**오픈소스 소개**

**1. 개요**

Open Source

FOSS(Free and Open-Source Software)

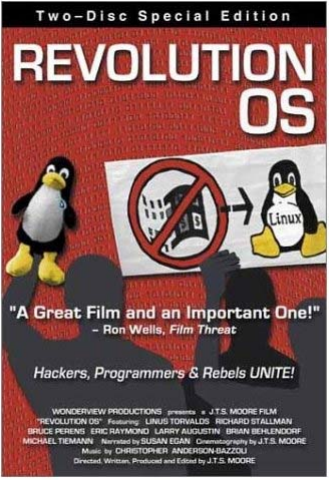
여기서 'Free'는 공짜가 아니라 자유의 의미로, 이 약자는 GPL 라이선스를 갖는 소프트웨어들을 가리킨다. 따라서 그냥 'Open Source Software'보다 조금 더 좁은 의미이다.

어떤 제품을 개발하는 과정에 필요한 소스 코드나 설계도를 누구나 접근해서 열람할 수 있도록 공개하는 것. 보통 소스가 공개된 소프트웨어를 '오픈소스 소프트웨어'라고 하고, 소프트웨어 말고도 개발 과정이나 설계도가 공개되는 경우 하드웨어에도 오픈소스 모델이 적용 할 수 있으며, 글꼴과 같은 데이터에도 오픈소스 개발 모델이 적용되는 경우가 있다. 다만 오픈소스지만 쓰려면 돈을 내야 하는 것도 있다.

단순히 소스를 공개만 하는 것이 아니라, 이를 2차 창작하는 것을 허용하기도 하고, 나아가 조건 없이 상업적 용도로까지 사용할 수 있게 하는 경우가 있다. 물론 공개만 하고 2차 창작이 안되는 경우도 있기는 하지만 이 경우는 보통 '오픈소스'라고 부르지 않는다.

**2. 오픈소스 이야기**

**2.1. Revolution OS (2001)** // 링크 영상



「I was at Agenda 2000 … and bumped in to him in an, in an elevator... in an elevator And uh, I looked at

his badge and said, "Oh, I see you work for Microsoft." And he looked back to me and said, "Oh, yeah

and what do you do?" And I thought he seemed just a sort of a tad dismissive I mean, here's the

archetypal, you know, guy in a suit looking at a scruffy hacker And so I gave him the thousand yard stare

and said,

"I'm your worst nightmare." – Eric Raymond on Revolution OS -」

「REVOLUTION OS」는 「Linux」와「오픈 소스 운동」에 의해서 마이크로소프트에 대항하는 해커들의 인사이드 스토리이다.

20년간 컴퓨터 해커들이 마이크로소프트의 독점에 대항해, 소프트웨어의 발전과 소유권의 있는 방법을 바꾸는 기술 혁명을 행해 왔다. 이 혁명의 집대성이「오픈 소스 운동」이며, 「Linux 운영체제」이다.

다큐멘터리 필름「REVOLUTION OS」는 오픈 소스 운동을 이끌고 온 주역들에 대한 인터뷰를 통해 그 기원을 밝히고 그 세력을 키워 주류로 성장해 가는 역사를 재조명한 작품이다. 또한,

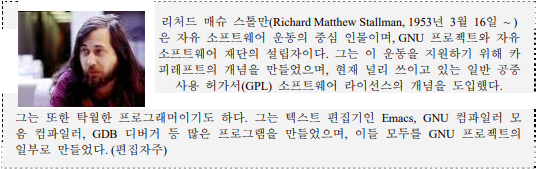
「REVOLUTION OS」는 Linux를 개발한 핀란드 엔지니어 ‘리눅스 토발즈’와 자유 소프트웨어 운동의 ‘리차드 스톨만’, OSI 창설자인 ‘에릭 레이몬드’ 등이 Linux의 탄생으로부터 곧 도달할 때까지를 중요한 인터뷰를 다수 수록하고 있다. 영상은 시네마스코프 35 mm필름을 사용해 실리콘밸리에서 촬영하였다.

**2.2. 자유 소프트웨어**

자유 소프트웨어(free software)는 복사와 사용, 연구, 수정, 배포 등의 제한이 없는 소프트웨어 혹은 그 통칭이다. 소프트웨어의 수정 및 수정본의 재배포는 인간이 해독 가능한 프로그램의 소스 코드가 있어야만 가능하며, 소스 코드는 GPL 등의 라이선스를 통하거나, 혹은 드물게 퍼블릭 도메인으로 공개되기도 한다. 자유 소프트웨어 운동은 초창기의 컴퓨터 사용자들이 이러한 자유를 누릴 수 있도록 하기 위해서 1983년에 시작되었다.

자유 소프트웨어는 완전히 무료로 또는 최소한의 금액만을 받고 자유롭게 배포되어야 하며 자유 소프트웨어를 통한 비즈니스 모델들은 대개 고객 지원이나 커스터마이징 등을 통한 것들이다. 반면 독점 소프트웨어를 이용한 비즈니스 모델들은 사용자가 합법적으로 소프트웨어를 이용하기 위한 허가를 위해서 반드시 일정 비용을 지불해야 하기 때문에, 자유 소프트웨어와는 맞지 않는다. 자유 소프트웨어는 이제 거대한 전 세계적인 움직임으로 확산되었으며, 개인 및 거대 단체와 정부 기관 등에서 사용하는 소프트웨어들이 만들어지고 있다. 아파치 웹 서버나 MySQL 데이터베이스, PHP 스크립트 언어 같은 자유 소프트웨어들은 서버 측 인터넷 어플리케이션 영역에서 강한 영향력을 지니고 있다.

완벽히 자유로운 컴퓨터 환경은 리눅스나 FreeBSD 등의 시스템 소프트웨어들을 기본으로 한 많은 패키지들을 통해서 구성할 수 있다. 자유 소프트웨어 개발자들은 웹 브라우저나 오피스 제품군 혹은 멀티미디어 플레이어 등의 대부분의 데스크톱 어플리케이션들을 자유 소프트웨어로 만들어냈다. 그러나 많은 영역에서 개인 사용자를 위한 이런 소프트웨어들은 경쟁 독점 소프트웨어들에 비해 미미한 시장 점유율만을 차지하고 있다. 대부분의 자유 소프트웨어들은 온라인으로 무료로 제공되거나, 오프라인으로 적당한 가격으로 배포된다. 그러나 이것이 필수적인 것은 아니다. 자유 소프트웨어의 경제적 가능성은 IBM이나 레드햇, 썬 마이크로시스템즈 등의 거대 회사들에 의해 인식되었다. 주력 산업이 IT 영역이 아닌 많은 회사들이 인터넷의 홍보 및 판매 사이트를 위해 비용이 적게 들고 어플리케이션을 쉽게 수정할 수 있다는 점에서 자유 소프트웨어를 선택했다. 또한, 소프트웨어 이외의 산업에서도 그 연구와 개발을 위해서 자유 소프트웨어의 개발과 유사한 방법을 사용하기 시작했다. 예를 들어 과학자들은 좀더 공개된 개발 과정을 생각하고 있었고, 마이크로칩과 같은 하드웨어들은 카피레프트 라이선스가 적용된 명세서와 함께 개발되기 시작했다. (오픈코어 프로젝트를 참조.) 크리에이티브 커먼스나 자유 문화 운동 등의 움직임들도 또한 자유 소프트웨어 운동의 영향을 크게 받은 사례이다.



