공개SW-09-03

공개SW Governance v1.0

공개SW 역량프라자 운영 사업

2009. 12. 15

주관연구기관 정보통신산업진흥원

지 식 경 제 부

요 약 문

1. 개요

“공개SW Governance v1.0"

“공개SW Governance v1.0"는 공공기관에서 정보화시스템 구축/전환시에 공개SW를 효율적으로 도입하고 사용ㆍ유지보수 함에 있어 지침서 역할을 하도록 개발되었다.

2. 공개SW Governance 개발 목적 및 의의

공개SW Governance는 공공기관이나 조직에서 정보화 담당자가 공개SW를 도입할 때 공개SW의 전반적인 이해를 돕고 효율적으로 도입 할 수 있도록 표준화된 프로세스를 제시하고 최적화된 관리방안을 제공하며, 도입, 사용, 유지보수 등에 관한 지침서로서 위험요소를 최소화하여 경영가치를 극대화 할 수 있도록 하였다.

또한, 공개SW Governance는 전문가 워킹그룹을 통하여 시시각각 변하는 국내외 공개SW 환경에 적절히 대응하고 올바른 방향설정을 위해 지속적인 자문과 검증 작업을 거쳐 작성이 되었으며, 공개SW Governance는 향후 공개SW 기반 정보시스템 구축사업을 위한 예산수립, 발주, 조직구성, 프로세스 등에 있어 최적화된 기준점을 제시 한다.

3. 주요 내용

공개SW Governance 개발의 추진 전략은 크게 4단계로 나누어 추진하게 된다. 첫째, 현황분석을 통하여 공개SW에 대한 이해를 높이며, 공개SW의 도입 실태, 국가정책, 기술 동향 등을 분석한다. 둘째, IT Governance Framework을 Tailoring 하여 공개SW Governance Framework을 수립 한다. 공개SW Governance Framework은 크게 공개SW 영역, 정책 및 가이드 영역, 운영영역으로 나뉘며 통제 등의 관리 기능보다는 지원과 정보제공을 위한 공개SW Governance Framework을 구성한다. 셋째, 수립된 공개SW Governance Framework에 따른 정책과 기본 원칙을 정의하며, 도입을 위한 프로세스를 정립한다. 넷째, 수립된 공개SW Governance의 수요처 확보방안과 수요처 특성에 따른 보급계획을 수립한다. 이러한 일련의 추진 과정에서 공개SW Governance의 올바른 방향설정 및 개발을 위해 워킹그룹을 구성하여 지속적인 자문과 검증업무를 수행하고 있다.

공개SW Governance는 조직이나 기관이 처한 환경, 비즈니스모델, 기술 수준 등에 차이가 있기 때문에 일률적으로 적용하기는 힘들지만, 각각의 내용들을 추상화시켜 일반적인 내용을 제시하고, 조직이나 기관들로 하여금 각자 처한 환경에 따라 구체화하여 사용할 수 있도록 하는 방안을 고려하여 이하와 같이 공개SW Governance를 개발 하였다.

공개SW Governance는 크게 공개SW 영역, 정책 및 가이드 영역, 운영 영역으로 구분이 되는데, 공개SW 영역에서는 공개SW의 3대 요소를 라이선스, 커뮤니티, 방법론으로 정의하고 이해 대한 상세한 정보를 제공하였다. 정책 및 가이드 영역에서는 공개SW Governance의 현황을 진단하고, 가이드 기본원칙이 되는 정책을 수립, 사례연구 등을 통해 공개SW를 안전하고 체계적으로 도입할 수 있도록 하였다. 운영 영역에서는 공개SW Governance의 프로세스, 컴플라이언스 팀, 리뷰보드, 공개SW 도입을 위한 전체 프로세스 등을 정립하여 공개SW Governance를 효과적으로 활용할 수 있도록 하였다.

이렇게 개발 된 공개SW Governance는 향후 자문위원과 실 사용자들의 지속적인 검증을 받아 유기적인 고도화가 필요하며 안정된 공개SW 사용 환경 조성을 위해 공개SW 라이선스 검증 툴 개발과 지원이 지속되어야 한다.

4. 공개SW Governance 활용에 대한 색인

목 차

1. 개요

가. 공개SW Governance 개발목적

본 공개SW Governance 개발의 목적은 공공기관이나 조직에서 공개SW를 도입할 때 체계적으로 도입할 수 있도록 도입에 관한 표준화된 프로세스를 제시하고, 공개SW를 사용함에 있어 최적화된 관리방안을 제공하며, 도입 및 관리에 관한 지침서로서 위험요소를 최소화하여 경영가치를 극대화함에 그 목적이 있다.

[그림1-1. 공개SW Governance 개발 목적]

나. 공개SW Governance 추진배경

(1) 공개SW Governance 개발 필요성

중앙행정기관, 지방자치단체, 정부투자기관 및 기타 공공기관, 민간기업 등이 공개SW 기반 정보시스템 구축 사업을 위한 예산수립 및 사업발주 시 전략수립, 조직구성, 프로세스 및 기술적용에 있어 체계적이고 최적화된 기준점을 제시한다.

공공기관이나 개별 기업에서 소프트웨어를 사용하고자 하는 경우 일반적으로 구매팀을 통해 특정 벤더와 계약을 체결하고, 동 소프트웨어의 라이선스 조건을 준수하면서 사용하게 된다. 그리고 대부분의 경우 공공기관이나 개별 기업들은 소프트웨어의 구매와 사용에 관한 일정한 프로세스를 가지고 있다. 그런데 이러한 일반적인 소프트웨어의 관리프로세스가 있음에도 불구하고, 공개SW에 대한 별도의 관리전략이 필요하다. 그 이유는 공개SW가 일반적인 소프트웨어와는 다른 특징들을 가지고 있기 때문이다. 예를 들어, 현재 공공기관이나 개별 기업에서 사용하고 있는 대다수의 공개SW가 구매팀을 통하지 않고 다른 경로를 통해 공공기관이나 개별 기업으로 들어오고 있다는 점이다. 많은 경우 제품개발 과정에서 개발자들이 직접 인터넷을 통해 가져오고 있으며, 또는 하청업체 등이 공개SW를 사용함으로써 공공기관이나 개별 기업으로 유입되는 경우도 많다. 그 결과 공공기관이나 개별 기업의 경영층이나 관리자들이 기업 내에서 사용되고 있는 공개SW를 파악하지 못하는 경우가 많다.

이 밖에 공개SW가 일반적인 소프트웨어와 다른 점은, 일반적으로 상용소프트웨어로부터 기대할 수 있는 특성들을 가지고 있지 않다는 점이다. 그리고 공개SW는 특별한 권리와 의무를 가진 공개SW 라이선스에 의해 배포되고 있다는 점을 지적할 수 있다.

그렇다면, 공공기관이나 개별 기업들에서는 이와 같이 특별한 취급을 요구하고 있는 공개SW를 어떻게 다룰 것인가? 이에 대해서는 각 공공기관이나 개별 기업이 처한 환경, 즉 비즈니스 모델, 기술 수준 등에 차이가 있기 때문에 일률적으로 말하기 어렵다. 다만, 각각의 내용들을 추상화 시켜 일반적인 내용을 제시하고, 각 공공기관이나 개별 기업들로 하여금 개별공공기관이나 개별 기업이 처한 환경에 따라 구체화하여 사용할 수 있도록 하는 방안을 고려할 수 있다.

다. 공개SW Governance 추진방법

(1) 공개SW Governance 추진 전략

공개SW Governance 개발의 추진 전략은 크게 4단계로 나누어 추진하게 된다. 첫째, 현황분석을 통하여 공개SW에 대한 이해를 높이며, 공개SW의 도입실태, 국가정책, 기술 동향 등을 분석한다. 둘째, IT Governance Framework을 Tailoring 하여 공개SW Governance Framework을 수립 한다. 공개SW Governance Framework은 크게 공개SW 영역, 정책 및 가이드 영역, 운영영역으로 나뉘며 통제 등의 관리 기능보다는 지원과 정보제공을 위한 공개SW Governance Framework을 구성한다. 셋째, 수립된 공개SW Governance Framework에 따른 정책과 기본 원칙을 정의하며, 도입을 위한 프로세스를 정립한다. 넷째, 수립된 공개SW Governance의 수요처 확보방안과 수요처 특성에 따른 보급계획을 수립한다. 이러한 일련의 추진 과정에서 공개SW Governance의 올바른 방향설정 및 개발을 위해 워킹그룹을 구성하여 지속적인 자문과 검증업무를 수행 할 계획이다.

(2) 공개SW Governance 워킹그룹 구성

공개SW Governance를 개발하는데 있어 국내의 개발사례는 도입 안내 가이드 정도의 수준에 머물러 있으며 체계적인 원칙수립 및 정책이 미비하여 시시각각 변화하는 국내외 공개SW 환경에 적절히 대응하기 못하고 있는 상황이다. 이에 본 사업을 통해 체계적인 공개SW Governance를 개발하였으며 더 나아가 각계 전문가 집단의 의견 수렴과 자문을 통하여 최적화된 공개SW Governance가 수립 되도록 하였다.

□ 워킹그룹 구성 방안

공개SW 워킹그룹은 공개SW 관련 각계 전문가들로 구성함을 원칙으로 한다.

□ 워킹그룹 수행업무

○ 추진방향 설정 :

- 공개SW Governance 추진방향에 대한 자문 및 검증업무 수행

○ 공개SW Governance 모델 자문

- 공개SW Governance 수립에 대한 자문, 상세 모델에 검토 및 관련자료 제공
- Governance 수립을 위한 각 단계별로 의견수렴 및 결과에 대한 감수

○ 선진사례 검토

- 국내외의 공개SW Governance 우수사례에 대한 적용성 검토

○ 의사결정 자문

- 성공적인 공개SW Governance 수립을 위한 의사결정에 대한 자문활동

2. 공개SW 이해

가. 공개SW 정의

공개SW(Open Source Software)는 저작권자가 소스코드를 공개하여 누구나 특별한 제한 없이 자유롭게 사용, 복제, 배포, 수정할 수 있는 소프트웨어 이다. 오픈소스는 자유 소프트웨어 재단(FSF, Free Software Foundation)의 자유 소프트웨어(Free Software)를 포함한 넓은 의미로 사용된다.

공개SW는 누구라도 소스코드를 읽을 수 있고 사용자가 능력이 있다면 각종 버그의 수정은 물론이고 그것을 개작하여 기능을 추가할 수 있으며, 누구나 소프트웨어 개발에 참여할 수 있다. 따라서 공개소프트웨어는 소스코드에 접근할 수 있는 권리, 프로그램을 복제하여 배포할 수 있는 권리, 프로그램을 개선할 수 있는 권리를 개발자에게 보장한다.

이러한 공개소프트웨어는 소스코드에 대한 접근성이 보장되므로 시스템간의 호환성을 확보할 수 있을 뿐 아니라 사용자의 요구에 부합하는 일관성과 함께 일치성을 보장받을 수 있다. TCO(Total Cost Ownership)[1]가 낮다는 점에서 경제적인 효과를 얻을 수 있다.

대표적인 공개소프트웨어운동 비영리조직인 Open Source Initiative(OSI)는 특정 소프트웨어가 공개 소프트웨어로 분류되려면 다음과 같은 10여 개 조건을 충족해야만 한다고 정의하고 있다.

① 자유 배포(Free Redistribution)
특정한 소프트웨어의 라이선스에는 해당 소프트웨어의 일부나 전부가 다수의 프로그램으로 구성되는 배포판의 일부로 포함되어 재 배포되지 못하도록 배포나 판매상의 제한을 설정할 수 없다. 또한 이러한 종류의 배포판에 대한 판매나 양도에 있어서 별도의 라이선스 비용을 징수할 수 없다.

② 소스코드 공개(Source Code Open)
프로그램 저작물에는 반드시 소스 코드가 포함되어야 하며, 컴파일 된 형태뿐만 아니

1) TCO(Total Cost Ownership)란 총소유비용을 정보시스템 구축에 필요한 하드웨어, 소프트웨어 등 초기 도입비용 뿐만 아니라 유지관리비용, 교육비용 등 정보시스템 구축과 운영에 필요한 모든 비용을 포괄하는 비용을 의미한다.

라 소스코드의 배포 또한 허용되어야 한다. 만약 소스코드를 제외한 상태로 배포하고자 한다면 일반적으로 통용되는 매체를 이용해서 제작 실비에 준하는 비용으로 소스코드를 제공해야만 한다. 이 경우 가장 바람직한 방법은 인터넷을 통해서 소스 코드를 무료로 다운로드 받을 수 있도록 하는 것이다. 소스코드는 프로그래머들이 개작하기에 용이한 형태로 제공되어야 하며, 고의로 복잡하고 혼란스럽게 만들어진 형태와 선행 처리기나 번역기에 의해서 생성된 중간 형태의 코드는 허용되지 않는다.

③ 2차적 저작물(Derived Works)
라이선스에는 프로그램 원 저작물의 개작이나 이를 이용한 2차적 프로그램의 창작이 허용되어야 하며, 이러한 파생적 프로그램들은 최초의 프로그램이 갖고 있던 라이선스의 규정과 동일한 조건하에서 재 배포될 수 있어야 한다.

④ 소스코드 수정 제한(Integrity of The Author's Source Code)
빌드 과정을 통해서 프로그램을 개작할 목적으로 소스 코드와 패치 파일을 함께 배포할 경우에는, 정상적인 빌드를 보장하기 위해서 라이선스 안에 소스코드의 수정을 제한하는 항목을 추가할 수 있다. 그러나 이러한 경우에도 수정된 소스 코드를 이용해서 만들어진 소프트웨어에 대한 자유로운 배포를 허용해야 하며, 수정된 소스코드를 통해서 만들어진 2차적 프로그램을 원래의 프로그램과 구별하기 위해서 별도의 이름과 버전을 사용할 것을 요구하는 항목을 추가할 수 있다.

⑤ 개인이나 단체에 대한 차별 금지 (No Discrimination Against Persons or Groups)
라이선스는 모든 개인과 단체에 대해서 동일한 기준으로 적용되어야 한다.

⑥ 사용 분야에 대한 제한 금지 (No Discrimination Against Fields of Endeavor)
라이선스 안에 특정한 분야에 종사하는 사람에 대한 프로그램 사용상의 제한을 설정할 수 없다. 예를 들면, 유전연구나 상용 사업체에서는 해당 프로그램을 사용할 수 없다는 등과 같이 특정한 분야에 대한 사용을 금지하는 제한을 설정해서는 안 된다.

⑦ 라이선스의 배포 (Distribution of License)
프로그램에 대한 권리는 반복되는 배포에 따른 별도의 라이선스 승인이나 양도 과정 없이도 프로그램을 배포 받은 모든 사람에게 동일하게 적용된다.

⑧ 라이선스 적용상의 동일성 유지 (License must no be specific to a product)
프로그램에 대한 권리는 반복되는 배포 과정에서 특정한 배포판에 포함되어 있는 상태로만 유효하지 않고, 모든 배포 단계에서 동일한 효력을 갖는다. 만약, 특정한 배포

판에 포함되어 있던 프로그램을 독립적으로 사용하거나 재배포한다면 해당 프로그램을 배포받는 사람은 프로그램이 포함되어 있던 최초의 배포판 상태에서 발생된 권리와 동일한 권리를 갖는다.

⑨ 다른 라이선스의 포괄적 수용 (License must not contaminate other software)
라이선스에 오픈 소스 소프트웨어와 함께 배포되는 소프트웨어에 대한 제한을 설정해서는 안 된다. 예를 들면, 동일한 매체를 통해서 배포되는 소프트웨어는 모두 오픈 소스 소프트웨어이어야 한다는 제한으로 인해서 다른 라이선스 기준을 따르는 소프트웨어가 함께 배포될 수 있는 형태를 금지해서는 안 된다.

⑩ 라이선스의 기술적 중립성 (License must be Technology-Neutral)
라이선스의 어떠한 규정도 개별기술 또는 인터페이스 형태에 기초하여 규정하지 말 것

나. 공개SW 특성

(1) 공개SW 장점

[표2-1. 공개SW 장점]

낮은 진입비용
공개SW는 무료로 다운로드 및 소스코드의 수정/재배포가 가능 하므로 초기 개발비용이 적게 요구 된다. 일반적으로 1/2 정도의 비용이 적게 드는 것으로 알려져 있다.
빠르고 유연한 개발
공개SW 커뮤니티는 보통 최신 기술 정보 및 문제점과 해결책을 공유하는 형태로 자유롭게 운영되기 때문에 독점 프로그램에 비해 기술 발전 속도가 빠르다.
오픈 포맷과 프로토콜 호환성
공개SW는 주로 오픈 포맷 또는 프로토콜을 사용하기 때문에 서로 다른 소프트웨어 간 상호 연동성이 보장된다. 모든 기기들이 서로 다른 네트워크를 통해 하나로 연결되는 유비쿼터스 시대에는 필수적인 요소로 볼 수 있다. 또한 공개SW 운동의 주원인 역시 상용 프로그램들 간의 비호환성을 해결하는 것이다.
신뢰성과 안정성
공개SW의 개발 과정을 볼 때, 전 세계에 있는 수많은 우수한 개발자들이 직접 개발과 디버깅 과정에 참여하기 때문에 In-house에서 폐쇄적으로 개발되는 독점 프로그램에 비해 비교적 안정적으로 동작한다는 평이 일반적이다. 하지만 이러한 신뢰성과 안정성은 많은 개발자들의 적극적인 참여가 있을 때에만 가능한 것이기 때문에 사용하고자 하는 공개SW의 개발 과정을 주의 깊게 살펴볼 필요가 있다.
네트워킹 지원
공개SW가 확산된 가장 큰 이유가 다양하고 강력한 네트워킹 기능 지원이라 해도 과언이 아닐 것이다. 대표적으로 아파치는 전 세계 웹 서비스의 절반 이상을 차지하고 있을 정도이다. 또한 공개SW 네트워킹 솔루션은 대부분 상용 프로그램과 호환되기 때문에 상품화에도 아주 잘 활용될 수 있을 것이다.

(2) 공개SW 단점

[표2-2. 공개SW 단점]

애플리케이션의 부족
대부분의 사용자들은 윈도우즈기반의 GUI(Graphical User Interface)를 갖고 있는 Application에 익숙해 있지만, 이에 버금가는 리눅스기반의 Application이 많이 부족한 것이 현실이다. 또한 리눅스기반으로 개발된 많은 Application들은 윈도우즈기반 Application들과 호환되지 않는 문제점도 있다.
빈약한 문서
상용프로그램에 비해 공개SW는 체계적인 문서를 갖고 있지 못한 단점이 있다. 이는

경우에 따라서는 개발과정의 지연을 초래할 수도 있기 때문에 활용하고자 하는 오픈소스가 얼마만큼 문서화가 잘 되었는지도 살펴보아야 한다.
불확실한 개발 로드맵
공개SW는 영리를 목적으로 하는 회사에서 개발되는 것이 아니라, 자발적 참여를 바탕으로 웹상에서 자유롭게 개발되는 것이 특징이다. 그렇기 때문에 독점프로그램에서 볼 수 있는 로드맵을 기대하기 힘든 면이 있다. 공개SW의 이러한 단점은 공개SW를 활용하는 개발 과제의 로드맵에 까지 영향을 미칠 수 있기 때문에 활용하고자 하는 공개SW의 향후 개발 계획이 얼마나 체계적으로 세워져 있는지도 고려해야 한다.
지적재산권
기업이 보유한 특허 및 소스코드가 오픈소스에 포함되는 경우 대부분의 공개SW 라이선스에서는 일반적으로 특허에 대한 사용료 없이 배포할 것을 요구하고 있다. 따라서 특허에 대한 사용료 없이 배포하기를 원치 않을 경우에는 해당 공개SW를 사용할 수가 없으며, 또한 사용 후 로열티를 주장하게 되면 해당 공개SW에 대한 사용권한이 박탈되는 경우가 일반적이다. 따라서 공개SW를 활용하여 특허를 구현하거나 기존 소스코드를 포함하고자 할 경우, 반드시 특허 사용료에 대한 입장을 명확히 하여야 할 것이다.

(3) 공개SW 비용

[표2-3. 공개SW 비용]

License fees
공개SW는 소스가 공개되는 있는 소프트웨어로, 일반적으로 공개SW는 무료라는 편견을 가지고 있다. 그러나 공개SW는 엄연히 라이선스를 가지고 있는 소프트웨어로 라이선스 비용이 발생할 수 있다.
Maintenance
업데이트 및 업그레이드 비용
Support
기술지원 비용
Training
공개SW에 대한 개발자, 관리자 교육 시간
Planning
분석, 자산 관리, 개발 비용
Management
유지보수(관리, 모니터링, 제어) 비용
IT overhead
설치, 사용, 관리 비용
Security

보안 관련 패치의 설치/개발 비용
Migration and integration
기업이 보유하고 있는 소프트웨어를 공개SW로 전환할 경우에는 Migration과 Integration에 비용이 발생한다. 특히 OS와 같은 Infra성 공개SW가 변경될 경우 해당 공개SW와 관련된 H/W, SW 등을 고려하여야 한다. 기업에 Migration 관련 무료 컨설팅을 통해 새로운 비즈니스 기회(H/W 판매, 인력 투입 등)를 창출하기도 했었다.
Hardware/maintenance
H/W 유지보수 비용

(4) 공개SW 이익

[표2-4. 공개SW 이익]

구분	내용
Architectural efficiency	필요한 IT 기능을 적시에 사용 가능
Business flexibility	필요할 때 필요한 기능을 가져 쓸 수 있음
Vendor leverage	벤더에 대한 영향력 강화
Higher quality	IT 아키텍처와 어플리케이션에 대한 결함율 감소
Fast problem solving	커뮤니티를 통한 빠른 문제 해결
Improved IT skills	IT 담당자의 역량 강화

(5) 공개SW 위험

[표2-5. 공개SW 위험]

구분	내용
Hidden costs	SW 라이프사이클 비용 발생 가능
Lack of support	오픈소스에 대한 지원의 보장이 부족
Missing features	필요한 기능의 부재
Lack of operational management	오픈소스 관리 도구의 부재, 유명한 오픈소스의 경우 관리 도구 또는 모니터링 도구를 판매하는 업체도 있다.
Unpredictable releases	고객의 요구에 부합하지 않는 릴리스 계획
Security	오픈소스의 경우 커뮤니티에서 제공하는 최신 보안 patch의 반영을 소홀히 할 경우 보안 위험이 발생할 수 있다.
IP liabilities	개발자의 라이선스 위반의 위험성 내포

(6) 비공개SW와 비교

앞서 공개소프트웨어에 대한 역사와 정의에 대하여 알아보았다. 이제 공개소프트웨어에 대한 현실적인 활용가치와 기능성 등을 확인하기 위하여 비공개소프트웨어와 공개소프트웨어를 차이점을 살펴보면 다음과 같다.[2)]

첫째, 가장 대표적인 차이점으로서 비용면에서의 차이점이다.

비공개소프트웨어는 라이선스에 대한 수수료가 발생하는 반면 공개소프트웨어는 라이선스 비용은 발생하지 않는다. 하지만 기술지원이나 서비스에 대한 수수료는 발생한다. 또한 비공개소프트웨어는 시스템에 대한 초기 적용비용이 높지만 공개소프트웨어는 초기 적용비용이 매우 낮거나 심지어는 전혀 들지 않는다.

둘째, 성능 면에서의 차이점이다.

비공개소프트웨어는 규모가 큰 시스템 환경에서 비교적 높은 성능을 나타내는 반면 공개소프트웨어는 비교적 규모가 작은 시스템 환경에서 높은 성능을 나타낸다. 리눅스커널 2.6.X이상에서는 중대형 시스템에도 안정적 성능을 나타내고 있으며 이러한 추세는 앞으로 더욱 활발해 질 것으로 예상된다.

셋째, 보안면에서의 차이점이다.

비공개소프트웨어는 프로토콜의 호환이 어려워 인증체계가 취약하며 폐쇄적인 운영으로 인한 공개되지 않는 취약점이 존재할 수 있는 반면 공개소프트웨어는 개발 시부터 공개되어 이미 많은 취약점이 개선 및 해결되었으며 안정된 상태에서 운영되고 있다.

넷째, 기술적인 면에서의 차이점이다.

2) 공개소프트웨어는 소스코드의 공개, 개작, 재배포가 자유로운 소프트웨어를 의미하는 것이며, 비공개소프트웨어는 이러한 권리가 제한된 소프트웨어를 의미한다. 공개소프트웨어를 상용소프트웨어와 반대되는 개념의 소프트웨어로 이해하는 경우가 있으나, 이는 공개소프트웨어를 무료 소프트웨어로 잘 못 이해하고 있기 때문에 생긴 오해이다. 공개소프트웨어도 유료 즉 상용으로 배포할 수 있다.

비공개소프트웨어는 프로젝트의 연속성 및 재사용성이 낮은 반면 공개소프트웨어는 소스코드의 공개로 인하여 언제든 재사용이 가능하기 때문에 프로젝트의 영속성이 보장되며 유지보수 및 업그레이드 비용이 낮다.

다섯째, 확장성에 대한 차이점이다.

비공개소프트웨어는 높은 적용비용과 제한된 시스템 운영환경으로 인하여 확장성이 완벽하게 보장되는 것은 아니다. 하지만 공개소프트웨어는 소스가 공개되어 있기 때문에 시스템의 확장성이 확실히 보장된다는 장점이 있다.

여섯째, 공급권에 대한 차이점이다.

비공개소프트웨어의 생산업체는 하나이므로 독점 및 단일공급 된다. 단, 유통회사는 여러 개 존재할 수 있지만, 원 공급 업체는 하나이다. 반면 공개소프트웨어는 동일한 솔루션에 대하여 다수 업체들로 부터 공급이 가능하기 때문에 사용자의 공급업체 선택권이 넓은 편이다.

일곱째, 저작권에 대한 차이점이다.

비공개소프트웨어는 일반라이선스를 채택하며 독점공급을 하므로 공급업체의 가격결정에 있어 매우 유리하다. 반면 공개소프트웨어는 거의 대부분 GPL 라이선스, BSD 라이선스, MPL 라이선스 등을 원하는 대로 적용할 수 있다.

여덟째, 경쟁력에 대한 차이점이다.

공개소프트웨어는 우수하고 핵심적인 소프트웨어에 대한 기술과 개발능력 및 저작권을 이미 보유하고 있는 소프트웨어 선진국에 매우 유리한 반면 공개소프트웨어는 소스공개로 인한 핵심기술이 공개되어 있기 때문에 라이선스가 아닌 기술과 개발 능력 뿐 아니라 서비스로 평가받을 수 있으며 소프트웨어의 핵심기술을 가질 수 있는 기회가 있기 때문에 소프트웨어 후발국에 매우 유리하다.

비공개소프트웨어와 공개소프트웨어의 차이점을 요약하면 다음 표와 같다.

[표2-6. 비공개소프트웨어와 공개소프트웨어 비교]

구 분	비공개소프트웨어	공개소프트웨어
비용분석	적용비용이 높음 유지비용 및 시스템 개선비용이 높음 소수의 관리자에 대한 관리비용 높음 라이선스 수수료에 대한 비용발생	적용비용이 낮음 유지비용이 낮고 기능 확장에 대한 추가비용이 낮음 다수의 공개된 사용자에 의한 관리로 관리자에 대한 관리비용이 낮음 서비스에 대한 비용발생
성능분석	규모가 큰 시스템 환경에서 비교적 높은 성능 고가의 장비로 인한 고성능 전체적으로 공개SW와 비슷한 성능	다양한 환경에 최적화된 설정으로 높은 성능 높은 안정성 및 비용효율이 높음
보 안 성	폐쇄적인 운영으로 인한 공개되지 않은 시스템 취약점 보유 최근에 다수의 취약점 발견으로 많은 보안 위협에 노출 프로토콜 호환이 어려워 인증체계가 취약함	개발 시부터 공개되어 이미 많은 취약점이 해결된 안전화 상태 공개키 기반의 인증 메커니즘 구현을 위한 통합패키지존재(적용용이) 다양한 암호화 알고리즘 및 키 관리에 대한 기능 제공
기 술 성	재사용성 없음	재사용성 높음 유지보수, 업그레이드용이 독점폐해방지
저 작 권	일반라이선스 독과점에 의한 가격결정우려	GPL라이선스 BSD라이선스 등
확 장 성	소프트웨어간의 호환성이 보장되나 높은 적용비용과 제한된 시스템 운영환경	소프트웨어간의 적용비용이 거의 들지 않고 낮은 수준에서의 기능추가 가능
경 쟁 력	소프트웨어선진국에 유리	공동개발상식에 따른 교육효과우수 우리나라와 같은 후발국에 적합
공 급 권	개발업체로 부터의 단일공급	다수 업체로 부터의 공급가능

다. 공개SW 기원

공개소프트웨어는(Open Source Software) 1984년 자유소프트웨어(Free Software)운동으로부터 분화하여 현재에 이르렀으며[3], 2000년 이후 전 세계의 정부기관과 민간단체에서는 공개소프트웨어의 효율성과 발전성에 대하여 확고한 의지를 가지고 정책추진과 사업화를 시도하고 있다. 국내에서도 2002년부터 공개소프트웨어 정책추진을 위한 본격적인 사업을 착수하여 현재 그 성과를 보이고 있다.

자유소프트웨어 운동(Free Software Movement)은 1984년 리차드 스톨만(Richard Stallman)이 GNU[4]라는 공개프로젝트를 시작하면서 부터 시작되었으며 리눅스와 함께 자유소프트웨어 운동도 발전하였는데, 이 과정에서 "자유(Free)"라는 용어에 대한 실용주의적 관점에서의 비판이 이루어지고 공개소프트웨어 운동(Open Source Software Movement)이 시작되었다.

자유소프트웨어 운동 진영은 소프트웨어 자체는 항상 윤리적, 도덕적, 사회적으로 타당해야함을 강조하면서 소프트웨어 사용자들에게 이를 바탕으로 한 사용상의 여러 가지 자유[5]가 주어져 있음을 또한 강조하고 있다[6].

그리고 공개소프트웨어 운동 진영은 이념적 측면보다는 개발방식에 초점을 맞춘 실용적 접근을 취하고 개발자에게도 경제적인 보상을 제공해야 한다는 실용적 측면을 강조한다.

자유소프트웨어는 원칙에 입각한 소스코드의 자유로운 사용이라는 측면을 공개소프트웨어는 사회적 실용성과 경제성에 중심을 두고 소스코드의 공개라는 측면을 강조한다고 할 수 있다.

공개소프트웨어가 정부의 정책에 반영되기 시작하고 이로 인하여 일반인들에게 알려지기 시작한 것은 불과 몇 년 전 부터이다. 공개소프트웨어가 현재에 이르기까지 어떻게 진화해 왔는가를 살펴보려면 먼저 그 역사를 살펴보아야한다. 공개소프트웨어에 관련된 주요한 사건들을 년대별로 간략하게 정리하였다.

□ 1950년대

3) 자유소프트웨어와 공개소프트웨어는 그 역사를 함께 해오고 있으며 같은 개념으로 보는 견해도 있다.(Debian GNU/Linux 개발자, What Does Free Mean? or What Do You Mean by Open Software? (visited Aug. 23 2002) <http://www.debian.org/intro/free>)

4) GNU는 "Gnu is not Unix"의 재귀적 표현임

5) 여기서 자유는 첫째, 어떤 목적으로도 프로그램을 가동시킬 수 있는 자유 둘째, 필요에 맞게 소프트웨어를 수정할 자유, 셋째, 무료 또는 유로로 복사본을 재배포할 수 있는 자유 넷째, 전체 공동체가 혜택을 볼 수 있도록 프로그램의 수정본을 배포할 수 있는 자유를 의미한다.

6) Richard Stallman, Why "Free Software" is better than "Open Source", <http://www.gnu.org/philosophy/free-software-for-freedom.html>

1950년대는 컴퓨터 프로그램의 초기 사용시기로써 모든 프로그램은 공개적으로 개발되고 또한 소스코드도 공개되었다. 즉, 여기서 중요한 것은 컴퓨터 프로그램의 역사는 공개적으로 개발되기 시작하였고 또한 개발된 소스코드는 공개되었다는 점이다. 이러한 초기 인식이 약 10년 정도 계속되었으나 1950년대 후반부터는 컴퓨터 프로그램이 일부이기는 하지만 판매되기 시작하였다.

□ 1960년대

판매되기 시작한 컴퓨터 프로그램이 1960년대 후반에 와서는 소프트웨어라는 상품으로 포장되어 정식으로 판매되는 상품으로 인식되기 시작하였다. 그리고 이때부터 소프트웨어는 개발한 회사의 일급비밀이 되어 소스코드를 공개하지 않게 되었다. 하지만 이때에도 UNIX를 포함한 공개소프트웨어는 소스코드가 계속 공개되고 있었다. 즉, 1960년대 중 후반기에는 공개소프트웨어와 비공개소프트웨어가 공유한 시기라고 볼 수 있다.

□ 1970년대

1970년대도 UNIX 소스코드는 많은 대학에서 자유로이 사용하고 수정하고 할 수 있었다. UNIX에서 사용될 수 있는 많은 공개소프트웨어들도 자유로이 수정되고 사용되었다. 하지만, 1980년대에 들어서면서 상용UNIX가 대두되기 시작한다. 즉, AT&T는 UNIX의 수수료를 높이고 그 배포를 제한하게 되었다. 이로써 UNIX 소스코드의 수정 본을 공유해왔던 많은 사용자들이 소프트웨어를 수정하는 것을 불가능하게 만들게 되었다.

