



공개소프트웨어 거버넌스 프레임워크 및 적용가이드

Open Source Software
Governance Framework and
Its Applying Guide



미래창조과학부
Ministry of Science, ICT and
Future Planning



nipa 정보통신산업진흥원
National IT Industry Promotion Agency



제1장 개요	6
1. 가이드의 목적	6
2. 활용방법	8
3. 적용범위 및 대상	10
(1) 내부사용	12
(2) 외부배포	12
(3) 외부서비스	12
제2장 프레임워크	14
1. 공개소프트웨어 거버넌스 프레임워크 개요	14
2. 전체 활동요소	17
(1) 컨설팅	17
(2) 정책수립	24
(3) 조직구성	31
(4) 요구분석	43
(5) 조사	46
(6) 분석	53
(7) 평가	58
(8) 계약	64
(9) 설계	70
(10) 개발	71
(11) 패키징	73
(12) 시험	75
(13) 배포	76
(14) 설치	77
(15) 운영	79
(16) 유지보수	80
(17) 기술지원	83
(18) 커뮤니티	84
(19) 컴플라이언스	87



Open Source Software Governance Framework and Its Applying Guide



(20) 교육	88
(21) 모니터링	89
제3장 적용가이드	92
1. 적용가이드 개요	92
2. 내부사용 관점	94
(1) 외부의 공개소프트웨어를 직접 가져와서 조직 내에서 사용하는 경우	95
(2) 특정 벤더사를 통해 공개소프트웨어 도입 및 기술지원을 받으며 사용하는 경우	99
3. 외부배포 관점	103
(1) 공개소프트웨어를 직접가져와서 개작 후 외부에 배포하는 경우	104
(2) 특정 벤더사를 통해 공개소프트웨어 도입 및 기술지원을 받아 외부에 배포하는 경우	120
(3) 직접 개발한 소프트웨어를 공개소프트웨어로 외부에 배포하는 경우	123
4. 외부서비스 관점	126
(1) 공개소프트웨어가 내부에 있고 외부에 서비스를 제공하는 경우	126
(2) 공개소프트웨어가 외부에 있고 외부에 서비스를 제공하는 경우	127
제4장 결론	142
첨부 자료	143
1. 공개소프트웨어 분류체계	143
2. 속성 별 정량화 공식과 적용 방법	146
3. 공개소프트웨어 조직관리 수준조사 양식	151
4. 공개소프트웨어 검증 권장 개선조치 보고서 샘플	159
5. Linux Foundation에서 국제표준으로 권장하는 SPDX 보고서	166
6. 용어집	167
7. 공개소프트웨어 기반의 개발 사업을 위한 위탁계약서 양식	172
8. 참고문헌 및 사이트	182



01 장

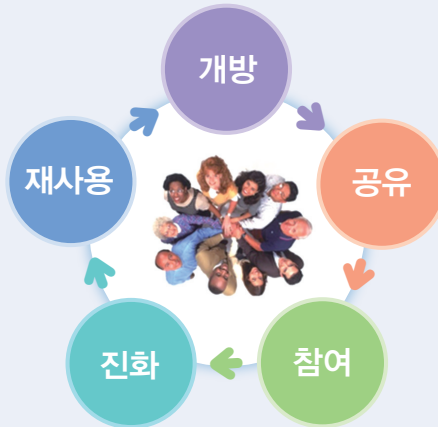
개요

1. 가이드의 목적
2. 활용방법
3. 적용범위 및 대상



1. 가이드의 목적

공개소프트웨어(Free and Open Source Software, FOSS)¹⁾는 낮은 획득비용, 개방형 표준 지향, 외부기업에 대한 의존성 감소 등 다양한 사용자의 요구를 충족시키며 급격하게 성장하고 있다. 특히, 과거와 다르게 상용소프트웨어와 비교하여도 기능이나 품질 측면에서 좋은 평가를 받고 있으며, 안정성, 유연성, 보안성, 투명성 등 많은 장점으로 인해 그 사용이 비약적으로 늘어나고 있다.



[그림 1] 공개소프트웨어 정의

전 세계적으로 그리고 클라우드, 콘텐츠 관리, 모바일 등 다양한 산업분야에서 공개소프트웨어는 핵심기술로 자리 잡고 있으며 최근 각광받고 있는 로봇, 사물인터넷 등의 미래 유망 산업분야에서도 기술의 중심이 되고 있다.

소스포지(Sourceforge.net), 깃허브(github.com) 등 하나의 사용 주체 내에서 관리되지 않고 다양한 저장소를 사용하는 데다 생성과 소멸을 반복하는 공개소프트웨어의 특성상 정확한 프로젝트 개수를 파악하기는 힘들지만, 블랙덱소프트웨어사에서 운영하는 공개SW 커뮤니티 온라인 공개 디렉터리인 Open HUB에서 확인해보면 66만여 개의 공개소프트웨어

1) FOSS(Free and Open Source Software) : 자유소프트웨어와 오픈소스소프트웨어가 결합된 단어로 국내에서는 공개소프트웨어로 통칭한다.

프로젝트 정보가 수집되어 있으며 다양한 분야의 공개소프트웨어 프로젝트가 꾸준히 증가하고 있음을 확인할 수 있다. (2014년 11월 현재)²⁾

그런데 공개소프트웨어가 다양한 산업분야에서 활용될 수 있다는 점은 분명하지만 기존 상용소프트웨어를 구매하여 사용할 경우와는 다르게 관리해야 하는 여러 가지 특성(라이선스 준수 의무, 기술지원 방식, 소스코드관리 등)이 있기 때문에 사용자가 공개소프트웨어를 활용 시에는 이에 대한 관리가 반드시 필요하다.

특히, 공개소프트웨어는 누구나 언제든지 무료로 다운로드 받아 사용할 수 있기 때문에 제작자에 대한 권리 존중이나 사용에 대한 대가 등을 생각하지 않는 경우가 충분히 발생할 수 있는데, 공개소프트웨어는 해당 소스코드에 대한 접근을 허용 할 뿐이지 엄연히 저작권으로 보호받고 있는 소프트웨어라는 점을 우선 주지할 필요가 있다.

공개소프트웨어는 사용 및 배포 등 소프트웨어를 사용하기 위한 의무사항을 규정하고 있는데 이를 라이선스라고 한다. 이를 어기게 되면 저작권을 침해한 것으로 간주되어 큰 낭패를 볼 수 있고, 소송에 휘말릴 경우 영세한 기업 같은 경우 존폐의 기로에 설 수도 있다.³⁾

또한, 내부에서 시스템을 사용하는 입장에서 사용이 무상이라는 개념만 가지고 공개소프트웨어를 적용하게 되면 조직이 보유한 해당 공개소프트웨어를 관리하는 역량이 부족하여 실제 개발이 어려울 수도 있고 적용하였다고 하여도 유지보수에 어려움이 발생할 수 있다. 이러한 내부 사용자 관점뿐 아니라 공개소프트웨어를 이용하여 시스템을 개발 후 타 업체에 판매하는 외부 배포자 및 외부에 기술을 지원하는 외부서비스 입장에서도 여러 가지 준수하고 적용해야 할 요소와 항목들이 있다.

따라서 공개소프트웨어를 올바르게 사용하기 위해서는 조직의 공개소프트웨어 적용 수준을 파악하고 그에 따라 정보화 계획 단계에서부터 일정한 기준과 절차에 의한 개발을 진행하고 지속적으로 관리하는 활동이 필요하다.

또한 이러한 내용들은 내부 시스템을 사용하는 입장과 공개소프트웨어를 사용하여 제품화하고 배포하는 입장, 그리고 외부에 기술을 지원하는 입장에서 서로 적용해야 할 내용이 다를 수 있다.

본 가이드에서는 공개소프트웨어를 개발 또는 사용하는 기업과 기관들이 공개소프트웨어 활용 라이프 사이클의 단계별로 공개소프트웨어 적용을 위해 취해야 되는 절차와 방법을 제공하고, 각 사용자 별 다른 관점을 적용하여 올바르게 공개소프트웨어를 사용할 수 있는 방법을 제시하고자 한다.

2) OpenHUB Projects : <https://www.openhub.net/explore/projects>

3) "공개SW 활용 법적 문제 대비해야", 디지털타임스, 2013

2. 활용 방법

본 가이드에서 제시한 공개소프트웨어 거버넌스 프레임워크는 사용 주체를 내부 사용, 외부 배포, 외부 서비스 관점으로 분류하고, 각 분류 내에서는 공개소프트웨어의 라이프 사이클에 따라 실행항목을 제시하고 있다. 따라서 사용자들은 자신에게 필요한 사용자 관점에 따라 해당되는 분류의 거버넌스 프레임워크를 참조하여 실전에 활용할 수 있다. 본 거버넌스의 사용 지침을 개괄적으로 설명하면 다음과 같다.

- ① 사용 주체는 자신이 어느 관점에 해당되는지를 확인
- ② 해당된 분류 항목의 거버넌스 프레임워크 맵에서 순차적인 실행 항목과 비순차적인 실행항목을 확인
- ③ 해당 분류 내에서 순차적인 실행 항목은 순서에 따라 진행
- ④ 순차적인 실행 항목이 비어있는 경우에는 다음 항목으로 이동
- ⑤ 비순차적인 실행 항목은 모두 동시에 진행
- ⑥ 비순차적인 실행 항목이 비어 있는 경우에는 무시
- ⑦ 비어있는 실행 항목일지라도 수행해야 하는 사항이나 점검 사항이 발견되면 프레임워크 발행 기관으로 보고(Feedback)

활용 극대화와 위험 극소화의 관점에서 공개소프트웨어를 가장 잘 사용하기 위해서는 공개소프트웨어가 쓰이는 장소와 시점에서 공개소프트웨어를 사용하는 주체가 어떻게 공개소프트웨어를 사용해야 하는지, 무슨 항목을 점검 해야 하는지, 어떠한 사항에 유의해야 하는지 등 모든 실행 항목에 대해서 전반적인 가이드가 필요하다. 따라서 이를 위해 거버넌스 프레임워크를 제공하게 되었다.

또한 본 가이드에서는 거버넌스 프레임워크를 사용하는 주체가 최대의 효과를 거둘 수 있도록 공개소프트웨어가 흘러가는 순서와 위치에 따라 크게 3개의 대분류와 7개의 소분류로 구분하였다. 여기서 일곱개의 소분류는 각 대분류의 상세 속성에 따라 세분화되었다. 만일 여기서 제시하는 어느 분류에도 해당되지 않는 사용 주체가 있다면 이러한 사항은 차기 프레임워크에 반영되어야 할 것이다. 사용 주체가 어느 분류에 해당되는지 정확히 판가름하기 위해서는 무엇보다도 사용 목적에 초점을 맞추어야 한다.

(1) 내부사용

먼저 내부사용이라는 분류는 사용 주체가 공개소프트웨어를 내부적인 이익을 목적으로 활용하는 경우이다. 예를 들어 포털, 그룹웨어, 인사관리시스템, 재무시스템 등 내부 임직원을 대상으로 공개소프트웨어를 활용하는 경우가 여기에 해당된다. 현실적으로 공개소프트웨어를 내부적으로 사용하는 주체가 모두 동일한 기술력을 보유하고 있다고 가정할 수는 없다.

즉, 내부 역량이 충분하여 자체적으로 설치부터 운영 및 유지보수까지 수행할 수 있는 사용 주체도 있고 그런 기술 수준이 못되거나 TCO(Total Cost of Ownership)⁴⁾ 이슈로 인해서 외부의 지원을 받게 되는 경우도 있다. 이러한 전제에 따라 내부사용을 세분화 시키면 다음과 같이 두 가지 세부항목으로 나누어진다.

- 1) 외부의 공개소프트웨어를 직접 가져와서 조직 내에서 사용하는 경우
- 2) 특정 벤더사를 통해 공개소프트웨어 도입 및 기술지원을 받으며 사용하는 경우

일반적으로 첫 번째 소분류는 ICT((Information and Communication Technology)⁵⁾ 기업 또는 연구소가 해당되고 두 번째 소분류에는 학교 또는 정부 기관 등이 해당된다고 볼 수 있다

(2) 외부배포

다음으로 외부배포라는 두 번째 분류에 해당되는지 판단하기 위해서는 공개소프트웨어가 자사의 밖으로 나가는지 살펴보아야한다. 공개소프트웨어를 내보내는 방법으로는 매체를 통해서 개발자가 사용자에게 직접 전달해 주는 적극적인 유형도 있지만 사용자가 인터넷을 통해 직접 다운로드 받는 경우도 있어 종류는 매우 다양하다. 또한 공개소프트웨어의 전달 매체를 살펴보면 CD, DVD, 외장 HDD, USB 등과 같이 소프트웨어 저장 용도로 만들어진 제품을 통해 가장 많이 외부로 전달되지만 스마트 폰, 가전제품, 자동차 제어기 등 공개소프트웨어가 저장되어 있는 특정 전자 장치도 전달 매체가 될 수 있다.

한편, 밖으로 유포되는 공개소프트웨어는 동일하지만 그 공개소프트웨어를 준비하는 작업은 상이할 수 있다. 따라서 공개소프트웨어를 외부로 배포하기 직전에 수행한 작업에 따라 외부배포를 다음과 같이 세 가지로 구분할 수 있다.

- 1) 공개소프트웨어를 직접 가져와서 개작 후 외부에 배포하는 경우
- 2) 특정 벤더사를 통해 공개소프트웨어 도입 및 기술지원을 받아 외부에 배포하는 경우
- 3) 직접 개발한 소프트웨어를 외부에 배포하는 경우

그러나 현실적으로 공개소프트웨어를 기반으로 기술지원 서비스를 제공할 목적으로 외부에 배포하는 경우가 대부분이므로 다음과 같이 정리할 수 있다

- 4) 공개소프트웨어를 직접 가져와서 개작 후 외부에 배포하는 경우
- 5) 특정 벤더사를 통해 공개소프트웨어 도입 및 기술지원을 받아 외부에 배포하는 경우
- 6) 직접 개발한 소프트웨어를 공개소프트웨어로 외부에 배포하는 경우

이 분류에서 배포되는 공개소프트웨어를 받는 수신 주체가 누구냐에 따라 거버넌스의 세부

4) TCO(Total cost of ownership) : 총 소유비용

5) ICT(Information and Communication Technology) : 정보통신기술을 지칭하며, 더 단순하게 IT로 사용하는 경우가 많다.

활동요소가 달라질 수도 있다. 주요한 수신 주체를 예로 든다면 소프트웨어 제작사, 소프트웨어 판매사, 전자제품 제작사, 전자제품 사용자, 공개소프트웨어 커뮤니티 등이 있다.

이러한 배포는 단일 행위로 끝나는 경우도 있지만 공급 체인의 구조에 따라 복수의 수신 주체가 순차적으로 이어지는 상황도 있다. 따라서 공개소프트웨어 배포 사슬의 각 단계에 따라 거버넌스 항목이 달라지는 것을 유의해야 한다.

(3) 외부서비스

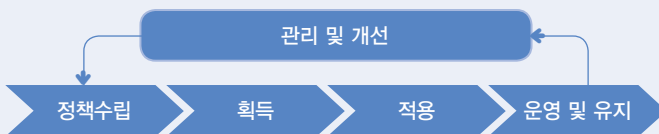
마지막으로 외부서비스라는 세번째 분류에 해당하는지 판단하기 위해서는 공개소프트웨어를 사용하여 외부로 기술 서비스를 제공하는지의 여부를 확인하여야 한다. 공개소프트웨어는 무료이기 때문에 사용료는 없으나 최상의 상태에서 문제 없이 사용하기 위해서 기술 지원이 필요한 경우가 있다. 주요한 기술 지원에는 설치, 업데이트, 운영, 유지 보수 등이 있으며 대부분 공수를 기반으로 서비스 대가를 산정하게 된다. 이러한 지원 서비스의 대상인 공개소프트웨어가 어디에 있는가에 따라 프레임워크의 실행항목과 점검항목의 내용이 달라지기 때문에 다음과 같이 두 가지 종류로 외부서비스 분류가 세분화 될 수 있다.

- 1) 공개소프트웨어가 내부에 있고 외부서비스를 제공하는 경우
- 2) 공개소프트웨어가 외부에 있고 외부서비스를 제공하는 경우

공개소프트웨어를 내부에 두고 외부로 서비스하는 경우는 인터넷을 기반으로 사용자에게 웹서비스를 제공하는 경우이다. 클라우드 컴퓨팅의 SaaS(Software as a Service)가 대표적이다. 그렇다고 공개소프트웨어를 내부에 두고 외부로 서비스를 제공하는 경우에는 운영 환경이 반드시 클라우드 컴퓨팅 환경일 필요는 없다. 이는 사업 모델에 따라 다양한 방법으로 공개소프트웨어에 관련된 서비스를 제공할 수 있기 때문이다. 한편 공개소프트웨어가 외부에 있고 외부서비스를 제공하는 경우에는 대부분의 공개소프트웨어에 기술 지원 서비스를 제공하는 업체가 이 분류에 해당된다.

3. 적용범위 및 대상

본 가이드는 공개소프트웨어를 활용하는 다양한 사용자들이 공개소프트웨어 활용 라이프 사이클 전체 단계에 걸쳐 관리해야 하는 위험요소 및 대응방안을 제공한다.