**2.3. 오픈소스 소프트웨어**

오픈소스 소프트웨어는 일반적으로 자유롭게 사용, 복제, 배포, 수정할 수 있으며, 소스코드가 공개되어있는 소프트웨어를 말한다. 오픈소스의 예로는 Linux 커널 및 관련 GNU 소프트웨어. 아파치, 웹서버, FireFox, 웹브라우저, MySQL, 데이터베이스시스템, Python/PHP/Perl 언어, Eclipse툴 등을 들 수 있으며, 그 외에도 전세계의 수많은 개발자들이 개발하고 있는 프로그램들이 있다.

**2.3.1. 정의**

OSI는 오픈 소스 소프트웨어로 인정받을 수 있는 10가지 '라이센스 조건'을 정의하고, 각각의 소프트웨어에 대한 라이센스를 분석하여 오픈 소스 소프트웨어라이센스에 해당하는지 여부에 대한 인증을 하고 있다. 주요 내용을 살펴보면, 소프트웨어에 대한 배포 및 수정의 자유를 인정해야하며 소스 코드를 제공할 수 있어야 할 것, 그리고 어떤 사람이나 그룹 또는 이용분야에 대한 차별이 없어야 한다는 조건 등을 두고 있다.

1. 자유로운 재배포

오픈 소스 라이센스(license)는 몇 개의 다른 출처로부터 모아진 프로그램들로 구성된 집합 저작물 형태의 배포판의 일부로 소프트웨어를 판매하거나 무상 배포하는 것을 제한해서는 안된다. 또한 그러한 판매에 대해 사용료나 그밖의 다른 비용을 요구해서도 안된다.

사용 허가에 자유로운 재배포를 규정하도록 강제함으로써 우리는 단기간의 적은 판매 수익을 얻기 위해 많은 장기적인 이익을 포기하는 유혹을 없앨 수 있다. 만약 이렇게 하지 않는다면, 협력자들에게 많은 변심의 압력이 있을 것이다.

2. 소스 코드

오픈 소스 프로그램에는 소스 코드(source code)가 포함되어야 하며, 컴파일된 형태 뿐 아니라 소스 코드의 배포도 허용되어야 한다. 만약 소스 코드가 함께 제공되지 않는 제품이 있다면 소스코드를 복제하는데 필요한 합당한 비용만으로 소스 코드를 구할 수 있는 널리 알려진 방법이 제공되어야만 한다.

이러한 경우에 있어 가장 권장할 만한 방법은 별도의 비용없이 인터넷을 통해 소스 코드를 다운 받을 수 있도록 하는 것이다. 소스 코드는 프로그래머가 이를 개작하기에 용이한 형태여야 하며, 고의로 복잡하고 혼란스럽게 만들어진 형태와 선행 처리기나 번역기에 의해 생성된 중간 형태의 코드는 인정되지 않는다.

소스 코드를 불분명하지 않은 형태로 제공하도록 규정하는 이유은 프로그램을 발전시키기 위해서 소스 코드에 대한 개작이 선행되어야 하기 때문입니다. 우리의 목적은 발전을 용이하게 만들기 위한 것이므로 개작이 용이하게 이루어 질 수 있는 방법을 요구한다.

3. 파생 저작물

오픈 소스 라이센스에는 프로그램의 개작과 2차적 프로그램의 창작이 허용되어야 하며, 이러한 파생 저작물들이 원프로그램에 적용된 것과 동일한 사용 허가의 규정에 따라 배포되는 것을 허용해야만 한다. 단순히 소스 코드를 열람할 수 있는 것만으로는 독립된 등위 검토(peer review)와 빠른 발전 경쟁에서의 생존을 지원할 수 없습니다. 프로그램을 빠르게 발전시키기 위해서는 사람들에게 개작된 프로그램을 실험하고 재배포할 수 있도록 허용할 필요가 있다.

4. 저작자의 소스 코드 원형 유지

오픈 소스 라이센스는 바이너리를 생성할 시점에서 프로그램을 수정할 목적으로, 소스 코드를 수반한 ``패치 파일''의 배포를 허용한 경우에 한해서 패치로 인해 변경된 소스 코드의 배포를 제한할 수 있다. 그러나 이 경우에도 변경된 소스 코드를 통해 만들어진 소프트웨어의 배포는 명시적으로 허용해야만 한다. 오픈 소스 라이센스는 파생 저작물에 최초의 소프트웨어와 다른 판 번호(version)와 이름이 사용되도록 규정할 수 있다. 소프트웨어에 많은 향상이 이루어지도록 장려하는 것은 좋은 일입니다. 그러나 사용자에게는 그들이 사용하고 있는 소프트웨어를 누가 책임지고 있는 지를 알 권리가 있다. 또한 저작자와 관리자에게도 반대 입장에서 사용자들이 그들에게 어떤 지원을 요구하고 있는 지를 알 권리와 그들의 명성을 보호할 권리가 있다. 따라서 오픈 소스 라이센스는 소스 코드가 쉽게 이용될 수 있도록 보증해야만 하지만 변형되지 않은 최초의 소스 코드가 패치 파일와 함께 배포되도록 규정할 수도 있다. 이러한 방법을 통해 “비공식” 수정들을 이용할 수 있으면서 소스 코드의 원형이 쉽게 구별될 수 있다.