□ 1980년

공개되어 왔던 UNIX 소스코드가 1980년대에는 상용 UNIX의 대두로 인하여 다시 회사의 일급비밀이 되기 시작하였으며 대학들에서도 UNIX에 대한 라이선스를 적용하기 시작하였다. 즉, 1980년도부터 공개소프트웨어의 개념이 희박해지면서 독점 소프트웨어가 본격적으로 출현하여 소프트웨어를 완전히 제품화하여 유통하고 판매한 시기라고 할 수 있다.

공개 및 공유되어 왔던 소프트웨어 소스코드가 완전히 상품화, 사유화, 제품화되어 기업의 수익을 위한 상품으로 전락한 것에 회의를 느낀 MIT의 리차드 스톨만은 "소프트웨어는 공유되어야 한다."라는 소신을 가지고 그 뜻을 펼치기 위하여 1984년도에 GNU라는 프로젝트를 시작하였다. 결국 이것은 1984년에 UNIX와 완벽하게 호환되는 자유소프트웨어 운영체제를 개발하기 위한 GNU프로젝트가 시작하게 된

계기가 되었다. 즉, 소스코드에 대한 사용자들의 권리를 되찾기 위한 운동의 시작이라고 할 수 있다. 이것이 리차드 스톨만의 자유 소프트웨어운동의 시작이다.

자유소프트웨어를 보호하고 자유 소프트웨어운동을 본격적으로 촉진하기 위하여 리차드 스톨만은 1985년에 자유 소프트웨어재단, 즉, FSF(Free Software Foundation)을 설립하였다. 자유소프트웨어 운동이란 모든 소프트웨어를 누구든지 자유로이 사용할 수 있고, 필요에 따라 변경을 할 수 있으며, 변경된 소프트웨어를 자유로이 배포할 수 있게 하기 위함이 목적이며 이를 위해서는 소프트웨어의 소스코드 공개가 필수적이라는 것이다. 그리고 이를 실현하기 위하여 GPL이라는 자유소프트웨어 라이선스를 만들게 된다. 그러나 이를 적용한 GPL 라이선스는 수익을 목적으로 운영되는 많은 기업들로 부터 사용을 꺼리게 하는 궁극적인 요인이 되었으며 그 결과로 자유소프트웨어 운동은 주로 학생들에 의해 주도 되었다. 기업들이 꺼리게 된 가장 큰 원인은 GPL 라이선스의 바이러스적인 속성 때문이다. 즉, 'GPL 라이선스를 따르는 소프트웨어를 개작하거나 또는 그 일부를 사용한 모든 소프트웨어는 GPL 라이선스를 따라야하며 소스도 공개되어야한다'라는 것 이였다. 이는 수익을 목적으로 해야 하는 많은 기업들에게는 당연히 환영받지 못하는 것이었다.

□ 1990년대

1991년도에 핀란드의 한 대학생인 리누스 토발즈가 80386칩에 최적화된 LINUX라는 시스템커널을 개발하였다. 그리고 이소스에 GPL라이선스를 부여하여 GNU프로젝트에 의해 개발될 수 있도록 공개하였다. 이때부터 LINUX커널은 전 세계의 수많은 개발자들과 함께 GNU프로젝트 하에서 이미 개발되었던 많은 프로그램 및 유틸리티들과 결합되었으며 매우 안정적인 운영체제로 놀라울 정도로 빠른 발전을 하게 되었다.[7]

GNU프로젝트의 많은 유틸리티들과 합쳐져, GNU/Linux운영체제가 탄생하게 되었으며 드디어 1996년도에 리눅스 커널 2.0이 발표되면서 Linux운영체제는 완전한 운영체제로 인정을 받게 되었다.

자유소프트웨어 운동 내부에서 이상주의적이고 도덕적인 스톨만의 입장에 대해서 실용주의적 관점에서 비판이 이루어지기 시작하였다. 에릭 레이몬드(Eric Raymond)는 리눅스를 개발하는 방식이 왜 성공적인 결과를 낳았는지를 분석하고 그 결과를 발표하였다.[8]

7) 리누스 토발즈는 운영체제의 일부분인 Linux커널의 개발을 시작한 선도개발자이며, 리누스 토발즈가 개발을 시작한 커널에 수많은 GNU프로젝트와 다른 프로젝트들의 결과물들이 합해져서 결국 완성된 하나의 Linux운영체제가 개발된 것이다.

8) Eric S. Raymond, The Cathedral and The Bazaar(http://www.oreilly.com/catalog/cathbaz/ 참조), 리누스 토발즈가 리눅스를 만들었다는 사실보다 불완전한 소프트웨어를 자주 빠르게 발표하여 수많은 사람들이 버그를 수정하는 '리눅스 개발 모델'을 만든 것에 큰 의미를 부여함

1998년 넷스케이프사가 레이몬드의 논리를 수용하여 '넷스케이프 커뮤티케이터'의 소스코드를 공개한다고 발표하였다. 레이몬드와 그의 주장에 동의하는 사람들은 공개소프트웨어 운동을 관리하기 위해 Open Source Initiative라는 조직을 결성하였다. 자유소프트웨어의 이념적 측면보다는 그것이 개발되는 방식에 초점을 맞추는 실용적 접근을 취하는 이들은 자유소프트웨어는 산업계는 물론이고 개발자에게도 경제적인 보상을 제공할 수 있어야 한다고 주장하였다.

라. 공개SW 올바른 이해

□ 기술지원

도입한 공개소프트웨어에 대한 기술 지원을 받을 수 없을 것이라는 오해는 직접 다운로드 받아 사용한 경우이며 현재는 리눅스 개발기업과 시스템 판매 업체, 일부 하드웨어 공급사가 상시 지원 체계를 갖추고 있으며, 기술지원을 전문적으로 하는 업체도 다수 있음

※ 공공기관에서는 기술지원 및 유지보수의 방법을 반드시 확보해야 함.

□ 안정성

공개소프트웨어는 대규모 기업에서 요구하는 중요한(Mission Critical) 업무에는 안정적이지 않을 것이라는 오해가 있으나 실상은 다른 운영체제보다 안정적임

□ 보안성

공개소프트웨어는 소스코드가 공개되어 있어 보안에 취약할 것이라는 오해가 있으나 보안의 허점은 전 세계에 흩어져 있는 수많은 공개소프트웨어 개발자들과 소스 공유에 의해 다른 운영체제보다 빠르게 수정 보완되고 있음

□ GUI 환경

윈도우에 비해 열악한 GUI를 제공하므로 사용하기가 직관적이지 못하다는 오해가 있으나 X 윈도우와 사용자 환경인 KDE(K desktop Environment), GNOME(GNU Network Object Model Environment)을 사용하여 직관적인 사용자 환경을 제공함

마. 공개SW 정부 정책

(1) 정책 추진 배경 및 사업 연혁

국가 경제에서 소프트웨어 산업의 중요성이 강조되고 있지만 국내 소프트웨어 산업은 수익기반이 취약한데다 국산 제품 경쟁력도 열악한 게 현실이다. 특히 국내 소프트웨어 시장은 특정 기업의 과점이 심각한 편이다. 이에 따라 공개SW를 통해 국내 기업들이 시장 경쟁력을 확보하기 위해 정부에서 공개SW를 도입하는 기관에 예산 가점을 주는 등 공개SW 정책을 적극 수립하고 지원에 나서게 된 것이다.

[표2-7. 공개SW 정부 정책 방향]

구분	내용
공공기관 중심 수요 창출	•전자정부 31대 과제, 공공정보화 사업 중심 수요 유도 •시범사업 및 우수 적용 사례 전파, 적용 우수 기관 가점 부여 •성공사례, 인식개선 등을 위한 마케팅 활동
공개SW 기술지원 불안감 해소	•공개SW 기술지원센터 통한 공공부문 기술지원 체계 구축 •리눅스에서 운영 가능한 서버용 솔루션개발 및 시험 지원
공개SW 기반환경	•구매관행 개선 등 공정경쟁 환경 조성 위한 법제도 개선 •국내외 시장 확대 위한 한중일 협력 및 글로벌 표준화 추진
조성 우수 공개SW 운영인력 양성	•시스템 엔지니어 등 리눅스 기반 정보시스템 운영인력 양성
공개SW 수요 확대	•공개 소프트웨어 전략 프로젝트 - 파급효과와 시장규모가 큰 특수 분야 공개SW 적용 •디지털교과서 등 특정업무의 공개SW기반 고정기능형 PC 도입 촉진
생산기반 강화	•기업, 대학(Lab 등) 등 공개SW 프로젝트 지원 및 공모대전 •공개 소프트웨어 커리큘럼 개발 지원 및 환경 구축
공개SW 국제 협력 및 저변 확대	•동북아 공개 소프트웨어 활성화 포럼 및 국제협력 강화 •공개 소프트웨어 정보 제공

(자료: NIPA(구 KIPA), 2009년)

(2) 부처별 공개SW 추진 일정

정부의 공개 소프트웨어 정책은 2004년부터 정부 시범사업을 통해 본격 추진됐으며 이때부터 2007년까지는 공개 소프트웨어에 대한 인식 전환과 실제 적용 사례를 통해 가능성을 타진하는 데 초점을 맞췄다. 그 결과 정부부처의 리눅스 서버에 대한 신규 도입 사례가 2004년 이후 꾸준히 증가하고 있으며 소프트웨어 시장의 주요 이슈로 부상했다. 그간의 공개 소프트웨어 활성화 지원 사업이 초기시장창출과 인식 개선에 초점이 맞추어졌다면 2008년부터는 공개 소프트웨어 생산기반 요소인 커뮤니티 활성화, 고정기능형 PC분야의 공개 소프트웨어 도입에 보다 주안점을 두고 추진 중이다.

□ 지식경제부

[표2-8. 지식경제부 공개SW 추진 일정]

연도	내용
2004년	4월, 국제 협력(동북아시아 공개 SW 활성화 포럼) 강화 -한중일 3국간 공개SW (NEA 공개SW Promotion Forum 협력) -한국 지식경제부, 중국 공업정보부, 일본 경제 산업성 포럼 회의 -정부, 기업, 대학 등 300여명 참석, 3개의 워킹 그룹으로 구성
2005	리눅스 최초 국내표준 제정 -ETRI(한국전자통신연구원)는 한국형 표준리눅스 플랫폼인 '부요(Booyo)'사업과정에서 리눅스의 표준채택으로 표준을 따르는 리눅스 배포판을 구현할 수 있게 됨
2006년	공개 SW 시범도시 사업 추진 -도시와 협약을 맺고 그 동안 진행한 공개 SW 활성화 사업의 성과를 해당 도시에 적용
2007년	공개SW 공모대전 -공개SW 분야 우수 인력 양성과 저변 확대를 위해 추진
2008년	5월, 공개 SW 기반 디지털교과서 서비스 환경 구축 사업 추진 공개 SW 기반 교육 혁신 프로젝트 지원 로봇용 오픈 플랫폼(uROSE) 공개 국제표준기반 가상화 관리 SW "바인(VINE)" 개발
2009년	2009년 공개SW 공모대전 공개SW 역량프라자 -오픈소스 산업 생태계의 선순환구조 형성을 위한 컨트롤 타워 역할

□ 행정안정부

[표2-9. 행정안전부 공개SW 추진 일정]

연도	내용
~ 2009년	Open API 기반의 "공유 서비스 발굴, 개발 사업" 추진 -공공과 민간에서 활용 수요가 높은 공공 정보 자원을 재사용이 용이한 표준화된 서비스 형태로 공유 -2008년 9월 : Open API 기반의 "나라 기록 통합 검색 서비스" 제공
	전자정부 표준 공통 서비스 및 개발 프레임워크 구축 사업에 오픈소스 도입 -2008년 12월부터 삼성 SDS, LG CNS, SK C&C 및 6개 중소기업이 공동으로 참여·개발 -전자정부 표준 프레임워크 임시 오픈, 2009.6.1 -전자정부 표준 프레임워크 발표, 2009.6.24 -"표준프레임워크 라이선스"의 OSI(Open Source Initiative) 국제인증을 추진 -한국정보화진흥원 내 프레임워크표준화위원회: 산·학·연 전문가로 구성 -2009년 하반기에 국가대표포털 등 5개 사업에 시범적용 -2012년까지 신규 정보화사업의 50% 이상적용

□ 문화체육관광부

[표2-10. 문화체육관광부 공개SW 추진 일정]

연도	내용
~2009년	공개 SW 검증/활용 시스템 -오픈소스 2만 건, 라이선스 70여종, 2009년 5월 1일 서비스 예정 -올해 12월까지 오픈소스SW를 800만 건으로 확대 -이를 기반으로 SW에 오픈소스 라이선스 사용 여부를 검증해 주는 시스템을 개발할 예정 정책 방향 -오픈소스SW 라이선스에 대한 인식제고 및 저작권문화 정착 기반을 마련 -오픈소스SW의 건전한 이용활성화를 도모 -국제 컨퍼런스와 공모전, 온라인 캠페인 등 다양한 대국민 교육 및 홍보 사업 추진 오픈소스SW저작권 인식 제고를 위한 공모전

□ 교육과학기술부

[표2-11. 교육과학기술부 공개SW 추진 일정]

연도	내용
~2009년	2008년 : 5월, 공개 SW 기반 디지털교과서 서비스 환경 구축 사업 추진 -교육 정보화 핵심 정책으로 시작된 "디지털 교과서 사업" -초/중등학교 PC 환경 및 교육 서비스의 획일화를 방지하기 위한 사업 -디지털 교과서 관련 표준 플랫폼, 공개 SW 기반의 운용체계(OS) 등 개발 --> 2013년까지 전국 일선 학교에 디지털 교과서를 적용 공개 SW인 리눅스를 활용한 디지털 교과서 시범학교 2009년도 디지털교과서 기능 고도화 및 추가 개발 시범 사업

□ 국토해양부

[표2-12. 국토해양부 공개SW 추진 일정]

연도	내용
2008년	국토공간계획지원체계(KOPSS) 구축에 오픈소스 적용 가능 -오픈소스를 포함한 개방형 GIS 엔진을 국토공간계획지원체계(KOPSS)에 적용할 수 있는 길 열어 -진입 장벽 제거 및 국산 SW의 점유율 확대와 관련 기술 발전 및 산업 활성화 효과

(자료 : 각 부처 사업 공고(RFP))

(3) 정부 시범사업을 통한 공개 소프트웨어 기반 구축사례

2004년부터 2007년까지의 공개 소프트웨어 활성화지원에서 추진된 사업은 크게 공개 소프트웨어 적용시범사업, 공개 소프트웨어 기술지원센터 설치 및 운영, 공개 소프트웨어 인력 양성, 한중일 공개 소프트웨어 포럼 참여, 공개 소프트웨어 인식 개선을 위한 홍보 사업 등이었다.

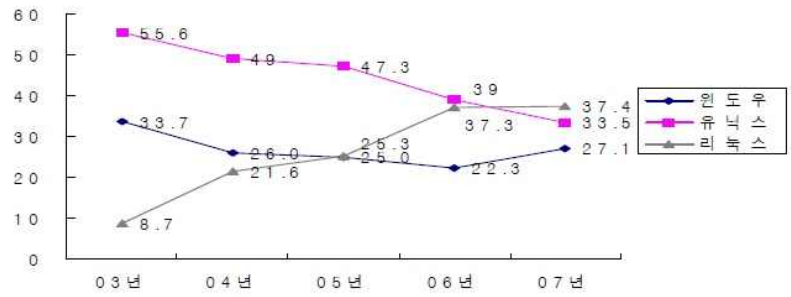
이 가운데 정부가 가장 주안점을 둔 분야는 시범사업으로, 정부는 공공 시장에서 성공사례를 만들어 공개 소프트웨어에 대한 인식 전환을 통해 민간으로의 확산을 기대하였다.

공개 소프트웨어 시범사업으로 우선 정부 부처 및 지방자치단체, 기관 등의 서버 운영체계(OS)를 기존 유닉스나 윈도우즈에서 리눅스로 전환하는 사업을 추진하였고, 이에 따라 공정거래위원회, 중앙인사위원회, 육군교육사, 도로교통안전관리공단 등이 리눅스 서버를 도입했으며, 국방부, 조달청 등은 리눅스용 데스크톱을 사용하기 시작하였다. 이 과정에서 공개 소프트웨어와 관련한 기술 지원의 필요성이 대두되면서 이를 위한 기술지원센터를 설치하는 한편, 관련 인력 양성까지 사업 영역을 넓히게 되었다. 기술지원센터는 주로 수십여 개의 협력사와 기술협력 체계를 구축해 지역 지원까지 가능하도록 하였다.

정보시스템에 공개소프트웨어를 도입한 가장 대표적인 사례가 바로 신교육정보화 시스템(NEIS)에 리눅스를 도입한 것이다. 당시 각 학교 통합정보관리를 위해 NEIS의 입학, 교무, 학사 부문 시스템을 위한 안정적인 시스템을 구축할 필요성이 제기됐으며 제한된 예산으로 시스템의 안정성과 확장성을 동시에 만족시켜야 하는 것이 이슈였다. 정부는 공개 소프트웨어 기반의 운영체제와 애플리케이션을 도입해 하드웨어 비용을 줄여 전체적인 예산 절감을 실현하게 됐다.

리눅스 서버 2,300여대를 도입한 NEIS 이외에 지방자치단체 공통 인프라에도 리눅스 서버가 700여대 도입됐다. NEIS 도입을 계기로 공공부문의 공개 소프트웨어 도입기관수는 크게 증가했다.

그림 3 중앙행정기관 서버 운영체제별 신규 도입률(대수기준)



자료 : 한국소프트웨어진흥원(KIPA), 2008년

[그림2-1. 중앙행정기관 서버 운영체제별 신규 도입률(대수기준)]

(자료: NIPA(구 KIPA), 2008년)

공개 소프트웨어 활성화 지원을 적극 추진한 결과, 시장 규모도 커지는 한편 공개 소프트웨어에 대한 기업의 인식도 변화하기 시작했다. 이에 따라 그동안 공개 소프트웨어 사업에 소극적이었던 IT서비스기업들도 최근에는 공개소프트웨어 전담 부서를 만들어 이 사업을 적극 추진하기 시작했다.

(4) 각국 정부의 공개 소프트웨어 지원 현황

외국 정부도 자국 내 공개 소프트웨어 활성화를 위한 법안을 마련하고 정부 부처의 사용을 독려하는 등 활발하게 지원하고 있다. 이들 정부들은 공개 소프트웨어를 비용 절감 측면으로 접근하려는 경향이 강하며 특정 기업(특히 마이크로소프트)의 소프트웨어를 사용한 것에 비해 공개 소프트웨어를 사용할 경우 비용절감 효과를 조사해 이를 반영하고자 하고 있다. 이러한 경향은 특히 유럽 국가들 사이에서 강하게 나타나며 한국을 비롯한 아시아 국가들은 소프트웨어 경쟁력 확보에 보다 주안점을 두는 것이 특징이다.

[표2-13. 주요 국가의 공개 소프트웨어 지원 현황]

국가	지원현황	비고
미국	• 주정부 및 기관을 중심으로 공개SW 및 관련 보고서·프로그램·시스템 개발에 집중 - 11개 지방 정부가 2004년도 결성한 GOCC(오픈소프트웨어 저장소), 국방부의OTD(개방형기술개발프로그램),	• 텍사스, 캘리포니아, 오리건주 주정부
유럽	• 2002년 6월 'eEurope 2005' 및 이의 후속 i2010에 근거, 개방형표준과 개방형 인터페이스지원을 통해 유럽전체 전자정부간 실현상호호환성 확보에 집중 - IDABC는 '03정보 포탈인 OSO 구축 및 '07 OSORepository 구축을 통해 eGovernment 응용프로그램 공유 지원 위한 플랫폼 구축	• 독일: 연방정부 차원의 공개SW 활성화 정책 자문 프로그램 운영, 지자체 차원의 공개SW 적용 활발 •프랑스: 오픈 오피스 전환 등 개방형 표준 선택 •핀란드: ODF 적용 •노르웨이: eNorway 2009 - digital leap'(국가정보화계획) •벨기에: ODF 적용 법안 통과(2006)
일본	• e-Japan Strategy 기반으로 정책·기술개발·해외시장개척 등 공개SW센터 중심으로 확산	•. 8개 지방정부, 49개 학교 대상 공개SW 현장적용 사업 추진

	적용 사업 추진 •정부조달의 기본지침에서 특정 상품명을 사용치 않고 Open Standard 우선 규정 - 가이드라인 및 SW 특허권 마련 - ODF와 기타 포맷간의 상호호환성 연구	
중국	• 공공분야 중심으로 국산 및 공개SW 시장 확대와 공개SW 테스트 및 인증프로그램을 통한 기술향상, 리눅스 교육 강화를 통한 기술인력 양성 추진 - CSIP 중심으로 공개SW 활성화 정책 추진	• 베이징 SW 테스트 센터 공개SW 평가 및 테스트 추진, 인증 마크 제공 •국가정보기술 인력양성 프로젝트 가동 및 40개 시범 대학 공개SW 정식 과목 선택

(자료: NIPA(구 KIPA))

(5) 시사점 및 향후 정책 방향

그동안 정부, 산업계, 학계가 공개 소프트웨어 개발 및 도입에 노력한 결과 공개 소프트웨어 사용국가에서 생산국가로 도약하고 있다는 점은 고무적인 현상이다. 그러나 오픈소스화한 소프트웨어들이 아직 국내 시장에서만 쓰이고 있어 글로벌한 활용이 미흡한 상황이다. 국내에서 제작된 공개 소프트웨어 제품을 해외 개발자나 기업들에게 알리려면 해외 커뮤니티와 연계한 활동이 활발해야 하며 국내 IT서비스 및 소프트웨어 기업들도 해외 프로젝트에 이를 적용해 사용자 저변확대에 나서야 한다.

공개 소프트웨어가 가진 본연의 취지는 개방과 공유를 통해 기술을 발전시키는 데 있다. 공개 소프트웨어를 내수용으로 사용하게 되면 기술 발전은 더딜 수밖에 없으며 신기술의 추격을 피할 수 없게 된다. 정부는 국내 공개 소프트웨어 커뮤니티를 설립 및 발굴해 지원한 다음에는 이들이 해외 커뮤니티와 연계해 활동할 수 있도록 지원해야 할 것이다.

또한 국제 협력을 위해 탄생한 한중일 공개 소프트웨어 포럼이 아시아 공개 소프트웨어 포럼으로 확대되기 위해서는 주변 국가들의 적극적인 참여를 유도해야 한다. 인도나 동남아시아 여러 국가들과도 협력을 강화하고 이후 이 포럼을 아시아에서 글로벌 포럼으로 확대 운영하겠다는 비전이 필요하다.

바. 공개SW 사업 환경

(1) 공개SW 시장 규모

국내 소프트웨어 시장규모는 63억 달러로 세계 대비 1%수준에 불과하여 국내 소프트웨어산업의 경제적 위상은 그리 높지 않다. 전 세계 소프트웨어시장으로 수출되는 한국 소프트웨어 규모는 7억불 정도로 주요 경쟁국인 아일랜드의 35분의 1, 이스라엘의 5분의 1 수준에 불과하며, 국내 소프트웨어시장도 외국기업이 상당부분을 차지하고 있는 실정이다. 국내 소프트웨어 시장규모는 63억 달러로 세계 대비 1% 수준에 불과하여 국내 소프트웨어산업의 경제적 위상은 그리 높지 않다. 전 세계 소프트웨어시장으로 수출되는 한국 소프트웨어 규모는 7억불 정도로 주요 경쟁국인 아일랜드의 35분의 1, 이스라엘의 5분의 1 수준에 불과하며, 국내 소프트웨어시장도 외국기업이 상당부분을 차지하고 있는 실정이다.

한편 국내 소프트웨어생산액은 2008년 기준으로 약 33.6조원으로 추정되어 국내 총 산업 생산액의 1.1%, IT생산액의 9.8%를 차지하여 다른 국가에 비해 상대적으로 낮은 수준으로 평가 받고 있다.

이제 전 세계 소프트웨어산업은 기존 시장구조에서 공개소프트웨어가 차지하는 비중이 점차 늘어나고 있는 추세이다. 특히 인터넷 인프라나 플렛폼 소프트웨어 개발 분야에서는 공개소프트웨어 비중이 급격히 늘어나고 있다

Gartner의 자료에 따르면 전체 소프트웨어시장에서 공개 소프트웨어시장이 차지하는 비중은 2005년 11%에서 2010년 24%까지 확대될 것으로 전망하고 있다.

이는 단순히 공개소프트웨어의 활용률(OS를 비롯한 애플리케이션 영역까지의 확대)이 늘어난다는 의미를 벗어나 소프트웨어에 대한 접근방법의 변화(참여, 공유, 개방)를 공개 소프트웨어가 주도한다는 것을 의미하며, 결과적으로 소프트웨어의 상품화(commoditization)에 기폭제 역할을 하고 있다.

우리나라도 지난 2003년부터 (舊)정보통신부를 중심으로 공개소프트웨어 활성화 정책을 추진해 왔으며 그 주요 정책은 공개소프트웨어 진입장벽 해소를 위한 법제도 개선, 인식개선 및 수요촉진을 위한시범사업 등 수요촉진정책과 중·고급 전문 인력

양성, 공개소프트웨어 기술지원센터 운영 등 공급촉진 정책이다. 이에 따라 정부주도의 공개소프트웨어 관련 시장은 매년 성장하고 있으나, 민간 기업을 아우르는 자생적인 공개소프트웨어 시장의 활성화는 미진한 편이다.

이제 공개소프트웨어는 세계적으로 각국의 소프트웨어 산업 발전을 위한 하나의 기회요인으로 인식되고 있으며, 공개소프트웨어 선도국들은 관련 기업들에 대한 지속적 투자와 함께, 공개소프트웨어 사용 활성화를 위한 입법과 정책 개발 등 범정부적 차원의 지원을 동시에 제공하고 있다. 특히 공개소프트웨어의 사용을 법으로 의무화하기 위한 입법화 움직임이전 세계적으로 본격화하는 가운데, 몇몇 국가들은 단독입법, 기존 관련법 조항의 일부 개정, 전자정부체제에의 편제 등 다양한 형태에서 그 입법 작업을 완료하는 등 상당한 진척을 보이고 있는 상황이다. 이는 공개소프트웨어가 하나의 유행으로서 끝나는 것이 아니라, 소프트웨어산업의 중요 축으로 자리매김하고 있다는 반증으로 이해할 수 있다.

공개소프트웨어의 산업적 가치공개소프트웨어가 산업적으로 주목받는 것은 다수의 개발자에 의한 고급 프로그래밍 기술의 적용이 가능하다는 점이다.

공개소프트웨어 개발과정은 인터넷의 커뮤니티를 통해서 진행됨으로 인터넷을 통해 프로젝트에 참여할 수 있는 숙련 개발자들을 많이 모을 수 있으며, 이를 통해 고급 프로그래밍 기술을 지닌 많은 사람들의 아이디어를 흡수하고 최신 선도 기술의 적용이 용이하다.

(2) 공개SW 사용현황에 관한 실태조사

□ 공개소프트웨어 도입률 조사결과[9)]

국내 기관 중 62.6%는 현재 공개소프트웨어를 사용하고 있는 것으로 나타났다. 물론 한 종류 이상의 공개소프트웨어를 사용하는 기관은 모두 도입한 것으로 간주했기 때문에 도입률 수치 자체의 해석에 상당한 주의를 요한다.3) 부문별로는 공공부문에서 공개소프트웨어를 사용하는 비중이 86.2%로 55%에 머문 민간부문과 비교하여 큰 차이를 보였다. 이렇게 공공부문에서 높은 도입률을 보이는 것은 정부의 정책과 교육목적에 따른 사용효과로 보인다.

9) 출처 : NIPA(구 KIPA) 2007.12.27, “국내 공개SW 사용현황 및 수요조사보고서”

[표2-14. 부문별 공개소프트웨어 도입률]

부 문	공공부문 (123개 정부/교육기관)	민간부문 (380개 기업)	비고
도입 기업 수	106	209	카이제곱값 = 3,357 p값< 0.001
도입률	86.2%	55.0%	

업종별로는 정부기관(88.5%) 및 교육기관(83.9%)에서 80%가 넘는 도입률을 보임으로서 단연 두드러지고 있다. 정부기관은 정부 정책으로 인해 높은 도입률을 보이고 있으며, 교육기관의 경우 교육 목적으로 사용되는 사례가 많은 것으로 판단된다. 반면 제조(52.0%), 유통/서비스(53.7%), 금융(54.3%)분야는 전체 도입률 62.6%보다 낮은 도입률을 보였는데, 이는 공개소프트웨어가 가지는 장점이 해당 업종에 크게 작용하지 못한 데 기인한 것으로 보인다. 금융권의 경우 보안 및 안정성에서 그다지 신뢰를 받지 못하고 있으며 유통/서비스와 제조업의 경우에는 공개소프트웨어에 대한 선입견이 남아있는 것으로 보인다. 기업 규모별로 보면, 매출액 1,000억 이상에서 41.6%의 도입률을 보이고 있어 41.3%에 머물러있는 1,000억 미만 기업보다 높게 나타나고는 있으나 그 차이는 통계적으로 유의하지 않은 것으로 나타났다. (p값 = 0.61)

[표2-15. 기업 규모별(매출액) 공개소프트웨어 도입률]

규모	1,000억 이상	1,000억 미만	Total	비고
응답기업	180	200	380	카이제곱값 = 0.26 p값 = 0.61
도입기업	101	107	208	
도입률	41.6%	41.3%	41.4%	

※ 주 : 규모별 현황은 민간기업 만을 대상으로 함

공개소프트웨어 도입 방법은 서비스계약을 통해 도입한 사례는 64.4%로 높게 나타났으며, 단순 다운로드 형태는 35.6%로 나타났다. 서비스계약을 통한도입 사례가 더 높게 나타난 것은 공개소프트웨어 도입률이 높은 공공부문에서 서비스 계약률이 높기 때문인 것으로 파악된다. 공공부문과 민간부문의 공개 소프트웨어 도입 방법은 정반대의 결과를 보였다. 정부 정책과 교육 목적으로 공개소프트웨어를 도입하는 공공부문의 경우 서비스 계약을 통한 도입 사례가 82.7%에 비해 제조, 유통/서비스 등 기업들이 포함된 민간부문의 경우 단순 다운로드 사례가 83.9%로 나타났다. 특히 1천 copy 이상 사용하는 (舊)정보통신부 등의 기관에서 해당 소프트웨

어를 서비스계약을 통해 운용하고 있기 때문에 copy별로 산정된 서비스계약 비중은 공공부문이 압도적으로 높게 나타날 수 있다. 이는 공개소프트웨어 시장이 아직 확대되지 않아 민간부문에서는 공개 소프트웨어를 사용한다고 해도 비용 부담이 없는 단순 다운로드를 통한 사례가 많기 때문인 것으로 판단된다.

[표2-16. 부문별 공개소프트웨어 구입/취득 방법]

구분	부 문	공공부문	민간부문	비고
Copy수	다운로드	1,411	2,600	카이제곱값 = 9,565 p값 < 0.001
	서비스계약	6,761	498	
비중	다운로드	17.3%	83.9%	
	서비스계약	82.7%	16.1%	

업종별 공개소프트웨어 도입 방법은 정부기관이서비스계약을 통한 사례가 95.0%인데 반해 통신/닷컴은 다운로드를 통한 사례가 89.2%로 극명한 대조를 이루었다. 제조, 유통/서비스 업종에서는 다운로드 도입 사례가 75% 이상으로 높게 나타났으며 교육기관, 금융업종의 경우에는 다운로드와 서비스계약 도입방법이 50% 내외로 비슷하게 나타났다. 대체로 공개소프트웨어의 도입은 다운로드로 이루어지고 있으며 공공부문이 많은 copy를 서비스계약으로 운용하고 있어 전체적인 서비스 계약률은 다소 과장된 부분이 있다고 할 수 있다. 참고로 교육업종과 금융업종 간의 구입/취득방법 차이와 제조와 통신/닷컴업종 간의 취득방법 차이는 통계적으로 유의하지 않은 것으로 나타났으며 그 외에는 모두 차이가 있는 것으로 나타났다.

[표2-17. 업종별 공개소프트웨어 구입/취득 방법]

구분	업종	정부 기관	교육 기관	금융	유통/ 서비스	제조	통신/ 닷컴	합계
Copy수	다운로드	292	1,119	44	608	1,387	561	4,011
	서비스계약	5,553	1,208	47	197	186	68	7,259
비중	다운로드	5.0%	48.1%	48.4%	75.5%	88.2%	89.2%	35.6%
	서비스계약	95.0%	51.9%	51.6%	24.5%	11.8%	10.8%	64.4%

참고 : 교육업종과 금융업종 간의 구입/취득방법 차이는 카이제곱 = 0.08(p값=0.96)이며, 제조와 통신/닷컴업종 간의 취득방법 차이는 카이제곱 = 1.382(p값=0.5)통계적으로 유의하지 않은 것으로 나타났으며 그 외에는 모두 차이가 있음(p값 < 0.0001)

민간 기업만을 대상으로 하는 기업 규모별 공개소프트웨어 취득/구입 방법에서는 매출 1,000억 원 이상 기업군이 19.2%로 13.8%에 그친 1,000억 원 미만 기업보다 서비스계약 방식의 도입이 좀 더 많이 나타나고 있으나 통계적으로 유의하지는 않는 것으로 나타났다. 결국 민간 기업에서는 서비스계약방식을 통한 도입사례가 20% 미만으로 매우 낮게 나타났다. 공개소프트웨어를 사용하더라도 기업들은 비용부담이 없는 다운로드 형태로 이용하는 것이라 할 수 있다.