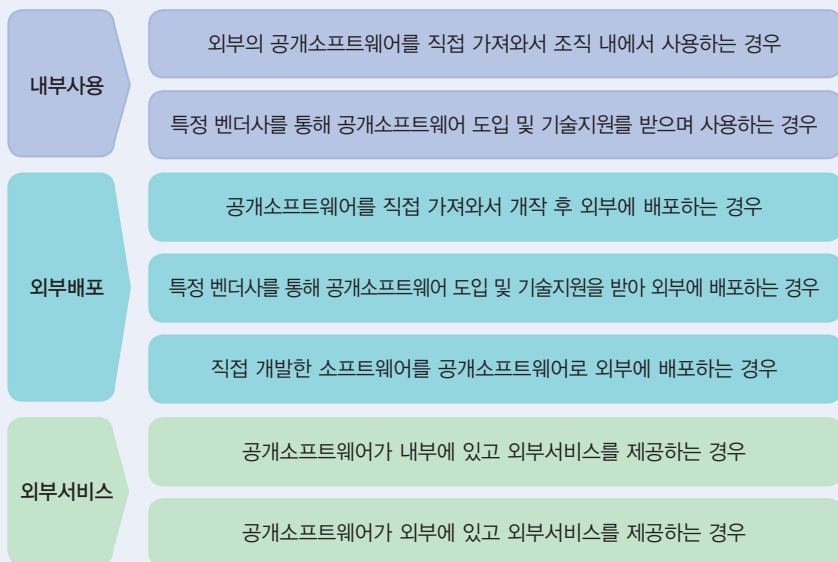


[그림 2] 공개소프트웨어 활용 라이프 사이클

- ① **정책수립 단계** : 해당 조직의 공개소프트웨어 적용 순응도 수준을 측정하여 그에 따른 정책과 전략, 그리고 운영계획을 수립한다.
- ② **획득 단계** : 공개소프트웨어를 조사하고 분석하여 적용을 위한 타당성을 평가하는 과정이다.
- ③ **적용 단계** : 공개소프트웨어를 조직에 적용하기 위한 활동요소 및 관리내용을 제시하고 있다.
- ④ **운영 및 유지 단계** : 공개소프트웨어의 설치 및 운영을 위한 유지보수를 포함하고 있다. 특히 공개소프트웨어 개발의 특징이라고 할 수 있는 커뮤니티 관련 사항이 포함되어 있다.
- ⑤ **관리 및 개선 단계** : 공개소프트웨어 활용 라이프 사이클 전반에 걸쳐 활동결과를 평가하고 성과를 측정하여 조직의 공개소프트웨어 거버넌스를 지속적으로 개선할 수 있는 방법을 제시하고 있다.

그리고 공개소프트웨어를 활용하는 사용자의 입장도 소스코드의 획득경로, 공개소프트웨어의 관리주체, 공개소프트웨어를 활용한 비즈니스 방식 등에 따라서 매우 다양하게 정의할 수 있다.

본 가이드에서는 공개소프트웨어를 활용하는 사용자를 내부사용, 외부배포, 외부 서비스의 3가지 관점으로 나누어 구분하고, 각각의 활용 케이스를 다음과 같이 구분하여 각각의 케이스에 대한 적용 가이드를 제시하였다.



[그림 3] 공개소프트웨어 사용자별 관점

(1) 내부사용

내부사용 관점이란 조직 내 공개소프트웨어를 적용해 정보시스템을 개발하고 적용하는 것으로 대부분의 공공기관 및 일반 업무조직은 내부사용 관점이라고 볼 수 있다. 내부사용 관점에는 2개의 활용 케이스가 있다.

- 1) 외부의 소스코드를 직접 가져와서 조직 내에서 사용하는 경우
- 2) 특정 벤더사를 통해 공개소프트웨어 도입 및 기술지원을 받으며 사용하는 경우

(2) 외부배포

외부배포 관점이란 공개소프트웨어가 사용 주체 밖으로 내보내지는 상황이다. 여기에는 공개소프트웨어 커뮤니티에 소스코드를 기여하는 상황, 사용 주체 간에 공급망에서 수급이 이루어지는 상황, 디지털 매체 또는 기기에 탑재되어 최종 사용자에게 전달되는 상황 등이 속한다. 특히 공개소프트웨어의 라이선스 위반에 따른 컴플라이언스 이슈가 가장 분명하게 나타나는 경우이다. 따라서 공개소프트웨어를 기반으로 패키지 소프트웨어를 제작하고 외부에 직접 판매 또는 배포하는 기업이 컴플라이언스 측면에서 가장 주의해야 하는 사용상 분류이다. 외부배포 관점에는 3개의 활용 케이스가 있다.

- 1) 공개소프트웨어를 직접 가져와서 개작 후 외부에 배포하는 경우
- 2) 특정 벤더사를 통해 공개소프트웨어 도입 및 기술지원을 받아 외부에 배포하는 경우
- 3) 직접 개발한 소프트웨어를 공개소프트웨어로 외부에 배포하는 경우

(3) 외부서비스

외부서비스 관점이란 공개소프트웨어를 활용하여 외부 고객을 대상으로 다양한 유형의 IT 서비스를 제공하는 비즈니스 모델의 경우를 의미한다.

외부서비스 관점에는 2개의 활용 케이스가 있다.

- 1) 공개소프트웨어가 내부에 있고 외부에 서비스를 제공하는 경우
- 2) 공개소프트웨어가 외부에 있고 외부에 서비스를 제공하는 경우

상기의 내용을 종합하여 본 가이드에서는 공개소프트웨어 활용 라이프 사이클에 따라 각 관점별 사용자들이 공개소프트웨어를 효과적으로 사용할 수 있도록 실행 항목과 점검 사항을 제시하였다.



02장

프레임워크

1. 공개소프트웨어 거버넌스 프레임워크 개요
2. 전체 활동요소



1. 공개소프트웨어 거버넌스 프레임워크 개요

지금까지 공개소프트웨어 거버넌스는 여러가지 의미로 사용되어 왔다. 그중에서도 “공개소프트웨어 적용 및 구현을 통치하기 위한 절차”⁶⁾ 또는 “공개소프트웨어 공급 및 수요체계를 구성하는 다원적 조직체계 내지 조직 네트워크의 상호작용 패턴으로서 구성원의 집단적 활동”⁷⁾ 등이 대표적인 정의가 될 수 있다. 또한 공개소프트웨어 거버넌스는 소프트웨어 라이선스 관점에서 역할과 책임을 중심으로 상호규약 측면에서도 정의될 수 있어 “공개소프트웨어 개발프로젝트에 기여하는 조직과 개인들의 조정, 통제 및 목표를 달성하기 위한 수단”⁸⁾으로도 볼 수 있다.

본 가이드에서는 이러한 다양한 정의를 바탕으로 공개소프트웨어 거버넌스 프레임워크란 “공개소프트웨어를 안전하게 사용·적용 및 배포하기 위해 필요한 사항을 다양한 관점에서 활용할 수 있도록 소프트웨어 라이프 사이클 단계별로 제시한 틀”로 해석하였다. 이를 위해 사용자 관점⁹⁾ 별로 활용방법을 구분하였고 그 사용자 관점에 따라 적용해야 하는 가이드라인을 공개소프트웨어 활용 라이프 사이클 단계별 수행 프로세스로 제시하고 있다.

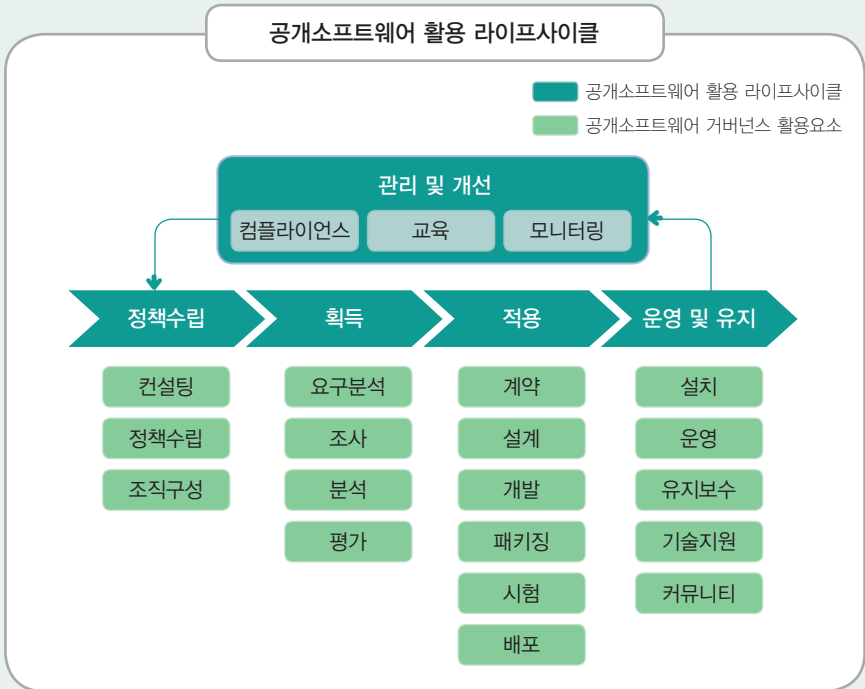
본 가이드는 [그림 4]에서 보는 바와 같이 공개소프트웨어를 활용하는 사용자의 라이프 사이클을 식별하고 그 흐름에 따라 공개소프트웨어 거버넌스 활동요소를 사용 유형에 적합하게 가이드 하고 있기 때문에, 어떤 유형이든지 자신의 상황에 맞는 활용 케이스를 선택하여 쉽게 적용할 수 있도록 하였다.

6) “Assessing Open Source Software Projects”, Gartner, 2005

7) “공개소프트웨어 거버넌스 가이드라인”, 김병선, 2013

8) “The governance of free/open source projects”, M. Lync Markus, 2007

9) “A framework for information systems architecture”에서 제시한 다양한 관점을 표현, J. A. Zachman, 1987



[그림 4] 공개소프트웨어 거버넌스 프레임워크

실질적인 공개소프트웨어의 활용을 돕기 위해서는 각각의 사용자 유형에 적합한 활동요소에 대하여 각각의 가이드라인을 제공하는 것이 필요하다. 예를 들어 공개소프트웨어 거버넌스의 주요 활동 요소 중 컴플라이언스 관련 활동에 대하여 가이드라인을 제공한다고 가정한다면, 동일한 컴플라이언스 활동에 대해서 각 단계별, 각 사용자 유형별로 상이한 적용 방안이 제시되어야 한다.

그러므로 본 문서에서는 명백하게 구분할 수 있는 모든 공개소프트웨어 활동을 단위 요소로 나열하고 각각의 공개소프트웨어 활용 라이프 사이클 단계에 따라 사용자 유형별 활용 방안을 제시하였다.

전반적인 이해를 돕기 위해서 이 거버넌스 프레임워크의 전체 활동 요소를 요약하면 <표 1>과 같다.

〈표 1〉 공개소프트웨어 거버넌스 프레임워크 활동 요소(21개)

활동요소	설 명	특성
정책수립	목표와 전략에 따라 반드시 지켜야 할 규정과 지침을 수립함	순차적 (18)
컨설팅	공개소프트웨어 적용과 전략수립을 위한 자문 서비스를 제공함	
조직구성	효율적인 인력 구성과 역할과 책임에 따른 운영 방안을 제시함	
요구분석	고객 또는 사용자의 고민, 요구사항 등을 분석함	
조사	새로운 공개소프트웨어 또는 특정 분야에 적합한 공개소프트웨어를 찾음	
분석	공개소프트웨어의 속성을 구분하고 상태나 수준을 정리함	
평가	각 속성에 가중치를 부여하고 평가 모델을 적용하여 채점함	
계약	공개소프트웨어의 도입 및 활용, 배포에 대한 일련의 책임과 의무에 대해 조건과 규정을 체결함	
설계	요구 분석 결과에 따라 기능과 사양을 미리 구성함	
개발	공개소프트웨어 프로그램을 변경 및 결합함	
패키징	공개소프트웨어의 설치가 편리하도록 단일 프로그램으로 묶음	
시험	요구 수준에 맞는지 품질과 성능을 확인함	
배포	공개소프트웨어를 저장매체, 웹사이트, 장비 등을 통해 전달함	
설치	공개소프트웨어를 운영할 수 있는 장비에 탑재함	
운영	공개소프트웨어를 실행시켜 정상적인 상태로 지속적으로 가동시킴	
유지보수	최상의 운영 상태를 유지하도록 제반 작업을 수행함	
기술지원	추가적인 요구 사항을 반영이나 문제 해결 등 공학적인 공개소프트웨어 서비스를 제공함	
커뮤니티	소스코드 기여, 재정적 지원, 활동 교류, 참여방법을 제시함	
컴플라이언스	라이선스 의무사항 준수 및 법적 문제를 해결함	비순차 · 비정기(3)
교육	공개소프트웨어의 도입, 활용, 배포에 대한 이해력을 높이기 위해 지식을 전달하고 스킬을 향상시킴	
모니터링	공개소프트웨어 적용 이후의 상황을 파악하고 피드백을 수렴함	

2. 전체 활동요소

(1) 컨설팅

컨설팅은 서비스 제공자가 공개소프트웨어를 원활히 도입, 활용, 관리할 수 있도록 조사, 진단, 분석, 개선 방안, 전략, 로드맵, 방법론 등을 고객에게 서비스로 제공하는 활동으로 자문 서비스와 산출물 제출이 포함된다.

컨설팅 활동은 일반적으로 수준 진단, 문제 분석, 해결 방안 도출, 실행 방안 제시 등으로 구분할 수 있다. 공개소프트웨어에 관련된 컨설팅은 공개소프트웨어의 조사, 도입, 적용, 투자 효과 분석, 거버넌스 체제 구축, 활성화, 위험 관리, 변화 관리 등 다양한 분야에서 수행되고 있다.

이러한 컨설팅은 내부사용 관점에서는 서비스를 받는 입장이고 외부 서비스 관점에서는 서비스를 제공해 주는 입장이 되기 때문에 관점에 따라 컨설팅의 준비와 실행에 큰 차이가 있고 유의해야 할 부분이 많이 다르다.

내부사용 관점에서는 외부의 도움으로 내부의 문제를 발견하고 해결책을 찾아야 하므로 최대한 정확하게 문제를 진단할 수 있도록 자료를 공개하고 지원을 아끼지 않아야 한다. 이를 위해서는 보안 사항에 대한 접근도 불가피하므로 비밀유지계약(Non-Disclosure Agreement, NDA)을 체결하여 상호 협력에 장애요소가 없도록 사전에 준비해야 한다.

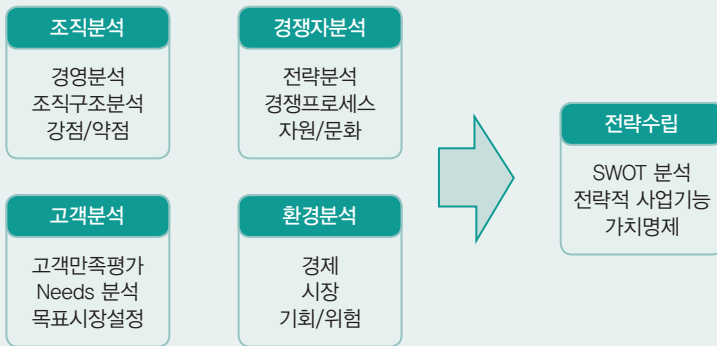
그러나 컨설팅을 받는 입장에서 매우 조심해야 할 사항은 내부 관련 조직이나 책임자의 당초 견해가 컨설팅의 최종 결과에 반영되도록 유도해서는 안 된다는 것이다. 이렇게 되면 객관적인 근거와 분석 결과에 따라 문제를 해결하지 못하고 도리어 외부의 힘을 빌어서 주관 조직의 주장을 강화시키는 오류가 생기기 때문이다. 부적절한 왜곡과 실패를 방지하기 위해서는 무엇보다도 컨설턴트에게 최대한의 권한과 자격을 부여해야 한다. 그래야만이 데이터가 조작되지 않고 객관적이고 공정한 결과를 얻어 낼 수 있다.

그리고 컨설팅의 결과만큼이나 중요한 사항은 그 결과의 효과적인 적용과 활용에 있다. 만일 컨설팅의 결과나 중간 산출물이 실질적으로 사용되지 못한다면 결과적으로는 시간과 비용만을 낭비했을 뿐이기 때문이다. 따라서 계약 체결 이전에 산출물을 정확하게 정의하는 것은 물론이고 이 산출물을 추후 어떻게 적용할 것인지에 대해 전략적인 활용 방안이 먼저 수립되어 있어야 한다.

공개소프트웨어 컨설팅은 다음과 같은 절차로 수행된다.

1) 사업 환경 및 전략 확인

- 조직의 경영전략, 조직 특성 및 환경, 조직 요구사항에 맞는 정보화계획을 수립하기 위해 프로젝트의 목표를 파악하고, 프로젝트의 범위 및 접근방법을 결정하는 단계
- 조직의 정보시스템과 관련된 전략이 도출된 경우 공개소프트웨어 시장 활성화 및 최신 기술동향을 반영한 개방형 정보시스템을 우선 고려
- 사업 환경 및 전략 확인은 전략정보 수집, 전략정보 조정, 전략정보 확인, 검토 및 승인의 절차로 진행된다.
- 그 결과 외부 환경 분석서, 내부 환경 분석서, 전략 분석서를 도출.