5. 개인 및 단체에 대한 차별 금지

오픈 소스 라이센스는 특정 개인이나 단체를 차별해서는 안된다. 오픈 소스의 공정으로부터 최대의 이익을 끌어내기 위해 최대한 다양한 개인과 단체에게 오픈 소스에 기여할 수 있는 동등한 자격이 부여되어야 한다. 따라서 우리는 어떠한 오프 소스 사용 허가도 특정인을 오픈 소스의 공정으로부터 제외시키는 것을 금지한다. 미국을 포함한 몇몇 국가에서는 특정한 종류의 소프트웨어에 대한 수출이 금지되고 있다. OSD를 준수하는 사용 허가는 피양도자에게 이러한 종류의 제한에 대해 경고하고 해당 법률을 준수해야 한다는 사실을 상기시킬 수 있다. 그러나 사용 허가 자체에 그러한 제한이 통합되어서는 안된다.

6. 사용 분야에 대한 차별 금지

오픈 소스 라이센스는 프로그램이 특정 분야에서 사용되는 것을 금지하는 제한을 설정해서는 안된다. 예를 들면, 기업이나 유전학 연구에 프로그램을 사용할 수 없다는 등과 같은 제한을 설정해서는 안된다. 이 조항의 주된 목적은 오픈 소스가 상업적으로 이용되지 못하게 방해하는 규정이 사용 허가에 포함되는 것을 금지하기 위한 것이다. 우리는 상업 이용자들도 오픈 소스 공동체에 동참하기를 원하며 이들이 공동체로부터 소외감을 느끼지 않기를 바랍니다.

7. 사용 허가의 배포

프로그램에 대한 권리는 배포에 따른 각 단계에서 배포자에 의한 별도의 사용 허가 없이도 프로그램을 재배포받은 모든 사람에게 동일하게 인정되어야만 한다. 이 조항은 비공개 계약을 요구하는 것과 같은 간접적인 수단을 통해 소프트웨어가 제한되는 것을 금지하기 위한 것이다.

8. 특정 제품에만 유효한 사용 허가의 금지

프로그램에 대한 권리는 프로그램이 특정한 소프트웨어 배포판의 일부가 될 때에 한해서만 유효해서는 안된다. 만약 특정 배포판에 포함되어 있던 프로그램을 별도로 분리한 경우라 하더라도 프로그램에 적용된 사용 허가에 따라 프로그램이 사용되거나 배포된다면 프로그램을 재배포받은 모든 사람에게 최초의 소프트웨어 배포판을 통해 프로그램을 배포받은 사람과 동일한 권리가 보장되어야만 한다. 이 조항은 또다른 형태의 사용 허가상의 제한을 방지하기 위한 것이다.

9. 다른 소프트웨어를 제한하는 사용 허가의 금지

오픈 소스 라이센스는 오픈 소스 라이센스가 적용된 소프트웨어와 함께 배포되는 다른 소프트웨어에 대한 제한을 포함해서는 안된다. 예를 들면, 사용 허가 안에 동일한 매체를 통해 배포되는 다른 소프트웨어들이 모두 오픈 소스 소프트웨어여야 한다는 제한을 두어서는 안된다. 오픈 소스 소프트웨어의 배포자들은 그들의 소프트웨어에 대한 스스로의 선택 권리를 갖고 있다. 물론, GPL은 이러한 규정을 충족시키고 있다. GPL 라이브러리와 결합되는 소프트웨어는 하나의 단일 저작물을 형성할 때에 한해서 GPL이 계승되는 것이지 단순히 함께 배포된다는 것만으로 GPL 소프트웨어가 되어야 하는 것은 아니다.

10. 라이센스 기술 중립성

오픈 소스 사용 라이센스는 특정 소프트웨어의 기술이나 인터페이스에 의존해서는 안된다. 이는 특정 소프트웨어만을 위한 라이센스가 오픈 소스로 인정 받을 수 없음을 의미한다.

**2.3.2. 장점**

1. 비용 절감

아마 가장 중요하고 큰 장점일 것이다. 오픈소스 소프트웨어는 프리웨어 소프트웨어와 마찬가지로 무료 이용이 가능하다. 여기서 더 나아가 소스코드가 공개 되어 직접 소프트웨어의 개선 또는 수정이 가능해지며, 이로 인해 개발 비용이 적게 드는 편이다. 실제로 오픈소스는 무료 다운로드와 수정/재배포가 가능하여 초기 개발 비용이 새 소프트웨어를 개발하는 것의 절반 정도 되는 것으로 알려져 있다.

2. 빠르고 유연한 개발

오픈소스 커뮤니티는 다양한 이용자들에게서 최신 기술 정보와 문제점의 해결책을 공유하여 운영되기 때문에 독점 프로그램에 비해 기술의 발전 속도가 빠른 편이다. 특히 개발자와 사용자가 일치하는 경우 클로즈드 소스 프로그램보다 뛰어난 고품질의 오픈소스 소프트웨어가 개발되기도 한다.

3. 호환성/유연성

오픈소스는 주로 오픈포맷 또는 오픈프로토콜(개방형 표준)을 사용하기 때문에 서로 다른 소프트웨어간의 연동이 쉽다. 서로 다른 플랫폼끼리의 상호 연동 또한 가능하다. 또한 특정 기기, 운영체제, 어플리케이션에 종속되지 않고 자유로운 변경이 가능하다. 여러 기기들이 네트워크를 통해 하나로 연결되는 유비쿼터스 시대에 아주 적합한 장점이라고 할 수 있다.

4. 신뢰성/안정성

전 세계의 수많은 개발자들과 전문가들이 오픈소스의 개발에 참여하기 때문에 폐쇠적으로 개발되는 독점 프로그램에 비해 안정적으로 작동한다. 단, 이는 많은 개발자들이 적극적으로 참여하는 프로그램의 경우에만 가능하므로 해당 오픈소스의 평판과 개발 과정을 주의 깊게 볼 필요가 있다.

특정 벤더에 의존하지 않아도 된다.

**2.3.3. 단점**

1. 애플리케이션의 부족

대부분에 사용자는 Windows 기반의 GUI(Graphical User Interface)를 가진 Application에 익숙해 있지만, 이에 버금가는 Linux 기반의 Application이 많이 부족한 것이 현실이다. 또한. Linux 기반으로 개발된 많은 Application은 Windows 기반 Application들과 호환되지 않는 문제점도 있다.

2. 빈약한 문서

상용 프로그램보다 오픈소스는 체계적인 문서를 갖고 있지 못하다.

3. 불확실한 개발 로드맵

오픈소스는 영리를 목적으로 하는 회사에서 개발되는 것이 아니라, 자발적 참여를 바탕으로 웹상에서 자유롭게 개발되는 것이 특징이다. 그러므로 독점 프로그램에서 볼 수 있는 로드맵을 기대하기 힘든 면이 있다.