[표2-18. 규모별 공개소프트웨어 구입/취득 방법]

구분	규모	1000억 이상	1000억 미만	합계	비고
Copy수	다운로드	252	246	498	카이제곱값 =16.233 p값 =0.0003
	서비스계약	1,062	1,537	2,599	
비중	다운로드	80.8%	86.2%	83.9%	
	서비스계약	19.2%	13.8%	16.1%	

□ 공개소프트웨어 제품별 사용현황

국내에서 사용되고 있는 공개소프트웨어 중 가장 많은 사용을 보이고 있는 제품은 <표 7>에 수록되어있는 바와 같이 Linux인 것으로 나타났다. 전체 기관 중 41.6%의 기관이 Linux를 사용

[표2-19. 공개소프트웨어 종류별 도입률(전체 조사대상 기준)]

공개SW	Linux	My SQL	Open Office	Apache	Tomcat	JB 공개SW	PH, Perl	Eclipse	FireFox	Others	합계
기관 및 기업 수	209	172	21	205	165	7	107	45	26	35	315
도입률	41.6%	34.2%	4.2%	40.8%	32.8%	1.4%	21.3%	8.9%	5.2%	7.0%	62.6%

이는 제조업을 제외하면 대부분의 업종에서 높은 사용률을 보이고 있다. 전체 도입률이 62.6%로 높은 것은 여러 공개소프트웨어 제품 중 하나라도 도입한 기관을 의미하기 때문이다. Linux 다음으로는 웹서버인 Apache인 것으로 나타났다. 전체 응답기업의 40.8%로 Linux와 함께 40%가 넘는 도입률을 보이고 있다. 이는 오랜 역사와 함께 충분한 검증을 거쳐 신뢰성을 가지고 있는 소프트웨어라는 점이 많은 사용을 보이고 있는 이유인 것으로 조사됐다. Apache 다음으로는 MySQL의 도입률이 34.2%로 높게 나타나고 있으며, Tomcat은 32.8%로 뒤를 이었다. PHP/Perl은

21.3%, Eclipse는 8.9%로 나타났다. 참고로 Linux의 경우 공공부문에서 가장 많은 사용을 보이고 있는 것으로 나타났다. 교육기관은 낮은 비용으로 높은 교육 효과를 거두기 위한 목적에서 비롯된 것으로 풀이되며 정부기관의 경우 정부 정책의 효과가 나타나기 시작한 것으로 보인다. Linux는 특히 공공부문과 민간부문의 차이가 뚜렷한 제품으로 금융, 유통·서비스, 제조업은 상대적으로 낮은 도입률을 기록했다. 특징적인 것은 Linux의 경우 제조업이 가장 낮은 도입률을 보였지만, MySQL의 경우 금융업종에서 가장 낮은 17.1%의 도입률을 보여 금융업종에서 사용하는 데이터베이스 SW는 높은 보안성과 안정성을 가지고 있을 뿐만 아니라 전문 업체의 안정적 지원을 받을 수 있는 상용제품의 선호도가 높은 것으로 나타났다.

□ 공개소프트웨어 사용 장애요인

공개소프트웨어 사용을 하지 않는 기업들에 대해사용하지 않는 이유를 조사한 결과 <표 8>에 요약되어 있는 바와 같이 호환성(24.7%)과 신뢰성(19.0%)을 가장 중요한 원인으로 꼽았다. 이는 공개소프트웨어에 대한 편견이 아직 존재하고 있는 것으로 판단되며 이 외에도 해당 영역에서의 적절한 엔지니어가 없다는 의견(12.3%)과 함께 유지보수에 대한 부담이라고 할 수 있는 유지보수업체 부실 (15.3%) 또한 크게 작용하고 있는 것으로 나타났다.

안정성 문제(18.3%)도 적지 않은 비율로 나타났으나, 성능문제는 6.7%로 나타나 다른 이유에 비해 상대적으로 낮은 비중을 보였다. 공공부문의 경우 기존 시스템과의 호환성 및 부조화 문제가 38.9%로 가장 큰 문제로 인식한 것과 비교하여 민간기업의 경우에는 23.8%로 조사되어 상대적인 차이를 보였다. 또, 공개소프트웨어에 대한 민간부문의 문제 인식은 매우 다양하게 나타나고 있으나 공공부문의 경우에는 호환성(부조화) 문제와 신뢰성 문제에 집중되는 경향을 보였다. 업종별로 기관들이 공개소프트웨어를 사용하지 않는 이유를 살펴보면 교육과 통신·닷컴 업종의 경우 타 업종에 비해 호환성이 50%이상으로 높은 비중을 차지하였다. 기존 시스템과의 연동에 대한 기술적 지원이 용이하지 않아 전면적 도입에 장애요소로 지적되고 있다. 또 기존 시스템을 안정적으로 사용하고 있는 상황에서 새로운 소프트웨어인 공개소프트웨어를 도입했을 경우 호환성 문제로 인한 어려움을 걱정하는 것으로 판단된다. 특히 금융 업종은 타 업종에 비해 안정성에서 27.0%를 나타내어 큰 비중을 차지한 것이 특징이다.

[표2-20. 공개소프트웨어 미사용 이유]

업종		정부 기관	교육 기관	금융	유통· 서비스	제조	통신·닷컴	합계
성능	기관수	0	0	3	4	12	1	20
	비중	0.0%	0.0%	8.1%	5.3%	8.1%	4.5%	6.7%
신뢰성부족	기관수	3	1	8	12	31	2	57
	비중	33.3%	11.1%	21.6%	16.0%	20.9%	9.1%	19.0%
안정성 문제	기관수	1	1	10	14	27	2	55
	비중	11.1%	11.1%	27.0%	18.7%	18.2%	9.1%	18.3%
엔지니어 부재	기관수	1	1	3	12	17	3	37
	비중	11.1%	11.1%	8.1%	16.0%	11.5%	13.6%	12.3%
유지보수 업체부실	기관수	2	1	7	12	22	2	46
	비중	22.2%	11.1%	18.9%	16.0%	14.9%	9.1%	15.3%
부조화 (호환성)	기관수	2	5	6	18	32	11	74
	비중	22.2%	55.6%	16.2%	24.0%	21.6%	50.0%	24.7%
기타	기관수	0	0	0	3	7	1	11
	비중	0.0%	0.0%	0.0%	4.0%	4.7%	4.5%	3.7%
합계		9	9	37	75	148	22	300

공개소프트웨어를 사용하다가 철회한 이유로는 <표9>에 요약되어 있는 바와 같이 호환성(25.0%)이 가장 높은 비중을 차지하였다. 이는 공개소프트 미사용 이유의 경우와 마찬가지이다. 다음으로는 안정성 문제/엔지니어 부재(18.3%), 신뢰성 부족/유지보수 업체 부실(15.0%) 순으로 나타났다. 공개소프트웨어 중단 이유와 미사용 이유는 대체로 비슷한 응답 결과를 보였다. 차이점은 실제 기관들이 공개소프트웨어를 사용하지 않는 이유 중 신뢰성부족이 큰 비중을 차지했지만 실제 사용을 중단한 이유는 신뢰성 부족보다는 엔지니어 부재가 더 큰 요인으로 꼽혔다. 이러한 결과를 공공부문과 민간부문으로 구분하여 살펴보면, 부문 간 차이는 나타나지 않았으며, 정부기관은 중단한 사례가 없고 교육기관의 경우 10개에 불과하여 의미 있는 통계결과는 나타나지 않았다.

향후 공개소프트웨어 사용을 하지 않겠다는 기관들은 지금도 사용하지 않는 경우가 많았으며, 사용 중인 기관들 중 상당수가 앞으로도 계속 사용하거나 또는 활용 영역을 확대하겠다는 응답을 나타내고 있다. 지금은 사용하지 않지만 향후 도입하겠다는 의사를 표시한 기관은 약 10% 수준에 불과해 향후 공개 소프트웨어에 대한 관심이 여전히 답보상태에 머물러 있는 것으로 조사됐다. 민간부문은 절반에 가까운 48.4%가 사용계획이 없다고 응답하여 14.8%에 불과한 공공부문의 응답과 비교할 때 관심 수준이 상대 적으로 낮은 것으로 조사됐다.

[표2-21. 공개소프트웨어 사용 중단 이유]

업종		정부 기관	교육 기관	금융	유통, 서비스	제조	통신,닷컴	합계
성능	기관수	0	1	1	0	1	0	3
	비중	0.0%	10.0%	14.3%	0.0%	3.8%	0.0%	5.0%
신뢰성부족	기관수	0	2	2	2	3	0	9
	비중	0.0%	20.0%	28.6%	11.8%	11.5%	0.0%	15.0%
안정성	기관수		02	2	2	5	0	11
문제	비중	0.0%	20.0%	28.6%	11.8%	19.2%	0.0%	18.3%
엔지니어	기관수	0	2	1	4	4	0	11
부재	비중	0.0%	20.0%	14.3%	23.5%	15.4%	0.0%	18.3%
유지보수	기관수	0	2	1	2	4	0	9
업체부실	비중	0.0%	20.0%	14.3%	11.8%	15.4%	0.0%	15.0%
부조화	기관수	0	1	0	5	9	0	15
(호환성)	비중	0.0%	10.0%	0.0%	29.4%	34.6%	0.0%	25.0%
기타	기관수	0	0	0	2	0	0	2
	비중	0.0%	0.0%	0.0%	11.8%	0.0%	0.0%	3.3%
합계		0	10	7	17	26	0	60

□ 결 론

공개소프트웨어의 도입률은 예상보다 높은 수준인 반면, 공개소프트웨어에 투자되는 비용은 여전히 낮은 수준을 보였다. 이는 여전히 공개소프트웨어가 언제든지 다운로드 받아 사용할 수 있는 무료 소프트웨어라는 인식이 존재하기 때문이다.

그렇지만 공개소프트웨어가 소위 'public domain'상의 무료 소프트웨어, 즉 개발자가 소프트웨어의 사용에 대해 아무런 주장을 하지 않는 소프트웨어와는 구별되어야 한다. 공개소프트웨어가 마치 이런 'public domain'상의 무료 소프트웨어와 유사할 것이라는 인식은 도입 여부와 상관없이 소프트웨어의 신뢰성 측면에서는 낮은 평가를 받고 있으며, 기존 시스템과의 호환성 또한 문제가 있을 것이라는 일반적 편견이 함께 존재하여 기관 IT영역의 핵심 분야에까지 파고들지 못하는 이유가 되고 있다.

또, 공개소프트웨어를 활용할 경우 공개SW의 장점(적은 초기개발비용, 커뮤니티방식에 따른 빠르고 유연한 개발, SW간에 상호연동성을 보장하는 오픈포맷과 프로토콜, 강력한 네트워킹 기능지원) 등의 장점이 있음에도 불구하고 여러 가지 단점(애

플리케이션의 부족, 빈약하고 체계적이지 못한 문서, 불확실한 개발로드맵)이 함께 존재하여 단순 도입률 향상이 아닌 질적인 향상의 걸림돌이 되고 있어 이에 대한 올바른 내용을 홍보하는 가이드가 정책적으로 필요하다.

공개소프트웨어의 보급도 중요하지만 기관의 정보화환경에 실질적인 효과를 보여주는 서비스모델이 함께 제공되어 고객의 인식을 전환시키는 노력이 필요할 것으로 판단된다.

특히 공개소프트웨어가 시장점유율을 혁신적으로 높이기 위해서는 기존의 독점형 소프트웨어(proprietary software) 사용자 집단보다는 공개소프트웨어 사용자가 전문적인기술을 보유하고 있어 니즈에 따라 적절히 보완할 수 있는 능력이 있어야 한다(Lin, 2008: 68-81). 그렇지 못한 경우 유지보수의 어려움과 비용에 대한 우려로 말미암아 도입을 꺼릴 수밖에 없다.

또 하나 공개소프트웨어의 가장 큰 장점이면서 단점으로 지적되고 있는 것은 다양한 소스코드에 기반 한 많은 제품의 범람이라 할 수 있다. 이미 국내에서도 여러 전문 벤더에서 여러 가지 버전의 Linux를 배포하여 일반 유저들은 선택의 즐거움보다는 혼란스러움을 겪어야 했다. 또 관련 서비스업체 또한 많은 인증 시스템과 표준화되지 않은 서비스 영역으로 인해 비용과 기술에 피곤함을 느끼고 사업을 추진할 여력을 잃어버리기도 했다. 공개소프트웨어의 특성상 다양한 개발자의 참여가 필수적이지만, 공개소프트웨어 커뮤니티의 덕목인 소스의 공유뿐만 아니라 개발자 상호간의 지원과 협력 등이 수반되지 않으면(Henkel, 2006: 960-967; Stewart & Gosain, 2006: 306-310; 디지털타임스, 2006), 비슷한 수준의 유사 소프트웨어의 양산만 초래할 가능성이 높다.

일관성 있는 정책과 표준화를 통한 신뢰성이 뒷받침 된다면 국내 공개소프트웨어 활성화는 상당히 진전될 것으로 예상된다.

□ 활용 참고 자료

○ 국내 공공부분 공개소프트웨어 도입 성공 사례

[표2-22. 국내 공공부분 공개SW 도입 성공사례]

기상청	문제	·대용량 기상정보를 예보관들이 손쉽게 볼 수 없음 ·이를 위한 워크스테이션 장비 비용 부담 ·단일 서버 구조로 안전성 강화 요구 증대
	해결	·리눅스 기반의 업무 애플리케이션 개발 ·워크스테이션 대신 PC에 리눅스 도입 ·장비 비용 및 운영체계 비용 절감
	계획	·리눅스 클러스터 기술이 따라 준다면 리눅스 슈퍼컴퓨터 도입
대한송유관공사	문제	·유닉스 기반 스카다 시스템의 비용 문제 대두 ·관련 장비 노후화로 대비 필요성 ·효율적인 복지 관리 시스템 필요
	해결	·리눅스 기반 스카다 시스템 개발 ·선택적 복지 시스템 도입
	계획	·리눅스 기반 스카다 시스템 테스트 후 현장 적용 ·복지 시스템 외부 기관 아웃소싱
도로교통 안전관리공단	문제	·교통안전 분담금 환급 시스템의 접속자 폭주로 인한 웹서버다운 및 접속 지연 ·기존 운영체제인 윈도우NT의 클라이언트 라이선스 추가 시 비용부담
	해결	·분담금 환급 시스템 서버 운영 플랫폼을 리눅스로 전환 ·향후 2007년부터 웹기반 애플리케이션 및 서버 시스템에 리눅스 운영 플랫폼 전사 도입
한국수출입은행	문제	·비용절감 압박 윈도우 시스템의 보안에 대한 불안 가중 ·기업 신용도 평가를 위한 무중단 환경 구현 필요
	해결	·공개SW인 리눅스 도입으로 윈도우 라이선스 보다 비용 절감 ·유닉스와 비슷한 환경으로 적응용이 ·보안 및 성능 탁월 ·리눅스 클러스터링으로 시스템 이중화해 무중단 환경 구축
	계획	·재해복구 시스템에 리눅스 도입
철도청	문제	·유닉스 플랫폼의 장비 비용 증가 ·유지보수 비용 증가 ·폐쇄적인 플랫폼으로 호환성 결여
	해결	·x86 기반의 리눅스 플랫폼 도입

		·비용 45~50% 절감 ·공개SW 기반인 리눅스로 호환성 향상
	계획	·향후 벤더 차원의 서비스 지원 고려
한국과학기술정보연구원	문제	·고가 장비의 장기간 사용으로 처리 성능 저하 ·업데이트, 구매주기에 벤더 종속화 심화 ·기관 통합으로 유닉스 장비증가하면서 관리 포인트 증가
	해결	·리눅스 기반 x86 서버 도입으로 비용 1/10로 절감 ·빠른 장비 교체 가능 ·최신 기술 반영 가능 ·벤더 독립적인 의사결정 가능
	계획	·시스템 교체 주기에 맞춰 유닉스 장비 리눅스 마이그레이션
한국생명공학연구원	문제	·기관 내 지식관리시스템의 유동성 확보 어려움 ·내부 고객 요구 반영 어려움 ·윈도우 라이선스 비용 증가
	해결	·리눅스 기반으로 지식 관리 시스템 마이그레이션 ·라이선스 비용 절감 ·공개된 소스 코드로 내부 고객 요구 반영용이 ·애플리케이션 유동성 확보용이
	계획	·리눅스 전문 벤더 통한 체계적인 업데이트와 서비스 지원 검토
행정자치부	문제	·시군구행정정보화 시스템완성 후 웹으로 공개하기 위한 시스템 구축 필요
	해결	·15개 광역지자체와 209개 시군구 지자체에 인터넷 민원 공개 서버 구축 ·리눅스 시스템을 통해 예산 절감
	계획	시군구행정종합정보화 시스템 고도화 작업을 위한 진단 작업 중 ·대용량 데이터 처리가 가능하다면 리눅스 도입 적극 검토
중소기업진흥공단	문제	·중소기업을 대상으로 한 홈페이지와 메일 서비스 제공 중 신청 업체들의 증가로 인한 예산 문제 봉착
	해결	·PC 서버에 리눅스 운영체제를 탑재해 서비스 제공 ·독자적인 리눅스 기반 메일 솔루션 개발
	계획	·리눅스 OS 서비스료 지불에 따른 예산 증액 검토
한국전산원	문제	·정부 기관의 지식정보를 연계해 메타 검색 사이트로서 역할하기 위해 중립성 있는 플랫폼 요구

		·저비용 고효율적인 시스템을 구축해 원하는 정보를 최대한 빨리 검색할 수 있는 성능 필요
	해결	·리눅스 기반으로 플랫폼을 구성해 구축비용을 유닉스의 30%수준으로 낮춤 ·전문 관리 아웃소싱 업체에게 위탁 운영해 유지보수 관리 및 문제 발생 시 해결
	계획	·민간 포탈 사업자를 통한 정보 검색 기능 추가해 일반인들의 활용도 향상 ·법제처 등 지식 정보 연계 기관 확대 ·메타 데이터 검색 스키마 강화

(자료: NIPA(구 KIPA))

○ 국내 민간부분 공개소프트웨어 도입 성공사례

[표2-23. 국내 민간부분 공개SW 도입 성공사례]

그리곤 엔터테인먼트	문제	·경쟁심화로 수익구조 악화와 이로 인한 비용 절감 압박 ·윈도우 라이선스 비용 증가 ·예기치 않은 서버 다운에 대한 대비 미비
	해결	·리눅스 기반으로 게임 서버 구축 ·라이선스 비용 50% 절감 ·서버 다운에도 최종 기록 보존이 가능해 안정성 배가
	계획	·향후 모든 신규 개발 게임에 리눅스 도입
대신증권	문제	·경기 침체로 인한 투자비용 회수 압박 ·HTS 시스템의 경쟁력 향상위한 안정성 확보 ·독점 플랫폼에 대한 해외 시장의 반감
	해결	·리눅스 플랫폼 도입으로 20% IT 비용 절감 ·벤더 독립적인 의사결정 ·리눅스 기반 HTS 개발로 대만 증권사에 수출
	계획	·향후 정보계 시스템의 점진적인 마이그레이션 테스트 계획
대한항공	문제	·클라이언트 서버 환경을 웹으로 전환 ·메인프레임 도입에 따른 유휴 자원 활용
	해결	·리눅스/390 플랫폼에 수입관리 ·승무원 스케줄 관리

		·임직원 정보, e-티케팅 ·국내선 수입 관리 등 20여개의 애플리케이션 개발 활용
	계획	·리눅스/390의 온 디맨드와 서버 온디맨드 사용 기준에 따른 애플리케이션을 관리 일원화
뱅크타운	문제	·ASP 서비스의 가격 경쟁력 개선 ·유닉스 플랫폼에서 서버 클러스터링 구성의 어려움
	해결	·유닉스 플랫폼 대비 구축비용을 1/10로 절감 ·클러스터링 구성으로 시스템 가용성 강화 및 장애 포인트 분산 ·은행별 독립 서버를 할당해 은행 고객에게 운용 ·관리 및 구조적 독립성 확보
	계획	·시장 추이 따라 리눅스 확대 적용 검토
아남전자	문제	·전자 결재, 게시판 등 웹 메일의 한계 극복을 위한 새로운 시스템 요구 ·그룹웨어 도입 후 직원들의 활용도가 높아지면서 하드웨어 증설 필요성이 생겼으나 하드웨어 추가 구매 비용에 대한 부담
	해결	·사내 그룹웨어를 리눅스 기반으로 구축
	계획	·스팸 메일 첨부, 바이러스 메일 등을 필터링하는 소프트웨어 검토 중 ·리눅스 환경 확대 고려중 ·ERP 오픈 예정
포스데이타	문제	·사내 유닉스 시스템의 장비 및 유지보수 비용 증가 ·원천 기술 개발로 경쟁력 확보 시급 ·비용 절감과 성능 배가를 동시에 고민하는 고객에게 새로운 솔루션 공급 필요성 대두
	해결	·리눅스 클러스터로 구성된 솔루션 자체 개발 ·사내 업무에 적용해 비용절감, 안정성, 성능 확보 ·병렬 처리 방식으로 자원 최적화 가능 ·특화된 솔루션 개발로 수익성 향상
	계획	·HPC 아닌 엔터프라이즈 시장에도 리눅스 클러스터 기술 공급
LG기공	문제	·고객사에 대한 서비스 제공에 있어 LG기공의 호스팅 서비스를 받는 고객사들이 리눅스 도입 요구 ·기존 시스템에 대한 효율화 요구 증대
	해결	·리눅스 기반 시스템 구축으로 비용 절감 효과 ·서비스 특성에 맞는 리눅스 도입
	계획	·향후 다른 영역으로의 리눅스 확산 고려
SK C&C	문제	·ERP 구축 시 기간 시스템 구축으로 상당한 자금 소요

		·특정 벤더 종속적인 운영체제 사용으로 사내 유휴 자원이 늘고 활용 방법이 없음 ·시스템 유연성이 떨어짐
	해결	·리눅스 도입으로 라이선스 비용 절감 ·장비 선택 자유로움 ·기존 유휴 자원 활용 가능 ·공개SW 특성으로 시스템 커스터마이징 용이
	계획	·리눅스 역량 강화 로드맵에 따라 전문 인력 양성 ·적용 기간 시스템 확대
엠게임	문제	·신속하고 유연한 게임 개발 환경 필요성 대두 ·운영체제 라이선스료가 고정 비용이 됨 ·한 장비 당 1000명 정도밖에 처리 못함
	해결	·리눅스 도입 ·오픈 소스 환경으로 개발 및 관리 편리
	계획	·신규 개발되는 게임 서버를 모두 리눅스 환경으로 구축
DMI	문제	·기업 합병으로 인한 시스템 효율성 대두 ·기존 시스템의 비용 문제 ·기존 시스템의 활용 극대화 요구
	해결	·리눅스 도입으로 비용절감 ·자체적인 지침서를 통한 안정성 확보 주력
	계획	·다양한 분야에 대한 리눅스 도입 예정
NHN	문제	·신규 서비스 제공에 따른 하드웨어 가격 폭증과 신속한 시장대응을 위한 애플리케이션 개발 문제 대두
	해결	·오픈소스 소프트웨어 적극 활용 ·리눅스 운영체제와 다양한 오픈 소스 소프트웨어를 통한 자체 개발
	계획	·리눅스 기반 게임 환경에 대한 검토
다음 커뮤니케이션즈	문제	·고객 수 증가에 따른 관리 비용의 부담 ·고객 변화에 따른 서비스 모색
	해결	·리눅스 활용으로 비용 절감 ·자체적인 시스템 개발 가능 ·서치자키 서비스로 검색영역 확보
	계획	·국제적인 시스템 구축 노력 ·검색부분 강화로 고객 관리
유리온	문제	·펀케이크닷컴 사이트 오픈과 암호화된 대량의 음원 콘텐츠에 대한 안정적인 서비스를 시행할 수 있는 시스템 구성 필요

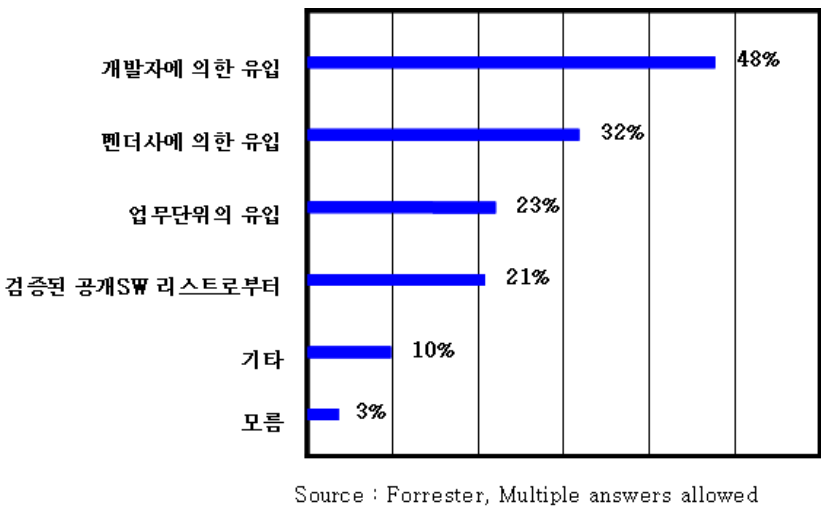
	해결	·안정적인 스트리밍 서비스를 위해 음원 콘텐츠를 서비스 요청 빈도에 따라 3단계로 나눠 시스템 구성 ·암호화된 스트리밍 서비스를 위해 리눅스 기반의 스트리밍 서버 자체 개발 ·백업 정책 차원에서 스토리지 이중화
	계획	·음원 콘텐츠와 가입 고객, 동시 접속자 증가 추세에 따라 서버 구성 변경과 서버/스토리지 증설 가능
숙명여자대학교	문제	·커뮤니티 서비스의 사용자가 늘어나면서 시스템 확장이 필요해졌으며 기존 서버에서는 DB와 애플리케이션을 함께 운영하고 있었으나 이를 분리해 더욱 강력한 성능의 DB 서버 필요 ·비용 대비 효율성과 시스템 안정성, 신속한 유지 보수까지 모두 요구
	해결	·세계 처음으로 리눅스 기반의 64비트체계의 클러스터 시스템 구현
	계획	·향후 사용자 및 데이터의 폭발적인 증가에 완벽하게 대비할 수 있는 체계 마련

(자료: NIPA(구 KIPA))

3. 공개SW 영역

가. 공개SW Governance 구축 개요

기관이나 조직 등에서 소프트웨어를 사용하고자 하는 경우 일반적으로 구매팀을 통해 특정벤더와 계약을 체결하고, 동 소프트웨어의 라이선스 조건을 준수하면서 사용하게 된다. 그리고 대부분의 경우 기관이나 조직 등은 소프트웨어의 구매와 사용에 관한 일정한 프로세스를 가지고 있다. 그런데 이러한 일반적인 소프트웨어의 관리프로세스가 있음에도 불구하고, 공개SW에 대한 별도의 관리전략이 필요하다. 그 이유는 공개SW가 일반적인 소프트웨어와는 다른 특징들을 가지고 있기 때문이다. 예를 들어, 현재 기관이나 조직에서 사용하고 있는 대다수의 공개SW가 구매팀을 통하지 않고 다른 경로를 통해 기관이나 조직으로 들어오고 있다는 점이다. 많은 경우 직원 등이 직접 인터넷을 통해 가져오고 있으며, 또는 하청업체 등이 공개SW를 사용함으로써 기관이나 조직으로 유입되는 경우도 많다. 그 결과 기관이나 조직의 경영층이나 관리자들이 기관이나 조직 내에서 사용되고 있는 공개SW를 파악하고 있지 못하는 경우가 많다.



[그림3-1. 공개SW 유입 경로]

이 밖에 공개SW가 일반적인 소프트웨어와 다른 점은, 일반적으로 상용소프트웨어로부터 기대할 수 있는 특성들을 가지고 있지 않다는 점이다. 예를 들어, 품질에 대한 보증, 면책의 범위 등에 있어서 차이가 있다. 그리고 공개SW는 특별한 권리와 의무를 가진 공개SW 라이선스에 의해 배포 되고 있다는 점을 지적할 수 있다.

그렇다면, 조직이나 기관들에서는 이와 같이 특별한 취급을 요구하고 있는 공개SW를 어떻게 다룰 것인가? 이에 대해서는 조직이나 기관이 처한 환경, 즉 비즈니스모델, 기술 수준 등에 차이가 있기 때문에 일률적으로 말하기 어렵다. 다만, 각각의 내용들을 추상화시켜 일반적인 내용을 제시하고, 조직이나 기관들로 하여금 조직이나 기관이 처한 환경에 따라 구체화하여 사용할 수 있도록 하는 방안을 고려하여 이하와 같이 공개SW Governance를 개발 한다.

공개SW Governance는 크게 공개SW 영역, 정책 및 가이드 영역, 운영 영역으로 구분이 되는데, 공개SW 영역에서는 공개SW의 3대 요소를 라이선스, 커뮤니티, 방법론으로 정의하고 이해 대한 상세한 정보를 제공한다.

정책 및 가이드 영역에서는 공개SW Governance의 현황을 진단하고, 가이드 기본 원칙이 되는 정책을 수립, 사례연구 등을 통해 공개SW를 안전하고 체계적으로 도입할 수 있도록 한다.

운영 영역에서는 공개SW Governance의 프로세스, 컴플라이언스 팀, 리뷰보드, 공개SW 도입을 위한 전체 프로세스 등을 정립하여 공개SW Governance를 효과적으로 활용할 수 있도록 한다.

□ 공개SW 전략의 수립

공개SW를 이용하여 제품을 개발하고 이를 제3자에게 판매하는 기업의 입장에서는 개별기업이 속한 시장의 상황, 해당 시장에서 공개SW의 기술수준, 경쟁기업의 동향, 기업의 기술수준과 R&D 전략, 공개SW 라이선스, 비즈니스모델 등 다양한 요소들을 고려하여 기업의 전략을 마련할 필요가 있다. 예를 들자면, 공개SW를 활용하여 기업이 달성하고자 하는 목표는 무엇인가? 단순한 비용절감인가? 아니면 관련 업계의 표준을 따라가기 위함인가? 이 밖에 보완상품의 판매, 기존제품의 end-of-life를 위한 전략, 시장에서의 기업 이미지 제고, 공개SW 유지관리 및 개발 비용의 분담 등을 목표로 설정할 수 있다.

공개SW 관련 전략을 수립하는데 무엇보다도 중요한 것은 기업의 비즈니스모델과 공개SW를 결합하는 것이다. 공개SW의 경우 GPL 등의 공개SW 라이선스가 요구하는 다른 비즈니스모델을 요구하고 있다. 그 결과 공개SW를 활용한 비즈니스 모델

은 상당히 다양하게 발전해 오고 있다. 이제까지 다양한 기업들이 다양한 비즈니스 모델로 공개SW와 관련된 사업을 수행해 왔다.

비즈니스모델을 포함한 전략을 수립할 때, 공개SW의 장·단점을 다시 한 번 명확히 이해할 필요가 있다. 일반적으로 공개SW는 낮은 진입비용, 빠르고 유연한 개발, 오픈포맷과 프로토콜에 의한 상호연동성 보장, 신뢰성과 안정성 등의 장점이 있는 반면, 애플리케이션의 부족, 빈약한문서, 불확실한 개발로드맵 등의 단점이 지적되고 있다.

나. 공개SW 3가지 영역

(1) 라이선스

□ 소프트웨어의 지적재산권과 라이선스 개요

현재 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, SW 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, 판매하는 것을 막을 수 있다. 특허는 무엇인가 유용한 것을 하도록 하는 방식(method)이므로 소프트웨어의 경우 특허 받은 방식을 구현하는 소프트웨어라면 프로그래밍 언어가 다르거나 소스코드가 다르더라도 해당 특허권자의 명시적인 허가를 받아야 하며 이는 오픈소스소프트웨어, 독점소프트웨어에 공통으로 해당된다.

○ 영업비밀 : 영업비밀이란 공연히 알려져 있지 아니하고 독립된 경제적 가치를 지니는 것으로서 상당한 노력에 의하여 비밀로 유지되는 생산방법, 판매방법, 기타 영업활동에 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. 하

지만 다양한 필요에 의해 이들 권리자가 다른 사람에게 일정한 내용을 조건으로 하여 특정 행위를 할 수 있는 권한을 부여할 필요가 있는데, 이와 같은 권한을 보통 '라이선스(license, 사용허가권)'라고 한다. 이러한 의미에서 라이선스는 물건을 판매하는 매매와는 차이가 있으며, 소프트웨어에 대한 지적재산권은 여전히 원래의 권리자에게 남아있고 일부 사용에 대한 권리만을 부여하는 것이다. 마이크로소프트, 오라클 등 일반적인 독점(proprietary)소프트웨어 업체의 라이선스는 고객이 소프트웨어 권리자에게 대금을 지급하고 소프트웨어의 '사용' 권한만을 허락하는 것이 일반적이다. 따라서 허락을 얻지 않고 소프트웨어를 복제, 배포, 수정하는 행위는 라이선스를 위반함과 동시에 불법에 해당한다.