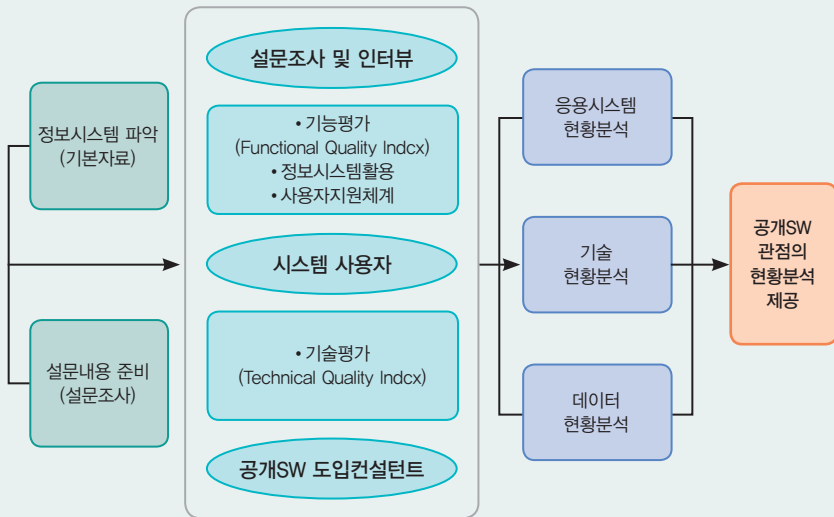


[그림 5] 사업 환경 및 전략 확인

2) 공개소프트웨어 현황분석

- 조직의 업무 및 환경 요건을 분석하고 조직의 전반적인 현황을 파악한 후 공개소프트웨어정보와 공개소프트웨어 기반기술을 활용할 정보시스템 구조를 수집
- 조직에 새로운 정보시스템이 필요한지를 결정하고 정의된 정보화 목적을 달성하기 위한 공개소프트웨어 도입 전략을 개발하기에 앞서 조직의 공개소프트웨어 관련 현황에 대한 이해 필요
- 조직의 정보화계획수립 프로젝트의 범위와 초점에 따라 현행 정보 및 시스템, 정보서비스 조직 및 관리관행, 정보기술 등에 대해 검토 하고 사용자의 요구사항 파악
- 조직 업무 및 응용시스템의 공개소프트웨어 적용 가능성 조사
- 현행 정보기술 및 자원 분석(하드웨어, 네트워크 등 시스템 구조 포함)
- 조직의 공개소프트웨어 보유 및 사용현황을 분석
- 현행 정보서비스 조직 및 관리관행을 공개소프트웨어 관점에서 분석
- 현황 평가를 통한 문제점의 식별 및 사용자 요구사항 수집

- 공개소프트웨어 기반 정보시스템의 도입 또는 전환가능 여부를 분석하기 위하여 조직의 정보시스템이 보유한 공개소프트웨어 현황파악이 필요
- 하드웨어가 공개소프트웨어 도입이 가능한지 현황파악 필요
- 조직의 업무시스템 현황을 분석하여 공개소프트웨어 도입이 가능한지 조사 필요
- 조직의 공개소프트웨어 보유 및 사용현황을 분석할 때 공개소프트웨어 도입에 적합한 비유계획의 수립을 위해서 반드시 공개소프트웨어 관련 라이선스의 유형파악이 필수
- 공급자에게 라이선스 위반에 대비한 소프트웨어저작권 협약서를 반드시 첨부하여 정보시스템을 구축하도록 계획

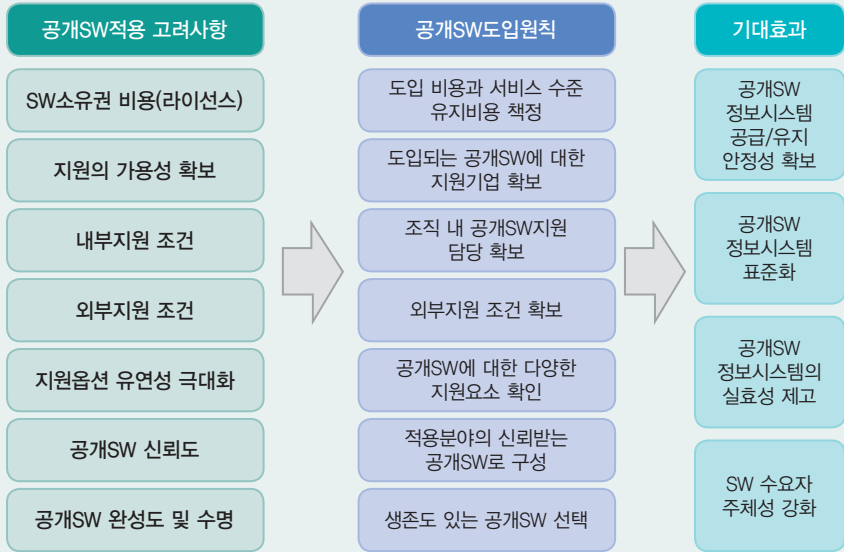


[그림 6] 공개소프트웨어 현황분석

3) 공개소프트웨어 도입원칙 수립

- 공개소프트웨어 도입 원칙은 이후 이루어질 추진과제 정의 및 공개소프트웨어 전환 계획 수립에 대한 원칙으로 안정적인 공개소프트웨어 활용을 위한 준수 사항이 된다. 공공기관 · 기업 등이 공개소프트웨어의 불안함을 해소하게 하기 위해 반드시 준수해야 할 원칙 항목들을 수립하는 단계
- 공개소프트웨어 정보시스템의 근간이 되는 공개소프트웨어 기반 정보 구조를 개발하기 위한 준비 작업으로 이전 활동에서 파악된 경영전략과 요구사항, 정보 및 시스템 요건을 효과적으로 지원할 수 있는 공개소프트웨어 정보시스템 구조의 전반적인 방향과 원칙을 수립

- 조직 전략과 요구사항, 공개소프트웨어 적용 고려 사항, 정보시스템을 효과적으로 지원할 수 있는 정보를 파악하고 시스템의 기본방향과 체계를 수립하며, 시스템 구축을 위한 도입원칙 정의



[그림 7] 공개소프트웨어 도입원칙 수립

(가) 라이선스 비용

대부분의 공개소프트웨어는 관련 라이선스 수수료가 없다. 개발업체들은 구현 및 지원 서비스를 제공하여 수익을 창출하고 있다. 일부 개발업체들은 라이선스를 부여받은 번들로 판매되는 공개소프트웨어와 비공개소프트웨어를 조합한 형태를 활용하여 솔루션을 판매하고 있다. 공급자들은 공개소프트웨어 제품을 서비스 수준 지원 계약과 함께 제공하고 있으므로 공개소프트웨어 도입에 관심이 있는 기관들은 이러한 형태의 가치를 평가해야 한다. 공개소프트웨어 적용의 일반적 방법은 하나의 기업으로부터 기본적인 제품을 획득한 후, 선택적인 전문위탁 과정을 통해 개별적으로 현재의 서비스 수준 지원 계약을 협상하는 것이 훨씬 더 비용 효과적이다.

(나) 지원의 가용성

공개소프트웨어의 도입과 관련하여 가장 빈번하게 나타나는 질문은 지원에 집중되어 있다. 도입을 고려하는 조직은 이 문제를 파악하여 다양한 소싱 옵션과 관련된 위험을 적절히 평

가해야 한다. 모든 공개소프트웨어는 사용자 공동체에 의해 유지되고 지원되므로 공개소프트웨어에 대한 지원은 임시 변동적이고 서비스 수준에 대한 보장이 없지만, 도입대상 공개소프트웨어의 시장이 존재하는 경우 주요 공개소프트웨어 제품을 지원할 수 있도록 다수의 개발업체들이 빠르게 대응하고 있다. 공개소프트웨어 제품에 대한 지원은 일반적으로 상용소프트웨어와 유사한 방법으로 제공되고 있다. 대부분의 개발업체와 판매자들은 서비스 수준 계약, 헬프데스크 전화지원 서비스 및 지원 패키지 구매를 제공하고 있으며 국내에는 상업적 조건에서 업계와 정부에게 지원을 제공해주는 중소기업 개발업체들이 몇 백개 규모로 갖추어져 있다.

(다) 내부 지원

내부 기술직원이 공개소프트웨어를 배포하고 유지·관리하는데 필요한 기술보유 또는 업무전담을 하고 있을 때 해당 조직은 공개소프트웨어의 적용을 결정할 수 있다. 하지만 위험 경감을 위해 내부 자원의 제한에 대해 인식하고 있는 것이 중요하다. 필요에 따라서는 외부 전문가의 지원을 요청할 계획을 세울 수 있다.

(라) 외부 지원

공개소프트웨어 도입 지원을 외부에 위탁하기로 결정한 경우, 기술 지원은 일반적으로 선정한 개발업체의 책임이다. 개발업체는 1차 및 2차 수준의 지원을 제공해야 한다. 또한, 개발업체는 3차 수준(버그 수정)의 문제가 해결될 수 있도록 소프트웨어를 제작한 개발자 공동체와 필요한 협력 및 대응을 수행해야 한다. 조직의 관점에서, 전체적인 과정은 상용소프트웨어 개발업체와의 계약과 유사해야 한다.

(마) 지원 옵션의 유연성 극대화

공개소프트웨어는 개발, 배포, 공급 및 라이선스 부여 계획을 통해 지원 옵션에 대한 상당한 유연성을 제공한다. 다수의 인기 있는 공개소프트웨어들은 경쟁적인 개발업체들로부터 제공되는 보다 광범위한 지원 서비스를 보유하고 있다. 각각의 개발업체들은 헬프데스크, 문제해결, 버그 수정 서비스를 비롯한 지원을 클라이언트들에게 제공할 수 있다. 이러한 다양한 방식의 공급 모델은 공개소프트웨어와 전통적인 소프트웨어간의 중요한 차이점으로서 유연성 있는 지원 옵션은 수요자에게 광범위한 선택의 폭을 제공하여, 공개소프트웨어의 도입으로 보다 많은 가치를 획득할 수 있게 해 줄 수 있다.

(바) 소프트웨어 신뢰도

공개소프트웨어에 대한 일반적인 고려사항은 품질 문제와 관련이 있다. 대부분의 공개소프트웨어들이 라이선스 비용 없이 사용이 가능하지만 조직은 그러한 소프트웨어를 신뢰할 수 있는지에 대해 신중히 판단해야 한다. 임의의 한 카테고리에는 수 백 개의 이용 가능한 공

개소프트웨어들이 있고 분야별 주요 공개소프트웨어들은 높은 완성도와 안정적인 성능을 바탕으로 광범위하게 사용되고 있지만 상당수는 바람직한 품질 및 신뢰도 수준에 도달하지 못하고 있기 때문에 신중한 선택이 필요하다.

(사) 완성도 및 수명

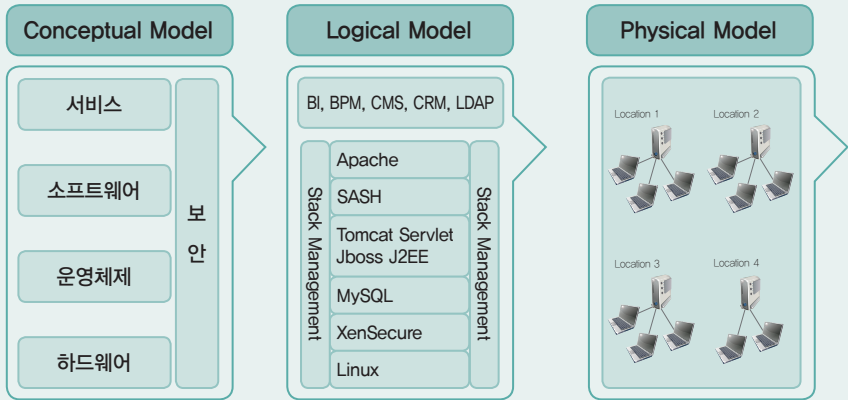
공개소프트웨어에 사용되는 사용자 참여에 의한 개발모델은 사용이 증가함에 따라 완성도와 신뢰도가 향상되는 피드백 기반의 개발 모델이다. 공개소프트웨어 수명주기의 초기 단계에서만 보면, 대부분의 공개소프트웨어 개발 공동체는 “빠르게 배포하고 자주 업데이트”라는 배포 철학을 가지고 있으므로 공개소프트웨어가 상용소프트웨어의 배포 버전보다 강력하지 못하며 신뢰도가 떨어질 수 있지만, 이러한 제품의 초기 버전들의 완성도가 전체적인 공개소프트웨어 완성도를 의미하는 것이 아님을 주의해야 한다. 성공적인 공개소프트웨어 프로젝트는 개발자 및 사용자 공동체의 지원이 뛰어나 소프트웨어 코드베이스, 문서, 웹사이트, 메일링 리스트 지원 및 토론 포럼 등 다양한 품질향상을 위한 활동이 자발적으로 전개된다. 대규모 사용자 공동체를 보유한 생존력 있는 공개소프트웨어를 선택하는 것은 매우 중요하다. 발전하고자 노력하는 과도기적 제품을 선택하는 것은 심각한 위험을 초래할 수 있으므로, 도입대상 공개소프트웨어의 분석을 통해 소프트웨어 완성도와 현재 수준을 판정하는 것이 중요하다.

4) 공개소프트웨어 추진과제 정의

- 전 단계에서 수행한 현황분석과 수립된 공개소프트웨어 도입원칙에 근거로 조직의 전략, 사용자의 요구사항과 공개소프트웨어기반 정보기술전략을 반영하여 개념모델 작성하고, 개념모델의 논리적 구체화 및 물리적 구체화 과정을 거쳐 최종 목표모델을 결정하는 과정
- 업무기능의 충족을 위해 공동으로 작용하는 데이터, 응용, 기술 및 조직 간 관계를 정의
- 데이터, 응용, 기술, 조직의 네 가지 구성요소로 구분
- 구조를 적절히 정의함에 따라 공개소프트웨어 정보시스템 구축 실패의 가능성을 최소화하고 성능, 통제, 복구 및 재개 등 공개소프트웨어 정보시스템 구축에 위험성이 큰 쟁점들의 해결 가능

5) 공개소프트웨어 도입계획 수립

- 목표모델로 설정된 공개소프트웨어 정보화사업의 수행에 필요한 정보기술 전략의 요약, 구현 계획 요약, 보안 계획, 이행 계획, 지속적 계획수립 접근방법 등에 대한 내용으로 구성
- 공개소프트웨어 정보화사업의 수행에 필요한 공개소프트웨어 기술 전략의 요약, 공개소프트웨어 정보시스템 구현 계획 요약, 보안 계획, 이행 계획 등에 대한 내용으로 구성



[그림 8] 공개소프트웨어 추진과제 정의

- 개념적 모델(Conceptual Model) : 사용자의 요구사항과 공개소프트웨어기반 정보기술 전략을 반영하여 개념모델 작성
- 논리적 모델(Logical Model) : 개념모델의 논리적 구체화
- 물리적 모델(Physical Model) : 물리적 구체화 과정을 거쳐 최종 목표모델 수립
 - 정보화기본계획에 따라 실제로 공개소프트웨어 정보시스템을 구축하는데 필요한 실행 항목을 도출하고 각각에 대한 비용/효과 및 자원요건을 분석한 다음 우선순위를 설정하여 세부 일정계획을 작성
 - 공개소프트웨어 정보시스템 구성요소를 구현하는데 필요한 작업, 데이터관리, 개발 및 유지관리 접근방법을 모두 통합하여 데이터 및 응용계획을 작성
 - 기술계획은 공개소프트웨어 기술 관점에서 데이터 및 응용구조 및 계획에서 파악된 응용시스템 요건과 기존의 정보처리환경에 부합되는 하드웨어, 시스템 소프트웨어, 데이터베이스 및 통신에 대한 자원 요건을 요약하고 공개소프트웨어 정보시스템 구성요소를 구현하는데 필요한 정책, 비용 및 실행항목들을 포함
 - 조직계획은 해당 조직이 공개소프트웨어 정보기술 전략을 구현하는데 필요한 조직 요건, 정책, 절차들을 포함하여, 조직 구조를 구현하는데 필요한 실행항목을 요약
 - 데이터 및 응용계획, 기술계획, 조직계획이 작성되면 이러한 계획들을 구현하는데 필요한 이행계획 작성 필요

(2) 정책수립

정책수립이란 기업 가치를 극대화시키고 저작권 침해 및 컴플라이언스 위험을 극소화시키기 위하여 기업 내의 공개소프트웨어 활용과 기업 간의 협업 체계에 대한 지침을 수립하는 활동이다.

이 과정은 조직 수준 진단결과를 바탕으로 해당 조직이 효과적으로 사내 정책을 수립하고 전개할 수 있는 포괄적인 지침을 제공하는 데에 목적이 있으며 공개소프트웨어를 전략적 자원으로 활용하는 기준을 수립함으로써 조직의 사업전략과 로드맵에 매우 중요한 역할을 수행하게 된다.

정책수립의 과정은 조직 내에 최적화된 공개소프트웨어 거버넌스를 구축 운영하기 위해서는 조직의 사업, 전략, 공개소프트웨어의 의존성 등을 파악하여 조직내부의 명확한 현황을 진단하는 것으로부터 시작하여, 공개소프트웨어 관리를 위한 조직진단 결과를 기반으로 조직 전반에 공개소프트웨어가 어떻게 관리되고 사용되어야 하는지에 대한 방향과 절차를 명시하여 조직 내에 최적화된 공개소프트웨어 관련 정책을 수립하고, 이 정책에 부합되는 공개소프트웨어 활용을 위해 주기적으로 공개소프트웨어 활용 실태 파악을 위한 검증, 교육, 모니터링과 정책운영을 위한 효과적 전개 지침을 포함하여야 한다.

정책수립을 통하여 공개소프트웨어의 선택, 승인, 사용과 운영에 관련된 조직 프로세스의 기반을 제공하게 되며, 모든 조직과 구성원이 공개소프트웨어 사용 의무사항들을 지속적으로 검토하고 확인할 수 있는 기반 제도 및 절차를 제공함으로써 공개소프트웨어와 관련된 현재 및 잠재적 위험요인을 최소화 할 수 있다.

공개소프트웨어 사용 및 절차를 공식화하기 위해서는 정책이 문서화되고 수립되어지는 체계적인 조직 프로세스가 필요하다. 구체적으로 정책을 문서화 하지 못하면 조직은 일관성 없는 공개소프트웨어 사용관리와 위험노출을 증가 시킬 수 있기 때문에 공개소프트웨어 정책수립 시 다음의 세 가지 구성요소를 반드시 포함해야만 한다.

1) 공개소프트웨어 사용 시 조직이 준수해야 하는 준법성 요구사항

- 공개소프트웨어 라이선스 목록
- 공개소프트웨어 라이선스가 소프트웨어 사용자들에게 부여하는 권한과 요구하는 의무사항들
- 공개소프트웨어 사용 유형에 따른 다양한 라이선스 의무사항들
- 공개소프트웨어 정책은 개인 구성원들이 개별적으로 라이선스를 해석하고 이해하고 판독하는 것 보다는 개별 라이선스에 대한 조직의 의무사항들을 식별해야 한다.
- 공개소프트웨어 정책은 조직의 독점 소프트웨어에서 분리되는 카피레프트

(copyleft)¹⁰⁾ 라이선스 소프트웨어에 대한 요구사항을 식별해야 한다.

- 조직이 라이선스에서 요구하는 의무사항들을 준수하기 위한 문서화된 의무사항들의 지침이 제공되어야 한다.

2) 공개소프트웨어 사용 혹은 활동을 위한 승인 절차

- 공개소프트웨어 라이선스 요구사항들을 준수하고 이해하기 위해서는 정책에 일반적인 승인 프로세스를 포함해야 한다.
- 승인 프로세스에는 구성원들이 아래의 경우에 어떠한 승인이 필요한지가 서술되어야 한다. 많은 조직들이 자동화된 승인 프로세스를 위해 공개소프트웨어 사용 요청방식을 온라인으로 지원하고 있다.
 - ① 공개소프트웨어 컴포넌트를 사용하기 위해
 - ② 외부 공개소프트웨어 프로젝트에 참여하기 위해
- 요청방식은 코멘트 뿐 아니라 승인·거부를 포함하며, 프로세스는 개별 엔지니어들에 의해 생성되어 직속 관리자들에 의해 검토 및 승인된다.
- 개별 관리자들에 의해 승인된 요청은 시니어 엔지니어링 관리자에 의해 검토되고 최종적으로 공개소프트웨어 리뷰 보드(Review Board)에 의해 승인된다.
- 정의된 승인 프로세스의 구축은 공개소프트웨어의 효과적 사용과 중요한 준법성 요구사항들을 준수하기 위한 기본 인프라 구축단계이다.
- 승인 프로세스의 구축은 개별 엔지니어들과 엔지니어링 관리에 필요한 부담을 덜어 주고 공개소프트웨어 검토에 대한 업무를 단순화 해 줌으로서 개발생산성을 증가시킨다.