4. 지적 재산권

오픈소스에 기업이 보유한 특허 및 소스코드를 포함할 경우 오픈소스 라이선스는 일반적으로 Royalty-free를 요구하고 있다. 따라서 Royalty-free를 원치 않을 때에는 해당 오픈소스를 사용할 수가 없으면, 또한 사용 후 Royalty를 주장하게 되면 해당 오픈소스에 대한 사용 권한이 박탈되는 경우가 일반적이다. 따라서 오픈소스를 활용하여 특허를 구현하거나 기존 소스코드를 포함하고자 할 경우, 반드시 Royalty에 대한 입장을 명확히 해야한다.

**3. 오픈소스 소프트웨어 라이선스**

**3.1. 라이선스 정의**

흔히 소프트웨어는 다음과 같이 저작권, 특허권, 영업비밀, 상표 등의 지적재산권법에 의해 보호받고 있다.

저작권

어떤 프로그래머가 특정 소프트웨어를 개발하게 되면 컴퓨터프로그램저작권이 자동적으로 발생하며, 프로그래머 또는 그가 속한 회사에 부여된다. 저작권(copyright)은 시, 소설, 노래 등 저작물에 대해 부여되는 권리로서 그 표현(expression)의 결과물을 보호하는 것이다. 누구도 원 저작자나 저작권자의 허가가 없이는 해당 저작물을 복사, 개작, 재배포할 수 없다.

특허권

특허는 하드웨어에 구현되거나 소프트웨어에 의해 동작이 구현되는 발명(invention)을 보호한다. 특허권은 자동으로 부여되는 것이 아니고 법에 정해진 절차에 의해 출원을 하여야 하며, 심사를 통해 부여되는 권리이다. 특허 기술을 구현(implementation)하기 위해서는 반드시 특허권자의 허락을 득하여야만 한다. 특허 소유자는 소유자가 허가하지 않은 사람이 해당 특허를 활용한 제품을 만들거나, 사용하거나, 판해하는 것을 막을 수 있다. 특허는 무엇인가 유용한 것을 하도록 하는 방식(method)이므로 소프트웨어의 경우 특허받은 방식을 구현하는 소프트웨어라면 프로그래밍 언어가 다르거나 소스코드가 다르더라도 해당 특허권자의 명시적인 허가를 받아야 하며 이는 오픈 소스 소프트웨어, 독점소프트웨어에 공통으로 해당된다.

영업비밀

영업비밀이란 공연히 알려져 있지 아니하고 독립된 경제적 가치를 지니는 것으로서 상당한 노력에 의하여 비밀로 유지되는 생산 방법, 판매 방법, 기타 영업 활동에 유용한 기술상/영업상의 정보로 정의되어 있다. 이러한 영업비밀은 "부정경쟁방지 및 영업비밀보호에 관한 법률"에 의하여 보호받고 있으며, 이와 같은 영업비밀을 부당한 수단으로 취득하거나, 비밀유지의무가 있음에도 다른 사람에게 누출하는 것은 처벌받게 된다.

상표

상표권이란 제품이나 서비스와 연계되어 마케팅에 활용되는 이름 등을 보호한다. 또한 상표는 시장에서 나의 제품과 타인의 제품을 구별해 주는 역할을 한다. 이상과 같은 지적재산권에 의해 권리자는 소프트웨어에 대한 배타적인 권리를 가지게 되며, 원칙적으로 권리자만이 소프트웨어를 사용, 복제, 배포, 수정할 수 있다. 하지만 다양한 필요에 의해 이들 권리자가 다른 사람에게 일정한 내용을 조건으로 하여 특정 행위를 할 수 있는 권한을 부여 할 필요가 있는데, 이와 같은 권한을 보통 '라이센스(license, 사용허가권)'라고 한다. 이러한 의미에서 라이센스는 물건을 판매하는 매매와는 차이가 있으며, 소프트웨어에 대한 지적 재산권은 여전히 원래의 권리자에게 남아있고 일부 사용에 대한 권리만을 부여하는 것이다. 마이크로소프트, 오라클 등 일반적인 독점(proprietary)소프트웨어 업체의 라이센스는 고객이 소프트웨어 권리자에게 대금을 지급하고 소프트웨어의 ‘사용’ 권한만을 허락하는 것이 일반적이다. 따라서 허락을 얻지 않고 소프트웨어를 복제, 배포, 수정하는 행위는 라이센스를 위반함과 동시에 불법에 해당한다.

**3.2. 오픈소스 라이선스 특징**

오픈 소스 소프트웨어 역시 독점소프트웨어(proprietary software)와 동일하게 저작권 등에 의한 법적 보호를 받고 있으며, 이와 같은 권리에 기반하여 이용자에게 라이센스를 부여한다. 그러나 오픈 소스 라이센스는 일반적인 독점소프트웨어 라이센스와는 많은 점에서 차이가 있다. 기본적으로 오픈 소스 라이센스는 다음과 같이 사용자의 자유로운 사용, 복제, 배포, 수정을 보장하고 있다.

- 라이선시는 해당 오픈 소스 소프트웨어를 자유롭게 사용할 수 있다.

- 라이선시는 해당 오픈 소스 소프트웨어를 자유롭게 복제할 수 있으며, 일정한 조건하에 재배포할 수 있다.

- 라이선시는 해당 오픈 소스 소프트웨어를 자유롭게 수정하여 사용할 수 있으며, 일정한 조건하에 수정된 내용을 재배포할 수 있다.

- 라이선시는 해당 오픈 소스 소프트웨어의 소스코드를 자유롭게 획득하고 접근할 수 있다.

오픈 소스 라이선스는 소프트웨어의 사용, 복제, 배포, 수정의 자유를 부여함과 아울러 다른 한편으로는 소프트웨어의 사용자에게 일정한 의무를 부과하고 있다. 구체적인 내용은 오픈 소스 소프트웨어와 함께 배포되는 라이선스의 내용을 통해 알 수 있다.

해당 오픈 소스 소프트웨어에 대한 라이선스는 주로 소스코드 내부나 홈페이지 등에 명시되어 있다. 소스코드에서는 주로 최상위 디렉토리에 COPYING이라는 독립된 파일에 라이센스 조항을 기록하기도 하며, 각각의 소스코드 파일 상단에 명시해 두기도 한다.

오픈 소스 라이선스에 서 요구하고 있는 준수사항을 라이센시가 이행하지 않으면 권리자로부터 저작권 위반 (또는 계약 위반)으로 소송을 제기 당할 수 있다. 만약 권리를 침해한 것으로 결론이 내려지면 소프트웨어의 배포가 더 이상 불가능할 뿐만 아니라 이미 배포한 소프트웨어에 대한 손해배상 등 막대한 책임을 부담할 수 있다.