□ 오픈소스 라이선스의 특징

오픈소스소프트웨어 역시 독점소프트웨어(proprietary software)와 동일하게 저작권 등에 의한 법적 보호를 받고 있으며, 이와 같은 권리에 기반하여 이용자에게 라이선스를 부여한다. 그러나 오픈소스라이선스는 일반적인 독점소프트웨어 라이선스와는 많은 점에서 차이가 있다. 기본적으로 오픈소스 라이선스는 다음과 같이 사용자의 자유로운 사용, 복제, 배포, 수정을 보장하고 있다.

- 라이선시는 해당 오픈소스소프트웨어를 자유롭게 사용할 수 있다.
- 라이선시는 해당 오픈소스소프트웨어를 자유롭게 복제할 수 있으며, 일정한 조건하에 재배포할 수 있다.
- 라이선시는 해당 오픈소스소프트웨어를 자유롭게 수정하여 사용할 수 있으며, 일정한 조건하에 수정된 내용을 재배포할 수 있다.
- 라이선시는 해당 오픈소스소프트웨어의 소스코드를 자유롭게 획득하고 접근할 수 있다.

오픈소스 라이선스는 소프트웨어의 사용, 복제, 배포, 수정의 자유를 부여함과 아울러 다른 한편으로는 소프트웨어의 사용자에게 일정한 의무를 부과하고 있다. 구체적인 내용은 오픈소스소프트웨어와 함께 배포되는 라이선스의 내용을 통해 알 수 있다. 해당 오픈소스소프트웨어에 대한 라이선스는 주로 소스코드 내부나 홈페이지 등에 명시되어 있다. 소스코드에서는 주로 최상위 디렉터리에 COPYING이라는 독립된 파일에 라이선스 조항을 기록하기도 하며, 각각의 소스코드 파일 상단에 명시해 두기도 한다.

오픈소스 라이선스에서 요구하고 있는 준수사항을 라이선시가 이행하지 않으면 권

리자로부터 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) 해야 하는 경우도 발생할 수 있으므로 라이선스의 의무사항을 명확히 이해하여 이와 같은 상황을 사전에 예방하는 것이 필수적이다. 그러나 이러한 위험 때문에 오픈소스 소프트웨어를 전혀 사용하지 않겠다는 결론을 내릴 필요는 없다. 독점소프트웨어 라이선스에서 규정하고 있는 의무사항에 비하면 오픈소스 라이선스가 요구하고 있는 내용이 결코 어려운 것이 아니므로, 오히려 이를 잘 이해하고 준수함으로써 오픈소스 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 라이선스의 요구 사항을 반드시 확인하여야 한다. 또한, 자체 판단이 불가능할 경우에는 외부 전문가에게 조언을 의뢰하여 개발 시작 전 해당 라이선스의 요구 사항과 오픈 소스 사용 목적을 확실히 분석하여야 한다. 이렇게 하는 것만으로도 충분히 올바르게 오픈 소스를 최대한 활용할 수 있으며, 나중에 발생할 수 있는 문제들을 사전에 차단할 수 있다.

□ 오픈소스 라이선스의 구체적 내용

공통적 준수사항

오픈소스 라이선스의 의무사항은 각각의 라이선스마다 조금씩 차이가 있지만 크게 나누어 보면 공통적으로 '저작권 관련 문구 유지', '제품명 중복 방지', '서로 다른 라이선스의 소프트웨어 조합 시 조합 가능 여부 확인' 등이 있고 선택적으로는 '소스코드 공개', '특허관련 사항 준수' 등이 있다.

아래는 모든 오픈소스소프트웨어에 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 오픈소스 소프트웨어들의 경우 거의 대부분 소스코드 상단에 개발자 정보와 연락처 등이 기록되어 있는데 만약 이러한 개발자 정보를 임의로 수정하

거나 삭제하여서는 안 된다. 특히 GPL등 수정된 결과물을 다시 공개하도록 규정하고 있는 '상호주의(reciprocal)' 라이선스들의 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 오픈소스 소프트웨어와 동일한 이름을 제품명이나 서비스명으로 사용하여서는 아니 된다. 특히 유명한 오픈소스 소프트웨어들일수록 해당 오픈소스 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 라이선스가 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 가리켜 라이선스의 양립성(Compatibility) 문제라고 한다. 따라서 어떤 오픈소스 소프트웨어에 다른 오픈소스 소프트웨어를 섞을 경우 반드시 두개의 라이선스가 서로 호환되는지를 확인하여야 한다. 양립성문제는 자유/오픈소스소프트웨어 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) 라이선스로 배포하는 라이선스 정책을 채택하여, 라이선스의 양립성과 관련된 불확실성을 제거하고 모질라 코드를 GPL 또는 LGPL 기반의 응용프로그램에 사용할 수 있도록 하였다. Trolltech도 Qt 라이브러리에 대한 오픈소스소프트웨어라이선스인 QPL과 GPL의 양립성 문제를 해결하기 위하여 QPL 및 상용라이선스 이외에 GPL을 추가하는 정책을 취하고 있다. 한편 최근 개정된 GPL 3.0은 Apache License 2.0과 양립가능하다.

아래는 라이선스에 따라 다르다. 어떤 라이선스의 경우는 아래 세 가지 사항 모두에 관계되는 경우도 있고, 어떤 라이선스는 아래 중 일부만을 요구하는 경우도 있다. 자세한 사항은 라이선스별 해설 부분을 참고하기 바란다.

○ 사용 여부 명시 : 많은 오픈소스 라이선스들은 소스코드를 자유롭게 열람하고 수정 및 재배포할 수 있는 권리를 부여하는 한편, 소프트웨어를 사용할 때 해당 오픈소스소프트웨어가 사용되었음을 명시적으로 표기하는 것을 의무사항으로 채택하고 있다. 이것은 마치 논문을 쓸 때 인용을 하는 것과 비슷하여, '이 소프트웨어는 오픈소스 소프트웨어인 무엇 무엇을 사용하였습니다.'라는 식으로 사용 여부를 명확히 기술하라는 것이다. 사용자 매뉴얼이나 기타 매뉴얼을 대체하는 매체가 있다면 그곳에 기술하면 된다.

○ 소스코드 공개 : 오픈소스 라이선스에 따라서는 수정하거나 추가한 부분이 있을 때 해당 부분의 소스코드도 공개하여야 한다고 명시하는 경우가 있다. 이에 해당하는 라이선스는 GPL이 가장 유명하다. 그러나 정확한 공개 범위는 각각의 라이선스에서 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 오픈소스이냐 아니냐에 상관 없이 공통적으로 해당된다. 그러나 어떤 특허를 오픈소스로 구현할 경우 해당 특허의 구현 결과는 오픈소스 라이선스를 따르게 되는 등, 오픈소스소프트웨어와 관련된 특허권의 문제는 보다 복잡하게 전개되고 있다. 특히 최근 소프트웨어특허가 급격히 증가하면서 문제가 심각해지고 있기 때문에, 새롭게 만들어지는 오픈소스 라이선스들에서는 특허관련조항을 포함하고 있는 경우가 많아지고 있다. 이와 관련된 자세한 내용도 다음 절의 쟁점부분을 참고하기 바란다.

□ 라이선스별 준수사항

이제 주요 라이선스 위주로 각각 준수해야 하는 사항들에 대해 살펴보기로 하자.

○ GPL 2.0

GPL은 현재 가장 많은 오픈소스소프트웨어가 채택하고 있는 라이선스이다. 오픈소스 라이선스들 중에서 가장 많이 알려져 있고 의무사항들도 타 라이선스에 비해 엄격한 편이다. GPL의 주요 내용은 다음과 같다.

- 소프트웨어를 배포하는 경우 저작권 표시, 보증 책임이 없다는 표시 및 GPL에 의해 배포된다는 사실 명시
- 소프트웨어를 수정하거나 새로운 소프트웨어를 링크(Static과 Dynamic linking 모두)시키는 경우 GPL에 의해 소스 코드 제공해야 함. 다만 리눅스를 기반으로 개발

된 Application은 소스코드 제공할 필요 없음
- 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 함
- 자신의 특허를 구현한 프로그램을 GPL로 배포할 때는 GPL 조건을 준수하는 이용자에게는 로열티를 받을 수 없으며, 제3자의 특허인 경우에도 특허권자가 Royalty-Free 형태의 라이선스를 제공해야만 해당 특허 기술을 구현한 프로그램을 GPL로 배포하는 것이 가능

GPL 소프트웨어를 사용하였을 경우 "본 제품(소프트웨어)는 GPL 라이선스 하에 배포되는 소프트웨어 XXX(사용한 GPL 소프트웨어 이름)를 포함 합니다"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 제품판매 시 함께 배포하거나, 매뉴얼에 소스코드를 요청할 수 있는 연락처를 기입하여 두거나, 혹은 FTP 서버, 웹서버 등에 소스코드를 업로드 해 두고 매뉴얼에 해당 주소를 기입하면 된다.
최근 특허에 관한 쟁점도 중요성이 증가하고 있는데, 자세한 내용은 다음 장의 쟁점 부분에서 설명한다.

○ LGPL 2.1

FSF가 일부 Library에 GPL보다 다소 완화된 형태인 GNU Lesser General Public License (LGPL)를 만들어 사용하고 있는 이유는 오픈 소스 소프트웨어의 사용을 장려하기 위한 전략적인 차원에서이다. 만일 상용 Library와 동일한 기능을 제공하는 Library에 GNU와 같은 엄격한 라이선스를 적용하게 되면, 개발자들이 Library의 사용을 꺼려할 것이다. 오히려 이미 널리 사용되고 있는 상용 Library와 동일한 기능을 제공하는 Library를 LGPL로 배포하여 그 사용을 장려하고 사실상의 표준으로 유도하는 한편, 관련된 다른 오픈 소스 소프트웨어를 보다 더 많이 사용할 수 있도록 하겠다는 것이 FSF의 전략이다. LGPL Version 2.1은 GNU ‘Library’ General Public License version 2.0의 후속 버전이다. 일부 한정된 Library에 대해서만 LGPL을 사용하려는 것이 FSF의 의도였으나 ‘Library’란 단어가 라이선스 이름에 포함되어 개발자들이 모든 Library를 위한 라이선스로 오인하는 경향이 있었다. 결국 이러한 오인을 방지하기 위하여 ‘Library’를 ‘Lesser’로 수정하였을 뿐 기본적인 내용은 동일하기 때문에 Version 2.1으로 표기한 것이다. LGPL의 주요 내용을 요약하면 다음과 같다.

- 소프트웨어를 배포하는 경우 저작권 표시, 보증책임이 없다는 표시 및 LGPL에 의해 배포된다는 사실 명시
- LGPL Library의 일부를 수정하는 경우 수정한 Library를 LGPL에 의해 소스코드 공개
- LGPL Library에 응용프로그램을 링크시킬(Static과 Dynamic Linking 모두)경우 해당 응용프로그램의 소스를 공개할 필요 없음. 다만 사용자가 Library수정 후 동일한 실행 파일을 생성할 수 있도록 Static Linking시에는 응용프로그램의 Object Code를 제공해야 함
- 특허의 경우 GPL과 동일함

LGPL은 링크하는 소프트웨어의 소스코드를 공개할 필요가 없다는 점이 GPL과 가장 큰 차이점이다. 여하한 경우에도 LGPL 소프트웨어 자체는 공개해야 하지만 LGPL 소프트웨어와 링크되는 부분의 소프트웨어 소스코드는 공개해야 할 의무가 발생하지 않으므로 기업의 입장에서는 LGPL 소프트웨어를 좀 더 선호하게 된다. 사용 여부 명시 등은 GPL과 동일하게 반영하면 되고 공개해야 할 소스코드의 공개 역시 GPL과 동일한 방식을 이용하면 된다.

○ BSD License

BSD(Berkeley Software Distribution) 라이선스는 GPL/LGPL보다 덜 제한적이기 때문에 허용범위가 넓다. 이는 BSD 라이선스로 배포되는 프로젝트가 미국 정부에서 제공한 재원으로 운영되었기 때문이다. GPL과의 차이점 중 가장 중요한 사항은 BSD 라이선스는 소스코드 공개의 의무가 없다는 점이다. 따라서 BSD 라이선스의 소스 코드를 이용하여 새로운 프로그램을 개발하여도 새로운 프로그램의 소스 코드를 공개하지 않고 BSD가 아닌 다른 라이선스를 적용하여 판매할 수 있다. 주요 내용을 요약하면 다음과 같다.

- 소프트웨어를 배포하는 경우 저작권 표시, 보증 책임이 없다는 표시
- 수정 프로그램에 대한 소스 코드의 공개를 요구하지 않기 때문에 상용 소프트웨어에 무제한 사용가능

○ Apache License

아파치 라이선스는 아파치 웹서버를 포함한 아파치 재단(ASF: Apache Software Foundation)의 모든 소프트웨어에 적용되며 BSD 라이선스와 비슷하여 소스코드 공개 등의 의무가 발생하지 않는다. 다만 "Apache"라는 이름에 대한 상표권을 침해하지 않아야 한다는 조항이 명시적으로 들어가 있고, 특허권에 관한 내용이 포함되어 BSD 라이선스보다는 좀 더 법적으로 완결된 내용을 담고 있다. 특히 Apache License 2.0

에서 특허에 관한 조항이 삽입되어 GPL 2.0으로 배포되는 코드와 결합되는 것이 어렵다는 문제가 있었는데, GPL 3.0(안)에서는 이 문제를 해결하여 아파치 라이선스로 배포되는 코드가 GPL 3.0으로 배포되는 코드와 결합하는 것이 가능해졌다.

○ MPL(Mozilla Public License)

MPL은 Netscape 브라우저의 소스코드를 공개하기 위해 개발된 라이선스이다. MPL은 공개하여야할 소스코드의 범위를 좀 더 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 훨씬 명확하다. 주요 내용을 요약하면 다음과 같다.

- 소프트웨어를 배포하는 경우 저작권 표시, 보증 책임이 없다는 표시 및 MPL에 의해 배포된다는 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 라이선스는 MPL이 아닌 다른 것으로 선택가능
- 특허기술이 구현된 프로그램의 경우 관련 사실을 'LEGAL'파일에 기록하여 배포

□ 주요 쟁점

○『소스코드 공개 여부』

앞에서 GPL/LGPL/BSD/MPL/Apache License 등 많이 사용되는 라이선스들을 간략히 살펴보았는데 이중에서 GPL/LGPL/MPL 등은 수정한 내용에 대한 소스코드를 공개하여야 하고 BSD/Apache License 등은 수정하더라도 소스코드를 공개할 의무가 발생하지 않는다. GPL/LGPL/MPL 등 소스코드 공개 의무가 발생하는 라이선스는 상호주의(reciprocal) 혹은 copyleft 라이선스라고 하며, 그 결과물이 원 소프트웨어의 파생물(derivative work)일 경우 소스코드를 공개해야 한다. 그런데 소스코드의 공개 범위를 기계적으로 판단할 수 있는 방법은 없으며 라이선스마다 서로 다르게 정의하고 있으므로 잘 판단하여야 한다. GPL을 중심으로 보다 자세히 살펴보도록 하자. GPL의 경우, GPL 프로그램의 소스코드를 이용자가 개발한 프로그램코드에 삽입하거나 링크시킨 후 함께 배포하고자 하는 경우, 이용자가 개발한 프로그램도 GPL 조건

으로 배포해야 한다. 반면 GPL 제2조 후단에서는 '원 프로그램으로부터 파생되지 않고 그 자체로 독립적이고 분리되어 있는 저작물(separate works)은 다른 라이선스 조건에 의해 배포가능하며, 단순 집합물(mere aggregation)로서 원 프로그램과 동일한 매체로 배포할 수 있다'고 규정하고 있다. 예를 들어 독립적으로 제작된 프로그램을 GPL 프로그램과 단순히 동일한 매체에 저장하여 배포할 경우에는 GPL이 아닌 다른 라이선스조건에 의해 배포할 수 있다. 하지만 구체적으로 어떠한 경우가 파생물에 해당하는지, 또는 독립된 프로그램의 단순한 집합물에 해당하는지를 구별하는 것은 쉽지 않다. 결국 최종적으로는 법원의 판단에 의해 결정될 문제인데, FSF는 GPL FAQ에서 몇 가지의 구별기준을 제시하고 있다. 예를 들어 두개의 모듈이 동일한 실행파일에 포함되어 있거나 공유주소영역(shared address space)에서 링크되어 실행되도록 설계된 경우에는 전자에 포함되고, 2개의 프로그램이 파이프(pipes), 소켓(sockets), command-line arguments 형태로 통신하는 경우에는 후자에 해당한다. 플러그인(plug-ins)의 경우 동적으로 링크되어 함수 호출을 하고 데이터구조를 공유하는 경우에는 전자에, fork와 exec를 이용하면 후자에 해당한다. 인터프리터(interpreter)나 컴파일러(compiler)의 경우에는 원칙적으로 컴파일 된 결과물과는 분리된 것으로 보지만, 컴파일과정에서 라이브러리나 클래스가 결과물에 추가된 경우에는 라이브러리 또는 클래스와 결과물은 하나의 프로그램으로 볼 수 있다.

이와 같은 GPL의 특징은 다른 오픈소스소프트웨어 라이선스와 비교할 때 명확히 드러난다. 예를 들어 LGPL은 일정한 요건을 충족시키는 경우 LGPL 라이브러리(Library)를 이용하는 프로그램(Work that uses the Library), 다시 말해 컴파일 또는 링크를 통해 LGPL 라이브러리와 함께 작동하도록 설계된 프로그램들을 배포할 경우에는 소스코드를 제공하지 않아도 된다.(LGPL 제5조). MPL도 한편으로는 GPL과 마찬가지로 수정된 코드의 소스코드를 제공할 것을 요구하면서, 다른 한편으로는 MPL 조건의 코드와 기타의 라이선스 조건의 코드를 결합한 프로그램(MPL에서는 이를 'Larger work'라고 표현하고 있다)을 만드는 것을 허용하고, 결합된 프로그램을 MPL이 아닌 다른 라이선스로 배포하는 것을 허용하고 있다(MPL 제3.7조). 예를 들면 별도의 파일로 함수(Function)를 추가할 경우 MPL은 기존 코드의 수정부분에만 적용할 뿐 추가된 함수에는 적용되지 않는다.

한편 원칙적으로 GPL 조건으로 배포하면서도 GPL 제2조와 관련해서는 다소 완화된 내용의 조건으로 프로그램을 배포하는 경우가 있다. 대표적 사례는 리눅스커널의 경우인데, 리눅스커널의 'COPYING' 파일에 포함되어 있는 라이선스 조건에는 '정상적인 시스템 콜에 의해 커널 서비스를 이용하는 사용자프로그램(user programs)은 GPL에 의해 배포하지 않아도 좋다는 내용이 포함되어 있다. 이와 같은 내용에 따라 리눅

스커널을 기반으로 한 라이브러리나 응용프로그램은 소스코드를 제공하지 않고도 배포할 수 있으며, 시장에서는 이를 확대해석하여 표준커널인터페이스를 이용하는 디바이스 드라이버나 동적 커널모듈(Loadable Kernel Module)도 사용자프로그램이 시스템 콜을 이용하는 것과 크게 다르지 않기 때문에 소스코드를 제공할 필요가 없는 것으로 보고 있다.

두 번째 사례는 GNU Classpath 프로젝트와 자바(Java) 플랫폼사례이다. GNU Classpath 프로젝트는 자바(java)언어의 가상머신(virtual machines) 및 컴파일러에서 사용되는 핵심 클래스라이브러리(core class libraries)를 자유소프트웨어로 대체하기 위한 프로젝트인데, 동 프로젝트의 결과물을 GPL로 배포하면서도 이와 링크된 다른 독립된 소프트웨어는 GPL로 배포할 필요가 없다는 내용의 예외를 인정하였다. 그런데 2006년 말 Sun이 향후 자바 플랫폼을 GPL 조건으로 배포하겠다는 선언을 하면서, 자바 플랫폼 중 특히 Java SE(Java Platform Standard Edition)와 Java EE(Java Platform Enterprise Edition)의 GPL 배포조건에 Classpath 예외를 추가한다고 발표하였다. 그 결과 Classpath 예외조항을 포함한 GPL 조건의 자바 플랫폼을 이용한 응용프로그램도 소스코드를 공개하지 않고 배포할 수 있다.

○ 특허권

GPL, LGPL, MPL, Apache 라이선스 등의 오픈소스 라이선스는 특허와 관련된 조항들을 가지고 있는데, 각각의 경우를 ⅰ) 라이선서(Licensor)의 특허인 경우, ⅱ) 제3자의 특허인 경우, ⅲ) 라이선시(Licensee)의 특허인 경우로 구분하여 설명할 수 있다. 다만 LGPL은 특허와 관련해서는 GPL과 동일하게 규정하고 있으며, BSD는 특허에 관한 규정을 두고 있지 않기 때문에 이하에서는 생략한다.

- 라이선서(Licensor)의 특허인 경우 : 소프트웨어에 대해 저작권을 가지고 있는 주체가 특허권을 함께 가지고 있는 경우이다. MPL과 Apache 라이선스는 이와 같은 경우 라이선서가 소프트웨어를 오픈소스 라이선스조건으로 배포하는 경우 관련 특허권의 라이선스도 무상으로 제공하는 것으로 규정하고 있다. GPL의 경우에는 명문으로 규정하고 있지 않지만 대체적으로 관련 조문(제7조 등)의 해석상 묵시적인 라이선스를 제공하는 것으로 보고 있다. GPL 3.0에서는 단순 재 배포자를 제외한 개발자 및 기여자(Contributor)의 경우 자신이 기여한 부분과 관련된 특허권 라이선스를 무상으로 제공하는 것으로 규정하고 있다. 한 가지 주의하여야 할 것은 특허권 그 자체는 여전히 유효하다는 것이다. 따라서 예를 들어 특허권자 특허 받은 정렬 알고리즘을 GPL로 배포되는 리눅스에 로열티 없이 사용 가능하도록 제공한다고

할지라도 독점 라이선스인 MS윈도우즈에는 해당 정렬 알고리즘을 사용토록 허가하면서 여전히 로열티를 받을 수 있다.

- 라이선시(Licensee)의 특허인 경우 : 프로그램을 사용하는 이용자가 특허권을 가지고 있는 경우이다. MPL의 경우 이용자가 자신의 특허권을 문제 삼지 않고 그냥 사용하는 경우에는 아무런 문제가 없지만, 만약 이용자가 MPL 프로그램을 사용하던 중 자신의 특허권을 근거로 소송을 제기하게 되면, 적절한 시일 내에 소송을 철회하지 않는 한 라이선스가 종료되고, 그 결과 MPL 프로그램을 더 이상 사용할 수 없거나, 그 동안 사용했던 부분에 대하여 로열티 산정을 하는 등 일정한 보복이 가해진다. Apache 라이선스 2.0도 MPL과 비슷한 취지의 조항을 추가하였으며, GPL 3.0에서도 관련 내용이 추가되었다.

- 제3자의 특허인 경우 : 특허 소유자와 이를 프로그램으로 구현한 주체가 다른 경우인데, GPL 제7조에 의하면 특허 소유자가 무상(Royalty-Free) 조건의 특허 라이선스를 허용하지 않는다면 구현자는 이 프로그램을 GPL 조건으로 배포할 수 없다. 예를 들면 甲회사가 乙회사의 특허기술을 바탕으로 A라는 프로그램을 만들었을 경우, 乙회사가 그 특허를 모든 사람에게 무상으로 허용하지 않는다면, 설사 甲회사가 라이선스를 무료로 받았다고 할지라도 A프로그램을 GPL 조건으로 배포할 수 없다. 나아가 GPL 3.0에서는 제3자인 특허권자가 이용자들을 차별하여 라이선스를 부여하는 것을 막기 위한 조항이 삽입되었다. 그 결과 2006년 말 MS와 노벨 사이에 체결되었던 형태의 계약은 향후 어려울 것으로 보인다. MPL은 제3자의 특허인 경우에도 일단 배포는 허용하되, 'LEGAL'이라는 이름의 파일을 추가하여 어떠한 특허가 문제되고 있는지, 어떤 사람이 클레임을 제기하고 있는지에 대한 사항을 자세히 기록하도록 하고 있다.

○ 듀얼 라이선스

또한 일부 오픈소스소프트웨어는 복수의 라이선스 하에 배포되는 경우가 있다. 이를 주로 듀얼 라이선스(dual license)라고 하며, 이런 경우는 주로 오픈 소스 소프트웨어를 상업적 목적으로 이용할 뿐만 아니라 오픈소스 커뮤니티와의 협력을 위한 경우가 대부분이다. 하나 이상의 라이선스가 있는 오픈소스소프트웨어를 이용할 경우, 이용자는 사용 목적에 가장 잘 부합하는 라이선스 하에 배포되는 소스 코드를 선택할 수 있다. 대표적인 사례로는 MySQL, Trolltech의 Qt 라이브러리 등이 있다.

□ 주요 오픈소스소프트웨어별 이슈

앞서 살펴본 오픈소스 라이선스에 대한 이해를 바탕으로 실제 많이 사용하고 있는 오

픈소스소프트웨어들 중 특별히 기억해 두어야 할 라이선스 관련 이슈에 대해 살펴보자.

○ Linux 커널

Linux 커널은 GPL v2로 배포된다. 그런데 리눅스커널의 'COPYING'파일에는 GPL v2 전문과 함께 다음과 같은 내용이 맨 위에 추가로 기재되어 있다.

> NOTE! This copyright does *not* cover user programs that use kernel services by normal system calls - this is merely considered normal use of the kernel, and does *not* fall under the heading of "derived work". Also note that the GPL below is copyrighted by the Free Software Foundation, but the instance of code that it refers to (the Linuxkernel) is copyrighted by me and others who actually wrote it.
>
> Also note that the only valid version of the GPL as far as the kernelis concerned is _this_ particular version of the license (ie v2, not v2.2 or v3.x or whatever), unless explicitly otherwise stated.

정상적인 시스템 콜에 의해 커널 서비스를 이용하는 사용자프로그램(user programs)은 GPL에 의해 배포하지 않아도 좋다는 뜻이다. 이와 같은 내용에 따라 리눅스커널을 기반으로 한 라이브러리나 응용프로그램은 GPL v2의 영향을 받지 않는 것이다. 그러나 Linux 커널은 커널의 기능을 커널 모듈이라는 형태로도 이용할 수 있는 인터페이스를 제공하는데 이 커널 모듈도 위의 예외사항에 속하는지에 대한 논란이 계속되고 있다. 즉, 리눅스 커널모듈도 모두 GPL v2로서 소스코드를 공개해야 하는지, 아니면 독점 라이선스를 적용하여 소스코드를 공개하지 않아도 되는지에 대한 논란이다. 이에 대해서는 리눅스 커널 개발자들 사이에서도 의견이 갈리고 있다.
최소한, http://www.kernel.org 에서 배포되는 공식 커널 버전을 기준으로 모듈 인터페이스를 임의로 수정하지 않은 상태에서 동작이 가능한, 자체 개발된 커널 모듈은 GPL이 아닐 수 있다고 주장할 수 있다. 그러나 모듈 인터페이스를 임의로 수정할 경우에는 설득력이 훨씬 약해지며, 커널 모듈이 GPL이냐 아니냐에 대한 논란은 매우 첨예하게 대립하고 있으므로 커널 모듈은 모두 GPL이라고 가정하고 공개가 불가능한 부분은 사용자프로그램(user program)에서 구현하는 것이 바람직하다.

○ FreeBSD

FreeBSD는 1993년에 처음 발표되었다. FreeBSD는 Unix 시스템인 BSD(Berkeley Software Distribution)를 기반으로 개발되었고, 다양한 Unix 버전 중 오픈 소스 Unix라 할 수 있을 것이다. FreeBSD는 가장 제약 사항이 적은 BSD License에 의해 배포되

고 있다. 따라서 FreeBSD 소스 코드를 상업적으로 아무런 제한 없이 이용할 수 있다. 단, 소스 코드 혹은 바이너리 형식으로 재배포할 때는 FreeBSD의 원저작권자인 The Regents of the University of California의 저작권을 명시해야 한다. 현재 FreeBSD의 소스 코드에는 다음과 같이 저작권이 명시되어 있다. "All of the documentation and software included in the 4.4BSD and 4.4BSD-Lite Releases is copyrighted by The Regents of the University of California" 또한, Free BSD를 이용하고 있음을 밝히기 위해서는 다음의 사항을 명시해야 한다. "This product includes software developed by the University of California, Berkeley and its contributors." 마지막으로 재배포 시에는 FreeBSD 소스 코드에 포함된 라이선스 내용을 그대로 포함시켜야 한다. 요약하면, FreeBSD는 소스 코드 공개를 요구하지 않으며, 만약 원본 혹은 수정된 소스 코드를 공개하고자 하면 위에서 언급한 사항들만 준수하면 된다.

○ MySQL

MySQL은 현재 가장 인기 있는 관계형 데이터베이스 서버로서 사용자는 GPL 라이선스나 일반 상용 라이선스 둘 중 한 가지를 선택할 수 있는데, 상용 라이선스는 GPL 라이선스의 여러 가지 요구사항들(소스코드 공개 등)을 지키기 어려운 경우에 선택할 수 있으므로 일반적인 상용 라이선스 판매를 통해 수익을 내고 있다. 이러한 라이선싱 모델을 듀얼 라이선스라고 하며 MySQL은 듀얼 라이선싱 모델의 대표적인 사례로서 종종 언급된다. MySQL의 이 같은 듀얼 라이선싱 모델에 대해서 좀 더 살펴보면 다음과 같다.

우선 오픈소스프로젝트 목적으로 MySQL을 이용할 경우를 살펴보자. 만약, 이용자가 MySQL 라이브러리를 이용하여 개발한 애플리케이션을 GPL 라이선스 하에 배포한다면, 이 경우 MySQL은 GPL 라이선스가 적용된다. 즉, 이용자가 기꺼이 직접 개발한 애플리케이션을 GPL하에 배포하기 때문에 MySQL 라이브러리 역시 GPL이 되더라도 문제가 발생하지 않을 것이다. 또한 이용자가 GPL이 아닌 Open Source Initiative에서 인정한 라이선스 하에 직접 개발한 애플리케이션을 배포할 경우에도 특별 예외조항을 두어 라이선스 충돌이 발생하지 않도록 하였다. 즉, 비록 GPL 라이브러리를 이용하더라도 직접 개발하였거나 GPL 라이브러리와 독립된 부분으로 인정이 되는 애플리케이션의 소스 코드는 GPL이 아닌 다른 오픈소스라이선스로 배포할 수 있도록 하였다. GPL의 원칙을 따를 경우, GPL 라이브러리에 기반한 애플리케이션 전체 소스코드가 GPL이 되기 때문에 소스코드 공개 의무가 발생한다. 한편 GPL 등 오픈소스 라이선스조건을 따르지 않는 형태로 사용하고자 하는 경우에는 MySQL AB 사로부터 Commercial License를 받아야 한다.

그러나 한 가지 주지하여야할 것은 GPL의 의무사항은 소프트웨어를 배포할 때 발생

하는 것이므로 만약 MySQL을 다운로드해서 MySQL과 연동되는 웹사이트 등을 만들어서 서비스만 하는 경우는 MySQL을 직접배포하지 않는 것이므로 GPL의 의무사항이 발생하지 않는다는 것이다. 예를 들어 인터넷 포털업체들은 MySQL의 상용버전을 구입하지 않고 GPL버전을 사용하면서 MySQL이나 관련소프트웨어의 소스코드를 공개하지 않고 있다.

○ Apache

Apache 소프트웨어들은 현재 ASF에서 자체적으로 개발한 Apache License Version 2.0하에 배포되고 있다. Apache License에 따르면, 누구든 자유롭게 Apache 소프트웨어를 다운받아 부분 혹은 전체를 개인적 혹은 상업적 목적으로 이용할 수 있다. 또한, 재배포 시에도 원본 소스 코드 혹은 수정한 소스 코드를 반드시 포함시킬 것을 요구하지 않는다. 다만, 재배포하고자 할 경우에는 Apache License, Version 2.0을 포함시키고 ASF에 개발된 소프트웨어임을 명확히 밝힐 것을 요구한다. 아울러 ASF가 승인하지 않는 ASF 공식 마크를 임의로 사용할 수 없다.

○ Mozilla Firefox

Firefox는 오픈 소스 소프트웨어이며 현재 MPL, GPL, LGPL 세 가지 라이선스 하에 배포되고 있다. 이 세 가지 라이선스는 모두 공통적으로 누구나 소스 코드를 보고 수정하며 재배포하는 것을 허용한다. 원래 Firefox는 MPL에 의해 배포되었다. 그런데 파생물의 상업적 이용을 제한적으로 허락하는 MPL은 GPL 혹은 LGPL과 호환될 수 없기 때문에 FSF로 부터 많은 비난을 받았다. 이 문제를 해결하기 위해 Mozilla는 Firefox를 MPL, GPL, 그리고 LGPL하에 다시 라이선싱 하였다. 이 후, 개발자들은 이 세 가지 라이선스 중 자신들의 목적에 가장 잘 부합하는 라이선스를 선택할 수 있게 되었다. 그런데 하나 유의할 점은, Firefox의 일부 상용 컴포넌트를 포함하고 있기 때문에, 이 상용 컴포넌트는 위에서 언급한 세 가지 라이선스의 적용을 받지 않는다. 대신, 이들은 Mozilla End User License Agreement(EULA)의 제한을 받고 있다.