3) 공개소프트웨어 관리를 위한 조직 자원의 역할과 책임 정의

- 조직의 특성에 따라 공개소프트웨어 승인가구와 조직의 전문 지식 자원을 제공할 공개소프트웨어 리뷰 보드(Review Board)를 구성해야 한다.
- 공개소프트웨어 검토위원회(OSRB, Open Source Review Board)는 사전에 논의된 공개소프트웨어 사용 요청들을 받고 요청관련 의사결정에 따라 피드백을 제공해야 한다.
- 개발자에 의한 공개소프트웨어 사용에 대한 의사결정은 일반적으로 거부되지 않도록 공개소프트웨어 사용에 대한 고려사항을 사전에 문서화 하여 배포해야 한다.
- 개발자에 의한 공개소프트웨어 사용에 대한 요청이 즉시 승인될 수 없다면, 공개소프트웨어 검토위원회는 공개소프트웨어 사용 형태에 따라 일부 변경 및 검토 후 승인되어 질 수 있도록 공개소프트웨어 사용에 대한 권장사항을 제공해야 한다.

10) 카피레프트(Copyleft) : 카피라이트(Copyright)의 언어적 유희로서 소스코드를 공개하는 한 자유롭게 복제, 개작, 배포가 가능한 저작권의 반어적 표현을 의미함.

정책수립 단계를 통해 조직의 사업전략과 공개소프트웨어 라이선스, 조직의 권리와 의무사항들에 대한 구성 요소가 공개소프트웨어 정책의 형태로 통합되어야 하며, 공개소프트웨어 정책에 기반하여 조직 및 구성원들이 공개소프트웨어 라이선스 요구사항을 인지하게 하고 모든 구성원들이 공개소프트웨어 사용을 위해 적합한 요구사항을 준수하기 위한 절차를 수립하고, 조직 전반에 어떻게 공개소프트웨어가 관리되어 질 것인가를 정의한 문서화된 공개소프트웨어 정책을 정의한다. 이 정책에 포함되어야 하는 항목들은 다음과 같이 예로 들 수 있다.

〈표 2〉 공개소프트웨어 정책 예시

정책예시	설 명
공개소프트웨어 사용	각 소프트웨어에서의 공개소프트웨어 사용 범위
소스코드 공개	소스코드 공개 범위, 자체 개발 소스코드, 소스코드 공개 방법
라이선스 검증	검증 방법, 라이선스 별 조치 대상, 외부 공개소프트웨어 라이선스 충돌의 해결
외부 커뮤니티 활동	공개소프트웨어 관련 외부 커뮤니티 활동을 지원하기 위한 정책
커뮤니티 생성 및 운영	공개소프트웨어를 위한 커뮤니티 수립 방안 및 이를 관리하기 위한 정책
공개소프트웨어 라이프 사이클	공개소프트웨어 도입에서 운영, 유지보수, 폐기에 이르는 전체 라이프 사이클을 관리하기 위한 절차
전사 R & R	공개소프트웨어 라이프사이클에서의 각 조직의 역할 및 책임 정립
공개소프트웨어 평가	유사 항목에서의 우수 공개소프트웨어를 선택하기 위한 평가 모델
공개소프트웨어 교육	공개소프트웨어를 도입, 사용하기 위한 교육
주요 라이선스별 적용 방안	공개소프트웨어 라이선스별로 안전하게 사용 및 적용하기 위한 정책
조직별 공개소프트웨어 활용 방안	조직의 특성에 따라 공개소프트웨어를 도입, 개발 또는 유지보수하기 위한 정책
공급업체의 공개소프트웨어 라이선스 의무사항	공개소프트웨어를 공급 받을시 공급업체로부터 받아야 하는 라이선스 준수 정책

공개소프트웨어 정책수립에는 제한이 없다. 사용하는 기업의 특성과 사업 특성에 따라 얼마든지 다양하게 정책을 수립할 수 있고 언제든지 변경이 가능하다. 일반적으로 통용될 있는 정책을 예시한다면 다음과 같다.

① 일반적인 정책

- 일반적으로 사용 주체의 목적에 부합되는 한 공개소프트웨어를 최대한 사용하여야 한다.
- 이는 공개소프트웨어를 통해 최대한 개발 역량 및 개발 생산성을 향상시키고 제품의 품질 수준을 높이며 개발 원가를 절감하고 수주 경쟁력을 확보해야 하기 때문이다.
- 공개소프트웨어를 도입, 사용, 배포할 경우 해당 부서장 또는 PM은 공개소프트웨어 전담부서에 이를 통보하고 확인을 받아야 한다.
- 공개소프트웨어 전담부서는 공개소프트웨어의 사내 인증을 수행하며 인증된 공개소프트웨어 목록을 주기적으로 또는 현장 요청에 따라 업데이트한다.
- 보통 사용 주체에서 인증된 공개소프트웨어는 공개소프트웨어 전담부서가 운영 및 관리하는 저장소와 포탈을 통해 제공되어야 한다.
- 공개소프트웨어의 라이선스 의무사항을 빠짐없이 준수하여 분쟁, 소송, 처벌 등 법적 위험을 사전에 제거하여야 한다.
- 공개소프트웨어를 사용할 때는 공개소프트웨어 라이선스의 공통 의무사항과 라이선스 별 의무사항을 준수하여야 한다.
- 공개소프트웨어 라이선스의 의무사항은 별도의 공개소프트웨어 라이선스 가이드라인을 통해서 상세히 설명하여야 한다.
- 프로젝트 또는 과제에서 사용될 공개소프트웨어의 라이선스는 프로젝트 시작 단계에서 확인한다.

② 특정 상황별 정책다음은 일반적인 내용이 아니라 특정한 상황에서 구체적으로 수립해야 하는 정책에 대해서 소개하고자 한다. 먼저 사용 주체의 독점 프로젝트를 공개소프트웨어로 변경하고자 할 때 수립할 수 있는 정책에 대한 예를 들면 다음과 같다

- 사용 주체 내에서 개발한 소프트웨어를 공개소프트웨어로 변경하고자 할 경우에는 공개소프트웨어 전담부서 또는 책임 임원과의 협의를 거쳐 라이선스를 결정한다.
- 개방화를 통한 시장 확대, 표준 선도, 독과점 견제, 기업 이미지 고취 등 특별한 목적이 있을 경우에만 사용 주체에서 개발한 소프트웨어를 공개소프트웨어로 변경할 수 있다.
- 독점 소프트웨어를 공개소프트웨어로 변경 시에 협업과 개작의 회수를 목적으로 할 경우에는 배포 후 개작 내용을 알 수 있도록 반환의 의무가 있는 라이선스(AGPL, GPL, LGPL, MPL, EPL, IBM, CPL, OSL, QI)를 채택하고 사업적 이윤추구를 위해서는 이중 라이선스(Dual License)를 사용하는 경우가 많으며, 확산과 활성화를 목적으로 할 경우에는 Apache, BSD, MIT 와 같이 허용적 라이선스(Permissive License)를 선택하는 것이 일반적이다.

- 공개소프트웨어로 변경 시 모든 소스코드에 저작권 관련 문구를 파일 단위로 기재한다.
- 사용 주체가 핵심 기술, 특허, 영업비밀 등 전략적인 지적재산으로 보존하는 부분은 공개소프트웨어로 변경하지 않는다.
- 공개소프트웨어 라이선스 충돌이 제거되지 않은 상태에서는 해당 소프트웨어를 공개소프트웨어로 변경하거나 배포하지 않는다.

③ 소스코드를 공개하기로 결정한 상황에서 부가적으로 수립할 수 있는 정책은 다음과 같다.

- 공개소프트웨어로 공개할 경우에는 인터넷을 통해서 적극적으로 배포한다.
- 공개소프트웨어 공개 시 회원/비회원 같은 차별적 제약을 두지 않는다.
- 공개소프트웨어 라이선스에 따른 공개의 범위에 유의하며 공개소프트웨어 전담조직은 공개에 대한 가이드와 시스템을 제공한다.
- 소스코드 공개의 의무조항이 명확하지 않은 공개소프트웨어 라이선스에 대해서는 공개 이전에 공개소프트웨어 전담조직 등에 문의하여 공개 여부와 범위를 확인한다.
- 원저작자에게 공개의 의무가 있는 공개소프트웨어를 배포할 경우에는 사전에 이메일을 통해서 공개용 URL(Uniform Resource Locator) 정보를 제공한다.
- 공개소프트웨어를 탑재한 제품을 출하할 경우에는 제품 매뉴얼이나 화면 메뉴에 공개소프트웨어 라이선스 정보와 소스코드 공개용 URL 정보를 제공한다.
- 전략적인 이유에서 소스코드의 공개를 지연시켜야 경우에는 written offer¹¹⁾로 대신한다.

④ 내부 프로세스에 대한 정책을 수립하는 경우에는 다음과 같이 공개소프트웨어의 흐름에 따라 수립할 수 있다.

- 공개소프트웨어 프로세스는 현행 부담의 최소화 원칙에 따라 효율적으로 수립한다.
- 공개소프트웨어 프로세스는 기존의 프로세스와 연계하여 운영한다.
- 공개소프트웨어 전담조직은 공개소프트웨어 프로세스를 수립하고 운영하고 개선하는 책임을 진다.
- 공개소프트웨어 프로세스를 전사 업무프로세스와 개발방법론에 반영한다.
- 공개소프트웨어 프로세스를 개발방법론에 반영하되 사업부 또는 조직의 특성에 따라 테일러링한다.
- 개발 단계에서 신고 없이 유입되는 공개소프트웨어의 라이선스 위험을 현장이 즉각적으로 발견하고 제거하기 위해서, 소프트웨어 개발자가 직접 공개소프트웨어 라이선스 검증을 수행하고 문제를 해결하는 프로세스를 정립한다.
- 요청 · 수행 · 종료 절차에 준하여 공개소프트웨어 도입 및 활용, 과제 생성 및 실행, 라이선스 검증, 공개소프트웨어 교육, 소스코드 공개, 조치 결과 보고 등의 지원업무는 반

11) Written Offer : 소스코드 제공을 요청할 수 있는 담당자 이메일등의 정보를 매뉴얼 등에 서면으로 제공함.

드시 공개소프트웨어 프로세스로 정립한다.

- 공개소프트웨어 프로세스의 실행 감리 및 결과 확인의 책임을 진다.

⑤ 공개소프트웨어의 저작권 침해와 라이선스 위반을 예방하기 위한 정책수립의 예는 다음과 같다.

- 소스코드 반환의 의무가 있는 공개소프트웨어 라이선스를 중점 관리 대상으로 한다.
- 중점 관리 대상은 AGPL, GPL, LGPL, MPL, EPL, CPL, OSL, Qt 등으로 정한다.
- GPL 계열의 라이선스(AGPL, GPL, LGPL)는 강력한 공개 의무를 요구하고 특허권의 행사도 제한하고 있으며, 국제적으로 활동하고 있는 감시 기관에 의해 저작권 소송 사례가 발생하고 있는 만큼 최우선적으로 모니터링하고 완벽하게 조치한다.
- GPL 계열의 의무사항을 위반한 소프트웨어는 절대로 외부 배포하지 않는다.
- 상용 프로그램과 AGPL 또는 GPL 라이선스의 공개소프트웨어를 연동하여 사용하지 않는다.
- 소스코드 반환의 범위를 명확히 하여 불필요한 소스코드의 공개가 발생하지 않도록 한다.
- 모든 공개소프트웨어의 유입과 배포 단계에서 라이선스 정보를 기록하고 보관한다.
- 공개소프트웨어를 상용 제품과 같이 사업에 이용할 경우에는 공개소프트웨어 라이선스와 기업의 비즈니스를 위한 상용 라이선스를 동시에 충족하는 이중 라이선스(Dual License)를 적용할 수 있다.

⑥ 공개소프트웨어의 라이선스 검증에 관련된 정책의 사례는 다음과 같다.

- 공개소프트웨어 라이선스 검증 대상은 사외로 배포되었거나 배포를 목적으로 개발된 모든 소프트웨어이다.
- 사외로 배포되는 모든 소프트웨어는 전수 검사한다.
- 협력사가 제공한 소프트웨어도 외부 배포 예정이면 라이선스 검증의 대상이다
- 개발 중에 있는 프로젝트에서는 코딩 단계에서 개발자가 직접 라이선스를 검증한다.
- 공개소프트웨어 라이선스의 위험 발생 가능성과 심각도에 따라 관리해야 하는 조직을 구분한다.
- 중점 관리 대상에는 단말기나 디바이스에 탑재되는 임베디드 소프트웨어, 해외 수출용 솔루션, 공공과 금융 시장에 납품되는 소프트웨어 개발에 관련된 조직 등이 속한다.
- 중점 관리 대상의 우선순위는 조치의 시급성, 인력 규모, 시간 등 다양한 제약 조건에 따라 등급을 결정한다.
- 시스템 유지보수와 시스템 통합 소프트웨어 개발 조직은 위험 발생 가능성과 심각도가 낮기 때문에 일반 관리 대상으로 분류할 수 있다
- 외부로 배포되지 않는 연구 프로젝트의 결과물 또는 개발 툴 등은 법적 이슈가 없으므로 관리 대상에서 제외된다.

- 공개소프트웨어 라이선스 검증 부담을 최소화시키기 위해서 예외 사항 또는 자동 필터링 규칙을 규정할 수 있다.
- 자동 필터링 규칙은 공개소프트웨어 검증 시 혹은 검증 작업 이전 단계에서 검증 대상이 아닌 소프트웨어 혹은 파일을 사전에 제외시킴으로써 검증 작업 시간을 줄이는데 목적을 둔다.

⑦ 공개소프트웨어 라이선스를 검증한 결과를 조치하는데 다음과 같은 정책을 수립할 수 있다.

- 공개소프트웨어 라이선스 검증 결과는 보안 사항이므로 대외비로 구분시킨다.
- 공개소프트웨어 라이선스 검증 결과는 공개소프트웨어 전담조직이 사업부 또는 부서 단위로 총괄해서 관리한다.
- 공개소프트웨어 전담조직은 공개소프트웨어 라이선스 검증 결과의 보고와 현장의 조치 현황 파악에 책임을 진다.
- 공개소프트웨어 라이선스 위반에 대한 조치는 현장 책임지도록 한다.
- 배포를 목적으로 개발된 소프트웨어는 출하 이전에 라이선스 위반 문제를 해결한다.
- 공개소프트웨어 라이선스 위반이 발견되면 중점 관리 라이선스부터 우선 조치한다.
- 중점 관리 대상 조직은 공개소프트웨어 라이선스 위반 사항을 즉각적으로 조치한다.

⑧ 공개소프트웨어를 관리하기 위한 조직을 구성하기 위해서 수립할 수 있는 정책은 다음과 같다.

- 공개소프트웨어 전담조직을 통해 전사적 통제 및 관리 업무를 수행한다.
- 현업은 공개소프트웨어 전담조직과의 원활한 업무 연락과 협업을 위해서 사업부 별로 공개소프트웨어 라이선스 관리 담당자를 임명한다.
- 공개소프트웨어 라이선스의 최종적인 법적 해석은 법무팀이 수행한다.
- 업무혁신그룹은 전사 업무프로세스 내 공개소프트웨어 프로세스를 검토하고 확정한다.

⑨ 공개소프트웨어를 전담하는 조직에 대한 정책으로 다음과 같은 역할을 부여할 수 있다.

- 전사 공개소프트웨어 정책수립 및 업데이트
- 전사 업무프로세스 및 개발방법론 개선
- 교육 교재 작성 및 업데이트
- 교육 강의 및 실습 지원
- 공개소프트웨어 라이선스 검증 지원
- 검증 유틸리티 개발 및 유지보수
- 이메일 전담 창구 운영
- 공개소프트웨어 라이선스 점검 툴 운영

⑩ 공개소프트웨어를 전담하는 조직에 대한 정책으로 다음과 같은 책임을 부여할 수 있다.

- 공개소프트웨어 라이선스 검증을 위한 현장 지원
- 전사 공개소프트웨어 라이선스 사용 및 위반 현황 파악
- 라이선스 검증 시스템 관리 및 운영

⑪ 공개소프트웨어의 법적인 이슈에 관련된 정책은 다음과 같이 수립할 수 있다.

- 라이선스에 관한 법리적 해석
- 전사적인 공개소프트웨어 라이선스 적용 지침 가이드
- 라이선스 사용의 적법성 검토
- 라이선스 검증에 대한 최상위 자문 수행

(3) 조직구성

조직구성이란 공개소프트웨어에 관련된 조직 간의 역할 정의 및 분담, 책임 소재 명시, 협업 체계 수립 등 단체 인력의 집합적 활용 활동을 의미한다. 공개소프트웨어를 도입하거나 활용하는 조직은 공개소프트웨어의 도입 및 활용의 목적과 조직과 사업 관점에서 공개소프트웨어의 의존성을 파악하여 조직의 수준에 부합된 공개소프트웨어 거버넌스 프레임워크를 구축·운영하여야 한다.

공개소프트웨어를 사용하는 조직이 목표와 전략에 부합하는 활동을 수행하기 위해서는 각 조직의 특성에 맞게 차별화된 조직운영 계획을 수립할 필요가 있다. 이를 위하여 조직 설계 및 운영 부문에서는 조직 설계 및 운영을 위한 고려사항, 조직 설계의 절차, 공개소프트웨어 거버넌스를 위한 조직 모델, 조직 운영에 대한 내용을 제공한다.

1) 공개소프트웨어 거버넌스 조직 설계 및 운영을 위한 고려사항

(가) 비즈니스 전략

조직의 변화를 통한 성공적인 공개소프트웨어 거버넌스를 위해서는 먼저 공개소프트웨어 비즈니스의 전략을 결정해야 한다. 그런 다음 그 전략을 실행할 수 있는 어떤 조직이 필요한지를 설계할 수 있고 적절한 조직 운영이 가능하다. 공개소프트웨어 비즈니스 전략이란 기업이 공개소프트웨어 자원을 어디에 집중할 것인지, 어디에서 경쟁할 것인지, 경쟁우위를 지속하기에는 어떤 행동이 필요한지를 서술하는 것이며 성공적인 비즈니스 전략은 기업의 비전으로부터 도출하게 된다. 따라서 해야 할 일을 명확하게 알지 못하는 채로 전략을 수립하는 것은 무의미하며 먼저 비전을 명확하게 정의하는 것이 필요하다.