특히 임베디드 소프트웨어의 경우 이를 내장한 제품까지 판매하지 못하거나 리콜(Recall) 해야하는 경우도 발생할 수 있으므로 라이선스의 의무사항을 명확히 이해하여 이와 같은 상황을 사전에 예방하는 것이 필수적이다. 그러나 이러한 위험 때문에 오픈 소스 소프트웨어를 전혀 사용하지 않겠다는 결론을 내릴 필요는 없다. 독점소프트웨어 라이선스에서 규정하고 있는 의무사항에 비하면 오픈 소스 라이선스가 요구하고 있는 내용이 결코 어려운 것이 아니므로, 오히려 이를 잘 이해하고 준수함으로써 오픈 소스 소프트웨어의 장점을 적극 활용할 필요가 있다. 또한 몇몇 라이선스만이 독자 개발한 소스 코드의 공개를 요구하고 있기 때문에 이를 잘 분석한 후 사용한다면 문제 발생 소지는 거의 없을 것이다.

따라서 인터넷 상에서 자유롭게 구할 수 있는 오픈 소스를 다운로드받아 개발에 적용할 때는 반드시 라이센스의 요구 사항을 반드시 확인하여야 한다. 또한, 자체 판단이 불가능할 경우에는 외부 전문가에게 조언을 의뢰하여 개발 시작 전 해당 라이센스의 요구 사항과 오픈 소스 사용 목적을 확실히 분석하여야 한다. 이렇게 하는 것만으로도 충분히 올바르게 오픈 소스를 최대한 활용 할 수 있으며, 나중에 발생할 수 있는 문제들을 사전에 차단할 수 있다.

**3.3. 라이선스 상세 내용**

오픈 소스 라이선스의 의무사항은 각각의 라이선스마다 조금씩 차이가 있지만 크게 나누어 보면 공통적으로 '저작권 관련 문구 유지', '제품명 중복 방지', '서로 다른 라이선스의 소프트웨어 조합시 조합 가능 여부 확인' 등이 있고 선택적으로는 '소스코드 공개', '특허관련 사항 준수' 등이 있다. 아래는 모든 오픈 소스 소프트웨어에 공통적으로 적용되는, 항상 지켜야 할 사항들이다.

저작권 관련 문구 유지

앞에서 저작권이란 표현된 결과물에 대해 발생하는 권리이며 자동적으로 부여된다고 기술하였다. 소프트웨어의 소스코드에 대해서도 마찬가지이며 잘 관리되는 오픈 소스 소프트웨어들의 경우 거의 대부분 소스코드 상단에 개발자 정보와 연락처 등이 기록되어 있는데 만약 이러한 개발자 정보를 임의로 수정하거나 삭제하여서는 안된다. 특히 GPL등 수정된 결과물을 다시 공개하도록 규정하고 있는 '상호주의(reciprocal)' 라이센스들의 경우 만약 소스코드 상에 개발자 정보가 수정/삭제된 채로 외부에 소스코드를 공개하였다가 그 사실이 밝혀질 경우 더 큰 문제가 발생할 수 있다. 상식적으로도 쉽게 판단 가능한 사항이므로 항상 준수하여야 한다.

제품명 중복 방지

사용하는 오픈 소스 소프트웨어와 동일한 이름을 제품명이나 서비스 명으로 사용하여서는 아니된다. 특히 유명한 오픈 소스 소프트웨어들일수록 해당 오픈 소스 소프트웨어의 이름이 상표로서 등록되어 있는 경우가 많기 때문에(예: 리눅스) 더욱 조심하여야 한다. 다만 이러한 제품명/서비스명에 대한 결정이 개발자들에 의해 이루어지는 경우는 많지 않으므로 역시 상식적인 수준에서 판단하면 될 것이다.

서로 다른 라이선스의 조합

소프트웨어를 작성하고자 할 경우 기존에 만들어진 코드를 재사용하거나 결합하는 경우가 많은데, 결합되는 각 코드의 라이센스가 상호 상충되는 경우가 있다. 예컨대 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. 이러한 문제를 가르켜 라이센스의 양립성(Compatibility) 문제라고 한다.

따라서 어떤 오픈 소스 소프트웨어에 다른 오픈 소스 소프트웨어를 섞을 경우 반드시 두개의 라이센스가 서로 호환되는지를 확인하여야 한다. 양립성문제는 자유/오픈 소스 소프트웨어 진영에 심각한 문제점을 제기하였으며, 이를 해결하기 위한 노력도 다양하게 진행되고 있다. 예를 들어 모질라 프로젝트(Mozilla.org)에서는 프로젝트의 결과물을 MPL, GPL, LGPL의 3가지(triple) 라이센스로 배포하는 라이센스 정책을 채택하여, 라이센스의 양립성과 관련된 불확실성을 제거하고 모질라 코드를 GPL 또는 LGPL 기반의 응용프로그램에 사용할 수 있도록 하였다.

Trolltech도 Qt 라이브러리에 대한 오픈 소스 소프트웨어라이센스인 QPL과 GPL의 양립성 문제를 해결하기 위하여 QPL 및 상용라이센스 이외에 GPL을 추가하는 정책을 취하고 있다. 한편 최근 개정된 GPL 3.0은 Apache License 2.0과 양립가능하다.

아래는 라이센스에 따라 다르다. 어떤 라이센스의 경우는 아래 세가지 사항 모두에 관계되는 경우도 있고, 어떤 라이센스는 아래 중 일부만을 요구하는 경우도 있다. 자세한 사항은 라이센스별 해설 부분을 참고하기 바란다.

사용 여부 명시

많은 오픈 소스 라이센스들은 소스코드를 자유롭게 열람하고 수정 및 재배포할 수 있는 권리를 부여하는 한편, 소프트웨어를 사용할 때 해당 오픈 소스 소프트웨어가 사용되었음을 명시적으로 표기하는 것을 의무사항으로 채택하고 있다. 이것은 마치 논문을 쓸 때 인용을 하는 것과 비슷하여, '이 소프트웨어는 오픈 소스 소프트웨어인 무엇무엇을 사용하였습니다.'라는 식으로 사용 여부를 명확히 기술하라는 것이다. 사용자 매뉴얼이나 기타 매뉴얼을 대체하는 매체가 있다면 그곳에 기술하면 된다.

소스코드 공개

오픈 소스 라이센스에 따라서는 수정하거나 추가한 부분이 있을 때 해당 부분의 소스코드도 공개하여야 한다고 명시하는 경우가 있다. 이에 해당하는 라이센스는 GPL이 가장 유명하다. 그러나 정확한 공개 범위는 각각의 라이센스에서 정하고 있는 범위가 다르고, 소프트웨어를 개발하는 방법에 따라서도 달라질 수 있다. 자세한 내용은 다음 절의 쟁점부분을 참고하기 바란다.