○ Sendmail

Sendmail은 1981년 Eric Allman에 의해 Mail Transfer Agent(MTA)로 개발되었다. 당시에 그는 UC Berkely에서 일하며 대학의 네트워크와 Arpanet 사이에 이메일을 주고받기 위한 목적으로 sendmail을 개발하였다. 처음부터 Sendmail은 메일 프로토콜의 개방성과 라우팅 기능성, 파일의 유연성에 초점을 맞추었기 때문에 오늘날 이메일 서버 시장의 1위 자리를 차지할 수 있었을 것이다. Allman은 여전히 Sendmail 개발의 중심에 있으며, Sendmail의 유지보수와 기능 및 성능 개선 등의 난관을 헤쳐가기 위해 Sendmail, Inc.이라는 회사를 설립하였다. Sendmail 8.9 이전 버전은 오픈 소

스 형식으로 개발되었으며, 8.9 버전부터는 Allman이 설립한 회사에 의해 개발 및 공개되었다. 이러한 역사적 배경 때문에 Sendmail은 두 가지 다른 라이선스 형식을 취하고 있다. 우선, Sendmail을 단순히 컴파일해서 바이너리만 사용하거나, 아니면 오픈 소스 형식으로 소스 코드를 제공하며 이용할 경우는 Sendmail License를 적용받는다. 이 라이선스는 Sendmail의 자유로운 이용, 수정, 및 배포를 허락한다. 단, 재배포할 경우에는 배포에 필요한 비용 이상을 부과할 수 없으며, Sendmail에 포함된 copyright notice를 반드시 포함시켜야 한다. 그리고 재배포시 소스 코드를 포함시키지 않을 경우에는, 최대 3년까지 소스 코드 제공을 보장하는 문서를 포함시켜야 한다. 반면, 소스 코드 제공 없이 상업적 용도로 이용할 경우에는 Sendmail, Inc.와 접촉해서 별도의 라이선스를 받아야만 한다.

(2) 커뮤니티

□ 공개SW 커뮤니티 현황 개요

공개SW(Open Source Software)는 저작권자가 소스코드를 공개하여 누구나 특별한 제한 없이 자유롭게 사용, 복제, 배포, 수정할 수 있는 소프트웨어 이다. 이러한 이유로 개발자, 사용자의 참여가 필수적인데, 이러한 개발자, 사용자의 참여를 “공개SW 커뮤니티”로 칭한다.

공개SW 커뮤니티는 보통 자발적으로 생성이 되어 산업현장에 끊임없이 새로운 정보와 기술을 발산하며, 실시간 정보교류와 의사소통이 진행된다. 실제로 개별업체에 소속된 개발자들도 특정 프로젝트를 개발할 때 어려움에 부딪히면 관련커뮤니티를 통해 해결한다. 또한 새로운 기술에 대한 검증과 트렌드를 접해 볼 수 있는 곳이 바로 커뮤니티다. 글로벌 SW업체의 경우, 제품을 출시하기 전에 대형커뮤니티에 평가판을 배포해 시장진출 가능성을 미리 타진하기도 한다.

국내 공개SW 커뮤니티의 경우 2004년 공개SW 시범사업 이후 공개SW의 저변 확대 및 인식 전환이 되는 계기를 마련하였으나 아직 중요도에 비하여 활성화 되고 있지 못한 실정이다.

국내 공개SW 커뮤니티는 100여 개가 활동하고 있는 것으로 추정 되며, 비교적 회원을 많이 확보해 대외적으로 알려진 국내 공개SW 커뮤니티는 50여 곳에 불과하다. 또한, 국내 공개SW 커뮤니티에서 활동하고 있는 회원들은 개인 사용자 중심으로 구성되어 있어 시장 수요를 견인하기에는 한계가 있다. 국내 공개SW 커뮤니티는 사용자 커뮤니티가 대부분이고 개발자 커뮤니티는 KLDP.net 등 2~3개 수준에 머물고 있다.

□ 국내 공개SW 커뮤니티 종류

공개SW는 웹 2.0, SaaS (Software as a Service), 가상화, 클라우드 컴퓨팅 등 이슈화된 주제에서 빼놓을 수 없는 중요한 위치를 차지하고 있다. 특히 웹 2.0의 개념이 잘 구현된 것이 공개SW라 할 수 있으며 공개SW는 웹 2.0에서 얘기하는 네트워크 효과의 이점을 잘 활용하는 구조를 가지고 있다.

공개SW 커뮤니티에서는 네트워크를 통하여 참여한 사용자간에 지속적인 상호 작용이 일어나고 이를 통해 디지털 데이터가 생성 된다. 공개SW 커뮤니티는 공개SW의 종류와 생성되는 디지털 데이터로 그 종류를 구분해 볼 수 있다.

- 공개SW 개발자 커뮤니티

  공개SW 커뮤니티에 의해 생성되는 디지털 데이터가 "공개SW의 소스 코드" 이다.

- 공개SW 사용자 커뮤니티

  공개SW 커뮤니티에 의해 생성되는 디지털 데이터가 "공개SW의 사용 정보" 이다.

- 공개SW 애플리케이션 사용자 커뮤니티

  MySQL, Tomcat 등과 같은 공개SW 애플리케이션의 사용과 관련된 정보를 교환하는 커뮤니티

- 공개SW 라이브러리 사용자 커뮤니티

  공개SW 라이브러리 사용법에 대한 정보를 교환하는 커뮤니티

- 공개SW 프로그램 언어 사용자 커뮤니티

  PHP, Java 등과 같은 공개SW 프로그램 언어에 대한 커뮤니티

  공개SW를 사용한 개발자들 간에 개발과 관련된 다양한 정보를 교환한다.

[표3-1. 국내 공개SW 사용자 커뮤니티]

사이트	사이트 설명
Debian (데비안)	• GNU 프로젝트에서 만든 운영체제 도구를 사용하는 Linux로 GNU/리눅스라고 불린다. • 2000년 10월, 공개SW 운영체제인 데비안 사용자들이 만든 홈페이지 • 공개 SW 세미나 및 데비안 컨퍼런스 개최

	• 회원제는 아니나 1일 약 5~600명 방문
GNOME 한국 사용자모임	• GNOME 한국 사용자 모임 • 2000년 4월에 개설된 그놈 한국 개발자 커뮤니티 • 그놈 환경의 번역 및 국제화 패치 작업 주도 • 회원 수 : 약 400여 명
KELP	• 한국 임베디드 리눅스 프로젝트 • 1999년 개설된 국내 대표적인 임베디드 SW 커뮤니티 • 초보에서 전문 수준에 이르는 다양한 강좌 진행 • 회원 수 : 약 1,500명
Korea Apache Group	• 아파치 유저 그룹 • 1997년 8월 웹서버에 대한 교육을 목적으로 개인 홈페이지 시작 • APM (Apache, PHP, MySQL)을 위한 사용자 커뮤니티
PHPSCHOOL	• PHP 개발자 커뮤니티 (장대익) • 1998년 9월 개설, 2005년 4월 법인화하여 PHP 자격 인증, 교육, IT 서비스 등 비즈니스 활동 • 1일 방문자 수 3만명, 1일 페이지 뷰 약 10만건
SUSE Linux 사용자 모임	• SUSE Linux 사용자 모임
김프 코리아	• 김프(GIMP)를 사용하는 한국 사용자들의 커뮤니티
데브피아	• 개발자 천국을 꿈꾸는 IT 포털, 2000년 6월 9일 설립 • MS(데브피아), 자바(자바누리), 공개SW(PHPSCHOOL) 3대진영이 연합 • 회원수 : 50만명
데이터베이스 사랑넷 (DSN)	• LDAP, 오라클, MySQL, PostgreSQL, MS-SQL, 인포믹스, Sybase, DB2, CUBRID 등 데이터베이스 관련 전자 게시판(BBS, Bulletin Board System) 운영, 멘토링 기반의 사용자 포털 • 1998년 정재익씨의 개인 사이트로 시작 (정재익, 문태준, …) • PostgreSQL, Gallery 사용
오픈 오피스 한국어 커뮤니티	• 오픈 오피스 한국어 커뮤니티
한국리눅스 유저그룹	• 한국리눅스유저그룹 - LUG Korea • 1998년 지역별로 소모임 형태로 활동하던 리눅스 사용자들을 통합해 전국 규모로 설립 • 2000년 3월 대구 계명대에서 첫 세미나를 개최한 이후 매년 리눅스 관련 기술 세미나를 개최 • 회원 수 : 약 7천 명
한국 모질라 커뮤니티	• 모질라의 공개SW(Firefox, Thunderbird) 한국 커뮤니티
한국 소프트웨어 커뮤니티 연합 (SCA)	• 2005년 7월 솔라시스 스쿨, 솔라리스 테크넷, 자바유저스넷, 자바크래프트, 파란자바동, KELP 등 국내 대표적인 15개 SW 개발자 및 엔지니어 커뮤니티를 중심으로 발족
한국 스프링 사용자 모임	• Java 개발 프레임워크인 Spring의 한국 사용자 모임

한국 자바 개발자 협의회 (JCO)	• 2000년 3월 자바라인, 자바랜드, 자바카페, 하이텔자바동, 자바스터디 등 5개 사이트 연합 • 자바스터디, 자바서비스넷, 모바일자바, JStorm,자바카페, 자바 모델링, 자바캔, 오브젝트월드, 한국스프링사용자모임 KSUG, 자바랜드, 파란 자바동, JavaCraft.Net, JSP School, 자바유저스넷, 아첸자바커뮤니티, OKJSP • 자바라인, 지니월드, 프로자바, OpenSeed Project

※ 순서는 ABC(영문), 가나다(한글)로 정렬

[표3-2. 국내 공개SW 개발자 커뮤니티]

사이트	사이트 설명
APM Setup	• Apache, PHP, MySQL 자동 설치 프로그램 사이트
KLDP	• Korean Linux Documentation Project • 한국의 공개SW 개발 프로젝트 지원 사이트 • 1996년 10월 개인 홈페이지로 출발, 현재 국내의 대표적인 리눅스 사이트, 리눅스 관련 문서의 한글화 작업이 시초 • 지금은 문서화뿐만 아니라 커뮤니티, 프로젝트 호스팅, CodeFest 개최 등 다양한 활동을 진행 • 회원 수 : 약4만 명
네이버(NHN), 개발자 센터	• 공개SW, 오픈 API 관련 네이버의 기술을 공유 한다. • Naver 오픈 API 공식 카페
다음, DNA 개발자 네트워크	• Daum DNA란 Daum의 개발자 네트워크 및 협력 파트너(Developers Network and Affiliates)를 위한 서비스로서 Daum 사외의 개발자 및 비즈니스 파트너를 대상으로 Daum이 축적한 서비스와 기술을 공유 • 오픈API 제공과 공개SW 지원
삼성 모바일 이노베이터	• 삼성 모바일 이노베이터, Samsung Mobile Innovator • 전 세계 휴대폰 소프트웨어(SW) 개발자들을 위해 기술 지원과 정보를 공유하는 휴대폰 소프트웨어 개발자 사이트
한국 공용 버그 질라	• 글로벌 오픈소스 커뮤니티에서 활동하는 한국 개발자들이 소스개발에 관심 있는 국내 개발자들을 위해 소스 코드 버그들을 미리 판별해 주고 이를 해외 프로젝트에서 버그로 처리하는 방식 및 패치 제작 등을 멘토링 해 주기 위해 만들어 사이트 (2008년 1월 18일 ~)

※ 순서는 ABC(영문), 가나다(한글)로 정렬

[표3-3. 국내 공개SW 기타 커뮤니티]

사이트	사이트 설명
Creative Open Source Hub - OPENYOURS	• Open Your Source (OYS)의 줄임 말 • 공개SW와 관련된 뉴스, 이슈, 그리고 파일 정보들을 실시간으로 빠르게 전달
Open Source Community Lab	• 공개SW 커뮤니티 연구소

※ 순서는 ABC(영문), 가나다(한글)로 정렬

□ 공개SW 커뮤니티 지원 전략

공개SW 커뮤니티는 그 자생력에 있어서는 자발적 활동이 매우 중요하지만 유지, 발전을 위해서는 정부나 SW서비스 기업의 지원이 필요하다. 개발자와 더불어 정부, 기업 모두 SW 시장의 중요한 구성원이기 때문이다. 커뮤니티 기반이 안정되어 있는 미국과 유럽의 경우만 하더라도 정부 주도의 공개SW 개발 프로젝트를 통하여 커뮤니티가 지속적인 자생력을 작도록 지원하고 있다. 선진국 뿐 아니라 커뮤니티 기반이 약한 아시아 국가들도 자국의 개발자들이 국제무대에서 활동 할 수 있도록 커미터를 대량 육성하고 있다. 커뮤니티가 자사 제품의 확대와 기술 발전을 이끈다는 판단 아래 글로벌 SW 기업의 개발자 지원 체계를 보면 개발자 커뮤니티는 일종의 동반자로 인식하고 있음을 알 수 있다. 오라클의 경우, 개발자와 커뮤니티를 위해 수백만 달러를 투자하고 있으며, 마이크로 소프트는 “커뮤니티 얼라이언스 프로그램”을 통해 체계적으로 개발자 커뮤니티를 지원하고 있다.

[표3-4. 커뮤니티를 연계한 정부 주도의 개발프로젝트 사례]

사례	주요내용
미국 국방부의 OTD(Open Technology Development	- 산업발전과 새로운 변혁에 따른 개발인력의 부족 현상을 타개하기 위해 추진 - “지구공간기반구조” 등의 분야에서 외부 공개SW 커뮤니티와의 협력체 구성을 추진
미국 대한 중심의 Sakai 프로젝트	- 대학 간 콘텐츠 공유를 위해 추진 - 미시간, 인디애나 주립대학을 포함하여 전 세계 105개 대학들이 커뮤니티에 참여
프랑스 지방 정부의 ADULLACT 프로젝트	- 지방정부 애플리케이션에 대한 공개SW 커뮤니티를 구축하기 위해 추진

(출처 : NIPA(구 KIPA) 정책연구센터, “해외 공개SW 정책분석 및 시사점”)

다행히 근래 들어 공개 SW 커뮤니티의 중요성을 인식하고 이를 활발히 도입하려고 하는 추세에 있다. 다음 커뮤니케이션스, NHN등 국내 포털사이트들은 필요한 SW를 개발하기 위해 커뮤니티를 활용하고 있으며, 삼성전자·LG전자 등도 멘토로 참여하고 있다. 정부 차원의 지원도 늘고 있다. 지식경제부는 지난해 4월부터 11월까지 대학·기업이 진행하는 공개 SW 개발자 커뮤니티의 연구 과제를 발굴, 지원하기로 했으며, 2007년 10개였던 지원 대상을 3배로 늘리고, 관련 예산도 과제별로 평균 5000만원씩 총 15억 원을 배정해 업계 사기를 올리고 있다. 공개 SW 커뮤니티에서 디지털교과서, 안드로이드 UI 개발 도구 같은 유망한 분야를 키운다는 전

략이다.

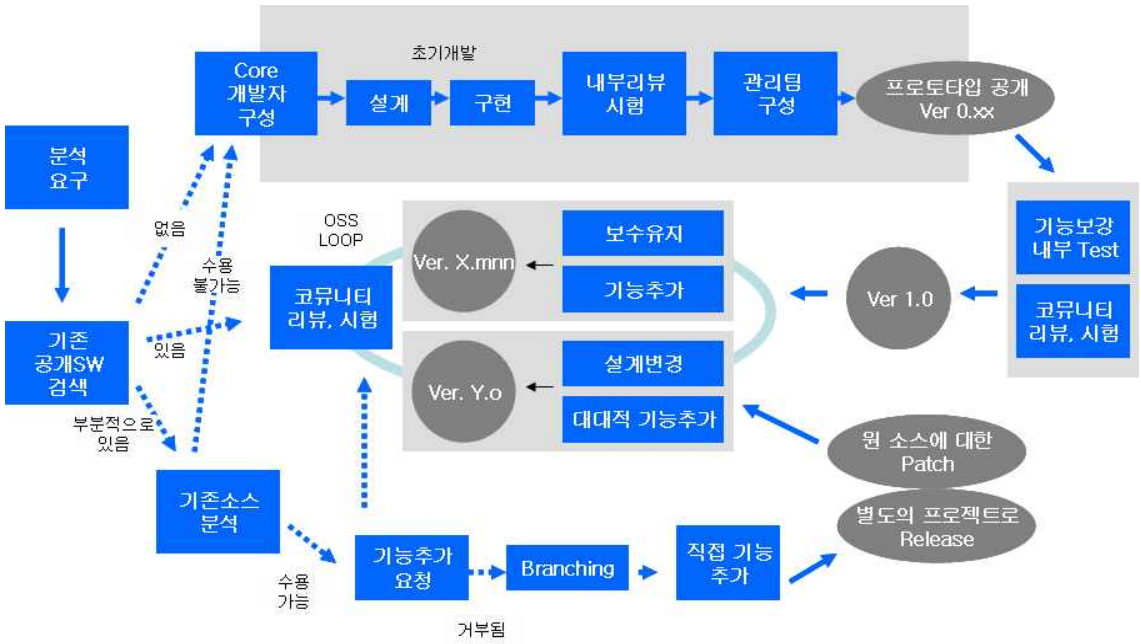
공개SW의 부가가치는 개발자 자체라고 해도 과언이 아니다. 경쟁력 또한, 개발자들이 다른 나라 종사자들에 비해 얼마나 실력이 있는가에 따라 판가름 난다. SW 산업의 핵심 인프라인 개발자들이 소통하는 커뮤니티가 끊임없이 기업과 SW산업 발전의 교량역할을 하기 위해서는 정부 및 SW기업들의 커뮤니티에 대한 인식과 체계적이고 지속적인 지원과 정책이 뒷받침 되어야 할 것이다.

(3) 방법론

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 리뷰 및 시험 과정, 기술 지원 방법, 기능의 확장, 새로운 프로젝트로의 따른 가지치기 과정 등이 상용 소프트웨어와 달리 매우 중요한 의미를 가지게 된다.

□ 기존 프로젝트 참여

어떤 소프트웨어를 개발하고자 하는 사람(그룹)은우선 개발하고자 하는 소프트웨어가 가져야할 기능에 대한 충분한 분석을 한 뒤, 이미 존재하는 오픈 소스 소프트웨어 프로젝트들 가운데 주어진 요구 사항을 만족하는 것이 있는지 확인한다.



[그림3-2. 공개SW 프로젝트 생명 주기]

검색 단계에서 유사 프로젝트가 발견되었지만, 기존 프로젝트 관리자 그룹에서 새로운 기능 추가 요청을 받아들이지 않은 경우에 개발자는 두 가지의 선택을 할 수 있다. 첫 번째는 검색 단계에서 요구 사항을 만족하는 프로젝트가 없었던 경우와 같이, 전혀 새로운 프로젝트를 시작하는 것이고, 두 번째 선택은 직접 기존 프로젝트에 기능을 추가하는 방법이다. 이를 프로젝트 가지치기(Branching)라 하며, 공개 소스 소프트웨어의 경우, 원 소스를 바탕으로 수정되거나, 추가된 소스를 모두 공개한다면, 소스의 사용과 배포가 자유롭기 때문에 아무런 법률적 문제없이 이런 결정을 내릴 수 있다.

□ 새로운 프로젝트 개발

새로운 공개 소스 소프트웨어 프로젝트의 시작은 요구사항을 만족하는 기존 프로젝트를 발견하지 못한 경우나 기존의 유사 프로젝트에 새로운 기능에 대한 수용 요구가 받아들여지지 않은 경우 등에 내리는 최후의 선택이라고 볼 수 있다.

새로운 프로젝트가 시작 되면, 상용 소프트웨어의 개발과 유사한 과정을 거치 게 된다. 이 과정은 같은 동기와 목표 의식을 가진 핵 심 개발자들로 개발팀을 구성하고, 요구 분석을 더욱 견고하게 한 뒤, 각종 위험 요인 분석, 일정 만들기 등의 절차적인 작업들로 시작된다.

그 가운데 위험 요인분석에는, 이 새로운 프로젝트가 기존 프로젝트들 에 비하여 경쟁력을 가질 수 있는지, 개발과 추후 관리를 위한 충분한 자원자를 확보할 수 있는지, 개발에 필요한 장비가 확보 가능한지 등을 포함한다.

많은 프로젝트의 경우, 표준적인 PC들로 개발 환경을 어렵지 않게 구축할 수 있으며, 소프트웨어 개발 도구 역시 공개 소스 소프트웨어를 사용하기 때문에 큰 비용을 유발하지 않으며, 여러 개발자들의 분산 개발을 지원하기 위한, 버전 관리 시스템(소스 저장소), 메일링 리스트 등도 개인PC를 이용하여 구축할 수 있다.

최근에는 다수의 오픈 소스 프로젝트들의 결과물이 마이크로소프트의 윈도우즈 계열 운영체제를 위해서 개 발되고 있다. cygwin과 같은 운영체제 적응 계층을 바탕으로 하는 경우에는 별다른 문제가 없지만, 윈도우즈 운영체제를 직접 지원하는 경우, 도구의 문제가 따른다.

프로젝트의 시작 단계에서부터 소스의 관리, 버그 의 관리, 개발자들 간의 원활한 의사소통을 위하여 SourceForge.net과 같은 공개 소스 소프트웨어 개발자 사이트를 이용할 수 있지만, 많은 초기의 프로젝트의 경우, 프로젝트의 시작 동기, 요구 사항, 설계 등이 기술된 공식적 문서의 부족이나, 다운로드가 가능한 소프트웨어 릴리즈가 없다는 이유로, 개발자 지원 사이트에 등록된다. 해도, 커뮤니티 형성 등의 파급

효과를 기대하기는 어렵다.

□ 개발자간 소통

개발자간 소통 방법에는 커뮤니티 참여자의 성격(개발자, 관리자, 사용 자 등)에 따른 메일링 리스트, 포럼과 그들의 아카이브, 버그 리포트, 프로젝트 관련 문서, FAQ 등이 있다. 버그 리포트는 사용자와 개발자의 공식적인 통신 방법으로, 오픈 소스 소프트웨어 개발자 사이트들이 공통으로 제공하는 버그 트래킹 시스템을 이용한다. 별도의 홈페이지를 이용하여 프로젝트를 공개한다면, Bugzilla와 같은 버그 트래킹 시스템을 설치하여 이용할 수 있다.

버그 트래킹 시스템은 버그를 등록 하고, 누가 그것을 담당하여 수정할 지를 할당하고, 현재 처리 상태는 어떤지, 그리고 그 버그에 대한 의 견 교환을 하는 등, 버그의 발생부터 해결까지의 전 과정에 대한 체계적인 관리 방법을 제공한다.

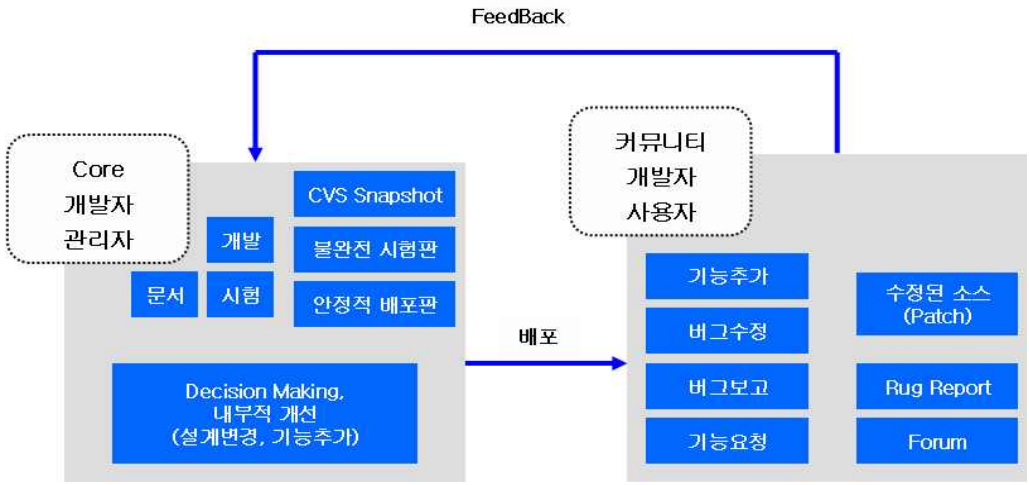
소스 코드의 경우 안정된 배포 버전, 흔히 베타 버전라고 하는 외부용 시험 버전, 그리고 현재 개발이 진행 중인 소스 코드 스냅샷, 세 가지를 모두 공개하는 것이 보통이다. 소스 코드 스냅샷 공개는 개발자들 이 소스의 버전 관리를 위하여 사용하고 있는 소스 코드 버전 관리 서버에 로그인하여 소스에 읽을 수 있도록 하는 것이다.

소스 코드의 공개는 오픈 소스소프트웨어의 가장 큰 미덕으로 누구나 쉽게 다운로드, 리뷰, 빌드를 할 수 있도록 하고, 궁극적으로 패치를 만들어 프로젝트관리자에게 보낼 수 있어야 한다.

많은 오픈 소스프로젝트들은 홈페이지 또는 배포된 소스에 프로젝트기여자들을 나열함으로써, 그들의 기여에 감사하고, 동시에 그 목록에 없는 개발자, 사용자들에게 동기를 부여한다.

□ 커뮤니티 기반 개발

일단 프로젝트의 프로토타입, 소스가 공개되어, 점차 알려지고, 사용자, 자원 개발자 등의 관심을 끌어, 커뮤니티가 형성되면 오픈소스 순환 구조에 의해 프로젝트는 진화한다. 공개에 앞서 결정된 의사 결정 구조 등에 의거하여 수정 버전의 릴리즈, 기능이 대폭 보강된 버전업 등이 이루어지며, 사 용 중에 발생하는 버그의 리포팅 및 수정, 기능 추가 요구 등이 커뮤니티 측에서도 이루어지며, 그 결과가 프로젝트 관리자 그룹에 의해 반영된다.



[그림3-3. 공개SW 순환 구조]

이 순환 구조가 얼마나 원활하게 운영되는지가 결국 오픈 소스 소프트웨어 프로젝트의 성패를 결정한 다. 이 구조의 원활한 순환과 효율성은 모든 참여자 사이의 의사소통 및 의사 결정 과정 등, 배포 전에 내려진 여러 결정에 의해 좌우된다. 오픈 소스 소프트웨어가 커뮤니티에 의해 진화한다는 일반적인 인식에도 불구하고, 그 프로젝트를 처음 시작하고, 많은 경우 결국 관리하게 되는 코어 개발자 그룹의 지속적인 관심과 개선 의지는 프로젝트 성공의 가장 중요한 요인이다.

다. 공개SW 적용 방안

(1) 리눅스가 일반적으로 선택되고 있는 환경

대표적인 공개소프트웨어인 리눅스는 웹서버, 메일서버 등 인프라서 서버와 1way ~ 4way 엔트리급 서버에서는 이미 기술적 안정성을 입증 받고 있으며, 리눅스 커널 2.6 출시 이후 DB서버, AP서버 등 기관 핵심 업무와 8 way 이상의 엔터프라이즈급 서버 운영체제로 폭넓게 활용되고 있는 추세이며, 특히 리눅스가 일반적으로 선택되고 있는 환경은 다음과 같다.

① 네트워크 인프라 : 도메인 네임 서버 (DNS), IP 주소 할당(DHCP), 웹 서비스, 어플리케이션 서비스, 프록시 서버, 디렉터리 패킷 쉐이핑 및 통신최적화를 위한 소프트웨어 포함

② 데이터베이스 서버 : 탁월한 오픈 소스 데이터베이스 서버로는 MySQL, Postgre SQL, MaxDB Firebird SQL(이전에는 Interbase), Ingres(이전에는 Adabas)가 있으며, 다수의 비공개 데이터베이스 서버도 리눅스에서 이용 가능

③ 보안 시스템: 방화벽, 침입 탐지 및 분석, 허니포트(honeypots), IPSEC 및 기타 가상 사설망(VPN) 시스템, 패킷스니핑 (packet-sniffing) 및 분석, 바이러스 백신 소프트웨어 및 스팸 방지 필터링 포함

④ 인터넷 및 인트라넷 출판: 웹 서버, 콘텐츠 관리 시스템(CMS) 플랫폼 및 워크플로우 관리 도구 포함

⑤ 문서 관리: 자동 전자 문서 캡처 시스템, 개정 관리 시스템, 데이터 캡처 기술, 문서보관 시스템 포함

⑥ 이메일 및 통신: 이메일, 일반 그룹웨어(그룹 캘린더 관리, 주소록 공유, 알리미 서비스, 공유 폴더) 및 인스턴트 메시지 교환 서버를 위한 다수의 솔루션 포함

⑦ 어플리케이션 서버: Java, PHP, Perl, Python 및 ZOPE 스크립트 툴, (Java 및) Java 2 Enterprise Edition(J2EE) 서버(예: JB공개SW), dotGNU .NET 공개소프트웨어 어플리케이션 서버에 기반한 광범위하게 사용되고 있는 웹 어플리케이션 서버 포함. 또한, 현재 다수의 비공개 어플리케이션 서버가 리눅스에서 이용 가능

⑧ 파일 및 프린트 서버: Unix NFS, Microsoft SMB/CIFS 및 Novell Netware NCP와 같은 대부분의 주요 파일 공유 프로토콜을 포함하는 툴

⑨ 저장 : 몇몇 네트워크에 부착된 저장 장치들은 주로 공개소프트웨어 플랫폼에 구축되어 있음

⑩ 제한된 기능의 워크스테이션 : 기본 웹, 이메일, 터미널 액세스 그리고 통화센터, 키오스크 및 이와 유사한 용도를 위한 사무실 생산성 관련기능을 제공하는 고정 용도의 워크스테이션

⑪ 고성능 컴퓨터: 여기에는 다수의 마이크로프로세서 (수직 평가), 다수의 저비용 시스템(수평 평가) 및 기타 유형의 슈퍼컴퓨터에 기반 한 클러스터를 장착한 단일 영상 시스템 포함

⑫ 고성능 기술 워크스테이션: 과학적 분석, 계량, 모델링, 3D 컴퓨터로 생성된 이미지 (CGI) 및 비디오 처리 기능과 같은 컴퓨터의 처리 기능을 강화한 어플리케이션용 멀티프로세서, 64-비트 및 대용량 메모리 시스템

(2) 3 Tier 기반의 구체적 적용 방안

□ 웹 서버

○ 개요

웹 기반 환경이 복잡해지면서 상대적으로 웹서버의 비중이 감소하는 추세이다. 현재의 웹 서버는 이미지 관리나 html의 출력만을 담당하며 transaction, clustering, fail-over 등의 기능은 Web Application Server(이하 WAS)를 통해 관리하고 있다. 대표적인 공개 SW 웹서버인 아파치의 경우 전 세계적으로 68% 이상의 점유율을 기록하고 있으며 성능 및 안정성에 대한 검증이 이루어졌다고 볼 수 있다.

○ 아파치 적용을 위한 고려사항

아래의 항목에 해당하지 않을 경우 공개SW 웹서버인 아파치를 적용할 것을 권고한다.

- 특정 웹 어플리케이션 개발 언어에 종속적이어야 함(예:ASP)
- 동시 접속자 수 최대 100,000 명을 이상 필요
- 웹 서버 단독의 TP Monitoring 기능이 필요함
- 웹 서버 단독의 SW를 이용한 HA 구성이 필요함

○ 웹 서버 적용을 위한 HW

웹 서버의 부하가 감소하는 추세이며, 단일 서비스 제공이라는 측면에서 볼 때 소형 혹은 중형 급 HW의 사용이 적합하다. 시스템 자원의 요구량이 증가하면 L4 스위치 등의 HW 사용으로 자원 분배 및 가용성의 확보가 가능하며, 필요에 따라 SW적으로 클러스터링을 구성한다.

○ 아파치 License

- 아파치는 아파치 License를 따르며, 이는 GPL license와 호환된다.

□ 웹 어플리케이션 서버(이하 WAS)

○ 개요

웹 기반의 3Tier 환경에서 웹 어플리케이션 서버의 비중이 점차 증가 추세이다. 현재 대표적인 공개SW WAS로 JB공개SW가 있으며, 구축하고자 하는 사이트의 규모에 따라 Tomcat 등의 제품으로 대체 구성이 가능하며. 구축 방법 및 특성에 따라 다양한 공개 SW를 이용 가능하다.

○ 공개SW WAS 적용을 위한 고려사항

아래와 같은 항목에 해당되지 않는다면 공개SW WAS의 적용을 권고한다.

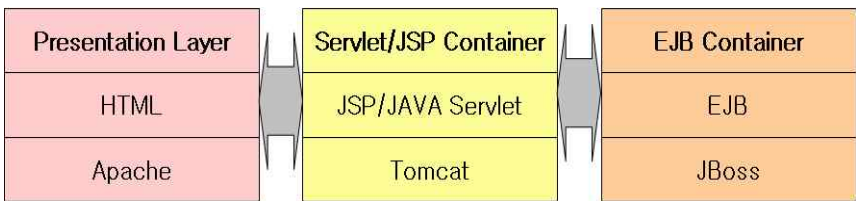
- 특정 운영체제 상에서만 운용해야만 함
- 특정 웹서버에 종속되어야 함
- J2EE 표준 스펙 이외의 기능이 필요함
- 특정 제품에 종속적인 코드의 사용이 필요함
- 시스템 전체 동시 사용자수가 최대 50,000 명을 넘음
- 특정 개발 툴이 필요한 환경임

□ 리눅스 기반의 공개SW WAS 적용방안

○ 개요

리눅스 기반 환경에서 공개SW WAS의 적용 방안에 대한 기준을 제시한다.