12) 출처 : "Beyond Entrepreneurship", James C. Collins, William C. Lazier, Prentice Hall, 1992

기업의 모든 구성원이 스스로 주도적인 의사결정을 하고 자신의 역량을 집중하는 성공하는 기업이 되기 위해서는 기업의 비전이 반드시 필요하다. 기업의 비전은 핵심가치와 믿음, 목적, 사명으로 구성¹²⁾ 되어 있으며 모든 사람이 공유하는 비전은 의사결정의 기준이 되고 구성원에게 업무 동기를 부여할 수 있다.

기업의 비전에 적합한 전략을 수립할 때는 기업이 가진 환경의 위협과 기회에 대한 분석, 자사의 강점과 약점의 확인, 시장과 경쟁 상황에서의 심층적인 통찰을 기반으로 목표와 자원의 배분을 조직이 시장 기회에 적합하게 대응할 수 있도록 수립해야하고, 목표에 맞게 전술과 통제 메커니즘을 수립하는 활동을 포함해야 한다. 고려해야할 요소는 다음과 같다.

- 제품 또는 서비스 : 생산 전략과 서비스 전략
- 고객 또는 시장 구성 : 서비스하고 있는 고객과 그들에게 다가갈 수 있는 방법
- 자금흐름
- 인력과 조직
- 인프라

아울러 성공적인 전략을 수립하기 위해서는 기업의 비전을 명확하게 하고, 기업이 가진 역량을 내부적으로 냉정하게 평가해야 한다. 또한 시장, 경쟁 업체, 업계의 추세 등 사업 환경을 철저하게 평가하여 자원분배와 제품의 개발수준을 결정해야 한다.

(나) 조직의 필수요소

기업의 적절한 전략이 수립되면 다음의 행동은 새로운 전략을 위하여 무엇이 필요인지 정하는 것이다. 공개소프트웨어 거버넌스를 위하여 기업이 바라는 인재상, 기업의 리더들이 갖추어야 하는 역량과 태도, 조직 구성원들의 업무 관련 능력과 스킬 등 기업의 전략을 수행하기 위한 조직의 필수요소를 도출해서 조직 설계에 반영해야 한다.

(다) 조직구조

조직구조는 공개소프트웨어 거버넌스를 위한 요소 중 매우 중요한 항목 중 하나이다. 조직의 변혁은 조직구조의 변화에서 시작되며 조직구조는 기업의 상황과 특성에 맞춰 기능조직, 사업부조직, 학습조직, 프로젝트조직, 매트릭스조직, 팀조직, 네트워크조직 등 다양한 유형으로 구성된다. 기본적 조직구조의 형태는 어떤 모형이 적절한, 조직은 어떻게 집중화되고 나누어져야 하는지에 대한 검토가 필요하다. 공개소프트웨어 관점에서는 조직의 구조를 유연하고 수평적으로 가져가야 한다. 이는 공개소프트웨어 관련 조직이 본받아야 하는 조직구조가 공개소프트웨어 커뮤니티이고 창의적 활동이 가능한 구조적 특성을 가지고 있기 때문이다.

(라) 조직 운영 시스템

조직 운영 시스템이란 많은 구성원이 해야 할 일이나 결정을 내려야 할 주요 문제를 판별하기 위한 양식 또는 과정이라고 할 수 있다. 조직 설계 및 운영 시 인사관리 제도와 운영방식, 의사소통 체제, 부서/팀 간, 부서/팀 내의 업무 협조와 팀워크 등의 요소도 조직 운영 시스템의 일부로 반드시 검토되어야 한다. 이처럼 조직 운영 시스템 관점에서 공개소프트웨어 업무를 추진하고 있는 조직이 공개소프트웨어에 관련된 미션과 목표를 달성하기 위해서는 기존과는 다른 운영 방식을 적용하여야 한다. 무엇보다도 평가와 보상 방법이 달라져야 한다. 즉, 내부적으로 공개소프트웨어에 관련된 활동을 업무로 보장하고 그 결과를 평가와 보상에 반영할 수 있어야 한다. 또한 커뮤니티 참여와 대외적인 활동이 기존의 업무 성격과 다를지라도 본연의 업무 범위에 포함시킬 수 있어야 한다. 이를 위해서는 공수를 산정하는 MH(Man-Hour) 시스템에 공개소프트웨어 활동을 입력할 수 있어야 한다. 한편, 조직 구성원을 총원하는 채용 체계에서도 변화가 있어야 한다. 즉, 지금까지는 학력, 근무 연수, 참여 과제, 인력 급수(초급, 중급, 고급, 특급) 등으로 인력을 평가하고 있었다면, 공개소프트웨어 인력을 채용할 경우에는 이러한 획일적인 평가 방식에서 벗어나 공개소프트웨어 관련 실적을 최우선적으로 인정하여야 한다. 공개소프트웨어 실적을 가늠하는 지표로는 커뮤니티 참여도, 프로젝트 수행 경력, 소스코드의 커밋 개수, 공개소프트웨어 생태계 내에서의 인지도 등이 대표적이다. 따라서 학교를 중퇴했거나 업무 경력이 없거나 인력 급수가 낮기 때문에 채용 대상에서 제외되거나 불이익을 당하는 상황이 발생하지 않도록 조직의 운영 시스템을 변경하여야 한다.

한편 조직 간의 협업의 문제가 발생하지 않도록 운영 방식도 변경되어야 한다. 공개소프트웨어의 특성 상 하나의 조직이 단독으로 업무를 처리할 수 없는 경우가 비일 비재하다. 비록 한 조직의 책임 하에 업무가 수행되더라도 여러 관련 조직이 협업을 해야 하는 상황이 발생할 수도 있다. 만일 이런 경우에 지원을 받아 성과를 내는 쪽만 유리하게 평가된다면 지원하면서 협업할 이유가 없어진다. 따라서 인력을 파견해 주거나 업무를 분담해서 수행하는 경우에 평가와 보상에서 피해가 없도록 전체적인 조직 운영 방식을 가져가야 한다.

(마) 조직 구성원

조직 구성원의 가치관, 자발적 동기, 성장가능성, 일에 대한 만족도, 성취감 등에 대한 고려가 필요하다. 조직은 사람들을 어떻게 채용하고 계발할 것인지, 조직의 각 부분별 구성원의 적절성은 어떠한지에 대해 검토하고 조직 설계에 반영해야 한다.

(바) 조직문화

일반적으로 조직문화는 무시되거나 소홀하게 취급되기 쉬운 부분이지만 공개소프트웨어 분야에서는 매우 중요한 핵심 요소이다. 만일 직급과 직무에 따라서 수직적인 명령 및 보고

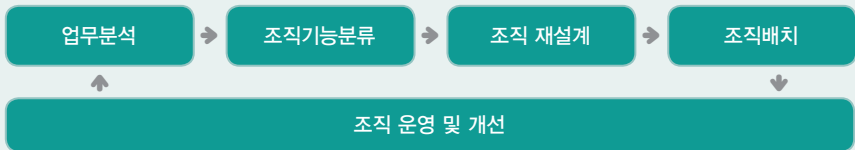
체계를 구성하고 있는 조직이라면 공개소프트웨어를 효과적으로 수용하기에 매우 불리하다. 이는 공개소프트웨어의 출생이 자유, 창의, 공유, 참여, 협업 등 수직적이기 보다는 수평적인 관계에 뿌리를 두고 있기 때문이다. 따라서 공개소프트웨어를 성공적으로 도입하고 적용하기 위해서는 개성을 인정하고 자유를 보장하는 조직문화로 바꾸어야 한다. 이는 파괴적인 창의력과 유연한 사고가 인정받도록 조직 풍토를 쇄신하는 변화관리를 의미한다. 이는 브레인스토밍 방식으로 토론하는 경우에 나이나 직급과는 무관하게 거침없이 의견을 교환하고, 어떠한 의견에 대해 비방이나 목살이 없는 분위기와 동일하다.

(사) 조직의 공유가치

공유가치는 조직 구성원이 어떤 결정을 해야 할 때 가장 먼저 의식하게 되는 중요한 가치를 의미한다. 모든 구성원이 일관성 있는 사고와 행동을 하도록 유도하기 위해서는 조직의 공유가치가 구성원에게 전파되어야 한다. 경영자는 비전 선포식 등을 통해 기업의 비전으로 공유하고, 조직의 구성원은 기업의 비전에 대하여 공유된 이해를 가져야 한다.

2) 공개소프트웨어 거버넌스 조직 설계 절차

공개소프트웨어 거버넌스를 위한 효율적인 조직을 설계하고 운영하기 위해서는 먼저 공개소프트웨어 사용유형에 따라서 각 기업에게 필요한 공개소프트웨어 거버넌스 업무 활동을 분석하고 각각의 활동에서 요구되는 하부기능에는 어떤 것들이 있는지 정의한 후 이런 활동을 수행하기 위하여 필요한 조직체계를 재설계하여 인력을 배치해야만 성공적인 공개소프트웨어 거버넌스 조직을 운영할 수 있다.



[그림 9] 공개소프트웨어 거버넌스 조직 운영 절차

(가) 업무 분석

현행 개발 및 정보서비스 조직 분석, 관리관행 분석을 통해 공개소프트웨어 거버넌스를 위한 조직의 성과 및 목표설정에 필요한 시사점을 도출하고, 이해관계자 식별 및 의사소통 촉진 방안에 대한 시사점을 도출한다.

조직에서 현재 활용되고 있는 공개소프트웨어 실태를 파악하기 위해서는 상용검증도구를

활용할 수 있으며 개발자 및 이해 관계자를 대상으로 한 Focus Area를 통한 인터뷰를 실시할 수 있다. 인터뷰는 공개소프트웨어 도입·활용의 수명주기에 포함되는 핵심인력을 포함해야 한다.

① 인터뷰를 통한 조직수준 진단을 위한 평가방법

사업현황과 공개소프트웨어관리를 위해 아래 8가지 주요영역에 대한 관리 실태를 파악하기 위해 질의 및 응답을 실시한다.

- [1] 서드파티와 공개소프트웨어 발견
- [2] 서드파티 공개소프트웨어 컴포넌트 검토와 선택
- [3] 공급망관리/공개소프트웨어 결합
- [4] 공개소프트웨어 코드관리
- [5] 공개소프트웨어 유지보수와 지원
- [6] 라이선스 준법성 프로그램
- [7] 공개소프트웨어 커뮤니티 상호작용
- [8] 의사결정자 검토

② 이상의 8가지 공개소프트웨어 주요 관리 영역별 조직수준에 대한 조사결과는 5단계의 공개소프트웨어 관리 성숙도 모델에서 제시하는 조직성숙도 단계에 적용하여 현재 조직수준을 파악하고 향후 개선을 위한 권장사항을 파악할 수 있다.

- [1] 위험노출단계(Exposed)
- [2] 측정단계(Measured)
- [3] 관리단계(Managing)
- [4] 참여단계(Participating)
- [5] 활성화 단계(Driving)

〈표 3〉 공개소프트웨어 관리 성숙도 모델

조직수준 생명주기	위험노출	측정단계	관리단계	참여단계	활성화 단계
검색(발견)	<ul style="list-style-type: none"> 공식적 가이드라인 혹은 프로세스가 없음 	<ul style="list-style-type: none"> 약간의 가이드라인 제공 	<ul style="list-style-type: none"> 수용 가능한 소스와 속성에 대한 명확한 정책이 있음 교육된 개발자 	<ul style="list-style-type: none"> 공개소프트웨어속성의 검증과 빠른 검색을 위한 도구가 있음 	<ul style="list-style-type: none"> 기업의 요구사항을 추진하기 위한 주요 커뮤니티참여
검토와 선택 및 공급망관리	<ul style="list-style-type: none"> 필요시 	<ul style="list-style-type: none"> 유입된 컴포넌트들은 확인되고 추적됨 	<ul style="list-style-type: none"> 명확한 정책과 프로세스 검토위원회에 의한 검토와 예외 처리 	<ul style="list-style-type: none"> 컴플라이언스를 위한 자동화된 프로세스 	<ul style="list-style-type: none"> 주요 커뮤니티와의 적극적 참여가 관심을 보이는 공개소프트웨어 공급자 관계 생성
코드관리	<ul style="list-style-type: none"> 공개소프트웨어가 독점 코드에 포함되어 관리됨 	<ul style="list-style-type: none"> 유입된 컴포넌트들은 확인되고 추적됨 	<ul style="list-style-type: none"> 내부정책기반의 개별 컴포넌트에 대한 담당과 책임을 수립 공개소프트웨어 저장소 추적 사용 	<ul style="list-style-type: none"> 자동화된 프로세스는 컴플라이언스 요구사항들과 사용, 속성, 소스를 추적 	<ul style="list-style-type: none"> 공개소프트웨어 저장소는 외부 릴리즈를 지원하는 것으로 확장됨
유지보수와 자원	<ul style="list-style-type: none"> 필요시 	<ul style="list-style-type: none"> 버그수정과 신규 출시에 대한 약간의 모니터링 	<ul style="list-style-type: none"> 내부정책기반의 개별 컴포넌트 담당자에 대한 책임 정의 강화된 자원 모델 	<ul style="list-style-type: none"> 자동화된 프로세스는 버전, 버그수정, 이슈를 추적 	<ul style="list-style-type: none"> 외부로 확장된 지원모델 외부지원을 처리하기 위해 확장된 자동화된 프로세스
준법 프로그램	<ul style="list-style-type: none"> 필요시 	<ul style="list-style-type: none"> 사용된 컴포넌트의 목록 관리 컴플라이언스 요구사항들은 수작업으로 처리 	<ul style="list-style-type: none"> 잠재위험을 방지하기 위한 검토와 코드 관리 자동화된 제품 출시 검사 문서화를 통한 컴플라이언스 프로세스 	<ul style="list-style-type: none"> 컴플라이언스 가능과 코드관리, 검토를 통합하는 자동화된 프로세스 관리와 고객을 위한 자동화된 보고서 	<ul style="list-style-type: none"> 기여 검토와 검사를 위한 정책과 자동화된 프로세스
커뮤니티 상호작용	<ul style="list-style-type: none"> 코드 다운로드 	<ul style="list-style-type: none"> 코드 다운로드 	<ul style="list-style-type: none"> 코드 다운로드 업데이트추적 사용 주체확인 없이 포럼에 참여 	<ul style="list-style-type: none"> 사용 주체특성에 부합된 포럼에 참여 버그수정 참여 	<ul style="list-style-type: none"> 신규 프로젝트/컴퍼넌트 참여 주요 커뮤니티 후원
의사결정자 감독	<ul style="list-style-type: none"> 없음 	<ul style="list-style-type: none"> 의사결정자는 사용 공개 소프트웨어 컴포넌트 목록을 받음 	<ul style="list-style-type: none"> 법무&관련부문이 검토위원회에 참여 	<ul style="list-style-type: none"> 커뮤니티 참여를 위한 정책 버그수정 참여를 위한 프로세스 	<ul style="list-style-type: none"> 컴포넌트 기여를 위한 정책과 프로세스 프로젝트 후원

대부분의 조직은 다음과 같이 공개소프트웨어 컴포넌트를 관리하는 세 가지 방법을 선택할 수 있다.

- ① **공개소프트웨어 사용을 방관하는 조직** : 공개소프트웨어 사용을 방관하면서 공개소프트웨어 관리를 위한 특정 계기가 되는 사건이나 사업 위기가 발생할 때까지 이슈를 무시한다. 이러한 조직은 사업투명도의 회손, 사업 리스크 증가에 대한 경영진의 책임, 잠재적 주주 및 고객 소송 등과 같이 관리되지 않은 공개소프트웨어로 인한 개발기간 연장, 지적재산의 회손 등 수 많은 부정적 결과를 초래할 수 있다.
- ② **공개소프트웨어 사용을 엄하게 금지하는 조직** : 공개소프트웨어 사용을 엄하게 금지하는 조직은 다음과 같은 이유로 권장되지 않는다.
 - 현재 환경에서 실행하기 어렵다
 - 외부적으로 개발된 컴포넌트들을 성공적으로 재사용하고 있는 조직과 비교하여 생산성과 개발속도가 떨어진다.
 - 개발자들에게 새로운 소프트웨어 개발을 위한 부족한 자원을 제공함으로써 동기부여를 할 수 없다.
- ③ **프로젝트 개발 생명주기의 중요한 포인트에 관리와 통제를 수반한 내부개발과 외부개발 컴포넌트 재사용을 권장하는 조직** : 이러한 조직은 새로운 개발 환경에 부합하여 권장되는 조직이며 다음과 같이 프로젝트 개발 생명주기의 포인트에 적절한 관리와 통제를 수반하여야 한다.
 - 컴포넌트가 처음으로 프로젝트에 첨부될 때
 - 내부 개발된 컴포넌트가 만들어지거나 수정 되었을 때
 - 모든 빌드 시에
 - 출시단계에서
 - 컴포넌트를 공개소프트웨어 커뮤니티에 공헌하는 것을 고려할 때 혹은 다른 서드파티에 소유권을 전환할 때
 - 소프트웨어 컴포넌트에 있어서 중요한 소유권 부분을 획득하기 전에

(나) 조직 기능 분류

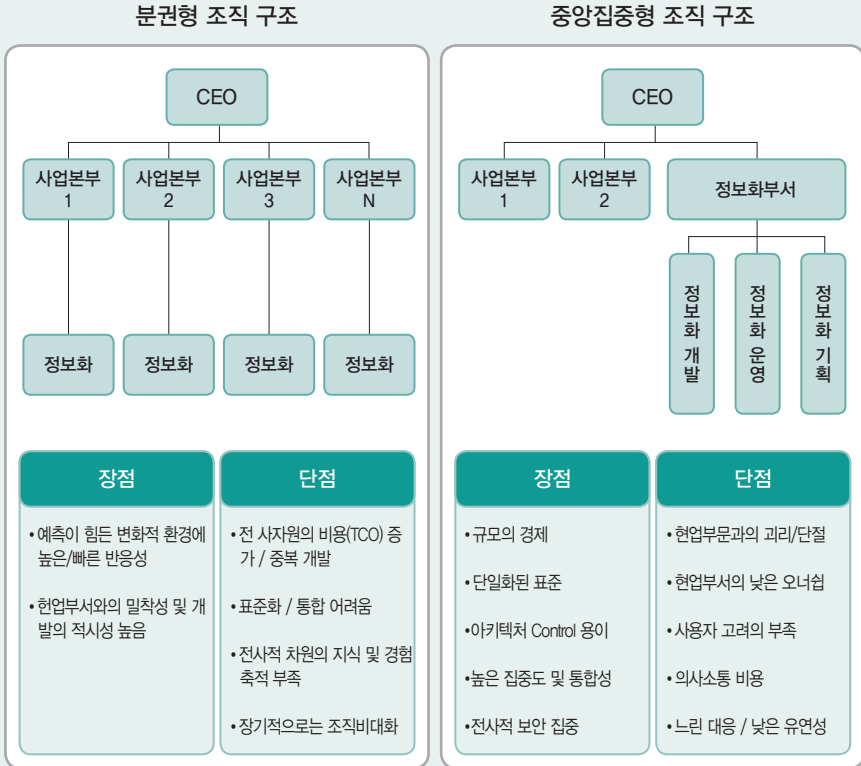
전략분석과 현행 업무조직 분석에서 도출한 시사점을 토대로 상위 조직 목표를 수립한다. 전략목표에 의해 수립된 비즈니스 모델을 수행하는 조직은 영업, 마케팅, 개발조직의 개선 사항과 현재 편성되어 있지 않는 커뮤니티 관리 조직 등의 개선을 목표로 조직 설계에 반영할 수 있다.