특허

특허에 대한 기본적인 내용은, 만약 어떤 기술이 특허로 보호될 경우 해당 기술을 구현할 때 반드시 특허권자의 허락을 받아야 한다는 것이다. 이는 오픈 소스 이냐 아니냐에 상관 없이 공통적으로 해당된다. 그러나 어떤 특허를 오픈 소스 로 구현할 경우 해당 특허의 구현 결과는 오픈 소스 라이센스를 따르게 되는 등, 오픈 소스 소프트웨어와 관련된 특허권의 문제는 보다 복잡하게 전개되고 있다. 특히 최근 소프트웨어특허가 급격히 증가하면서 문제가 심각해지고 있기 때문에, 새롭게 만들어지는 오픈 소스 라이센스들에서는 특허관련조항을 포함하고 있는 경우가 많아지고 있다.

**3.4. 각 라이선스 소개**

**3.4.1. GNU 일반 공중 사용 허가서(GPL)**

GPL은 현재 가장 많은 오픈 소스 소프트웨어가 채택하고 있는 라이센스이다. 오픈 소스 라이센스들 중에서 가장 많이 알려져 있고 의무사항들도 타 라이센스에 비해 엄격한 편이다. GPL의 주요 내용은 다음과 같다.

\* 소프트웨어를 배포하는 경우 저작권 표시, 보증책임이 없다는 표시 및 GPL에 의해 배포된다는 사실 명시해야 한다.

\* 소프트웨어를 수정하거나 새로운 소프트웨어를 링크(Static과 Dynamic linking 모두)시키는 경우 GPL에 의해 소스 코드 제공해야 한다.

\* Object Code 또는 Executable Form으로 GPL 소프트웨어를 배포하는 경우, 소스 코드 그 자체를 함께 배포하거나 또는 소스코드를 제공받을 수 있는 방법에 대한 정보 함께 제공해야 한다.

\* 자신의 특허를 구현한 프로그램을 GPL로 배포할 때는 GPL 조건을 준수하는 이용자에게는 로열티를 받을 수 없으며, 제3자의 특허인 경우에도 특허권자가 Royalty-Free 형태의 라이센스를 제공해야만 해당 특허 기술을 구현한 프로그램을 GPL로 배포하는 것이 가능 하다.

GPL 소프트웨어를 사용하였을 경우 "본 제품(소프트웨어)는 GPL 라이센스 하에 배포되는 소프트웨어 XXX(사용한 GPL 소프트웨어 이름)를 포함합니다"와 같은 문구를 매뉴얼 혹은 그에 준하는 매체에 포함시키고, GPL 전문을 첨부해야 한다. GPL에서 가장 논란이 되는 부분은 소스코드 공개 범위이다. 실제 소스코드 공개 범위는 다음 장의 쟁점 부분에서 확인하기로 한다. 소스코드를 공개하기 위해서는 소스코드를 CD Rom 등의 매체에 담아서 제품판매시 함께 배포하거나, 매뉴얼에 소스코드를 요청할 수 있는 연락처를 기입하여 두거나, 혹은 FTP 서버, 웹서버 등에 소스코드를 업로드해 두고 매뉴얼에 해당 주소를 기입하면 된다. 최근 특허에 관한 쟁점도 중요성이 증가하고 있는데, 자세한 내용은 다음 장의 쟁점 부분에서 설명한다.

**3.4.2. 크리에이티브 커먼즈 라이선스(CCL)**

Creative Commons License. 줄여서 CCL. 크리에이티브 커먼즈에서 만든 저작물에 대한 이용허락 규약(라이선스)를 말한다.

라이선스의 취지는 자신의 창작물을 공유하고 싶은데, 법적인 해석이 모호해지거나 의도된 바와 다르게 사용될 우려로 그러지 못하는 경우를 방지할 수 있도록 하기 위함이다. 또한, 저작권법의 철퇴를 맞을 우려 없이 사람들이 공유된 저작물을 사용할 수 있도록 해주는 기능도 한다. 미국의 저작권 기간 연장 법안에 대한 위헌 소송인 엘드리드 대 애시크로프트 사건에서 에릭 엘드리드 쪽에 참여했었던 하버드 대학교 로스쿨의 로렌스 레식(Lawrence Lessig) 교수가 그 사건에서 영향을 받아 CC(Creative Commons)를 설립하고 CCL을 만들었다. 간단하게 말하면 원저작자가 설정한 조건만 지키면 저작권법에 걸릴 일이 없다는 것이다.

CCL이 뭔지 잘 모르는 사람은 아예 CCL로는 자신의 저작물 배포를 하지 않는 편이 좋다. 일단 배포하고 나면 저작자가 배포에 대해 제약할 수 있는 바는 오직 그 저작을 한 것이 자신인 것을 밝히라는 것과 CCL 로 걸었던 조건이 전부이기 때문에 공유를 위한 라이선스인 CCL로 배포해놓고 나중에 딴 소리를 하는 건, 다른 사람들에게 피해를 주는 행동이다. 물론 저작권자가 다른 라이선스로 재배포하거나 돈 받고 파는 것도 가능하다. 단지 다른 사람들이 CCL 조건에 따라 무료로 공유하며 쓰는 걸 못 막을 뿐. 한 마디로 말하자면, 공유는 간지나라고 하는 것이 아니고, CCL은 간지나라고 다는 물건이 아니다! 그러니 제대로 알고 현명하게 판단하여 달자.

반대로 위키에서는 문서 내에 CCL을 채택하는 경우가 많다. 위키는 특성상 하나의 문서에도 수많은 사람들이 기여하기 때문에 문서를 퍼가고 싶을 경우 각 기여분의 저작자들에게 일일이 허락을 구하기 대단히 힘들고, 따라서 이럴 필요가 없도록 문서 공유 조건을 미리 명시해놓는 CCL을 채택하는 것이다.

CC BY-SA인 위키백과나 리브레 위키의 내용은 CC BY-NC-SA인 나무위키로 퍼올 수 없으며, 반대로 나무위키의 문서를 위키백과나 리브레 위키로 퍼갈 수도 없다. 단, 원 저작자 자신이 쓴 내용에 한해서 세 곳 동시에 올릴 수는 있다.

CCL은 미국 저작권법에 맞춰서 개발된 것이라 미국 외의 국가의 저작권법과는 맞지 않을 수 있다. 따라서 국가별 저작권법에 맞춘 현지화 작업도 진행되었다.