- 공개 SW로 웹 기반의 환경을 구축하는 경우 웹 어플리케이션 서버로 Tomcat, JB공개SW를 이용 권고
- 웹서버로 아파치, 어플리케이션 서버로 Tomcat을 사용하여 구축하며 EJB를 사용하는 경우 JB공개SW 추가 사용 권장
- 최근에 EJB에 대한 논란이 발생하면서 JB공개SW의 EJB 컨테이너를 대체하기 위해 Spring Framework를 사용
- 본 문서에서는 아파치와 Tomcat을 사용하여 웹 기반 환경의 시스템을 구축하는 방법을 제시하며, EJB 사용을 위해 JB공개SW 사용 방법을 제시함



[그림3-3. 공개 SW 적용 시스템]

□ Tomcat

○ 개요

아파치 소프트웨어 재단의 자카르타 프로젝트의 일환으로 개발된 소프트웨어이다. 독립적으로 웹서버의 기능을 수행할 수 있지만, 보통의 경우 성능 향상을 위해 아파치와 연동을 통해 서비스를 제공한다.

○ Tomcat의 License

- Tomcat의 경우 아파치 License를 따름

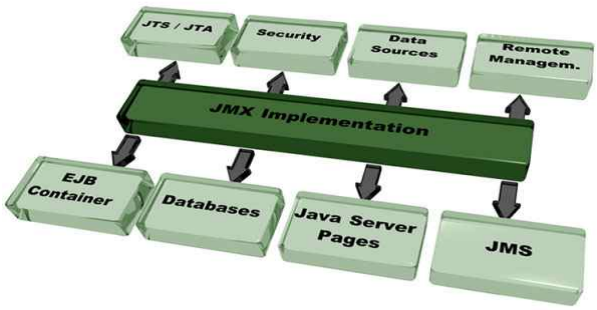
○ 기타

- Tomcat의 설치 및 설정에 대한 상세한 내용은 별첨으로 제공되는 문서를 참고

□ JB공개SW

○ 개요

JB공개SW의 경우 http://www.jb공개SW.org을 통해 공개 SW로 개발되고 있으며 자유롭게 사용이 가능하다. 최근의 SDTimes의 조사에 따르면 JB공개SW의 점유율이 2002년 15%에서 2003년 25%로 급성장 되고 있는 것을 알 수 있다.



[그림3-4. JB공개SW Architecture]

○ 특징

- JB공개SW는 유일하게 Microkernel-based Architecture를 사용하고 있으며, 이를 통해 향상된 class-loading이 가능함
- Service-Oriented Architecture의 이용으로 쉽게 서비스가 추가/삭제 될 수 있으며, 모든 서비스 들이 패키지화가 될 수 있으며 hot-deploy의 특징이 있음
- AOP(Aspect-Oriented-Programming)의 특징을 가짐
- J2EE 1.4의 표준을 따름(JB공개SW 4.0 기준)
- JDBC 2.0, JCA 1.0, Servelets/JSP 2.3 (Jetty + Tomcat)

- JNDI : Java Naming and Delivery Interface
- EJB 2.0, full CMP 2.0 Engine : Container Managed Persistence
- JMS 1.1 : Java Messaging Service
- JTS/JTA : Java Transaction Service/Java Transaction API
- JavaMail,Clustering

○ JB공개SW의 License
- JB공개SW는 LGPL을 따름

□ WAS 적용을 위한 HW
- 웹서버의 기능이 줄어드는 반면 WAS의 기능이 점차 증가하는 추세임
- 따라서 WAS에 적용 가능한 HW로는 규모에 따라 소형, 중형 급 서버가 적합하다고 판단됨
- 시스템 자원의 요구량이 증가하면 자체 클러스터링 기능을 이용하여 확장이 가능함

□ 데이터베이스 서버(이하 DB 서버)
○ 개요
리눅스 기반에서 공개 및 상용 Database 서버의 구축 방법에 대해 기술한다. 사이트 규모나 특성에 따라 공개 및 상용 Database 서버를 구축하는 방법에 대해 기술한다.
- 공개SW의 경우 가장 많이 사용되고 있는 MySQL을 기준으로 설명
- 트랜젝션 처리를 강화하기 위해 InnoDB와 병행하여 사용하며, SAP를 사용하는 경우 MAX DB를 사용 권고

○ 공개SW DBMS 적용을 위한 고려사항
아래의 항목을 해당하지 않는다면 공개SW 데이터베이스서버인 MySQL의 적용을 권고함
- 특정 운영체제 상에서만 운용이 가능해야 함
- 이미 구축되어 있는 DBMS를 사용해야 함
- 멀티미디어(동영상/오디오/그래픽)파일을 제외한 예상 데이터베이스크기가 1TB를 초과함
- 시스템 전체 동시 사용자수가 최대 10,000 명을 초과함
- 최대 테이블 ROW의 길이가 64KB 이상이 필요함
- 단일 쿼리 크기가 1MB 이상임
- index의 길이가 1KB를 초과함
- 테이블 당 최대 32개 이상의 index가 필요함

- 테이블 당 3,000개 이상의 컬럼이 필요함
- XML Database가 필요함
- Bitmap Indexing 기능이 필요함
- Partitioning이 필요함
- 이기종 간의 데이터 연동이 필요함

□ 리눅스 기반의 공개SW DBMS 적용지침

○ 개요

- 대표적 공개 SW Database 서버임
- 현재 출시된 4.X대를 기준으로 살펴보면 상용 SW DBMS에 비해 기능적으로 부족하지만 이는 5.X대 버전 출시 이후(내년 상반기내) 해결될 것으로 전망됨
 - 이와 같은 이유로, 현재 이미 다른DBMS로 구축되어있는 사이트의 경우 MySQL로의 전환은 추천하지 않으며, 신규로 구축되는 경우에 한해서 MySQL의 적용할 것을 권고함

○ MySQL의 License

- MySQL은 Dual Licese 정책을 따름(GPL License와 상용 License를 병행)

□ 구축 시스템 대상 규모에 따른 적용 지침

○ 중소 규모 시스템에 적용

- DBMS의 기본기능에 충실하며 최고의 'select' 속도를 보이는 MySQL의 적용이 가장 적합할 것임
- 별도의 수정을 하지 않은 MySQL은 동시접속자 500명의 시스템에 적합하도록 컴파일 되어 있음
- 이를 넘어서는 규모의 사이트에 구축하는 경우 아래의 작업을 수행하거나, clustering 기능을 이용하여 시스템 확장을 고려
- 500이상의 connection 처리를 위한 패치
 · glibc를 rebuild해야 하며, 재 컴파일 된 libpthread.a를 링크하여 MySQL을 rebuild 해야 함
 · 자세한 사항은 http://www.mysql.com/doc/L/i/Linux.html을 참고

○ 대규모 시스템에 적용

- MySQL Clustering을 이용하거나 별도의 3rd party Clustering 제품 사용 권장

□ DBMS 적용을 위한 HW

- 시스템 규모가 증가함에 따라 처리하는 데이터의 양도 증가하고 있는 추세임
- 따라서 DBMS 적용을 위한 HW로는 중, 대형 급 서버가 적합하다고 판단e됨
- 대규모의 사이트에서 요구량이 증가하면 DBMS 자체 클러스터링 기능을 이용하여 확장 가능
- MySQL의 경우 내부적으로 64bit 정수를 많이 사용하고 있으므로 64bit CPU를 사용하면 좀 더 나은 퍼포먼스를 기대할 수 있다. 2GB가 넘는 큰 테이블이 필요하다면, AMD Opteron 등의 64bit HW를 고려 권장

□ MySQL의 Replication 기능

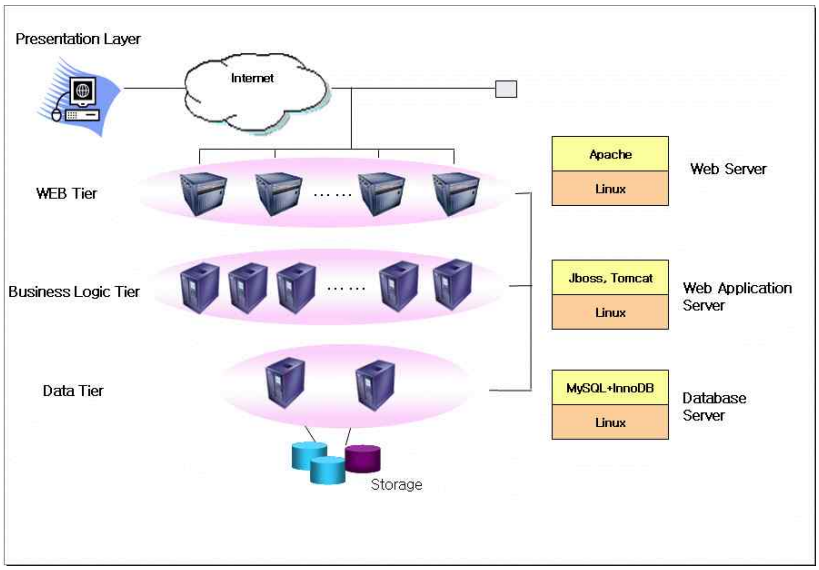
- MySQL에서는 Replication을 위해 Master와 Slave라는 말을 사용함
- Master는 실제로 업데이트가 발생하고, 동기화를 할 서버의 기준이 되는 주체를 말함
- Master와 Slave의 상세한 설정 방법은 아래 사이트 참고
  (http://dev.mysql.com/doc/mysql/en/Replication.html)

(3) 공개SW 도입 유형(예시)

다음에 제시하는 3 Tier 기반의 웹 정보 서비스 시스템 모델의 다양한 공개소프트웨어 전환 시나리오를 참고하여, 추후 정보시스템의 공개소프트웨어 전환 시 적합한 유형을 선택하여 적용 할 수 있다.

이 시스템이 다음과 같은 조건들을 가지고 있다고 가정

- 웹 기반의 업무 환경은 웹서버, 웹 어플리케이션 서버, DB서버의 3계층구조를 기본으로 하며, 각각의 계층은 웹 브라우저/웹서버, 웹 어플리케이션 서버, 데이터베이스 서버의 순서로 처리 한다.

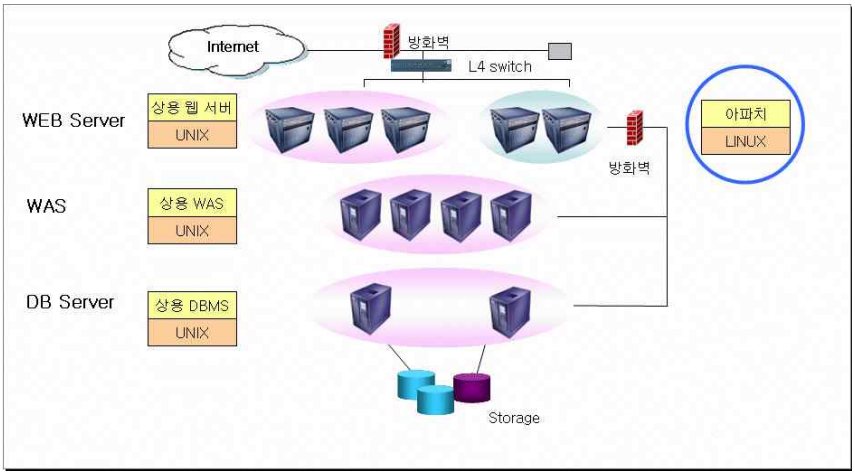


[그림3-5. 3 Tier 기반의 웹 정보 서비스 시스템 구성도 1]

이 시스템을 공개소프트웨어로 전환하는 방법은 다음의 3가지 유형으로 적용 가능하다.

□ 웹서버 추가

- '소극적 공개SW 도입' 유형으로 기존 시스템을 변경하지 않고 시스템만 추가하여 공개SW 환경을 일부 구축
- 일반적으로 공개SW 처음 도입할 경우에 사용
- 본 시나리오와 같은 모델을 우선 취하여 테스트 후 실제로 사용하고 있는 시스템들을 전환하여 업무에 적용한다.



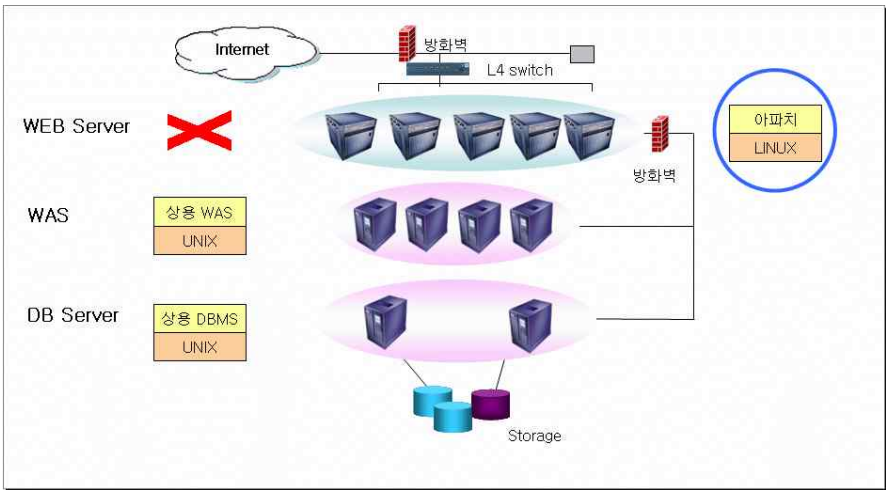
[그림3-6. 3 Tier 기반의 웹 정보 서비스 시스템 구성도 2]

○ 특징

- 그림과 같이 기존의 Unix 계열의 OS와 상용SW를 이용하여 구축된 사이트에 추가적으로 공개SW OS와 공개SW 웹서버를 도입하는 경우
- 기존에 구축된 환경의 내용 연한이 지나지 않았으나 시스템 확장이 필요한 경우도 해당됨
- 현재 시스템에 영향을 미치지 않고 공개SW 기반 시스템 추가 도입 시 권고

□ 웹서버 전환

- 시나리오1의 과정을 거친 후 실질적인 전환 구축을 하는 단계
- 우선 구축이 가장 쉬운 웹서버부터 전환



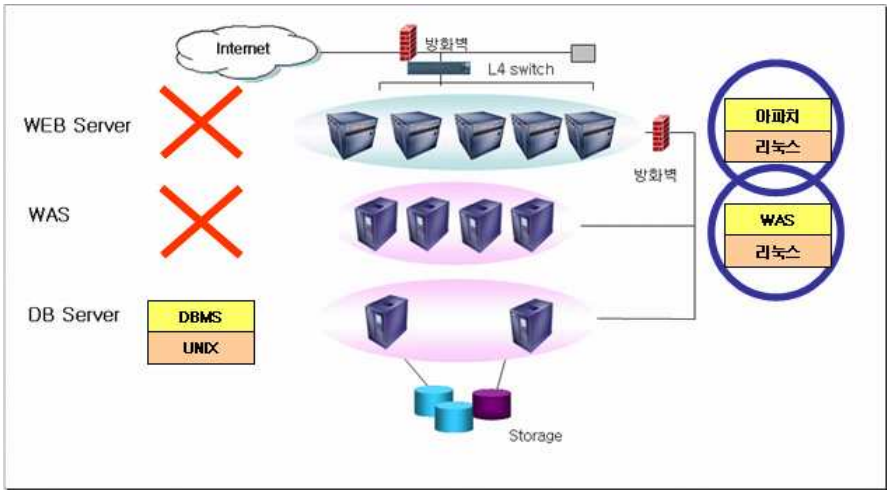
[그림3-7. 3 Tier 기반의 웹 정보 서비스 시스템 구성도 3]

○ 특징

- 기존 3계층으로 구성된 환경에서 웹 서버 계층을 공개SW로 완전히 대체하는 경우
- 웹 서버에 공개SW를 도입하는 경우는 단순히 서버만을 교체하는 수준의 작업이 요구되므로 전환용이

□ 응용프로그램 서버 전환

시나리오2의 구축이 성공적으로 끝나면 이제 바로 윗 단에 있는 응용프로그램서버를 전환하도록 함.



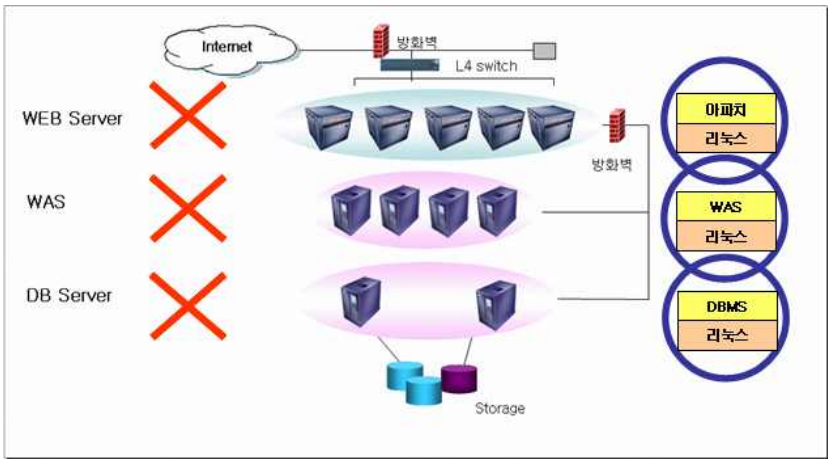
[그림3-8. 3 Tier 기반의 웹 정보 서비스 시스템 구성도 4]

○ 특징

- 기존의 3계층으로 구성된 환경에서 웹 서버 계층에 공개SW를 도입하며, WAS 서버의 운영체제를 리눅스로 전환 하는 경우
- WAS의 운영체제만을 교체하는 작업으로 비교적 전환이 용이
- 업무유형별 웹 어플리케이션의 전환 작업 필요

□ 응용프로그램 전체 전환 및 DB 서버 전환

시나리오3에서 전환하지 못한 복잡한 응용프로그램 서버 및 DB 서버를 다음과 같이 전환하여 시스템 전체 전환 시나리오를 완료



[그림3-9. 3 Tier 기반의 웹 정보 서비스 시스템 구성도 5]

○ 특징

- 기존의 3계층으로 구성된 환경에서 웹 서버 계층에 공개SW를 도입하며, WAS 서버의 운영체제를 리눅스로 전환 하는 경우
- WAS와 DB 서버의 운영체제만을 교체하는 경우로 비교적 전환 용이
- 운영체제 교체에 따른 유지보수 비용과 초기 도입 비용을 절감 가능

(4) 공개SW 역량프라자 기술 지원

○ LSB 인증

공개SW 역량프라자 공개SW LSB 인증 지원을 통한 신뢰성 있는 공개SW 활용을 제시하며, 국내 공개SW LSB 인증 테스트 도구 및 활용 정보를 제공하고, LSB 인증을 획득하도록 지원한다.

- 공개SW LSB 인증 지원 방안 제시
- LSB Tool 활용 정보 제공 및 LSB 인증 절차 안내
- 공개SW 라이선스 검증 도구 및 공개SW 테스트 도구 지원 방안제시
- 공개SW LSB 인증 지원 수요처(기관 및 기업) 확보 방안제시
- 대표적인 공개SW의 LSB 인증 연계 방안제시

○ 공개SW 적용 Test

공개SW의 안정성과 신뢰성 인식의 변화를 위하여 공개SW의 기능, 성능, 정합성 등을 테스트하여 공개SW의 적용 효과를 높일 수 있도록 한다.

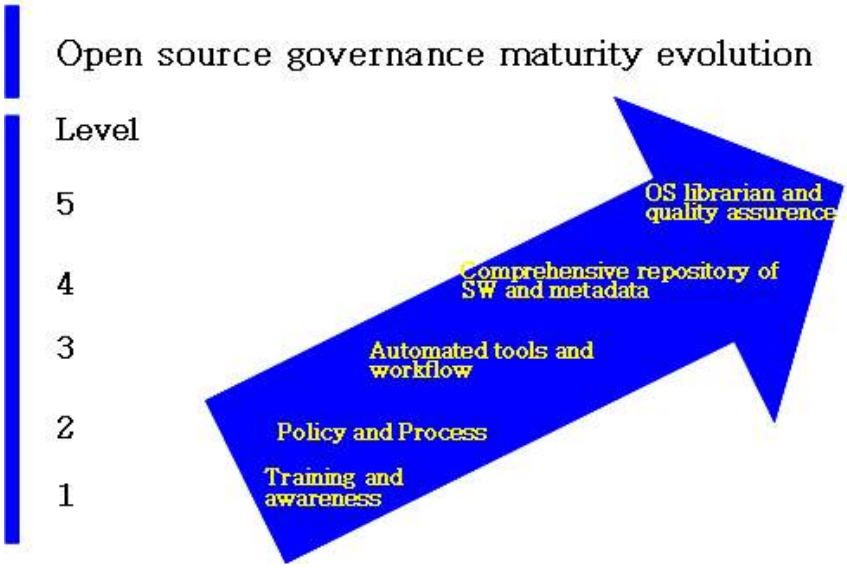
- 공개SW 기능·성능·정합성 테스트
- 공개SW 기반 시스템 모의 테스트 및 검증
- 정보화 심의의 공개SW 도입 예정 시스템의 안정성 테스트

4. 정책 및 가이드 영역

가. 공개SW Governance 현황 진단

공개SW Governance 전략을 위한 첫 번째 단계는 해당 조직의 공개SW 관련 활동에 대한 전반을 진단하는 것이며, 통상 조직 내의 공개SW 사용에 대하여 포괄적으로 조사하는 것을 포함한다.

사용하는 공개SW가 기관이나 조직 내부용 IT 인프라인지, 고객들에게 배포되고 있는 것인지, 오픈소스 프로젝트의 일부인지 또는 다른 제품에 임베드 된 것인지 등에 대한 전반적인 조사를 통해 조직 내의 공개SW 활용에 관한 정보를 확인하고 수집할 필요가 있다. 아울러 사용하는 공개SW에 통합 되거나 함께 배포되고 있는 제3자 제품에서의 공개SW 사용여부도 체크할 필요가 있으며, 그 과정에서 공급사와도 협력하는 것이 중요하다. 이와 같은 분석을 토대로 해당 조직의 공개SW Governance 성숙도를 평가해볼 필요도 있다.



[그림4-1. 공개SW Governance 성숙도 모델(HP)]

나. 공개SW Governance 정책수립(가이드 기본원칙)

비즈니스모델을 포함한 기관이나 조직의 전략을 수립한 후에는, 공개SW와 관련된 기관이나 조직의 정책(Policy)를 구체적으로 결정하게 되는데, 이 경우 기관이나 조직의 목표를 명확히 선언하고, 공개SW 정책을 기관이나 조직의 목표와 조화시킬 필요가 있다. 공개SW 정책에서는 공개SW 코드의 사용, 생성, 릴리즈를 위한 공식적인 프로세스를 정의하고, 기관이나 조직 내부 및 외부 사용자를 위한 교육프로그램을 만들 필요가 있으며, 관련자들로 하여금 규정된 절차를 준수하게 하고, 아울러 프로세스를 주기적으로 체크하고 조정할 필요가 있다.

(1) 공개SW의 선택과 획득

중요한 정책결정요소중의 하나는 어떠한 공개SW를 선택할 것인가의 문제이다. 이 경우 일반적인 상용 소프트웨어처럼 공개SW의 품질이 중요하다. 아울러 공개SW는 관련 라이선스 조건, 커뮤니티의 크기 · 역사 · 활동수준, 현재 및 장래의 소송가능성을 포함한 법적 상황 등도 고려하여야 한다.

필요한 공개SW를 선택한 이후에는 해당 공개SW를 누가 어떤 경로를 통해 획득할 것인지를 결정해야 한다. 획득절차의 주체와 관련해서는 정보화 담당자 책임 하에 할 수 있으며, 구매/공급망 관리부서 등을 통해 할 수도 있다. 그리고 공개SW는 SourceForge, FreshMeat, Krugle, Google, OSDir 등 인터넷상 어디에서든 확보가능하기 때문에 기관이나 조직의 입장에서는 커뮤니티 소스로부터 직접 확보할 수 있다. 그러나 최근 공개SW와 관련된 특허침해소송 등을 고려한다면 한글과 컴퓨터, IBM, Montavista, Red Hat, SKC&C등 제3의 공급자로부터 확보하여, 필요한 정도의 보증(Warranty)이나 면책(Indemnification)을 받아 두는 것도 고려할 사항이다.

Open Logic 사례 인용

- 나의 조직은 새로운 공개SW를 조사하고 평가하는 공식적인프로세스를 가지고 있다.
- 나의 조직은 공개SW의 기술적 평가를 수행하는 표준기준(criteria)을 가지고 있다.
- 나의 조직은 공개SW와 관련된 커뮤니티의 크기, 지속가능성 등을 평가하는 표준기준(criteria)을 가지고 있다.
- 나의 조직은 새로운 공개SW 패키지에 대한 공식적인 승인 프로세스를 가지고 있다.
- 나의 조직은 “승인” 공개SW 패키지 리스트를 관리하고 있다.

2) 공개SW 사용현황 관리

기관이나 조직의 입장에서는 현재 사용하고 있는 공개SW의 현황을 파악하는 것도 중요한 일이다. 이와 관련하여 기관이나 조직 내부에서의 공개SW를 어떻게 배포하고 관리할지에 대한 정책결정이 필요하다.

예를 들어 관련된 정보화 담당자 모두에게 관리할 수 있도록 하거나, 기관이나 조직내부의 모든 공개SW를 한곳에 통합하여 관리하고 이를 통해서만 사용하도록 제한할 수도 있다. 즉 특정한 데이터베이스를 통해 조직 내에서 사용되고 있는 공개SW의 inventory를 만들고, 사용 중인 공개SW를 추적 및 관리하며, 다수의 버전들을 관리할 수 있다.

아울러 기관이나 조직 내에서 누가 공개SW 컴포넌트의 최초 획득, 라이프사이클 관리를 책임질 것인가를 결정하는 것도 중요하다. 예를 들어 개별적인 최종 사용자 모두에게 책임지도록 하거나, 또는 각각의 공개SW 컴포넌트에 지정된 책임자를 둘 수 있으며, OSRB 등의 중앙 부서/팀에서 관리할 수도 있을 것이다.

Open Logic 사례 인용

- 나의 조직은 기술 공급자가 공개SW를 설치, 설정, 통합하는데 얼마의 시간이 필요한지를 이해하고 있다.
- 나의 조직은 승인된 공개SW를 위한 중앙 Repository를 가지고 있다.
- 나의 조직은 오픈소스 커뮤니티 버전 업데이트에 대한 모니터링, 업데이트 버전의 품질에 대한 평가, 적절한 업데이트버전의 획득을 위한 공식적인 절차를 가지고 있다.
- 나의 조직은 전사에서 공통적으로 사용되는 공개SW 스택을 공유하는 공식적인 절차를 가지고 있다.
- 나의 조직은 기업에 설치된 공개SW에 대한 지식을 공유하는 공식적인 절차를 가지고 있다.

(3) 공개SW의 지원(Support)

일반적인 상용소프트웨어와 마찬가지로 공개SW에 대한 지원(Support)도 중요한 일이다. 기관이나 조직의 입장에서는 커뮤니티와의 관계를 유지하면서 직접 지원을 받을 수도 있으며, 상용기업이나 SI업체를 통한 지원을 받을 수 있다.

Open Logic 사례 인용

- 나의 조직은 공개SW의 기술지원에 대한 공식적인 절차를 가지고 있다.
- 나의 조직은 하나이상의 오픈소스 기술지원 공급자들과 컴포넌트 레벨의 지원 계약을 체결하였다.
- 나의 조직의 정보화 담당자는 기술지원문제들을 해결하기 위해 오픈소스 커뮤니티 메일링 리스트를 이용하고 있다.
- 나의 조직의 경영자들은 오픈소스 커뮤니티 메일링 리스트를 통해 회사의 비밀정보가 공개되는 문제에 대해 관심을 가지고 있다.
- 나의 조직은 커뮤니티 메일링 리스트 및 오픈소스 지원 공급자등 외부지원 자원들과 기술지원 문제를 해결하는 공식적인 정책을 가지고 있다.
- 나의 조직은 공개SW와 관련된 제품버전의 분기(forking)를 피하기 위해 소스코드를 변경시키는 공식적인 절차를 가지고 있다.

(4) 공개SW 및 관련 프로젝트에 대한 추적

공개SW 관련 커뮤니티를 기반으로 계속해서 변화해간다. 따라서 기업의 입장에서는 해당 공개SW 및 프로젝트의 커뮤니티 상황을 지속적으로 추적할 필요가 있다. 이를 통해 취약성 및 치명적인 결함이 드러나지는 않는지, 프로젝트의 건전성 및 로드맵은 적절한지, 혹시 주요 기여자들이 떠나고 있지 않은가에 대해 파악할 필요가 있다. 예를 들어 관련 프로젝트가 분기되거나 포기되는 경우에 기업은 어떻게 대처할 것인가, 벤더/공급자의 대안이 있는가 등에 대한 대책이 필요하다.

(5) 조직원들의 공개SW 기여에 대한 정책 및 가이드

공개SW를 적극적으로 활용하고 있는 기관이나 조직들에서는 많은 개발자들이 공개SW 커뮤니티와 직간접적으로 관계를 맺고 있으며, 필요한 경우 개발결과물의 일정부분을 기여(contribution)하고 있다. 그런데 기업은 중요한 지적자산을 관리해야 하기 때문에, 이와 같은 종업원들의 기여에 대한 정책을 마련할 필요가 있다. 예컨대 개발자들이 공개SW 프로젝트에 관계하고 있는 내용을 제출하거나, 기여를 하기 위해서는 승인을 받으려고 하는 프로세스를 마련할 수 있다. 그리고 이러한 정책내용을 문서화하고 교육할 필요가 있다.

정책결정시 중요한 요소들로는, 누가 저작권을 가지고 있는가, 기관이나 조직의 정보와 영업비밀의 보호, 이해관계의 상충, 공개SW 커뮤니티와의 경쟁, 조직의 시간

과 조직원 개인 시간의 배분 등이다. 특히 커뮤니티와의 관계를 어떻게 가져갈 것인가가 중요한데, 커뮤니티에 충분한 기여를 하지 못할 경우 자칫 커뮤니티로부터 신뢰를 잃을 수 있다.

(6) 라이선스의 관리

공개SW 라이선스와 관련하여 기관이나 조직은 어떠한 라이선스들의 사용을 허용할 것인가를 결정할 필요가 있다. 예를 들어 모든 오픈소스 라이선스 또는 OSI 인증 라이선스를 폭넓게 인정할 수 있으며, 이와는 달리 상호주의(Copyleft) 라이선스를 제외한 모든 라이선스를 허용하거나, 특정한 리스트에 포함된 라이선스로 제한할 수도 있다.

한편 조직 내에서 누가 또는 어느 부서에서 공개SW 라이선스 조건에 대한 준수의무를 이해하고 책임지게 할 것인가도 중요하다. 법률부서, 감사, 엔지니어링 매니저, 개별 개발자, IT 관리부서 등을 고려할 수 있을 것이다.

다. 유지보수 가이드라인

(1) 목적

유지보수 가이드라인은 공개SW의 특징을 고려하여 공개SW 유지보수 서비스의 범위 및 대가 산정방식을 명확히 하고 유지보수 서비스 구매 시 필요한 고려사항을 제공하여 공개SW의 활용을 높이고 효과적인 예산편성과 집행을 도모하는데 그 목적이 있다.

(2) 공개SW의 유지보수 특징

일반적으로 저작권자가 공개하는 소스 코드 형태의 공개SW는 비공개 패키지SW와 달리 그 자체로서는 사용자가 편리하게 이용하기 쉬운 형태라고 하기는 어렵다.

이에 따라, 개발자가 공개한 소스코드를 활용하기 쉬운 형태로 제공하거나 제품 설치, 제품오류 및 결함에 대한 수정, 업그레이드 등 제품의 활용을 위해 기술지원을 필요로 하는 고객 수요가 발생하여 공개SW 시장이 형성되고 이와 같은 기술지원 서비스(유지보수 서비스)를 전문으로 제공하는 공개SW 업체가 발달하고 있다.

즉 '공개SW=무로'라는 공식은 소스코드의 무료사용에 한정된 것이고, 공개SW 활용을 위해 공개SW 업체로부터 제공받는 유지보수서비스는 유상으로 제공되는 것이다.

(3) 적용범위

본 유지보수 가이드는 SW산업진흥법 제 19조의 국가, 지방자치단체, 국가 또는 지방자치단체가 투자하거나 출연한 법인 또는 기타 공공단체 등(이하 '발주기관'이라 함)의 공개SW의 유지보수 서비스 구매에 적용한다. 단, 소스 코드는 공개되지 않지만 SW를 무료로 제공하는 상용SW에 대한 기술지원 서비스(유지보수 서비스) 구매에도 준용할 수 있다.

(4) 공개SW 유지보수 서비스의 정의

가) "공개SW 유지보수 서비스"라 함은 공개SW를 최적의 상태로 활용, 유지하기 위한 기술지원 서비스다.

