에서 수신하거나 이용할 수 있도록 저작물을 무선 또는 유선 통신의 방법에 의하여 송신(Send)하거나 이용에 제공(Upload)하는 것

제품명 중복 방지:

오픈소스 SW의 제품명은 상표의 의미이므로 동일 이름으로 다른 제품명 및 서비스명에 사용하는 것을 금지하는 조항

주석 (Annotations):

어떤 낱말이나 문장의 뜻을 이해하기 쉽도록 풀이함, 또는 그 글. 주해(注解).



지적재산권(Intellectual Property):

인간의 창조적 활동 또는 경험 등을 통해 창출하거나 발견한 지식·정보·기술이나 표현, 표시 그 밖에 무형적인 것으로서 재산적 가치가 실현될 수 있는 지적 창작물에 부여된 재산에 관한 권리를 말 한다

카피레프트(Copyleft):

독점적인 의미의 저작권(카피라이트, copyright)에 반대되는 개념이며, 저작권에 기반을 둔 사용 제한이 아니라 저작권을 기반으로 한 정보의 공유를 위한 조치이다.

컴파일 (Compile):

프로그래머가 CHILL, FORTRAN, COBOL 등의 고급 언어(high level language)로 작성한 프로그램을 번역하고, 컴퓨터가 실행(execute)할 수 있는 형식인 기계어(machine language)의 (목적)프로그램으로 변환하는 동작.

컴파일러 (Compiler):

컴파일을 실제로 행하는 프로그램(소프트웨어)을 컴파일러(compiler)라고 한다.

컴퓨터 프로그램 (Computer program):

컴퓨터 실행을 목적으로 만든 명령문(선언문, 주석문, 실행문 등)의 집합으로 소스코드와 실행 코드로 구분됨

특허권 (Patent):

특허권(patent)은 발명에 관하여 발생하는 독점적·배타적 지배권으로 법에 정해진 절차에 의해 출원을 하여야 하며, 심사를 통해 부여되는 권리이다. 특허기술을 사용하기 위해서는 반드시 특허권자의 허락을 얻어야만 한다. 특허권자는 자신이 허락하지 않은 사람이 해당 특허를 활용한 제품을 만들거나, 사용하거나, 판매하는 것을 막을 수 있다. 특허는 무엇인가 유용하게 하는 방식(method)이므로 특허 받은 방식을 구현하는 SW라면 프로그래밍 언어가 다르거나 소스코드가 다르더라도 해당 특허권자의 명시적인 허락을 받아야 하며 이는 오픈소스SW 뿐만 아니라 상용 SW에도 마찬가지이다.

파생저작물 (Derivative works):

다른 기존 저작물에 기초를 둔 저작물을 말한다. 이차적 저작물의 독창성은 기존 저작물의 개작이나 다른 언어로 번역하는 경우와 같이 창작적인 요소에 있다. 이차적 저작물을 보호되며 기존 저작물의 저작권에 영향을 미치지 않는다.



파일 (File):

컴퓨터에서 프로그램을 저장하는 최소 단위

편집 저작물 (Collective work):

많은 국가의 저작권법에서 기존의 저작물, 자료 등을 수집, 선정, 배열, 조합, 편집 등의 행위를 통해서 전체로서 하나의 저작물이 되는 것을 말한다. 선택과 배열 등에 독창성을 인정하여 저작권 보호를 받도록 한 것이다. 우리나라 저작권법과 같이 편집 저작물이라고도 한다. 미국 저작권법에서는 편집물에 수집 저작물이 포함되도록 규정하고 있다. (제101조) 우리 법의 해석으로도 편집물은 수집물을 포함한다.

프로그램 (Program):

기계가 읽을 수 있는 매체에 담아, 정보 처리 능력을 가진 기계로 하여금 특정의 기능이나 결과를 지시, 수행 또는 완성할 수 있도록 한 일련의 명령(명령의 집합)을 말한다. 독창적인 프로그램은 보호받는 저작물이 된다.

프리웨어 (Freeware):

프리웨어는 실행파일의 형태로 무료로 배포되는 SW를 말한다. 일반적으로 프리웨어의 형태로 제공되는 경우에는 SW의 계속적인 단순사용과 재배포는 허용하지만 변경이나 2차 저작물 작성은 허용되지 않는다. 국내에서는 흔히 공개판이라고 부르기도 한다. 국내에서 공개판이라고 하는 Winzip, realaudio 등이 이에 해당한다.

허용 라이선스:

Permissive 오픈소스 SW 라이선스의 경우는 제한 없이 마음대로 사용이 가능하고, 해당 소스코드를 이용하여 상업적인 용도의 사용도 가능하다는 것을 의미한다.