한국 버전의 경우 사단법인 한국정보법학회가 2003년 '디지털 정보의 공유와 전유'라는 주제로 콘퍼런스를 준비한 것이 계기가 되어 만들어졌다. 한국정보법학회측에서 미국 크리에이티브 커먼즈 재단 사무총장에게 연락했을 때 사무총장이 한국에도 CCL을 보급하는 게 어떻겠냐고 제안했고, 판검사 및 변호사, 법대 교수로 이뤄진 단체답게 바로 CC측과 양해 각서를 체결하고, 번역 및 한국 법에 맞게 수정 작업에 들어갔다. 이렇게 만들어진 CCL 2.0 KR은 2005년 3월 21일 크리에이티브 커먼즈 코리아를 설립하여 배포하였다.

기본적으로 CCL은 저작권자의 저작물 이용에 대해 제한을 두지 않으므로 자신이 저작권자이거나 저작권자에게 별도의 허락을 받았다면 NC를 달고도 영리적 행위를 하는 것 등이 가능하다. 마찬가지로 CCL이 있더라도 NC 조항이 함께 있는 경우 저작권자에게 허락을 받지 않고 경제적 이득을 위해 기여자가 올린 내용 및 사진을 무단 도용하였다면 저작권법 위반으로 법적인 처벌을 받을 수 있다. 하지만 CCL에 NC 조항이 없는 경우 저작권자의 별도의 허락 없이 상업적 이용이 가능하다.

**3.4.3. 모질라(Mozilla) 퍼블릭 라이선스**

MPL은 Netscape 브라우저의 소스코드를 공개하기 위해 개발된 라이센스이다. MPL은 공개할 소스코드의 범위를 좀더 명확하게 정의하였다. GPL에서는 링크되는 소프트웨어의 소스코드를 포함하여 공개하여야 할 소스코드의 범위가 모호하게 정의되어 있지만 MPL에서는 링크 등의 여부에 상관없이 원래의 소스코드가 아닌 새로운 파일에 작성된 소스코드에 대해서는 공개의 의무가 발생하지 않는다. 따라서 MPL 소프트웨어 그 자체는 어떻게 하든 공개를 해야 하지만 원래 소스코드에 없던 새로운 파일들은 공개하여야 할 의무가 발생하지 않으므로 GPL에 비해 훨씬명확하다. 주요 내용을 요약하면 다음과 같다.

\* 소프트웨어를 배포하는 경우 저작권 표시, 보증책임이 없다는 표시 및 MPL에 의해 배포된다

는 사실을 명시

\* MPL 코드를 수정한 부분은 다시 MPL에 의해 배포

\* MPL 코드와 다른 코드를 결합하여 프로그램을 만들 경우 MPL 코드를 제외한 결합 프로그램에 대한 소스코드는 공개할 필요가 없음

\* 소스코드를 적절한 형태로 제공하는 경우, 실행파일에 대한 라이센스는 MPL이 아닌 다른 것으로 선택가능

\* 특허기술이 구현된 프로그램의 경우 관련 사실을 ‘LEGAL’파일에 기록하여 배포

**3.4.4. 아파치(Apache) 라이선스**

Apache License. 아파치 소프트웨어 재단에서 만든 소프트웨어 라이선스이다.

아파치 라이선스 2.0은 GPL과는 달리 소스 코드 공개의 의무가 존재하지 않고, 2차 라이선스와 변형물의 특허 출원이 가능하다. 라이선스 적용 시 아파치 재단의 이름과 라이선스의 내용을 명시해야 하며, 아파치 라이선스 2.0이 적용된 소스 코드를 수정했을 경우 그 사실을 밝혀야 한다.

아파치 라이선스는 특허 출원이 된 소스 코드의 사용자에게 특허의 무제한적 사용을 허가한다. 따라서 개발자는 그 사용자에 대해 특허권 행사를 할 수 없다. 그러므로 아파치 라이선스 하에서 특허를 보호받고 싶다면 해당되는 소스 코드를 공개하지 말아야 한다. 이는 소스 코드를 공개해 놓고 누군가가 그 코드를 가져다 쓰면, 그걸 빌미삼아 특허권 침해 소송을 제기하는 행위를 방지하기 위함으로 보인다. 반면 아파치 라이선스가 적용된 소프트웨어에 의하여 제3자의 특허가 침해되었을 경우, 해당 소프트웨어에 대한 라이선스는 무효화된다.

**4. 오픈소스 프로젝트 구성 및 개발**

**4.1. 구성원**

대부분 오픈소스 프로젝트 구성원은 다음과 같다.

**컨트리뷰터** : 컨트리뷰션을 하는 모든 사람들을 말한다.

**커미터** : 컨트리뷰션의 내용을 리뷰하고 프로젝트에 반영할지 결정하는 사람. 커미터가 없는 경우도 있다.

**메인테이너** : 프로젝트 방향 설정, 관리하는 사람. 보통 커미터 중 일부가 맡으며, 없는 경우도 있다.

**저작자** : 프로젝트 만든 사람 또는 조직을 말한다.

**컨트리뷰션**

오픈소스 프로젝트에 도움이 되는 모든 활동을 컨트리뷰션이라고 한다. 버그 수정, 기능 추가, 소스 코드 수정, 코드 테스트와 같은 중요한 활동 이외에도, 오타 수정, 번역, 가이드 문서 작성, 디자인 작업, 의견 제시와 같은 사소한 활동도 컨트리뷰션에 해당한다.

사람들이 컨트리뷰션을 하는 이유는 여러 가지가 있다. 첫째로 사용하던 오픈소스를 개선한 경우 자신만 사용하면 버전업이 될 때마다 추가로 패치해야하는 불편함이 있다. 이 불편함을 해소하기 위해 컨트리뷰션을 한다. 둘째로 개인의 개발 실력 향상과 영어능력 향상을 위해서 하는 경우도 있다. 오픈소스 커뮤니티로부터 피드백을 받을 수 있어 특히 개발 실력 향상에 도움이 된다. 셋째로 오픈소스 활동은 공개 기록으로 남기 때문에, 구직 활동 시 자신의 이력으로 사용할 수 있다. 이 외에도 명성을 높이려고 기여하기도 하는 등 다양한 이유로 사람들은 기여를 한다.