나) 공개SW 유지보수 서비스는 공개SW의 제품상 오류 및 결함의 수정(상용SW의 하자보수), 정보시스템 운영 시 발생하는 문제점, 기타 사용자 요구사항 해결(상용SW의 유지보수) 등을 포함한다.

(5) 공개SW 유지보수 서비스 내용

공개SW 유지보수 서비스에는 일반적으로 다음과 같은 항목들이 포함된다.

[표4-1. 공개SW 유지보수 서비스 내용]

항목	내용
설치	목적 소프트웨어를 고객이 원하는 시스템 환경에 옮기고 정상적으로 작동 되도록 하는 작업
패치제공	새로운 기술의 적용이나 운영체제의 변화 등으로 발생하는 불일치 조정
업데이트	목적 소프트웨어뿐만 아니라 이에 관련된 라이브러리, 도구, 인터페이스 등 기존 설치 환경을 최신 버전으로 갱신 시키는 작업
업그레이드	목적 소프트웨어의 버전을 향상시키는 작업으로 메이저 업그레이드와 마이너 업그레이드로 분류 예) 버전이 x.y.z 형식으로 표기된다면 x에 해당하는 숫자를 높이는 작업이 메이저 업그레이드이고 y나 z를 향상 시키는 작업이 마이너 업그레이드에 해당
최적화	시스템 또는 DBMS의 성능을 향상시키거나 저장장치를 효과적으로 사용하기 위해서 파일, 네트워크, 인덱스, 캐쉬, 버퍼에 관련된 파라미터를 변경하는 작업

튜닝	시스템 성능 향상을 위한 환경변수를 조정하는 작업
문제해결	목적 소프트웨어 자체의 문제(오동작, 에러, 버그, 해킹) 또는 운용 환경상의 문제(연동, Configuration) 등을 기술적으로 해결해 주는 작업
모니터링	설치된 소프트웨어의 실시간 운용 상황을 동적으로 관찰하여 통계적 자료를 제공하는 업무
온라인도움	포탈이나 이메일을 통해 질문이나 지원 요청을 접수하여 지식 베이스 또는 전담 기술 인력을 동원하여 접수된 요청사항을 온라인으로 지원해주는 업무
기술자문	마이그레이션, 커스터마이제이션, 백업 등 공개SW 서비스에 관련된 사항을 지도해 주는 업무
개발자지원	아키텍처링, 파라미터 구성, 성능 튜닝, 최적화 등의 개발 업무에 대해 조언하는 업무
보증	공급한 공개SW제품의 소스코드에 아무런 법적인 문제가 없도록 지원하는 업무

(6) 공개SW 유지보수 서비스 가이드라인

□ 기본원칙

가) 공개SW 유지보수 서비스는 공개SW 구입과 동시에 공개SW 업체로부터 유상으로 구매한다.

나) 공개SW는 제품가격 별도로 존재하지 않는 경우가 대부분이므로 공개SW 유지보수서비스는 요율제가 아닌 정액제 또는 콜 베이스(Call-Base)방식으로 비용을 지불한다.

- 정액제는 유지보수 서비스 수준에 따른 비용을 일정기간 단위로 지불하는 형태를 말한다.
- 콜 베이스제는 사용자의 서비스 요청에 따른 유지보수 서비스 제공 건수마다 비용을 지불하는 형태를 말한다.

□ 공개SW 유지보수 서비스 대가 산정방법 및 절차

- 발주기관은 '공개SW 유지보수 서비스 내용' 및 설치된 제품의 특성 또는 시스템 환경 등을 고려하여 필요한 공개SW 유지보수 서비스의 항목 및 서비스 수준을 결정한다.

- 공개SW의 규모, 정보시스템에 미치는 영향 등을 고려하여 정액제, 콜베이스제 등의 공개SW 유지보수 서비스 형태를 결정한다.
- 공개SW 유지보수 서비스의 대가는 공개SW 업체가 서비스 수준에 따라 책정한 금액을 기준으로 발주 기관과 업체가 협의하여 결정한다.
- 공개SW 유지보수 서비스의 대가에 대한 추정가격 산출은 2개 이상의 공개SW 업체가 제시하는 견적가격을 비교하거나, 투입인력, 기술료, 제경비 등의 분석이 가능한 경우 소프트웨어사업 대가의 기준(정통부 고시)의 '투입인력수와 기간(M/M)에 의한 산정방식'을 적용한다.

□ 고려사항

공개소프트웨어를 도입 방법, 절차 그리고 적용 가능한 시스템에 대하여 살펴보았으며, 공개소프트웨어를 도입할 경우 다양한 요구사항들이 도출되는 것을 확인했다. 따라서 이번 절에서는 공개소프트웨어를 도입함에 있어서 고려해야 할 사항들을 검토하면 다음과 같다.

첫째, 도입하고자 하는 공개소프트웨어에 대한 라이선스를 정확하게 확인하고 이해해야 한다. 공개소프트웨어로서 완전한 권리를 획득하기로 하였다면 해당 공개 소프트웨어는 소스코드를 모두 공개하고 해당권리를 사용자에게 양도해야 한다. 하지만 대형 프로젝트 등에서는 공급자가 모든 소스에 대한 권리가 없는 경우가 많으므로 소스의 일부는 비공개소프트웨어거나 타인의 저작권에 속하는 기술로 개발된 부분일 수 있으므로, 이러한 부문의 검토도 간과하지 말아야 한다.

둘째, 시스템 호환성 확보를 위해 개방표준을 지원하는 제품을 우선적으로 고려한다. 정보시스템을 설계할 때에는 유연한 상호호환성 확보를 위해 개방 표준을 지원하는 제품을 우선적으로 고려해야 한다. 모든 사람들이 읽어야 하는 자료를 웹을 통하여 제공할 경우에는 가능한 모든 정보통신 환경 사용자가 접근 가능하도록 제공하여야 한다.

셋째, 도입하는 공개소프트웨어에 대한 기술지원 및 유지보수에 대하여 고려해야 한다. 만약 간접공급 방식으로 공개소프트웨어를 도입하기로 결정하였다면 사용자 지침서, 온라인 튜토리얼, 교육훈련, 헬프 데스크 및 유지보수가 원활해야하며 또한 여러 플랫폼이나 사용 환경에 맞추어 버전관리가 잘 이루어질 수 있는지 확인해야

한다. 직접선택 방식으로 공개소프트웨어를 도입한다면 위 사항에 대한 구체적인 대안이 마련되어야 한다.

넷째, 사업의 취지에 가장 적합한 솔루션인가를 깊이 있게 고려해 보아야한다. 제도적 또는 절차적 공정성을 확보하여 독점소프트웨어와 공개소프트웨어의 장단점을 고려하여 가장 적합한 솔루션을 선택해야 한다. 독점소프트웨어라고 하여 소프트웨어의 완성도가 반드시 높은 것은 아니며 공개소프트웨어가 완성도가 높은 경우도 흔히 있다. 따라서 사업의 취지에 맞는 소프트웨어의 장단점을 검토하여 가장 적합한 솔루션을 선택해야 한다.

다섯째, 도입되는 솔루션에 대한 합리성을 확보해야 한다.

소프트웨어 구매 시 같은 성능일 경우에는 가격이 낮은 제품을 선택하고, 같은 가격일 경우에는 소스코드의 확보 및 확보된 소스코드의 자유로운 변경이 가능한 제품을 우선적으로 고려해야 한다. 성능과 가격이 모두 같은 경우라면 소스코드의 확보가 가능한 것이 향후 프로그램의 유지보수에 대한 보장성과 사업의 영속성을 위하여 보다 합리적이므로 소스코드의 확보가 가능한 것을 선택하도록 한다. 단, 소스코드의 확보는 독점소프트웨어이든 공개소프트웨어이든 구분하지 않고 확보가능성 여부를 고려해야 한다. 여기서 주의할 것은 소스코드의 확보가 가능하다고 하더라도 이것의 자유로운 변경이 허락되어 있지 않은 경우에 추가비용이 필요할 수 있으므로 이러한 경우에도 소스코드의 자유로운 변경이 가능한 제품을 선택해야 한다. 여기서 주의할 것은 소스코드의 확보가 가능하다고하더라도 한 번 수정이 가해지면 원래의 공개소프트웨어 공급처(또는 커뮤니티)의 소프트웨어와 호환성이 떨어져 향후 업그레이드가 어려워지므로 소스코드의 수정작업은 신중히 이루어져야 한다.

여섯째, 도입되는 공개소프트웨어의 소스코드에 대한 완전한 권리를 획득하였는가를 확인해야 한다. 도입하는 공개소프트웨어에 대한 소스코드의 확보는 매우 민감한 문제이다. 소스코드를 확보하지 못한다면 향후 수정, 패치, 업데이트 등과 같은 작업뿐만 아니라 시스템 증설작업이 필요할 경우에 사업자의 폐업과 같은 극한 경우에도 사업의 영속성을 보장받을 수 있기 때문이다. 따라서 공개소프트웨어 도입 시에는 소스코드의 확보와 함께 확보한 소스코드의 변경 가능여부와 변경한 소스코드를 적용가능한가를 반드시 확인해야 한다.

일곱째, 비공개소프트웨어를 선택하였을 경우와 마찬가지로 선택한 소프트웨어에 대한 기술적인 분석이 이루어져야 한다. 공개소프트웨어를 선택하든 독점소프트웨어를 선택하든 도입하기로 선택한 소프트웨어에 대해서는 기능성, 성능, 효율성, 그리고 확장가능성 및 시스템의 운용에 따르는 부가적인 문제로 구분하여 분석해야 한다.

여덟째, 도입하는 공개소프트웨어에 대한 법적인 분석이 이루어져야 한다. 어떤 무형의 권리를 “지적재산권”으로 총칭하여 언급하는 것이 편리하기는 하지만, 세부적인 내용을 살펴보면 이것은 많은 관점에서 서로 다른 법률들을 포함하는 집합적인 개념(collective term)이다. 따라서 실제로 무엇이 쓰여 지거나, 창조되거나, 개발되었는지, 그리고 어떤 종류의 지적재산권법이 관련되는지를 개별적으로 파악할 필요가 있다.

마지막으로, 웹 클라이언트 측면에서의 접근성에 대한 준수여부를 고려해야한다. 현재의 웹 기반 개발 환경이 MS Internet Explorer를 기준으로 개발되고 있어 리눅스, 맥킨토시 등 비 윈도즈 운영체제 및 모질라, 사파리 등 비 IE 환경 사용자들의 인터넷 사용에 제약을 가하고 있다.

따라서 웹을 통한 정보공개, 민원서비스 등의 시스템을 구축할 경우 다음의 기준을 준수하는 것이 필요하다

① HTML 문서 작성 시 표준 태그만을 이용하여 작성
② 자바스크립트 작성 시 표준을 따르는 코드만 이용
③ 사용자 인증 등을 위하여 Plug-in 기술을 사용해야 하는 경우 다양한 운영체제 및 웹브라우저에서 접근 가능하도록 설계
④ 기타 특정기업에 의존성 있는 코드 사용 배제
⑤ 위에 언급된 대표적인 브라우저(MS IE, 모질라, 오페라, 넷스케이프)에서 동일하게 접근가능한가 테스트를 실시(W3C의 Validation Test(validator.w3c.org) 참조)

다양한 플랫폼 사용자의 웹 콘텐츠 접근성 보장을 위한 기술지침은 정보통신접근성향상 표준화포럼의 웹 콘텐츠 접근성 지침 1.0과 한국 소프트웨어진흥원의 크로스브라우징 가이드를 통해 세부적인 내용을 참고할 수 있다.

5. 운영영역

가. 가이드 활용

공개SW를 이용하여 정보시스템 구축 시 필요한 지침을 기술하여, 차후 발생하는 정보화 시스템 구축 시 활용하도록 한다.

전자정부사업등 공공정보화 사업 추진 시 공개SW 기반 시스템 구축 적용 방법에 대한 상세한 가이드를 제시함으로써 공공기관 중심의 공개SW 확산에 필요한 지침서로 사용한다.

공개SW를 적용하고자하는 개발 사업에 본 과제의 산출물인 지침서를 사용함으로써 시행착오를 피하고 기획, 개발, 운영, 등의 단계에서 비용을 절감하고 효율성과 안정성을 높일 수 있다.

(1) 공개SW 프로세스 구축

오픈소스를 활용하기 위해서는 독점소프트웨어와 마찬가지로 반드시 해당 오픈소스의 라이선스에 대한 준수가 필수적이다. 하지만 오픈소스 라이선스에서 강제하고 있는 내용에 대해서 개발자 및 관리자들의 이해가 아직도 많이 부족한 것이 현실이다. 자칫 잘못하면 라이선스 위반으로 이미 판매중인 제품을 리콜하거나, 소스코드를 공개해야 하며, 개발 중인 제품을 아예 처음부터 다시 개발해야 하는 상황을 초래할 수도 있으므로 체계적인 프로세스를 수립하고 이를 담당할 관련 조직을 구축하는 것이 필요하다.

개발이 끝난 이후에 오픈소스 라이선스관련 문제가 발견된다면 수정에 많은 시간과 비용이 소요되므로 과제 계획 단계부터 오픈소스 라이선스 문제를 고려할 필요가 있다. 또한 개발이 진행되면서도 단계별로 준수해야 할 사항들을 정의하고 반드시 체크해야만 한다. 본 문서에서는 이러한 준수 사항에 대해 구체적으로 설명하기 위해 SW 개발프로세스를 다음과 같이 표준화/단순화하였다.

'기획(SW Design)' -> '구현(Implementation)' -> '검증 (Verification)' -> '제품화 (Production)'

□ 기획(SW Design)단계

오픈소스 라이선스 관련 문제를 피하는 가장 좋은 방법은 개발 기획 시점부터 이를 고려하는 것이다. 우선 해당 과제에 오픈소스를 활용할 것인지의 여부를 판단하

여야 하며, 구체적으로 어떤 프로그램을 사용할 지를 판단하여야 한다. 오픈소스의 특성상 Web 상에 여기저기 흩어져 있지만, 쉽게 오픈소스에 관한 정보를 찾을 수 있는 곳으로는 Freshmeat.net, SourceForge, OSDir.com, BerliOS, Bioinformatics.org 등을 들 수 있다. 이와 같은 사이트들은 대부분 License별 오픈소스 분류 항목을 두고 있기 때문에 쉽게 해당 프로그램의 라이선스를 확인할 수 있을 것이다.

기획 단계의 마지막으로 해당 SW Component별로 소스코드 공개가능여부를 판단하여야 한다. GPL등 소스코드 공개 의무가 발생하는 오픈소스소프트웨어를 사용할 경우에는 과제 결과물의 소스코드 공개가 요구되기 때문에, 경우에 따라서는 SW 구현 방법을 달리해야하기 때문이다. 소스코드의 공개가능 여부에 대한 판단 기준으로 다음의 사항을 참조할 수 있을 것이다.

○ Maintenance : 소프트웨어의 경우 하드웨어와 달리 개발 후 지속적 Upgrade 및 Debugging과 같은 Maintenance 과정이 중요하다. 이러한 Maintenance 과정은 상당한 Resource를 요하기 때문에 Maintenance를 직접 할 것인지에 대한 고려가 필요하다. 개발한 소스 코드를 오픈소스 커뮤니티에 공개하고, 이를 바탕으로 오픈 소스 커뮤니티를 통한 Maintenance 방법 역시 경우에 따라 아주 효율적일 수도 있을 것이다.

○ Fast Development : 오픈소스의 개발 모델 중 가장 특징적인 것이 바로 'Release Early and Often'을 통한 'Parallel Development and Debugging'이 가능한 것이다. 이를 통해 오픈소스는 빠른 개발 속도를 가능케 하고 있다. 이러한 모델을 Resource가 부족한 개발 과제에 적용하면 보다 효율적이고 빠른 개발이 가능할 것이다.

○ Reliability 확보 : SW의 신뢰성 확보의 가장 좋은 방법은 다양한 사용자들이 다양한 환경에서 해당 프로그램을 사용하면서 발견되는 문제점을 신속히 수정하는 것이다. 이런 측면에서 볼 때 오픈소스 커뮤니티를 잘 활용하면 SW Reliability 확보에 상당한 도움을 얻을 수 있을 것이다.

○ 차별화 유지 어려움 : 소스 코드를 공개하게 되면 그 소스 코드는 경쟁사에게도 공유되는 것이기 때문에 결국 제품의 차별화 확보가 불가능하게 되는 단점이 있다.

○ 지적재산권 확보의 어려움 : 기업이 보유한 특허를 구현하여 소스코드를 공개하는 것은 결국 모든 사용자들에게 Royalty-free의 조건으로 특허를 공개하는 것이나 마찬가지가 된다.

○ 특허 침해 소송 제기 가능성 증가 : 소스 코드가 공개되어 있으면 누구든 그 소스 코드를 볼 수 있기 때문에 특허 침해 소송 제기 가능성이 증가하게 되는 문제점이 발생할 수 있을 것이다.

□ 구현(Implementation)단계

자체 개발한 소스코드를 공개해도 무방한 경우는 특별히 구현 방법에 신경 쓸 필요가 없다. 단, 소스코드를 공개할 경우 회사보유의 지적재산권을 포함시키지 않도록 주의할 필요가 있다. 그러나 소스코드 공개를 원하지 않을 경우는 사용하는 오픈소스SW의 라이선스 의무사항과 활용하고자 하는 형태(Kernel, Application, Device Driver 등)에 따라 다양한 경우가 발생할 수 있기 때문에 상당한 주의가 요구된다.

○ 라이선스 삭제 금지

프로젝트 매니저는 개발자에게 오픈소스SW를 사용한 경우에는 해당 라이선스를 삭제하지 않도록 해야 한다. 거의 대부분의 오픈소스SW의 경우에는 소스코드 시작부분에 해당 라이선스를 표시하고 있다. 이러한 라이선스는 오픈소스SW를 검증하는 차원에서 검색을 통해 해당 오픈소스 SW의 사용이 적절한지 판단하기 위해서 꼭 필요하기 때문에 개발자에게 이를 꼭 주의시켜야 한다. 만약 이러한 라이선스를 따르지 않을 경우 저작권 침해로 인한 법적인 문제가 발생한다. 따라서 개발자들이 오픈소스SW를 사용하는 경우 해당 SW에 사용한 오픈소스SW의 라이선스를 반드시 추가하고 이를 삭제하지 않도록 하여야 한다.

○ 오픈소스 사용목록 작성

SW를 구현하는 단계에 있어서 오픈소스SW를 사용할 경우에는 오픈소스SW 사용목록을 작성하여 제출하도록 하여야 한다. 이렇게 함으로써 개발자에게 오픈소스SW를 사용하는데 있어 법적인 문제가 있음을 인식시킬 수 있으며, 향후 오픈소스SW 사용으로 인한 법적인 문제가 발생했을 때도 해당 목록을 통해 적극적인 대응을 할 수 있다.

특히 SW의 일부를 아웃소싱하는 경우 SW를 개발하는 업체에 대해 오픈소스 사용목록을 작성하도록 해야 나중에 이로 인한 법적인 문제가 발생 시 책임문제를 명확하게 할 수 있다.

○ 오픈소스 라이선스 확인

일부 오픈소스SW의 경우 라이선스의 확인이 어려운 경우가 있다. 이 경우에는 다음과 같은 방법을 통해서 라이선스를 확인해야 하고 이를 소스와 오픈소스 사용목록에 추가하여야 한다.

- 구글을 통한 방법
구글은 오픈소스 SW 코드를 검색하는 코드서치(http://www.google.com/codesearch) 사이트를 운영하고 있다. 이 사이트를 이용하여 자신이 인용한 코드의 파일명이나 패키지 명을 입력하면 해당 라이선스를 화면에 보여주기 때문에 자신이 사용한 프로그램의 라이선스를 쉽게 확인할 수 있다.

- Freshmeat이나 Sourceforge를 이용한 방법
오픈소스SW의 대부분의 프로젝트가 Freshmeat이나 Sourceforge에서 이루어지고 있기 때문에 이 사이트를 통하여 검색하면 해당 오픈소스SW의 라이선스를 쉽게 확인할 수 있다.

- 소스코드를 통한 방법
소스코드를 다운받아 COPYING, README, LICENSE 등의 파일에서 확인하거나 소스코드의 상위 코멘트에서 확인이 가능하다.

□ 검증(Verification)단계
개발이 완료된 후에는 개발 결과물인 소스코드에 대해 실질적인 검증 작업이 필요하다. 개발 계획서 그 자체로는 라이선스 이슈가 없었더라도 실제 구현과정에서 개발자가 오픈소스SW 라이선스에 대한 검증 없이 사용한 경우가 있을 수 있기 때문이다. 최근에는 특정한 소스코드가 오픈소스SW 코드와 일치하는지를 검증하여 주는 프로그램을 활용하는 사례가 증가하고 있다.
그러나 개발자들이 오픈소스SW의 라이선스를 지우지 않았다면 다음과 같이 간단한 스트링(string) 검색으로 라이선스를 확인할 수 있다.
또한 구현과정에서 설명한 오픈소스SW 라이선스 확인방법을 이용하면 된다.

```
$ grep -r 라이선스이름.
```

현재 디렉터리(.) 아래서 recursive하게(-r) 라이선스이름(GPL, LGPL, BSD) 등이 들어간line을 찾는 것이다.

□ 제품화(Production)단계
이 단계에서는 사용된 오픈소스SW들을 라이선스별로 분류하고 각 라이선스에서 준수해야 할 사항들이 실제로 제품에 반영될 수 있도록 하여야 한다. 앞에서 오픈소스SW의 라이선스 의무 사항은 크게 '저작권 관련 문구 유지', '제품명 중복 방

지', '서로 다른 라이선스 조합', '사용 여부 명시', '소스코드 공개', '특허' 등이 있다고 기술하였는데, 이중에서 '저작권 관련 문구 유지', '제품명중복방지', '특허' 등은 기획 및 구현단계에서 확인되어야 할 사항이고, '소스코드 공개', '사용여부명시'등은 제품화 단계에서 확인되어야 할 사항이다.

○ 소스코드 공개 방법

소스코드의 공개 방법으로 두 가지 방법이 있다. 첫째는 제품판매 시 제품과 함께 소스코드를 제공하는 것이다. 둘째는 제품과 함께 저작권 정보만을 제공하며 여기에 소스코드에 대한 정보와 실제 소스코드는 웹사이트를 통해 공개하거나 CD-ROM 등과 같은 매체를 통하여 우편으로 전달하는 것이다. 이 중 첫 번째 방법은 소스코드 제공에 있어서 확실한 방법이 되겠지만 라이선스를 유지하고 관리하는데 있어서는 비효율적이라 할 수 있다. 따라서 일반적으로 제품의 라이선스를 제공할 수 있는 웹페이지를 운영하는 것이 체계적이고 효율적인 방법이라 하겠다.

○ 소스코드 공개방법 사례

[표5-1. 소스코드 공개방법 사례]

구분	하드웨어	SW
제품명	소니 디지털카메라 DSC-G1	곰플레이어
저작권 문구	**On GNU GPL/LGPL applied software** The software that is eligible for the following GNU General Public License (hereinafter referred to as "GPL") or GNU Lesser General Public License (hereinafter referred to as" LGPL") are included in the camera. This informs you that you have a right to have access to, modify, and redistribute source code for these software programs under the conditions of the supplied GPL/LGPL. Source code is provided on the web. Use the following URL to download it. http://www.sony.net/Products/Linux/ We would prefer you do not contact us about the contents of source code. Read" license2.pdf"in the" License"folder on the CDROM. You will find licenses (in English) of" GPL,"and "LGPL"software. To view the PDF, Adobe Reader is needed. If it is not installed on your computer, you can download it from the Adobe Systems web page: http://www.adobe.com/	※ 해당부분 내용 - 이 프로그램에 포함된 비디오 디코더는 FFmpeg(http://ffmpeg.sourceforge.net)을 수정하여 제작하였으며, FFmpeg의 라이선스는 LGPL을 따릅니다. LGPL 라이선스는 설치 디렉터리의 LGPL.TXT를 참고하시기 바랍니다. 수정된 FFMPEG의 소스는 http://gomdevel.gomtv. com을 통해서 다운로드 받으실 수 있습니다.

○ 새로운 오픈소스SW 라이선스를 만드는 경우

제품을 상품화하면서 오픈소스로 개발은 하지만 자신의 회사에게 유리하도록 라이선스를 규정할 수가 있다. 그것은 바로 OSI에 정하는 오픈소스SW의 10가지 조건14)에 맞추어 승인을 받으면 된다. 11월 1일 현재 OSI에는 60여개의 오픈소스SW 라이선스가 등록되어 있다. 등록절차는 다음과 같다.

---

1. 다른 라이선스의 제목과 구별되는 자신만의 고유한 라이선스 이름을 만든다. 즉, 기존에 승인 받은 라이선스(제목 또는 카테고리가)와 구별되도록 한다.

2. 라이선스를 HTML과 플레인 텍스트의 두 가지 형식으로 제작하고 HTML 형식의 라이선스를 OSI 웹페이지에 올린다. 그러면 OSI에서 이미 승인한 라이선스와 같은 형식으로 변환시켜준다.

3. 라이선스가 오픈소스의 정의규정에 부합하도록 법적인 검토(의견)를 덧붙인다. 라이선스의 각 단락들이 각 오픈소스 정의 규정과 어떻게 부합하는지에 대한 해설이 있어야 한다. 그러한 법적검토(의견)는 공인된 변호사로부터 작성된 것이야 한다. 변호사의 법적검토가 담긴 이메일을 OSI로 보낸다.

4. 아래 설명에 따라 세 부분으로 나누어진 이메일을 라이선스-토론 메일링 리스트로 보낸다. 메일 제목에는 라이선스 제목과 함께“승인요청(승인용)”이라고 적는다.

- OSI에서 승인받은 라이선스 중 본인의 라이선스와 가장 유사한 라이선스를 표시하고 왜 기존의 라이선스가 본인의 필요를 만족시켜주지 못하는지에 대한 설명을 덧붙인다.

- 본인의 라이선스의 적용을 받아 배포되는 SW는 다른 오픈소스 라이선스 하에 배포되는SW와 어떻게 결합될 수 있는지 설명한다.

- 본인 라이선스의 텍스트 버전을 첨부한다.

5. license-discuss-subcribe@opensource.org에 가입하여 라이선스 논의에 참여한다. 만약 라이선스 토론 메일링 리스트의 회의들이 라이선스가 OSI정의에 부합하지 않는다는 것을 발견 한다면 그 문제를 토론을 통해 해결할 수 있다.

6. OSI와 라이선스-메일링리스트를 구독하는 사람들 또는 다른 검토자들이 본인의 라이선스가 오픈소스정의와 부합함이 분명하고 더 이상 특이사항이 없다는 확신이 생기면 라이선스가 승인되었음을 공지하고 OSI 웹사이트의 오픈소스SW 라이선스 리스트에 등재될 것이다.

○ OSI에서는 오픈소스가 되기 위한 10가지 조건을 만들어놓고 있는데 그 조건은 다음과 같다.

1. 자유로운 재배포 2. 소스코드 제공 3. 개작 허용 4. 저작자의 소스코드 원형유지 5. 개인 및 단체에 대한 차별 금지 6. 사용분야에 대한 차별 금지 7. 라이선스의 재배포 8. 특정제품에만 유용한 라이선스 금지 9. 다른 SW를 제한하는 라이선스 금지 10. 기술 중립적인 라이선스 제공

(2) 공개SW Governance 컴플라이언스 팀 구성

기업이나 조직에서는 공개SW Governance 이슈의 전반적인 관리를 위한 조직 또는 담당자를 둘 필요가 있다. 이러한 담당자는 공개SW 정책의 문서화, 공개SW 리뷰 프로세스, 공개SW 데이터베이스, 컴플라이언스 툴, 조직의 교육 프로그램 등을 담당하고, 조직 내에서 공개SW의 사용에 대한 포괄적인 감사(audit) 업무, 파트너 및 공급사들과의 협력, 법률부서와 함께 모든 공개SW 라이선스 분석업무를 담당하게 된다.

○ 공개SW 담당자의 주요 업무

- 주간/월간 리뷰 프로세스 수행
- 리뷰 및 승인 프로세스 수립 및 최적화
- 기업에서 사용되는 모든 공개SW를 추적 및 기록
- 리뷰 프로세스를 진행하기 위해 법률이나 IT 등 다른 부서 담당자들과 협력
- 정책 토론 및 결정을 진행
- 기업의 공개SW 정책을 문서화하고 관리
- 신규 라이선스, 제품, 법률 사례 등 공개SW 관련 사건들을 수집하고 기업 내 전파
- 기업이 공개SW를 사용함으로써 발생할 수 있는 리스크를 완화하려는 방안 마련

공개SW 컴플라이언스 담당자, B기업 사례

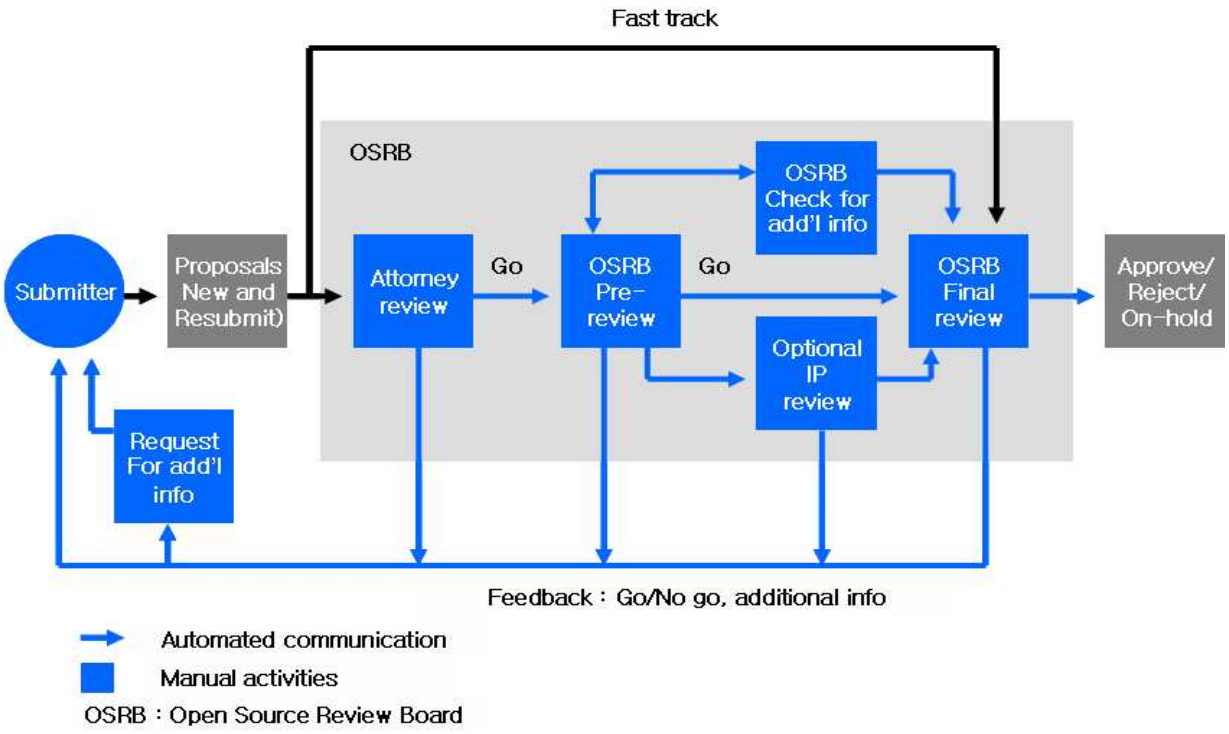
공개SW를 활용하다 보면 여러 가지 애매하고 의사 결정이 쉽지 않은 상황에 봉착하게 된다. 이러한 경우 컨설팅을 해 줄 수 있는 사내 전문가를 육성해 컨설팅을 해 주어야 할 것이다. 이러한 사내 전문가는 다양한 상황에 대해 경험을 가지고 있고 소프트웨어 개발도 어느 정도 있는 사람이어야 할 것이다. 사내 전문가는 또한 주요 오픈소스 라이선스 관련 메일링 리스트를 지속적으로 모니터링 하는 업무도

병행하는 것이 좋다.

(3) 공개SW Governance 리뷰보드 구성

IT, 법률, 엔지니어링 및 커뮤니티 멤버로 구성된 오픈소스 전문가들의 가상조직(virtual team)으로서 가칭 OSRB(Open Source Review Board)를 구축한다.

조직의 제품, 솔루션, 서비스에 공개SW를 적절하게 사용하도록 한다.



[그림5-1. 오픈소스 리뷰보드 구성]

(4) 공개SW 도입을 위한 전체 프로세스 정립

□ 요구사항 분석

공개 소프트웨어의 도입을 계획하기 전에 도입 기관의 요구 분석을 위한 사전 조사 필요

- 사용자 및 인적 요소
- 시스템의 기능 및 특성
- 제약사항 등

사전 조사 과정을 통해서 도입 과정 문제 발생을 최소화

○ 사용자 및 인적 요소

누가 어떠한 시스템을 어떻게 사용할 것인가 하는 사항은 전환 이전에 중요하게 생각되어야할 관점 중 하나이다. 이 과정에서 조사되어야 할 사항의 예를 들면 다음과 같다.

- 누가 시스템을 사용할 것인가?
- 사용자의 유형은 다양한가?
- 사용자가 시스템을 이해하고 사용하기 쉬운가?
- 시스템을 사용하는데 필요한 교육은 어떤 종류인가?