[그림 10] 조직 기능 분류

(다) 조직 재설계

일반적인 기업의 기본적인 조직구성은 영업, 마케팅, 기획관리, 소프트웨어 기술개발, 소프트웨어 기술개발 및 대외(공개소프트웨어 커뮤니티, 고객) 기술협력 조직, 정보서비스 조직, 정보인프라 구축/운영조직, 경영지원 조직 그리고 외주 운영 조직으로 구성되는데 이런 9개 기본적인 조직구성에 대한 조직의 목표정의와 변화관리를 포함해야 한다. 이 단계에서 기업의 상황과 특성에 맞는 거버넌스 조직 구조가 도출된다.

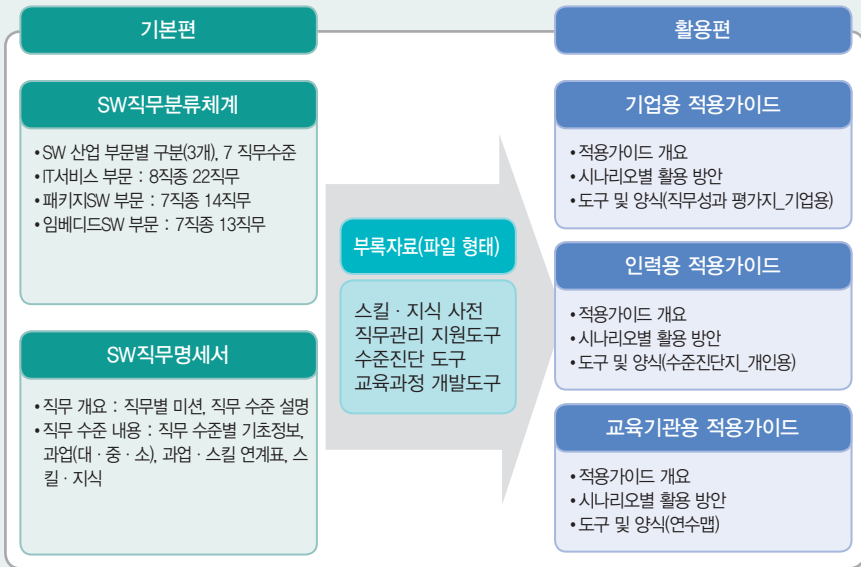


[그림 11] 조직구조의 예

(라) 조직 배치

단위 조직별 요구되는 업무에 대한 적정 인원을 산정하고, 적절한 인원을 그 수준에 맞는 지위에 배치하려면 활동분석을 기초로 하여 각각의 직무를 기술하고 직무명세서를 작성하여야 한다. 직무명세서는 주어진 지위에 적합한 사람에게 요구되는 교육적 배경, 경험, 기술, 적성 그리고 기타 중요한 자격요건을 규정하고 개인과 지위를 연결시켜서 작업상의 기초를 제공하며 조직구조에 성공적으로 구성원을 배치할 수 있도록 도와준다.

정보통신산업진흥원의 소프트웨어공학센터(<http://www.sw-eng.kr>)에서는 소프트웨어 직무수행능력표준에 대한 가이드와 컨설팅을 제공하고 있다.



[그림 12] 소프트웨어 직무수행능력 표준 기본 구조

〈표 3〉 직무명세서 예시

수행 업무 세부 내역			
대분류	중분류	레벨	세부 수행내역
SW개발	개발환경 구축	Level 2	1. 설정된 개발환경 명세서 따라 스스로 업무 수행용 PC환경을 설정한다.
SW개발	SW 이해 및 적용	Level 1	1. 대상 플랫폼의 HW 특성에 기인하는 프로그래밍의 제약사항을 이해하고 프로그래밍을 수행한다. 2. 대상 OS의 동작 특성을 숙지하고 오동작, 시스템에러에 대한 기본지식을 갖추고 프로그래밍을 수행한다. 3. 대상 응용영역의 업무 특성을 이해하고 구현/적용을 수행한다. 4. HW, 유틸리티, 개발도구 등 SW개발과정과 실행환경에 대해 이해하고 구현 및 적용을 수행한다. 5. 데이터베이스SW의 필요 기능과 용도에 대해 이해하고 데이터베이스SW의 개발·적용을 수행한다. 6. SQL 등 데이터베이스 관리·조작에 필요한 언어를 활용한다. 7. 네트워크상의 컴퓨터간 상호작용에 대해 이해하고 프로그래밍을 수행한다. 8. 상호작용에 참여하는 장비들의 플랫폼 특성을 이해하고 프로그래밍을 수행한다.

			9. 오프라인 상태 및 온라인 상황에서의 SW 보안 기능에 대해 이해하고 보안SW를 구현한다.
SW개발	프로그래밍	Level 2	1. 프로그래밍 언어를 사용하여 단위 모듈의 프로그램 구현을 수행한다.
SW개발	디버깅	Level 2	1. 디버깅도구를 활용하여 작성한 프로그램의 디버깅을 수행한다.
SW 테스트 계획 수립	SW테스트 실행계획 수립	Level 2	1. 테스트에 필요한 SW 및 HW의 사양 및 수량을 파악한다.
SW 테스트 계획 수립	SW테스트 합격기준 정의	Level 2	1. 선임자의 지도하에 테스트 항목과 특성 별 척도 정의 및 각 척도에 대한 합격기준치 정의 등 테스트 합격에 대한 기준문서를 작성한다.
SW 테스트 실행	설계 요소별 테스트	Level 2	1. 테스트 시나리오 및 테스트 데이터를 준비하는 등 테스트 환경을 구축하고, 테스트 시나리오를 기반으로 각 설계요소별 단위 테스트를 실행한다. 2. 코딩이나 로직 결함 시 오류 수정 및 보완, 재테스트를 실시하여 기대되는 결과를 도출한다.
SW 테스트 실행	부하 및 장애 테스트	Level 2	1. 명확한 테스트 목표와 지표를 설정하고, 운영 시와 비슷한 시스템 환경을 구성한다. 2. 부하를 일으킬 수 있는 테스트 케이스(Through put 성능 테스트, 응답시간 성능 테스트, 한계 테스트, 내구 테스트)를 준비한다.
기술 지원 및 보급	버전 관리	Level 2	1. 버그리포트 검토 및 프로그램 수정에 의해 보완된 프로그램을 리패키지(Repackage)한다.

출처 : 소프트웨어직무수행능력표준 적용가이드¹³⁾

(마) 조직 운영 및 개선

성공적인 공개소프트웨어 거버넌스를 위한 조직 운영을 위해서는 중장기적인 조직 변화를 위한 환류체계 수립(Cycling)이 필요하다. 이를 위하여 조직 운영 실태에 대한 지속적인 모니터링을 통해 조직 구조 설계 및 운영방안 수립, 직무 구성 및 업무분장, 정보화 교육 강화, IT 관리의 효율성 제고 방안 수립, 업무 프로세스 정립 방안 수립 등의 개선 과제들을 도출하고 개선을 위한 지속적 활동을 수행해야 한다. 그 내용을 정리하면 다음과 같다.

13) 출처 : http://www.software.kr/mbs/swkr/jsp/board/view.jsp?page=1&boardId=143&boardSeq=1254707&mcCategoryId=&id=swkr_040100000000

- 조직 기능 및 역할 재정립
- 필요 신규 직무 및 인력 확보 방안 도출
- 인적자원 관리 체계 정립

3) 공개소프트웨어 거버넌스 조직 모델

본 가이드에서는 일반적인 기업의 조직구성으로 영업, 마케팅, 기획관리, 소프트웨어 기술 개발, 대외(공개소프트웨어 커뮤니티, 고객) 기술협력 조직, 정보서비스 조직, 정보인프라 구축 및 운영조직, 경영지원 조직 그리고 외주 운영 조직으로 9개 영역을 기준으로 공개소프트웨어 거버넌스를 위한 조직의 전략사업단위(Strategic Business Unit)의 구성방안을 다음과 같이 제시하고자 한다. 제시하는 전략사업단위 별 구성방안을 토대로 기업의 전략과 특성에 따라 공개소프트웨어 거버넌스 조직을 구성하여 적용할 수 있다.

〈표 4〉 공개소프트웨어 거버넌스 조직모델의 예

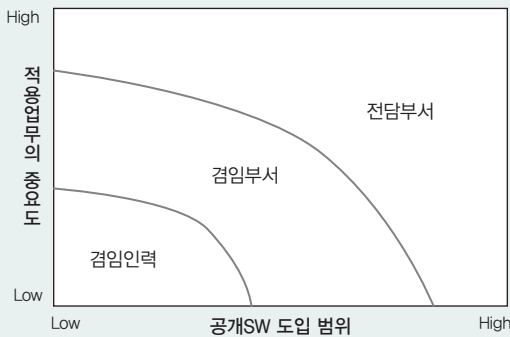
조직	개요 및 업무 내역
공개소프트웨어 검토위원회(Open Source Review Board, OSRB)	<ul style="list-style-type: none"> • 공개소프트웨어 정책 수행을 위한 핵심 기구 • OSRB는 공개소프트웨어사용에 대한 최후의 의사결정권한을 가지며 조직내 공개소프트웨어 사용에 대한 전문가 센터 • OSRB는 조직 부서 및 구성원을 대표할 수 있는 대표자로 구성(법무, 개발, 운영, 마케팅, 재무 등)
법무(Legal)	<ul style="list-style-type: none"> • 라이선스와 계약 준수를 포함하여 법적 요구사항들을 준수해야 하는 조직의 제품과 솔루션에 대한 궁극적인 책임을 가지고 있음
개발 (Engineering/ Development)	<ul style="list-style-type: none"> • 개발부서는 실제로 공개소프트웨어를 사용하고 공개소프트웨어와 결합된 제품 및 솔루션을 창출하는 조직 • 어떤 공개소프트웨어 컴포넌트들이 엔지니어링 프로세스에 포함되어 사용되어져야 하는 지에 대한 의사결정을 하는 그룹으로서 OSRB에서 매우 중요한 역할을 수행
운영(Operations)	<ul style="list-style-type: none"> • 조직의 개발시스템 환경을 담당 • 이 그룹에 있어서 승인된 소프트웨어 스택의 구성은 매우 중요 • 개발 공개소프트웨어 스택 및 컴포넌트가 내부적으로 사용된다면, 해당 소프트웨어 컴포넌트들이 승인되었는지 혹은 적절한 버전인지를 확인 • 개발 공개소프트웨어 스택 및 컴포넌트가 외부 조직으로 서비스를 제공하는 데 사용된다면, 모든 소프트웨어 컴포넌트들이 라이선스 요구사항들을 준수하고 추적되고 있는지를 확인하는 것이 매우 중요
마케팅(Marketing)	<ul style="list-style-type: none"> • 공개소프트웨어 컴포넌트들과 결합된 제품 및 솔루션을 위한 사업계획 뿐 아니라 해당 공개소프트웨어 라이선스 의무사항을 준수하기 위한 모든 필요 속성들을 확인 • 마케팅 대표의 OSRB 참여는 공개소프트웨어 의사결정에 대한 사업영향에 대한 평가에 도움을 주고, 개발 그룹에서 생각하지 못하는 모든 요구사항들을 준수할 수 있게 함

재무/감사 (Finance and/or Compliance)	<ul style="list-style-type: none"> 경우에 따라 몇몇 조직들은 소프트웨어 스택 혹은 데이터 추적 준수사항들을 증명해야 되는 어떠한 규제 조건들을 따라야 할 경우가 있음 재무/감사 대표의 OSRB참여는 조직의 특정 상태에 따라 준법성이 적절히 준수되고 있음을 증명해야 하는 책임이 있음
--------------------------------------	---

4) 공개소프트웨어 거버넌스 조직 운영

(가) 비즈니스 전략에 따른 조직의 차별화

공개소프트웨어를 사용하는 조직이 명확한 역할 수행과 공개소프트웨어 자원의 효율적 활용을 위한 체계를 구축하기 위해서는 공개소프트웨어 거버넌스를 위한 기업의 주요기능들에 대한 전담 조직을 구성하는 것이 가장 좋다. 하지만 모든 기업이 전담조직을 보유할 필요는 없으며 공개소프트웨어 사업 전략에 따라 조직 설계를 다음과 같은 유형으로 차별화할 수 있다.



[그림 13] 공개소프트웨어 거버넌스 조직 구성 방안

- 전담부서 : 기관 및 기업 내 공개소프트웨어가 광범위하게 도입되어 투자관리, 성과측정, 라이선스 관리, 공개소프트웨어 기반 개발 등의 전사적 차원의 모니터링 및 통제
- 겸임부서 : 현재 운영/유지하고 있는 정보 시스템 관련 기획, 개발, 운영(지원) 부서에 공개소프트웨어 관련 책임과 역할의 할당
- 겸임인력 : 공개소프트웨어 운영/유지를 위한 최소한의 인력으로 커뮤니티 활동 및 전문 기술지원 업체와의 의사소통 및 관리

(4) 요구분석

요구분석은 공개소프트웨어 사용자 및 개발자의 고민과 요구를 기반으로 기능, 성능, 연동, 호환, 보안 등의 관점에서 필요한 사항을 분류하고 해석하는 활동이다. 공개소프트웨어의 요

구분석이라고 해서 일반적인 절차를 많이 벗어나지는 않는다. 즉 공개소프트웨어와 상용소프트웨어의 요구분석 활동은 매우 유사하다. 그러나 공개소프트웨어의 요구 분석을 진행하기 이전에 요구사항을 충족시키는 방법이 서로 다르다는 점을 충분히 인식하고 있어야 한다.

상용소프트웨어에서는 요구 사항을 수용할지 배제할지 여부를 자체적인 판단으로 결정한다. 공개소프트웨어의 경우에는 요구 사항이 커뮤니티에 의해서 채택되지 않으면 의미가 없다. 만일 채택되지 않는 요구 사항을 반드시 반영하려면 커뮤니티와 다른 버전을 새로 만들게 되고 이를 위한 개발과 유지 보수를 감수하여야 한다. 공개소프트웨어 커뮤니티의 원래 취지가 자체적인 개발을 피하고 협력해서 그 결과를 공유하자는 의도이므로 자체적으로 요구 사항을 공개소프트웨어에 반영하는 것은 매우 바람직하지 않다. 따라서 공개소프트웨어의 요구사항을 분석하기 이전에 반드시 커뮤니티에 의해서 채택되도록 하는 전략을 가지고 요구 사항 수집을 진행하여야 한다. 따라서 커뮤니티를 움직일 수 있는 역량이 없는 경우에는 사용자의 필요와 고충 사항이 있더라도 커뮤니티 버전에 맞추어 불편을 감수하는 자세를 취할 수밖에 없다. 그러나 공개소프트웨어가 주는 혜택이 이러한 불편에 비해서 월등히 크기 때문에 공개소프트웨어를 사용하고 개선에 참여하는 것이다.

요구사항을 정의하기 위해서는 관련 이해관계자(최종사용자, 운영 담당자, 기타 의사결정 관련 담당자 등) 목록을 참조하여 인터뷰(또는 설문) 대상자를 선별하고, 인터뷰를 효율적으로 진행하기 위해 사전에 주요한 의사결정이 필요하거나 확인해야 할 사항을 중심으로 인터뷰 질의서를 작성한다. 인터뷰는 현재 시스템을 사용하고 있거나 혹은 향후 시스템을 적용할 업무 범주에 포함되는 현업 및 IT담당자에 대한 인터뷰를 수행하여야 한다. 이를 위해 사업 범위 및 방향성 수립 단계에서 파악된 이해관계자 목록을 참조하고, 업무 및 기술 현황 분석단계에서 파악된 이해관계자를 반영하여 인터뷰 계획을 수립한다.

사전에 정의된 인터뷰 담당자들을 대상으로 향후 구축할 시스템의 장비구성, 기능, 성능, 인터페이스, 데이터, 테스트, 보안, 품질, 제약사항, 프로젝트 관리, 프로젝트 지원 요구사항 등의 요구사항을 도출한다. 인터뷰 수행 시 이해관계자별로 질문 항목을 차별화하고, 그에 맞는 요구사항을 도출해내도록 한다. 예를 들어 최종사용자의 경우 기능, 사용자 인터페이스 및 사용성과 같은 품질 요구사항 위주로 질의하며, 시스템 운영자일 경우 시스템 장비구성, 성능, 보안, 테스트, 시스템 인터페이스와 같은 요구사항이나 향후 운영 및 기술지원 같은 프로젝트 지원 요구사항을 중심으로 인터뷰를 하도록 하여 요구사항이 구체화될 수 있게 한다.

요구분석 활동은 공개 소프트웨어의 도입을 계획하기 전에 조직의 요구 분석을 위한 다음과 같은 항목에 대하여 사전 조사가 필요하다.

- 사용자 및 인적 요소
- 시스템의 기능 및 특성

- 제약사항 등

1) 사용자 및 인적 요소

누가 어떠한 시스템을 어떻게 사용할 것인가 하는 사항은 전환 이전에 중요하게 생각 되어야 할 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?

2) 시스템의 기능 및 특성

시스템의 수행업무 및 특성에 대한 고려가 필요 하다.

- 시스템의 수행업무 파악
- 시스템의 특성 파악

이 과정에서 조사되어야 할 사항의 예를 들면 다음과 같다

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

3) 제약 사항

운영 시스템의 특성상 공개소프트 도입에 제약이 있을 수 있으므로 사전 검토, 제약 사항은 별도 정리해 두어야 한다.