**4.2. 개발**

오픈 소스 소프트웨어 프로젝트에서는 ‘공개’라는 단어가 함축하듯이 여러 명의 개발자가 참여하는 분산 개발, 기존에 공개되어 있는 많은 소프트웨어 자원의 이용, 다양한 부류의 자원자들에 의한 소프트웨어 리뷰및 시험 과정, 기술 지원 방법, 기능의 확장, 새로운프로젝트로의 따른 가지치기 과정 등이 상용 소프트웨어와 달리 매우 중요한 의미를 가지게 된다.

새로운 프로젝트의 경우, 비교적 폐쇄적인 초기 개발 단계를 거쳐, 공개된 뒤에, 커뮤니티와 호흡하는 오픈 소스 소프트웨어 순환 구조에 들어간다. 일단 오픈 소스 소프트웨어 순환 구조에 들어가면, 프로젝트 관리자들뿐만 아니라 커뮤니티의 모든 참여자들이 공개된 소스에 접근 가능하며, 기능 추가, 버그 리포트 및수정, 새로운 기능의 요구 등을 함으로써 지속적인 소프트웨어의 개선이 그 안에서 이루어지게 된다.

**5. 오픈소스 개발 도구**

**5.1. 버그 트래커**

버그 트래커(Bug Tracker) 혹은 이슈 트래커는 오픈 소스 개발에서 없어서는 안될 중요한 소프트웨어이다. 이는 제품의 문제점을 발견하고 해결하기 위한 과정을 시스템화 한 것으로 문제 발견 / 증상 규격화 / 원인 규명 / 재구현 / 문제 해결의 과정을 거친다.

**5.2. 버전 컨트롤**

버전 관리(version control, revision control), 소스 관리(source control), 소스 코드 관리(source code management, SCM)란 동일한 정보에 대한 여러 버전을 관리하는 것을 말한다. 공학과 소프트웨어 개발에서 팀 단위로 개발 중인 소스 코드나, 청사진 같은 설계도 등의 디지털 문서를 관리하는데 사용된다.

그러한 문서의 변경 사항들에 숫자나 문자로 이뤄진 ("개정판 번호"나 "개정판 레벨"이라고도 불리는) "버전"을 부여해서 구분한다. "버전"을 통해서 시간적으로 변경 사항과 그 변경 사항을 작성한 작업자를 추적할 수 있다. 간단한 버전 관리 방법으로는 처음 작성한 코드에 버전 번호 1을 부여한다. 변경 사항이 생기면, 버전 번호를 2로 증가 시킨다. 이처럼 추후 변경 사항이 발생 시마다 버전 번호를 1씩 증가시킨다.

**6. 오픈소스 소프트웨어 공학**

**6.1. Mozilla의 개발 프로세스**

6.1.1. 코드 리뷰 절차

앞서 언급한 대로 모질라의 소스 코드는 모듈로 구성되어 있다. 각 모듈의 소스 코드 수정에는 대개 두 단계의 코드 리뷰 규칙을 따른다. 먼저 수정된 패치나 소스가 버그질라를 통해 제출되었을 때, 각 모듈의 소유자가 먼저 분석을 한다. 이 과정에서 허가가 나면 모질라 프로젝트를 움직이는 사람들로부터 슈퍼 리뷰(Super Review)라고 하는 단계를 거친다. 양쪽 모두의 리뷰를 받으면 대부분의 코드는 소스 트리에 커밋(Commit) 된다. 많은 커밋이 수행되기 때문에 코드 수정에 따라 어플리케이션이 동작하지 않거나 컴파일 되지 못하는 상황에 도달할 수 있다.

만일 소스 트리에 문제가 생겼을 경우, 커밋은 중단되고 문제 해결 단계로 넘기는 규칙을 가지고 있다. 슈퍼 리뷰를 하는 사람들은 지금까지 참여 과정에서 그 코딩 기술의 우수성이 잘 검증된 ' 중요 해커(very strong hacker)'로서 꼭 모질라 프로젝트를 이끌어가는 리더일 필요는 없다. 이들은 모듈 소유주가 모듈에 특정한 사항을 중점적으로 검토한 후에 혹시 놓쳤을 수도 있는 문제점을 검토해 준다. 이러한 소스 코드는 몇 주에 한번씩 새로운 스냅샷(Snapshot)을 만들어 버그를 수정하게 된다.

개발 로드맵에 따라 알파와 베타 단계를 거쳐 제품으로 출시해야 할 때는 최상위 드라이버

(Driver)라고 불리는 프로젝트 관리자들의 승인을 받아야만 소스를 고칠 수 있다. 이 때 소스를 고칠 때는 패치가 고치는 버그의 심각성, 영향을 받는 사용자 및 플랫폼, 복구 가능성 등을 잘 설명해서 드라이버를 설득해야 한다. 또한, 드라이버들은 소스 코드가 목표로 한 개발 로드맵에 따라 제대로 준비되었는지 소스의 상태를 점검하고 일정을 만드는 작업까지 하고 있다.

6.1.2. 제품 출시 및 QA 절차

모질라의 제품 출시는 기본적으로 원래 소스 코드 저장소인 트렁크(Trunk)로부터 저장소 분기(Branch)를 하는 것부터 시작한다. 각 저장소 소스 코드는 매일 자동 컴파일된 완성 버전인 일일빌드(Nightly Build)가 나와서 개발자들이 소스 수정에 대한 검증 작업을 하는 데 용이하다. (일일빌드는 수 시간 단위로 나오기도 한다.)

모질라의 제품은 로드맵에 따라 우선 순위 기능이 해결되는 단계에서 알파 버전을 내게 된다. 대략 3~5회 정도의 알파 버전이 나오게 되며 신 기능에 대한 테스트 및 버그 수정을 완성하게 된다. 베타 버전은 신 기능 탑재가 거의 완료되고 소소한 버그와 성능 이슈, 안정성, 지역화 메시지 고정 등을 위해 작업을 하며 대략 3~5회 정도의 과정을 거친다. 각 단계 마다 소스 코드 트리를 막고 빌드를 만든 후 품질 검증 과정(QA)을 거치게 되는 데 대략 2주 정도가 소요된다.

QA과정은 기능 검증과 보안 검증 등 크게 두 가지로 나누어 실행한다. 기능 시험은 대개 QA팀에 자원 봉사를 하는 테스터들이 시행을 하고 하루를 정해 리트머스(Litmus.mozilla.org)라는 QA웹 사이트 도구를 이용한다. 이 도구는 기능에 대한 테스트 단계를 제시하고 예상 결과에 맞는지 여부를 테스트들이 확인 해서 기입하도록 하고 있다. QA 단계에서 버그가 발견되면 대개 리그레션(Regression)이라는 방식으로 이전 기능으로 원상 복구하여 제품 출시 사이클에는 문제가 생기지 않도록 하고 있다.