이 외에도 각 기관의 특성에 따라서 조사되어야 할 항목은 다양해짐.

○ 시스템의 기능 및 특성

시스템의 수행업무 및 특성에 대한 고려가 필요 하다.

- 시스템의 수행업무 파악
- 시스템의 특성 파악

이 과정에서 조사되어야할 사항의 예를 들면 다음과 같다

- 하드웨어적인 기능 및 성능은 어떠한가?
- 어떠한 운영체제를 사용하는가?
- 시스템이 하는 일은 무엇인가?
- 시스템은 어떻게 운영, 관리되고 있는가?

○ 제약 사항

운영 시스템의 특성상 공개소프트 도입에 제약이 있을 수 있으므로 사전 검토, 제약사항은 별도 정리해 두어야 함

[표5-2. 제약사항]

구분		내용	비고
사용자 및 인적요소	누가 사용하는가	기관 내 근무자	
	사용자 유형은 다양한가	팀별로 권한 설정	
	시스템을 사용하는데 교육이 필요한가	접속 방법	
시스템 기능 및 특성	어떠한 운영체제를 사용하는가	Window / Unix	
	시스템이 하는 일은 무엇인가	파일 서버	
	시스템은 어떻게 운영, 관리되는가	사용자 및 그룹별 접근 제한	
제약 사항	공개 소프트웨어 기반에서 불편사항은 없는가? 전환할 때 주의해야 할 사항은 없는가?		

□ 도입유형 결정

공개소프트웨어를 도입하는 방법에는 크게 3가지 방법이 있다. 첫째 기관이 직접 선택하여 도입하는 직접선택, 둘째 민간기업 등 외부의 추천에 의한 간접공급, 셋째 기관내부에서 공개소프트웨어를 개발 및 수정하는 내부개발 등이다.

○ 직접선택

직접선택("내부조달"이라고도 함)이란 공개소프트웨어 리소스 사이트에서 사용자가 특정 공개소프트웨어를 직접 다운로드하여 활용하는 것을 의미한다. 물론 다운로드 행위자체가 기관 내부의 직원에 의해 수행되며 기관 내부에서 공개소프트웨어 관련 정보가 어느 정도 확보되어 있어야 가능한 방법이다. 이 방법은 비용발생이 거의 없다는 장점이 있는 반면 위험부담이 크다는 단점이 있다.

특정 공개소프트웨어를 직접 다운로드 한다는 것은 독점소프트웨어를 공급업체로부터 구매하는 것에 비한다면 기술검증에 대한 책임을 스스로 부담한다는 단점이 있다. 즉, 다운로드하는 소프트웨어 내에 트로이목마와 같은 바이러스 코드가 들어있다거나 키 로거(key-logger)와 같은 악성 소프트웨어에 의한 감염을 포함한 소프

트웨어가 될 수 있기 때문이다. 그러나 세계적으로 사용빈도가 높은 공개소프트웨어를 공식 사이트로부터 다운로드받는다면 이러한 부담을제거할 수 있으며, MD5 등의 무결성 체크를 선행하는 경우 대부분해결이 가능하다.

그리고 또 다른 문제점은 다운로드한 공개소프트웨어의 보상과 보증이 보장되지 않는다는 것이다. 즉, 공개소프트웨어 솔루션을 기관내부에서 직접 선택하여 도입하려고하는 경우에는 소프트웨어의 보증과 보상을 보장받지 못하므로 이에 대한 위험감소 대책을 고려해야만 한다. 기관에서 공개소프트웨어 도입 시 각 분야에서 신뢰도가 높은 소프트웨어는 별첨을 참고할 수 있다.

이러한 위험에 대한 대책으로는 지역 내에 기술지원이 가능한 개발 업체가 하나이상 존재하는 공개소프트웨어를 선택하는 것이다. 이와 같은 대책을 마련해 둔다면 공개소프트웨어를 도입한 기관이 직접 해결하지 못하는 기술적 문제에 대해 안정성을 보장 받을 수 있다. 즉, 도입기관의 시스템 운영자는 대부분 매일의 주기적이고 반복적인 업무는 수행할 수 있다 하더라도 기술지원업체를 미리 확인하고 점검해 두는 것은 2중 3중의 안전장치를 가지는 것이며 시스템 가용성을 더욱 높이게 되어 위험요소를 감소시킬 수 있다. 직접선택 방법으로 공개소프트웨어를 도입할 경우 다음의 절차를 따를 것을 권고한다.

[표5-3. 직접선택 공개소프트웨어 도입 절차]

단계	내용
1단계	공개소프트웨어 솔루션 조사
2단계	유력한 공개소프트웨어 솔루션 선택
3단계	솔루션의 목적에 맞는 적합성과 비용적 가치 검토
4단계	솔루션의 품질과 보안성 분석
5단계	예비 프로젝트 수행
6단계	예비 프로젝트 결과를 통하여 기본적 수준의 예측사항 검토
7단계	프로젝트 수행 계획 수립
8단계	프로젝트 수행

특정 공개소프트웨어 제품이 기관의 요구조건을 충족시킬 경우 도입에 앞서 해당 기관에서 보다 상세한 검토를 수행하기로 결정하기로 하였다면 다음과 같은 체크리스트를 작성하고 평가과정을 거치도록 해야 한다.

직접선택방식 도입 전 체크리스트

① 기관에서 사용되는 운영체제 플랫폼에서 해당 공개소프트웨어가 실행되는가?
② 해당 공개소프트웨어가 기존의 운영체제 플랫폼에 대해 획득, 시험, 배치해야하는 추가적인 시스템 구성요소, 라이브러리 또는 모듈을 필요로 하는가?
③ 해당 공개소프트웨어의 설치 절차가 이해하기 쉽고 명확하게 정의되어 있는가?
④ 도입기관이 목적에 대한 적합성 여부를 결정할 수 있도록 공개소프트웨어를 설치, 배치, 시험할 수 있는 내부 전문지식을 갖추고 있는가?
⑤ 해당 공개소프트웨어의 설치 및 제거 절차가 명확하게 정의되어 있는가?

○ 간접공급

간접공급("외부조달"이라고도 함)이란 공개소프트웨어를 업체로 부터 공급받는 것을 의미한다. 즉, 공개소프트웨어 공급 전문기업으로부터 특정 공개소프트웨어를 공급받는 것으로 기술지원과 서비스 비용이 발생한다. 비용이 발생하는 대신 직접선택 방법에 비해 위험부담을 줄일 수 있는 안전한 방법이라고 할 수 있다.

간접공급 방식으로 공개소프트웨어를 도입하였다면 라이선스에 관련된 법적인 위험요소에 대한 책임은 원칙적으로 공급업체에 있다. 하지만 이 경우에도 공급업체와의 계약서 내용에 따라서 법적인 책임소재가 달라질 수 있으므로 이 부분에 대한 확인을 반드시 해야 한다. 즉, 기관은 기술적인 위험요소와 변경관리 위험요소를 감소시키기 위한 적절한 실사조사를 수행해야 한다. 즉, 기관이 공개소프트웨어 솔루션을 위해 간접공급 방식을 선택한 경우, 해당 업체 즉, 공급업체가 모든 기술지원에 있어 책임을 져야하며 기관은 공급업체가 해당 공개소프트웨어에 대한 적절한 위험경감 절차를 수행함을 확인해야한다는 의미이다.

이외의 사항은 비공개소프트웨어를 도입할 때와 고려사항 및 절차가 유사하다. 예를 들어 기관이 네트워크 및 시스템에 도입된 소프트웨어 보안표준에 대한 정책을 이미 보유하고 있을 경우에 개발업체는 위험분석의 일환으로 그러한 정책을 반드시 준수해야 한다. 또한 모든 절차가 끝난 후 검수작업 시에도 보안표준과의 일치성 여부를 확인하도록 해야 하며 계약서 내에 이러한 모든 절차들이 포함되어 있어야한다.

또한, 기관은 개발업체에게 모든 소프트웨어 구성요소에 대한 공급을 확인해야 하며 소프트웨어 출처와 라이선스 계약, 그리고 위험평가 문서를 확인해야 한다. 그리고 비공개소프트웨어를 도입할 때와 마찬가지로 도입된 공개소프트웨어의 전체적인 통합문제를 분석하고 이를 파악하여 반드시 문서화해야 하며 개발업체의 수행능력을 확인 할 수 있도록 자체평가를 수행할 수도 있다.

비공개소프트웨어를 업체로부터 도입할 때와 마찬가지로 간접공급방법은 개발업체의 적격성, 확장성, 소프트웨어의 성숙도 등에 관련된 분석이 선행되어야 하며 이러한 문제는 소규모 개발업체와 거래할 때 특히 신중히 검토되어야 한다. 따라서 앞서 설명한 바와 같이 잠재적인 위험요소를 줄이기 위하여 개발업체에 대한 적절한 실사조사 및 평가는 반드시 이루어져야하며 또한 평가 결과에 따른 조치가 이루어져야 한다. 다음은 이러한 평가에 반드시 포함되어야 할 내용들이다.

- 재정적 보장 및 안정성
- 위험관리의 적격성
- 내부 관리의 평가
- 업무 지속성 계획의 검토
- 일반적인 기능 개요
- 서비스 전달 및 관리
- 기타 기관의 특성에 맞는 평가요소

실제로 기관들은 공개소프트웨어 제품을 직접선택(내부조달)을 하기 위한 기술적인 능력을 갖추지 못한 경우가 대부분이므로 외부의 서비스 제공업체를 통해 공개소프트웨어 솔루션을 도입 받는 간접공급(외부조달)방법을 선택하는 것이 바람직하다.

그리고 잘 알려져 있고 인기 있는 대부분의 공개소프트웨어 제품들은 상용 솔루션 제공업체들을 통해 공급이 가능하다. 하지만, 외부의 어떤 업체를 통해 공급받더라도 공개소프트웨어와 비공개 소프트웨어(독점소프트웨어)의 공급에는 차이점이 있다. 즉, 기관이 외부 서비스업체를 통해 공개소프트웨어 솔루션 또는 제품을 공급받는 경우 일반적으로 관련 공개소프트웨어는 무료로 제공받고 이에 대한 서비스를 구입한다. 이것은 라이선스를 구입하고 부가가치 서비스로 지원을 제공받는 비공개 소프트웨어(사유소프트웨어)의 경우와 차이가 있다.

그리고 기관이 외부 공급업체를 선택하기에 앞서 후보업체들에 대한 실사로서 재정상황, 안정성, 기술능력 등을 평가해야 한다. 이러한 평가 작업들은 커뮤니티의 지원이 사라질 수 있는 위험한 제품선택에 대하여 위험을 줄여준다. 그리고 도입 후 어느 시점에 외부 공급업체가 사라지더라도 다른 공급업체에게서 지원을 계속 받을 수 있는 안정성과 유연성을 제공해 준다.

○ 내부개발

내부개발이란 기관내부에서 직접 개발하는 방법을 의미하며 공개되어 있는 수많은 공개소프트웨어들 가운데 도입하고자 하는 용도에 가장 적합한 공개소프트웨어를 선택하고 다운로드하여 기관내부의 개발자들에 의해 수정 개발하는 것을 의미한다. 아무것도 없는 상태에서 처음부터 개발하는 것이 아니라 공개되어 있는 수많은 공개소프트웨어들 가운데 도입하고자 하는 용도에 가장 적합한 공개소프트웨어를 선택하고 다운로드하여 기관내부의 개발자들에 의해 수정 개발하는 것을 의미하는 것이다.

기관내부 개발 방법으로 공개소프트웨어를 도입할 때에 고려해야 할 사항들을 간단히 정리해 보면 다음과 같다.

① 가장 적합한 공개소프트웨어 선택기준 마련
② 다운로드한 공개소프트웨어의 안정성 검토
③ 적용되는 라이선스 정책마련
④ 가장 적합한 개발방법론 마련
⑤ 개발 직원의 개발능력 검토
⑥ 사용직원의 사용법 교육 안 마련
⑦ 개발소프트웨어의 문서화 작업
⑧ 개발된 공개소프트웨어의 안정성과 영속성 대책 마련
⑨ 기타 특정업무의 종속되는 대책 마련 등

다음은 실제 내부 개발을 추진한 경우 필요한시에 단계별 개발절차이다.

- 서비스의 정의 및 적용업무 분석

가장 먼저 해야 할 작업이 어떤 업무에 적용하기 위하여 공개소프트웨어를 도입할 것인가를 정의해야하는 작업이다. 즉, 이러한 작업을 "서비스 정의"작업이라고 할 수 있으며 웹기반으로 개발할 것인지 아니면 기관내부 솔루션으로 사용하기위한 개발인지 아니면 유관기관과의 정보공유 및 협동 네트워크를 위한 개발인지를 정확하게 구분하고 이를 정의해야 한다.

- 기반 기술 정의

다음 단계는 개발기술에 대한 분석과 정의이며, 공개소프트웨어의 개발에 필요한 기반기술들을 간단히 정리하면 다음과 같다.

① 운영체제, 웹서버, DBMS, 사용 언어 등에 대한 기반기술 분석 및 정의
② 공용 framework 개발 및 업무 서비스 요구 분석 및 설계 기술
③ 분산 DB 구축 설계 및 관리 기술
④ 그룹웨어의 확장 및 공유 수준 지정에 의한 access 범위 설정 기술 공개소프트웨어 안내서
⑤ 유관 기관 간 정보 DB 표준화 (XML 기반) 이와 같이 관련 기반기술들이 정의가 되어 있어야 하며 개발 직원에 대한 교육과도 연계되어야 할 항목이다.

- 적용 업무에 대한 분석

개발하려는 공개소프트웨어가 적용될 업무에 대한 기본적인 분석이 이루어 져야한다. 즉, 공개소프트웨어를 직접 개발하기 이전에 다음과 같은 사항들이 이미 검토되어야 한다.

① 활용성 : 개발 이후의 당 기관 또는 유관기관에서의 활용성
② 수요의 규모 : 개발한 공개소프트웨어의 수요정도에 대한 검토
③ 재활용성 : 개발한 공개소프트웨어가 타 업무에 재활용 가능 여부
④ 표준성 및 표준의 준수여부

- 개발기간 및 투입인력 분석

공개소프트웨어를 직접 개발하기 위해서는 개발기간과 투입인력에 대한 분석과 정의가 이루어져야 한다. 개발기간에 대한 정의는 다음을 고려하여 작성하도록 한다.

① 5단계 개발기간 분석 : 요구분석기간, 분석기간, 개발기간, 수정 및 검토기간, 적용 및 안정화기간에 대한 분석 및 각 기간별 위험관리 대책 마련(개발기간의 준수여부에 따라 기간초과에 따르는 위험관리)
② 개발인력에 대한 대책

- 기반 기술(주로 Language)에 가장 적합한 기술인력 활용
- 개발경험이 있는 내부 개발 직원 활용
- 개발책임자(PM)을 선정하여 프로젝트로 진행
- 개발 분야에 따라 외부 개발 직원을 투입대책 마련
- 개발자 교육 대책 마련

- 개발한 공개소프트웨어의 사후 관리문제

내부개발에 의해 개발 완료된 공개소프트웨어는 일회성 프로젝트로 끝나지 않도록 사후관리가 이루어져야 한다. 사후관리에는 개발 직원에 대한 사후관리를 포함한다.

사후관리는 개발한 공개소프트웨어를 관리대장에 등록하여 지속적인 관리가 이루어지도록 한고, 개발 시에 사용한 기술을 종합 정리하여 이를 문서화하고, 개발한 공개소프트웨어의 설치문서, 사용법문서, 업그레이드문서 등을 표준화하여 작성한다.

지금까지 설명한 내부개발 방법을 설명하였다. 기관 내부에 공개소프트웨어 개발자를 확보하기 어려울 뿐만 아니라 프로젝트 진행을 위한 관리자 확보가 어렵기 때문에 현실적으로 내부개발 방법으로 개발하는 경우는 거의 없다. 개발자나 프로젝트 실무자를 확보하고 있다하더라도 실제 진행을 위해서는 보직변경 및 장기간 동일업무 종사여건등과 같은 조직내부적인 환경이 갖추어져야 할 것이다. 결론적으로 내부개발의 방법으로 프로젝트를 진행하기란 현실적인 어려움들이 존재하는 것이 사실이고 이러한 현실적인 어려움을 극복하고 프로젝트를 진행한다 하더라도 개발 및 프로젝트 노하우의 축적이 지속되기란 더욱 어려운 실정이다.

□ 공개SW 도입 절차

정부기관 및 공공부문에서 시스템을 도입하거나 소프트웨어를 도입 하는 일반적인 절차는 다음 표와 같다. 각 단계에서 공개소프트웨어 도입을 위한 주요 검토사항은 다음과 같다.

[표5-4. 공개SW 도입을 위한 주요 검토 사항]

단 계 (절 차)	내 용
사업계획서 작성	◊ 과제 발굴(대부분 사업 전년도), 예산편성 ◊ 사업 타당성, 적정성 검토 ◊ 사업 추진계획서 작성(최종 사업계획서 작성)
제안요청서 작성	◊ 제안요청서 작성 (객관성 및 타당성 검토가 절실히 요구되는 단계이며 도입되는 공개SW가 조직 내부용인지 도네이션 용인지 사업 목적과 범위를 명확하게 정의)
사업공고 및 제안서 접수	◊ 입찰공고 ◊ 제안 설명회 ◊ 사업제안서 접수
제안서평가 및 선정	◊ 제안서평가(평가위원회) ◊ 원가분석 ◊ 기술협상 및 가격협상
계약체결	◊ 최종계약체결
사업수행	◊ 사업수행(계약체결일로 부터 사업 진행)

	◊ 진도보고(착수, 주간, 중간보고 등) ◊ 완료보고
결과보고	◊ 최종 결과보고서 평가(검수)

- 사업계획서 작성 단계

추진 과제를 발굴하여 사업 타당성과 적정성을 검토한 후 사업계획서를 작성한다. 주관기관은 사업계획 수립 시 비공개소프트웨어와 함께 공개소프트웨어 도입을 고려하되, 시스템 개발 시 개방 표준(OpenStandard)을 지원하고 개방형 플랫폼(Open Platform)과 상호 호환성을 가지는 제품을 우선 고려해야 한다.

- 제안요청서 작성 단계

사업자 선정을 위한 제안요청서(RFP, Request For Proposal)를 작성한다. 제안요청서는 사업 프로젝트 수행에 있어서 매우 중요한 역할을 하는 만큼 제안요청서 작성 시 비표준 조건과 특정기술 등의 제약을 두지 않도록 매우 주의해야하는 단계이다. 즉, 고의, 부주의 또는 업무관습 등으로 인하여 주관기관에서 제안요청서 작성 시 운영체제와 하드웨어 플랫폼 등과 같이 매우 중요한 부분의 요구사항에서 특정 기업 제품 또는 기술표준만 제안 가능한 기술요건을 명시할 경우 불공정경쟁을 초래할 수 있으므로, 다음 표를 참고하여 비 표준적이거나 특정 기술조건을 명시하지 않도록 주의하여야 한다.

[표5-5. 비표준 특정기술 조건 예]

구 분	요구조건	특정기술조건 예
공통	특정업체 특정제품만 가능한 사양한정	-PCI Slot -48개 특정 locking 명시
하드웨어	특정 운영체제 아키텍처기반의 CPU를 명시	-RISC -SPARC
	특정 제품만 가능한 성능, 품질 등의 인증요구	-특정업체 인증 tpmC요청
주변기기	특정 운영체제만 지원하는 주변기기 채택	-Windows XP 전용 스캐너
운영체제	특정 운영체제만 지원하는 소프트웨어 개발언어 및 기술사용	-ActiveX, ASP, WMV로 개발
	특정 운영체제를 명시	-UNIX 플랫폼지원

		-Windows 플랫폼지원
미들웨어	특정 개발 툴에 종속된 개발환경요구	특정 세션관리 기법제시
웹서버	특정 운영체제상에서만 운용이 가능한 제품 채택	-IIS 도입
	특정 웹 어플리케이션 개발 언어의사용 요구	-ASP 사용 개발
데이터베이스	특정 운영체제상에서만 운영이 가능한 제품 채택	-특정 Locking 기법 제시
어플리케이션	특정업체에 기술 종속적이며 사용자의 선택권을 제한하는 소프트웨어 지정	-Windows Media Player

그리고 제안요청서 작성 시에 기관은 시스템 계층별 독립성을 확보하기 위하여 제안요청서상의 '도입대상 장비 내역 및 구성요건' 작성 시 다음 사항을 요구할 필요가 있다.

첫째, 특정 하드웨어에 종속된 특정 운영체제를 선택할 수밖에 없는 상황을 초래하지 않도록 하기 위하여 하드웨어와 운영체제를 별도의 항목으로 명시하고, 별도의 비용으로 계상한다.

둘째, 특정 운영체제에 종속된 특정 응용 소프트웨어를 선택해야만 하는 상황을 초래하지 않도록 하기 위하여 응용 소프트웨어와 운영체제를 별도의 항목으로 명시하고, 별도의비용으로 계상한다. 그리고 포털사이트 구축 사업 또는 웹 기반의 서비스 구축 사업을 진행할 경우 주관 기관은 제안요청서 작성 시 국민의 "정보접근권 보장"을 위하여 "다양한 컴퓨팅 환경에서 접근 가능하도록 국제표준을 준수하는 제품도입 및 개발"에 대한 항목을 명시하도록 한다. 그리고 BPR/ISP사업의 제안요청서 작성 시에는 공개소프트웨어 적용 가능성 분석을 포함시킴으로써 사업 초기 단계부터 공개소프트웨어기반의 시스템 구축 가능성을 검토함으로써 시스템의 상호운용성을 확보할 필요가 있다.

마지막으로 공개소프트웨어를 제안하는 사업자는 사업완료 이후 공개소프트웨어 유지보수 방안 제시'에 대한 항목을 반드시 명시하도록 함으로써 공개소프트웨어에 대한 기술지원 및 유지보수 서비스를 보장 받아야 한다.

- 제안서(제안업체) 평가 및 선정 단계

일반적으로 사용하는 제안서 평가항목은 다음과 같다.

① 유사분야에서의 사업수행 경험
② 개발대상 업무의 이해도
③ 개발전략
④ 기능 및 성능
⑤ 개발방법론
⑥ 기타 주관기관의 평가항목

각 항목별 세부 평가요소들은 주관기관의 사업추진 방향에 따라서 다소 달라질 수 있다. 특히, 소프트웨어 평가 시 동등 성능일 경우 공개소프트웨어를 제안한 업체를 우선 고려할 경우 정보화예산을 절감할 수 있는 장점이 있다.

- 사업 수행 및 결과보고 단계

공개소프트웨어를 적용하여 시스템을 구축하였을 경우 제공하는 소프트웨어의 소스코드를 확보하여야 한다. 특히, 제안업체에서 소스코드를 일부 수정하였을 경우에는 수정부분에 대한 문서화와 필요한 경우 해당 소프트웨어 개발 커뮤니티에 정보를 제공함으로써 공개소프트웨어 발전에 기여할 필요가 있다.

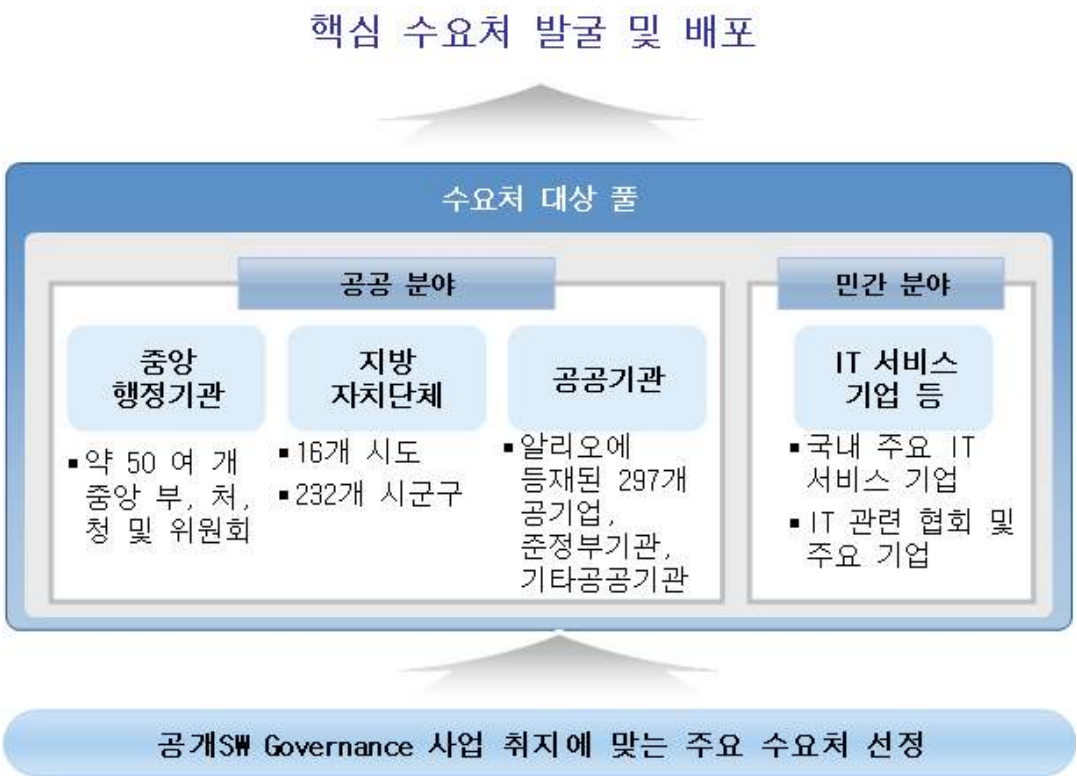
6. 공개 SW Governance 수요처 확보 및 보급 계획

가. 기본 방향

본 공개SW Governance의 수요 대상은 1차적으로 중앙행정기관, 지방자치단체, 공공기관 등의 공공분야의 정보화 담당관으로 정의하였다. 이에 1차 적으로 중앙행정기관 약 50여 개 중앙부, 처, 청 및 위원회와 지방자치단체 16개 시, 도, 232개 시군구를 대상으로 수요처를 확보하고 보급 할 예정이다.

2차 적으로 알리오에 등재된 297개 공기업, 준 정부기관, 기타 공공기관을 대상으로 수요처를 확보하고 보급 할 예정이다.

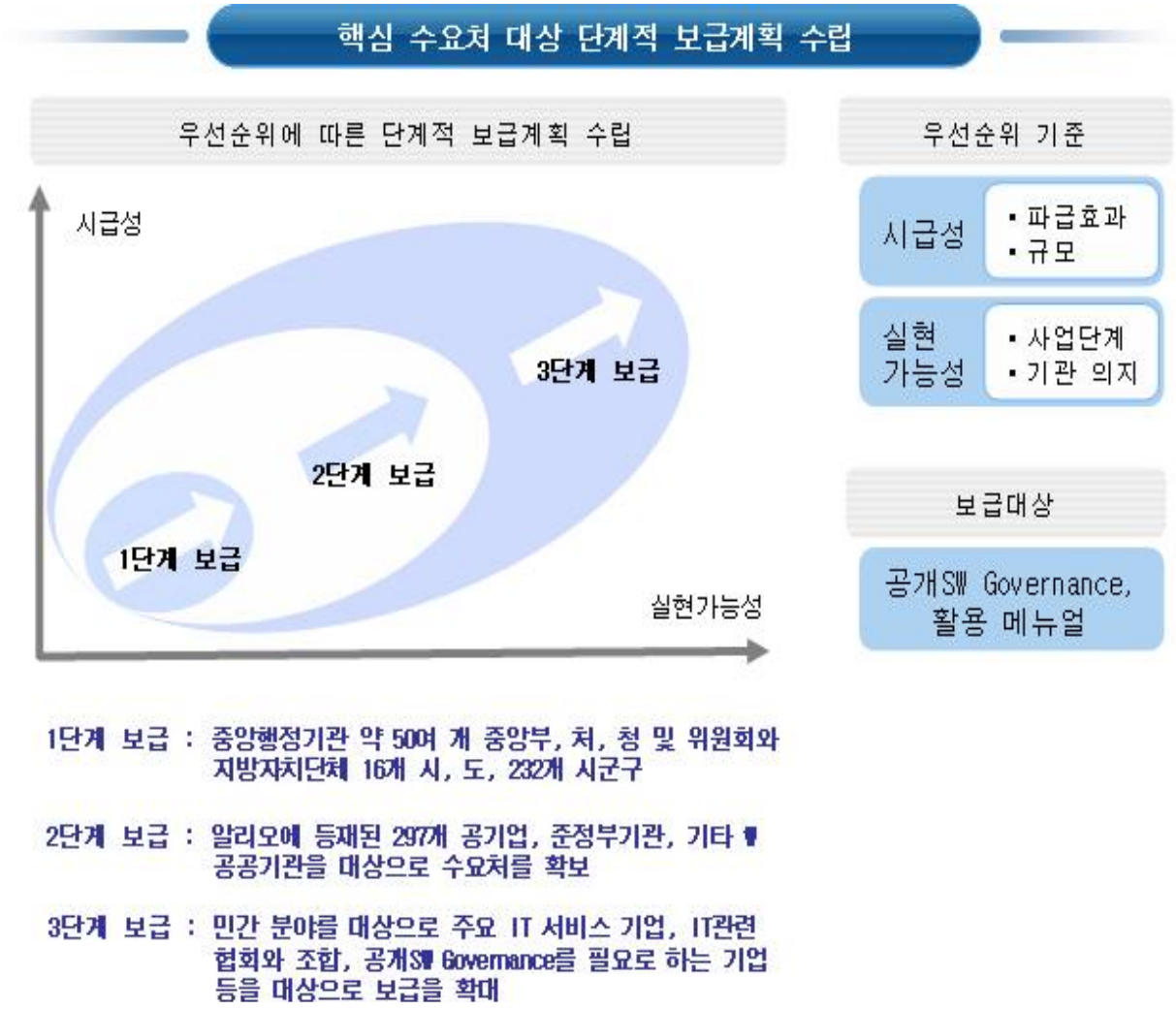
3차 적으로 민간 분야를 대상으로 주요 IT 서비스 기업, IT관련 협회와 조합, 공개SW Governance를 필요로 하는 기업 등을 대상으로 보급을 확대할 예정이다.



[그림6-1. 핵심 수요처 발굴 및 배포 계획]

나. 단계적 보급 계획 수립

공개SW Governance 사업 취지에 맞는 수요 수요처를 선정한 후 핵심 수요처를 대상으로 분야(중앙행정기관, 지방자치단체, 공공기관, 민간기업 등)별, 단계별 보급을 수립한다.



[그림6-2. 핵심 수요처 대상 단계적 보급계획]

다. 보급 방안 수립

공개SW Governance의 보급은 향후 공개SW 역량프라자 내에서 전담팀을 구성하여 홈페이지 등을 통한 홍보 및 마케팅을 담당하고, 공공기관의 공개SW Governance 신청 시 발급프로세스를 수립하여 배포한다.

[그림6-3. 공개SW Governance 보급 계획]

7. 맺음말

공개SW는 매년 성장세를 거듭하고 있고, 세계적으로 도입과 활용이 빠르게 확산되고 있으며, 국내에서도 정부의 적극적인 정책과 지원으로 공개SW에 대한 인식 재고와 민간분야로의 파급이 진행되고 있다.

이러한 빠른 속도의 발전에도 불구하고 공개SW는 개발환경에 따르는 구조적 취약점을 안고 있다. 공개SW는 다수의 개발자에 의한 비경제적 협업으로 문서화가 취약하며, 라이선스, 유지보수, 체계적인 관리체계 등에 있어 사용자로 하여금 공개SW 사용을 망설이게 하는 원인을 제공하고 있다.

이에 본 공개SW Governance는 공공기관이나 조직에서 사용자가 공개SW를 도입할 때 공개SW의 전반적인 이해를 돕고 효율적으로 도입 할 수 있도록 표준화된 프로세스를 제시하고 최적화된 관리방안을 제공하며, 도입, 사용, 유지보수 등에 관한 지침서로서 위험요소를 최소화하여 경영가치를 극대화 할 수 있도록 하였다. 또한, 향후 공개SW 기반 정보시스템 구축사업을 위한 예산수립, 발주, 조직구성, 프로세스 등에 있어 최적화된 기준점을 제시하려고 하였고 실제로 공개SW 유지보수가 가능한 기업을 조사해 제시하였고, 기관이나 조직에서 언제든 필요한 솔루션 정보를 확인 할 수 있도록 공개SW 대표적 리눅스 운영체제와 분야별 대표 공개SW 솔루션 리스트를 제시하였다.

본 공개SW Governance는 범용적인 성격을 가지고 있으나 1차적으로 공공기관을 대상으로 작성하여 향후 SI기업, 개발자, 일반 사용자 등을 위한 공개SW Governance의 수평적인 확장이 필요할 것으로 보인다. 또한, 본 공개SW Governance에서는 공개SW 검증 툴에 Fossology를 적용하여 테스트 해 본 결과 검증 툴로서 기능이 완벽하지 않아 향후 공개SW 라이선스 검증에 관한 안정적인 범용 검증 툴을 개발해야 할 필요가 있다.

마지막으로 공개SW Governance는 시시각각 변하는 국내외 공개SW 환경에 능동적으로 대응하고 올바른 방향설정을 위해 지속적인 자문과 검증 작업, 고도화가 필요할 것이다.