요구분석의 과정을 통하여 조직 내 공개소프트웨어 활용을 위한 필수요구사항과 제약사항을 정의하게 되며 이를 기반으로 조사를 수행한다.

구 분		내 용
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	공개소프트웨어 기반에서 불편사항은 없는가? 전환할 때 주의해야 할 사항은 없는가?	

(5) 조사

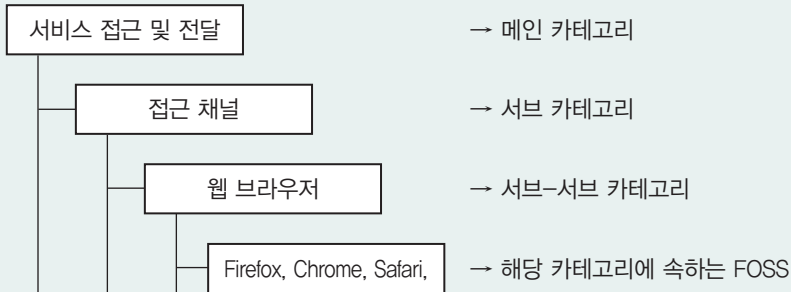
조사는 공개소프트웨어의 속성과 커뮤니티 정보를 취합하는 활동이다. 이 활동을 통해서 반드시 취합해야 하는 속성으로는 기능, 용도, 라이선스, 프로그래밍 언어, 적용 분야, 연동 및 연계 소프트웨어, 유스케이스 등이 있으며 공개소프트웨어 자체의 특성에 대한 정보를 제공한다.

커뮤니티에 관련된 정보로는 최초 등록일, 소스코드의 증가 속도, 참여 기업, 다운로드 횟수, 레퍼런스 개수, 핵심 개발자 및 커미터 등이 있다. 한편 공개소프트웨어는 상용소프트웨어와는 달리 소프트웨어 자체에 대한 조사뿐만 아니라 해당 공개소프트웨어를 개발하는 커뮤니티와 이 커뮤니티에 참여하거나 커뮤니티를 지원하는 단체에 대한 조사도 필요하다. 상용 제품의 경우에는 판매 회사가 고객 유치와 영업을 위해서 마케팅을 통해 적극 홍보활동을 진행하지만 공개소프트웨어의 경우에는 영업이나 마케팅 인력이 별도로 배정되어 있지 않고 개발자가 중심이 되어 소프트웨어 자체의 기능과 품질에 대한 활동만 집중적으로 진행되기 때문이다. 공개소프트웨어에는 보증 제도가 없다는 점을 고려해 볼 때 일차적으로 공개소프트웨어를 제작하는 커뮤니티의 발전성과 영속성을 중요 시 해야 한다. 그리고 이러한 판단에 신뢰성을 배가시키기 위해서는 공개소프트웨어의 설계와 개발에 기여하는 개인이나 단체 또는 재정적으로 후원하고 있는 기관에 대한 조사는 반드시 필요하다. 한편 공개소프트웨어를 외부의 기술지원 업체를 통해서 사용하고자 하는 경우에는 해당 업체에 대한 조사도 불가피하다. 그러므로 공개소프트웨어의 조사 활동에 대상이 되는 객체를 구분하자면 공개소프트웨어 자체, 공개소프트웨어 커뮤니티, 공개소프트웨어 커뮤니티 참여/지원/후원 기관, 기술지원 서비스 업체 등으로 나누어 볼 수 있다.

사용 주체가 공개소프트웨어를 가장 잘 사용하기 위해서는 적용할 수 있는 공개소프트웨어에는 어떤 종류가 있는지 먼저 조사하여야 한다. 이는 공개소프트웨어의 개수가 수십만개 이상이 되기 때문에 수적으로도 너무나 많고 그 종류도 다양하기 때문이다. 그러므로 철저한 조사가 수행되지 않으면 몰라서 못 쓰거나 잘 못 사용하는 사태가 발생하기도 한다. 특히 새롭게 탄생한 공개소프트웨어를 발견하는 것도 중요하며, 짧은 기간 동안에 급격히 성장하는 공개소프트웨어를 유심히 관찰하여 유용성 여부를 조기에 판별할 필요도 있다.

일반적으로 새롭게 등장한 공개소프트웨어는 초기부터 적극적으로 커뮤니티에 참여하여 리더십을 장악할 수 있어서 중요하고, 급성장 공개소프트웨어는 미래 사업 발굴 및 전략 수립에 차별적인 가치를 제공한다. 그러나 어떠한 공개소프트웨어도 커뮤니티 개설 이전부터 본 거버넌스 프레임워크를 사용하는 사용주체에게 먼저 신고하거나 변동 사항에 대한 지속적인 업데이트를 제공하지 않는다. 더구나 이미 알고 있는 커뮤니티라고 할지라도 최신에 갱신된 버전이나 정보에 대해서 먼저 알려 주지 않기 때문에 면밀하게 조사하지 않으면 긴요하게 사용할 수 있는 기회를 놓치기 십상이다.

이러한 조사 활동을 체계적으로 수행하기 위해서는 분류 체계가 필요하다. 상용 제품도 마찬가지이지만 이러한 분류 체계에는 표준이 있을 수 없기 때문에 어느 정도 권위가 있는 기관에서 출간한 공신력 있는 분류체계를 참조할 필요가 있다. 특히, 공개소프트웨어 자체적인 특성과 공개소프트웨어 간의 차이점만을 분류한다면 신규 도입뿐만 아니라 대체 및 전환 시 정확한 공개소프트웨어의 선택이 어렵기 때문에 동일한 분류 항목에 해당하는 유사한 상용 제품도 같이 분류해야 한다.



[그림 14] 표준 공개소프트웨어 분류체계의 예시

[그림 14]는 공개소프트웨어를 분류하는 체계를 예시하고 있다 자체적인 기술 역량을 보유하지 않으면 공개소프트웨어를 자체적으로 사용할 수 없는 것이 당연하기 때문에 이 분류에는 공개소프트웨어에 관련된 전문 기술력을 보유하고 있는 기업이나 연구소가 해당된다. 특히 기업 입장에서 조명하여 볼 때 이러한 조사 활동의 최대 효용성은 독점 솔루션의 개발과 신규 사업의 적용에 있다. 따라서 조사 자체만을 목표로 삼아서는 안 되며 사업전략으로 연결시켜야 한다. 이러한 목적으로 조사 활동을 더욱 세분화시키면 4S(Seeking, Sensing, Seeding, Sourcing) 활동으로 나눌 수 있다.

〈표 6〉 조사 활동의 단계와 내용

단 계	활동 내용
Seeking(탐색)	초기에 급성장하는 공개소프트웨어를 관찰하고 미래 사업에 핵심적인 기술 후보를 발견한다.
Sensing(감지)	기술 흐름과 고객 니즈에 따른 기술 적정성과 시장성을 파악한다.
Seeding(육성)	사업 가능성이 있는 기술을 비즈니스 모델과 연결시키고 육성시킨다.
Sourcing(확보)	필요한 인력을 확보하고, 커뮤니티를 통해 기술을 확보한다.

따라서 기업 내에서 수행되는 4S 활동은 기술 기반으로 기업 솔루션을 만들어 내는데 목표를 두어야 하며, 신규 기술이 사업을 만들 수 있도록 또한 시장의 수요를 충족하는 기술을 만들어 내도록 수행되어야 한다. 일반적으로 기술과 시장이 분리되어서는 어떤 사업도 성공할 수 없기 때문에 이 둘은 항상 진행되어야 하는 것이 당연하다. 그러나 중요한 문제는 아무도 사업이 될 만한 기술을 미리 알려 주지 않는 데 있다. 다행히도 공개소프트웨어 커뮤니티에서 장래성 있는 미래 기술을 찾을 수 있으며 이러한 이유로 많은 글로벌 선진 기업들이 공개소프트웨어 커뮤니티를 주의 깊게 관찰하고 있다¹⁴⁾. 이러한 관점에서 기술 전문성이 필요한 기업은 목표 시장과 사업 방향에 부합하는 신규 공개소프트웨어를 찾아내야 하며 미래 기술 확보와 연결시켜야 한다. 이를 위해서 기여자 수, 소스코드의 라인 수, 다운로드 수, 방문자 수 등 객관적인 평가 지표를 사용하여 최근 급격하게 성장하고 있는 공개소프트웨어 커뮤니티를 찾고 그 핵심 기술을 분석하여야 한다. 또한 기술의 적정성과 시장성을 검증하여 미래 비즈니스 모델과도 연결시켜야 한다.

그리고 이렇게 정의되고 검증된 미래 핵심 목표 기술을 내재화시켜야 하며 미래의 유망기술이 될 만한 공개소프트웨어 커뮤니티에 매우 초기 단계부터 진입하여 주도적으로 리드하는 것이 중요하다. 즉, 개방과 공유 그리고 협업이라는 독특한 커뮤니티의 문화로 인하여 기술 난이도가 낮은 시작 단계부터 적극적으로 참여해야만 핵심 모듈에서도 배타적인 우월성을 확보해 진입 장벽을 높일 수가 있기 때문이다. 결국 공개소프트웨어 기술의 진입 장벽은 사람이기 때문에 핵심 개발자를 보유하는 순간 공개소프트웨어 커뮤니티를 소유할 수 있게 된다. 지금까지 설명된 4S의 각 단계별 활동 내용을 시각적으로 요약하면 다음 그림과 같다.

〈표 7〉 4S의 단계적 활동 내용

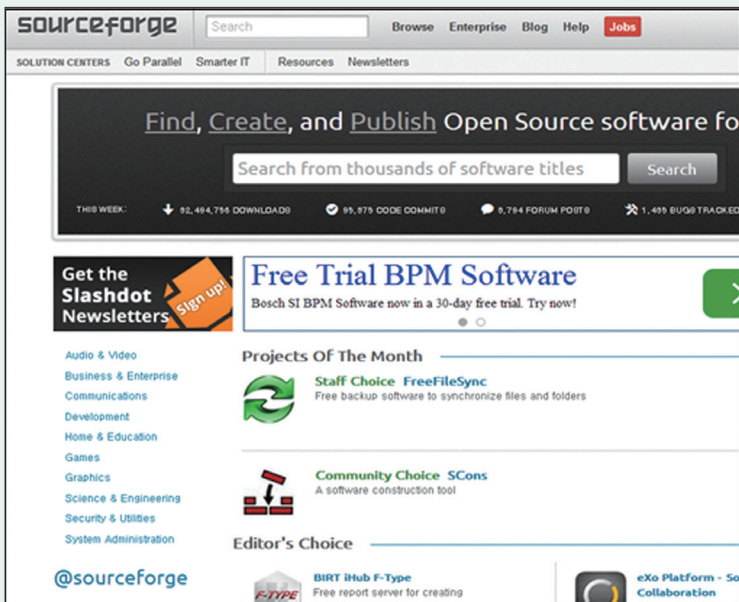
1. 탐색(Seeking)	2. 감지(Sensing)	3. 육성(Seeding)	4. 확보(Sourcing)
<ul style="list-style-type: none"> • 인터넷의 공개 사이트를 통하여 새로운 공개소프트웨어를 찾음 • 커뮤니티 규모, 발전 속도, 활용 사례 등 평가 지표에 관련된 정보를 수집함 	<ul style="list-style-type: none"> • 세계적인 트렌드를 인식 • 고객과 시장을 분석 • 테스트를 통한 특성 분석 • 활용성이 뛰어난 후 보군 선정 및 추천 	<ul style="list-style-type: none"> • 감지된 공개소프트웨어 기술과 목표 시장을 매칭 • 핵심 기술을 정의 • 요구사항을 구체화 • 목표 사업 모델을 설계 	<ul style="list-style-type: none"> • 인적 자원 획득 및 팀구성 • 프로토타입 설계 • 공개소프트웨어 커뮤니티 참여 • 내부 공개소프트웨어 커뮤니티 육성

14) 공개소프트웨어 프로젝트는 생성과 소멸이 빈번하기 때문에 정확한 개수는 알 수 없으나 Openhub.net, Sourceforge.net, Github.com 등 공개소프트웨어 네트워크를 사용하여 조사할 수 있다

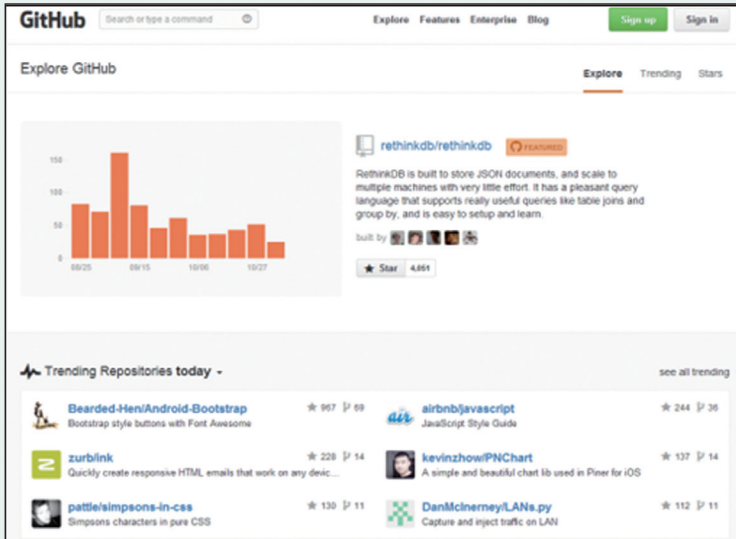
이처럼 이전 단계에서는 도출된 요구 사항을 토대로 공개소프트웨어의 프로그램 자체에 대한 정보뿐만 아니라 공개소프트웨어에 관련된 기술 문서와 웹 사이트 등을 조사하고 최신의 시장 변화와 발전 방향을 감지할 수 있는 정보도 확보하여야 한다.

이제 요구분석에서 도출된 명세서를 토대로 요구하는 기능에 부합되는 공개소프트웨어를 어떻게 검색할 수 있는지 실례를 들어 설명하고자 한다. 전 세계 모든 개발자들이 자유롭게 참여하는 공개소프트웨어는 영업이나 마케팅 활동이 미약한 관계로 통합된 검색 수단이 제공되지 않는다. 그러므로 원하는 소프트웨어를 찾기 위하여 직접 조사하거나 미리 선정한 우수 공개소프트웨어 목록을 참고하여야 한다.

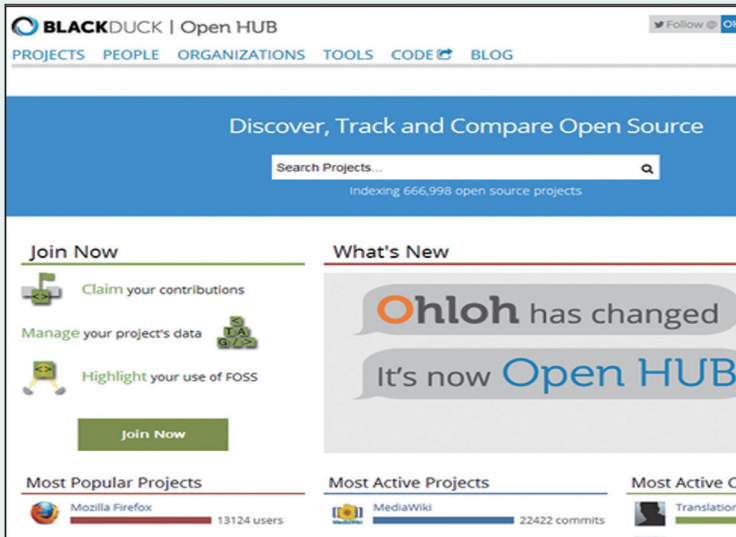
직접 조사하는 경우 원하는 공개소프트웨어를 검색하기 위하여 SourceForge.net, GitHub.com, Openhub.net 등의 서비스를 사용할 수 있다.



[그림 15] SourceForge (<http://sourceforge.net>)



[그림 16] GitHub (<https://github.com>)



[그림 17] OpenHUB (<https://www.openhub.net>)

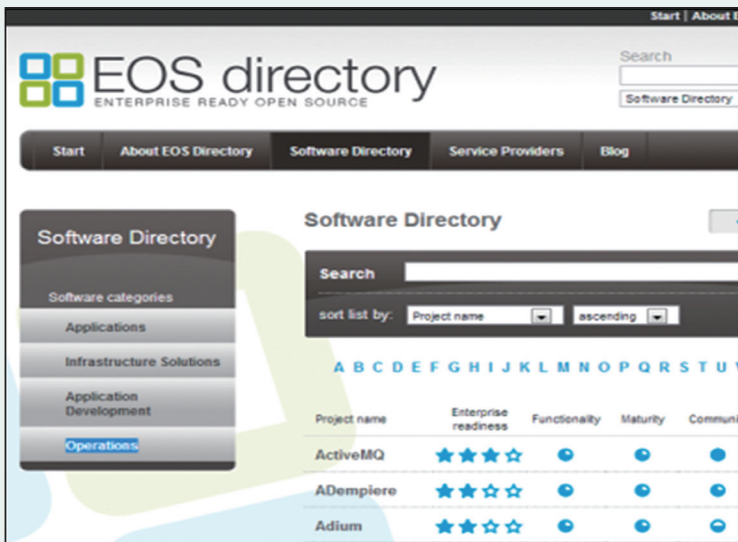
각 공개소프트웨어 네트워크에 등록된 프로젝트의 현황은 다음과 같다.

〈표 8〉 공개소프트웨어 프로젝트 현황 (2014.12)

Site	URL	Projects	Contributors
SourceForge	http://sourceforge.net	324,000	3,400,000
GitHub	https://github.com	16,600,000	7,400,000
OpenHUB	https://www.openhub.net	666,998	3,627,589

원하는 요구사항에 적합한지 검토하기 위하여 이처럼 많은 숫자의 공개소프트웨어를 조사하는 것은 많은 노력을 필요로 하게 되는데, 이런 경우 미리 신뢰성 및 성숙도에 대해서 검증한 다양한 자료를 참고할 수도 있다.

해외의 경우 EOD directory(<http://eos.osbf.eu/software-directory/>), OLEX(<http://olex.openlogic.com/library/certified>) 등의 사이트에서는 다른 기업에서 사용하는 공개소프트웨어들의 선호도를 확인할 수 있기 때문에 직접 검색하는 것보다는 쉽게 신뢰할 수 있는 공개소프트웨어를 선정할 수 있다.



[그림 18] Enterprise Open Source Directory

국내의 경우 공개소프트웨어 역량프라자(<http://www.oss.kr>)에서 제공하는 공개소프트웨어 목록(http://www.oss.kr/oss_repository12)과 공개소프트웨어 테스트 결과자료(http://www.oss.kr/oss_repository6)를 참고하거나 유망 공개소프트웨어 프로젝트 목록(http://www.oss.kr/oss_repository12/103649)을 참고할 수 있다. 유망 공개소프트웨어 프로젝트 목록은 공개소프트웨어 프로젝트의 커밋 수, 기여자 수, 최근 3년 평균 커밋 성장률, 커뮤니티 성장성, Github fork, Github favorite, 주관 및 후원 조직, 커뮤니티 활동 정도 등을 고려하여 선정한 목록으로 기업의 비즈니스에서 미래 사업 발굴 및 전략 수립에 차별적인 가치를 제공할 수 있다.

순번	분류	솔루션명	라이선스	기술지원	홈페이지	제품개요
1	BPM	OpenProj	CPAL	community	http://sourceforge.net/projects/openproj/	프로젝트 관리 소프트웨어
2	BPM	ProcessMaker	AGPL v3	prof/community	http://www.processmaker.com	비즈니스 프로세스 소프트웨어
3	BPM	ProjectLibre	CPAL	community	http://www.projectlibre.org	Java 기반의 BPM 솔루션
4	BPM	uEngine BPM	LGPL	prof/community	http://www.uengine.org	비즈니스프로세스관리(BPM) 솔루션
5	CMS	Drupal	GPL v2	community	http://drupal.org	다양한 기능의 웹사이트 구축 솔루션
6	CMS	Geeklog	GPL v2	community	http://www.geeklog.net	웹 콘텐츠 관리 CMS 솔루션
7	CMS	GNU Board4	GPL	prof/community	http://www.sir.co.kr	PHP와 MySQL을 사용하는 웹사이트 구축 솔루션
8	CMS	Joomla	GPL	community	http://joomla.org	PHP와 MySQL을 사용하는 웹사이트 구축 솔루션
9	CMS	KimsQ Rb	LGPL	prof/community	http://www.kimsq.com	블로그/커뮤니티/기업 웹사이트를 만들 수 있으며 모듈시스템을 통해서 소핑몰 그룹웨어
10	CMS	MediaWiki	GPL	community	http://mediawiki.org	위키백과(Wikipedia)에서 사용하고 있는 웹 기반 위키 저작 솔루션
11	CMS	OpenCMS	코어:LPGL v2.1 일부모듈:GPL v2.1	community	http://www.opencms.org	자바기반의 CMS
12	CMS	PHP-Nuke	GPL	community	http://www.phpnuke.org/	PHP와 MySQL 기반 CMS 솔루션

[그림 19] 공개소프트웨어 목록

공개소프트웨어 역량프라자 에서는 공개소프트웨어 뿐만 아니라 공개소프트웨어 제공 벤더 및 기술지원 기업에 대한 목록(http://www.oss.kr/index.php?mid=oss_techsupport) 도 제공하고 있기 때문에 이를 참고할 수도 있다.

○ 공개SW 유지관리 서비스 개요

- “공개SW 유지관리 서비스”는 공개SW를 최적의 상태로 활용·유지하기 위한 기술지원 서비스를 의미합니다.
- 공개SW 도입 이후 운영에 필요한 기술지원 서비스를 제공하는 것으로 제품지원(기능향상, 제품수정 및 업데이트), 유지관리(장애지원, 예방지원 등), 컨설팅 등의 서비스가 포함되어 있습니다.
- 계약기간과 유지관리 서비스 레벨(Service Level)의 조합에 의한 정액제 방식으로 계약을 체결하며, 공개SW 유지관리 서비스는 계약 기간 내에서 도입된 공개SW가 폐기 또는 변경되기 전까지 제공(계약기간과 시스템 수명주기의 불일치에 의한 기간 및 대가의 조정은 별도 약정) 합니다.

○ 공개SW 유지관리 서비스 가이드라인 소개

- 공개SW 유지관리 서비스 가이드라인은 공개SW의 특성을 고려하여 공개SW 유지관리 서비스의 범위 및 대가 산정 방식을 명확히 하고 유지관리 서비스 구매 시 필요한 서비스의 항목, 내용 등을 표준화함으로써 발주자의 예산편성 및 집행 용이성을 도모하고 공개SW의 안정적인 서비스 제공을 촉진하는데 그 목적을 둔 가이드 문서입니다.
- 최초 2008년 1월에 “공개SW 유지보수 서비스 가이드라인”이 제정되어 2012년 6월 13일 “지식경제부 예규 제 41호, 공개SW 유지관리 서비스 가이드라인”으로 개정되었습니다. **[지식경제부 예규 제 41호, 공개SW 유지관리 서비스 가이드라인 상세보기]**

○ 공개SW 제품 및 서비스 기업 리스트

- 아래 공개SW 제품의 등록 기준은 OSI(Open Source Initiative)가 정한 OSD(Open Source Definition) 조건을 충족해야 하며, 새로운 솔루션의 등록 신청은 솔루션을 직접 개발한 개발자 및 개발자, 커뮤니티의 신청을 통해 등록을 진행합니다.

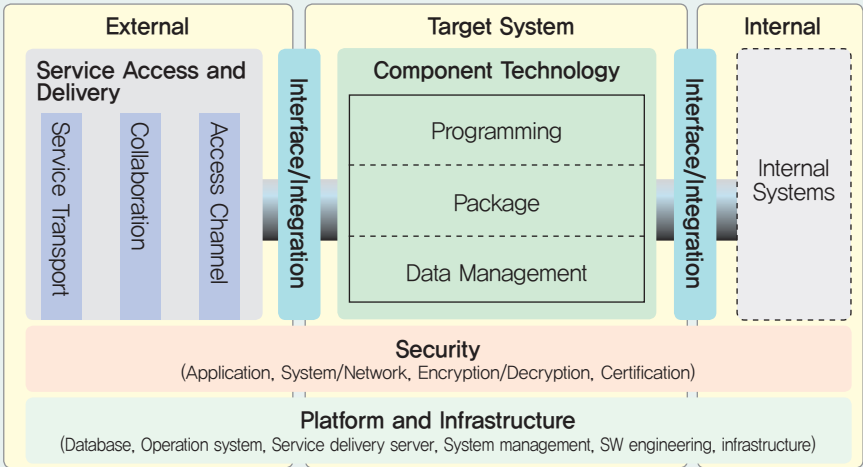
[OSI 기준확인] * OSI(Open Source Initiative) : 1998년 오픈 소스 소프트웨어 사용을 장려하기 위하여 만들어진 단체.

[그림 20] 공개소프트웨어 기술지원기업 안내

(6) 분석

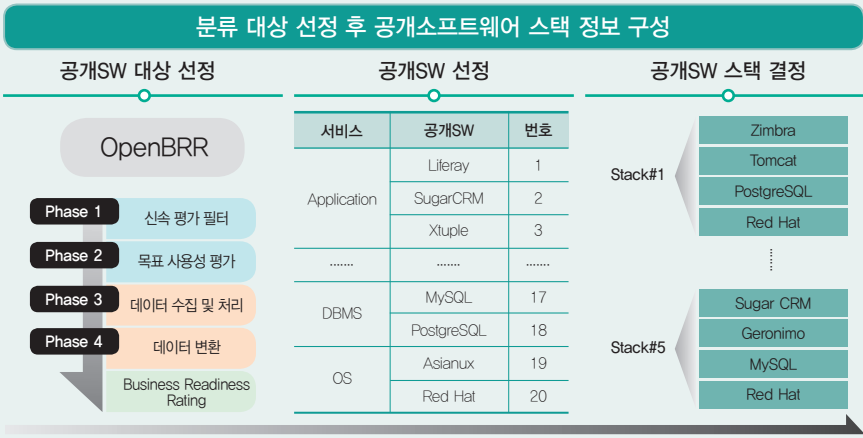
분석은 공개소프트웨어의 성능도와 적용성을 평가하기 위해서 필요한 속성과 척도를 정의하고 평가 항목을 정량화시켜 기능성, 이식성, 신뢰성, 사용성, 유지 보수성, 커뮤니티 영속성 등 공개 소프트웨어의 수준과 가치를 알아 낼 수 있는 데이터를 취합하고 정리하는 활동이다. 대부분의 공개소프트웨어는 비영리의 개발 커뮤니티의 특성을 가지고 있기 때문에 상용 소프트웨어와는 달리 마케팅 활동이나 영업망 또는 고객 인벤토리가 없는 상태이다. 이에 따라 공개소프트웨어의 품질 수준을 가늠할 수 있는 정보가 빈약할 수밖에 없다. 결국 품질 수준을 측정할 수 있는 평가 지표의 부재로 인하여 막상 공개 소프트웨어의 도입이 검토될 때 관계자의 막연한 불안감이나 두려움이 적극적인 공개소프트웨어의 활용에 지장을 초래하고 있다. 따라서 공개소프트웨어가 사용되는 영역 별로 성능, 품질, 지원, 인프라 등 다양한 각도에서 상용 제품 대비 어느 정도의 수준인지를 분석할 필요가 있다.

이러한 문제를 해결하기 위해서 가장 먼저 수행하여야 하는 활동은 공개소프트웨어를 분류하는 작업이다. [그림 21]은 대표적인 공개소프트웨어의 분류 체계를 보여 주고 있다. 새로운 공개소프트웨어가 생기게 되면 기존의 분류 체계에서는 찾을 수 없는 영역이 생길 수 있으므로 이러한 분류체계는 끊임없이 갱신될 수밖에 없으며 사용자에게는 항상 최신의 버전이 제공되어야 한다.



[그림 21] 공개소프트웨어 분류체계

공개소프트웨어 자체에 대한 분석도 중요하지만 공개소프트웨어는 특이한 생태계에서 독특한 라이프 사이클을 가지고 있으므로 공개소프트웨어 커뮤니티에 대한 분석도 병행하여야 한다. 상용 제품은 사용 주체가 품질을 보증하지만, 공개소프트웨어의 경우에는 전적으로 커뮤니티에 의해서 개발되고 배포되므로 커뮤니티 자체의 활동성, 로드맵, 견실성, 영속성 등이 분석되어야 한다. 예를 들어 아래 그림과 같이 OpenBRR(Open Source Business Rating)은 이러한 분류 및 분석 활동의 대표적인 예로 적합하다.



[그림 22] 공개소프트웨어 평가 모델의 예(OpenBRR)

그러나 일반적으로 공개소프트웨어의 유용성을 정확하게 판단할 수 있도록 일반적인 품질 특성에 더하여, 유사한 분류에 속한 상용 제품과 비교할 수 있는 척도는 제시되지 않고 있다. 이러한 척도를 제공하기 위해서 데이터베이스 관리시스템 분야에 있어서는 TPS(Transaction Per Second), 웹 서비스 분야에서는 웹페이지 응답속도, 고가용성, 클러스터링에 대해서는 노드 구성 방식 등과 같은 항목들이 특화된 제품 특성을 비교하는 지표로 사용되어야 한다. 따라서 공개소프트웨어의 성숙도와 적용성을 평가하는데 필요한 속성을 정의하고 이 속성을 최대한 정량화시켜야 한다. 정성적인 방법으로 평가와 비교가 불가능하지는 않지만 비교 속성이 정량화되지 못하면 최종적인 점수와 순위를 얻어 낼 수 없기 때문에 거버넌스 차원에서의 전략적인 판단이 불가능하다. 그러므로 조사 단계에서 수집한 기초 데이터를 근거로 분석 단계에서 정의할 수 있는 속성 항목과 분류를 아래 표와 같이 정리하였다.

〈표 9〉 공개소프트웨어 평가지표 기본속성 예시

속성군	기본 속성	설 명
기능성	기능 적합성	분류 체계의 해당 카테고리에서 마땅히 제공해야 하는 목표 기능을 충실히 수행하는 수준
	지원성	설치 툴, 패치, 관리, 모니터링 등 목표 기능을 최상의 조건으로 수행하는데 필요한 보조 기능을 다양하게 제공하는 수준
	상호운용성	다양한 운영체제(Linux, Unix, Windows)에서 설치 및 작동이 가능한 수준
이식성	대체성	동일한 기능의 다른 공개소프트웨어 제품에서 전환 및 대체(migration)를 용이하게 수행할 수 있는 수준(표준 수용성)
	대체후기능성	유사 공개소프트웨어 제품으로 대체한 이후에도 동일한 기능을 수행할 수 있는 수준
	설치성	다양한 플랫폼에 이식될 수 있도록 구성 파라미터(configuration parameter)의 조작이 용이하고 설치가 간단하고 편리한 수준
신뢰성	가용성	에러, 버그, 정지, 종료 등 비정상적인 동작이 없이 정상적으로 운영되는 정도
	회복성	문제 및 장애 발생 시 복구 및 대응이 잘 되는 정도
	최신성	최근 일정 기간 동안 신속하게 발전하는 정도
	성숙성	커뮤니티의 인력 구성, 역할 분배, 운영 및 관리 체제가 얼마나 안정적이고 체계적인지 나타내는 수준
사용성	이해성	매뉴얼, 가이드, 튜토리얼 등 제품 사용 및 이용에 필요한 문서 및 자료의 제공 수준

속성군	기본 속성	설 명
	학습성	제품 구성, 설치, 운영에 필요한 자문, 컨설팅, 교육, 인증(자격증) 등에 관련된 서비스를 제공하는 수준
	운용성	사용, 운영, 관리에 편리한 기능 수준 (예 GUI 환경)
유지 보수성	분석성	에러 또는 문제를 해결하는데 도움이 되도록 원인과 상태를 상세히 분석할 수 있는 메일링, 버그 리포팅, 이슈 트래킹 등 소통 수준
	전문기술	해당 공개소프트웨어에 대해서 전문 업체 또는 커뮤니티의 기술 지원 서비스가 가능한 수준
	시험성	패치 또는 업데이트 버전에 대한 품질 측정 수준
커뮤니티	나이 및 규모	오랫동안 활동이 왕성하게 지속되고 최근에도 활동이 활발하여 발전하고 있는 수준
	주체	커뮤니티가 쇠약하지 않고 발전할 수 있도록 기업 및 단체로부터 지속적인 경제적, 인력적, 사업적 지원이 있는 정도
	접근성	상위의 인터넷 검색이 가능하고, 커뮤니티 참여와 지적 자산의 공유에 편리한 인터페이스를 제공하고 있는 수준 (이메일, 게시판, 페이스북)
	관리체계	커뮤니티 내에서 프로그램 개발, 소스코드 기여, 수용 여부 심사, 품질 테스트, 로드 맵 수립 등 개발과 품질에 관련된 활동이 체계적으로 진행되고 있는 수준
라이선스	소스코드공개 범위	해당 공개소프트웨어를 활용하여 배포할 경우 발생하는 소스코드공개 범위 확인
	저작권	해당 공개소프트웨어를 활용하여 배포할 경우 저작권 문제 발생 시 법적 위험으로부터 자사 혹은 고객을 보호할 수 있는 정도
	특허	해당 공개소프트웨어를 활용하여 배포할 경우 특허위반여부 및 특허권리 확보에 대한 제약조건
내·외부 고객 요구사항	수준 정의	내부 혹은 외부 고객의 서비스가 수준별로 정의되어 있고 지원되는 정도
	기술지원	문제가 발생했을 시 신속한 기술지원을 제공할 수 있는 수준
	품질	기술지원을 통해 제품의 품질이 보증되는 수준
	선호도	내·외부 고객이 선호하는 정도

분석은 일반적으로 평가를 전제로 하기 때문에 객관적이고 공정한 지표가 필요하다. 또한 지표들은 측정과 비교가 가능하도록 정량적으로 산출되어야 한다. 그렇지 않으면 평가 단계에서 비교하고자 하는 공개소프트웨어 간의 순위 또는 수준을 알 수 없기 때문이다. 따라서 분석 단계에서 공개소프트웨어의 속성을 정량적으로 표현할 수 있는 지표를 도출하는 작업이 매우 중요하다.

그럼 어떻게 이 지표를 만들고 정량적으로 측정할 수 있는지 공개소프트웨어의 속성 하나를 선정하여 실질적으로 설명해보도록 하겠다. 다만 이해력을 높이기 위해서 아래와 같이 예를 들어서 단계적으로 평가 모델을 설명하도록 한다.

- 1) 공개소프트웨어를 평가하기 위해서 “지원성”을 평가 속성으로 선택한다.¹⁵⁾ 지원성의 정량적 지표로 “지원 가능한 서비스의 종류(개수)”를 선택한다. 이 지표는 공개소프트웨어를 사용하는 주체에게 얼마나 많은 기술 지원 서비스를 제공할 수 있는지 측정한다.¹⁶⁾
- 2) 최대로 지원 가능한 서비스의 종류가 몇 개나 되는지 정한다. 이 최대 개수에 비해 현재 얼마나 많은 서비스가 제공되는지에 따라서 평가를 받게 된다. 최대 서비스 개수를 분모에 두고 현재 지원되는 서비스 개수를 분자로 두면 지표를 백분율로 표현할 수 있다.
- 3) 여기서는 현실적인 상황과 계산의 편리성을 위해서 최대 지원 가능한 서비스의 종류를 5개로 한정한다. 이 서비스에는 툴, 패치 및 가이드, 모니터링, 알람, 콘솔 등이 선정된다. 이들의 지원 서비스의 내용을 살펴보면 다음과 같다.
 - 별도의 설치 툴 제공
 - 패치 및 가이드의 다운로드 제
 - 제품 모니터링 툴 제공
 - 장애 알람 제공 (SMS, Push e-Mail)
 - 웹 콘솔 제공
- 4) 지원성의 백분율을 등급 점수로 변경한다. 현재는 최대 지원 서비스가 5개로 지정되어 있지만 그 개수를 얼마든지 변경할 수가 있다. 그러면 백분율 기반으로 계산하게 되어 복잡성이 높아지므로 아래와 같이 범위를 기반으로 지원성을 5개 등급으로 정량화시켰다.¹⁷⁾

15) 지원성은 기본 속성 중 기능성에 속함

16) 지원 가능한 서비스의 종류(수) 이외에도 정량화시킬 수 있는 다른 지표를 선택할 수 있다. 예를 들어 지원 인력의 수, 지원 인력의 수준, 인력 별 지원 경력 등 다양한 지표를 생각할 수 있다.

17) 등급의 개수가 많으면 많을수록 정밀한 측정치를 반영할 수 있지만 여기서는 5개의 등급으로도 측정에 문제가 없다고 가정함

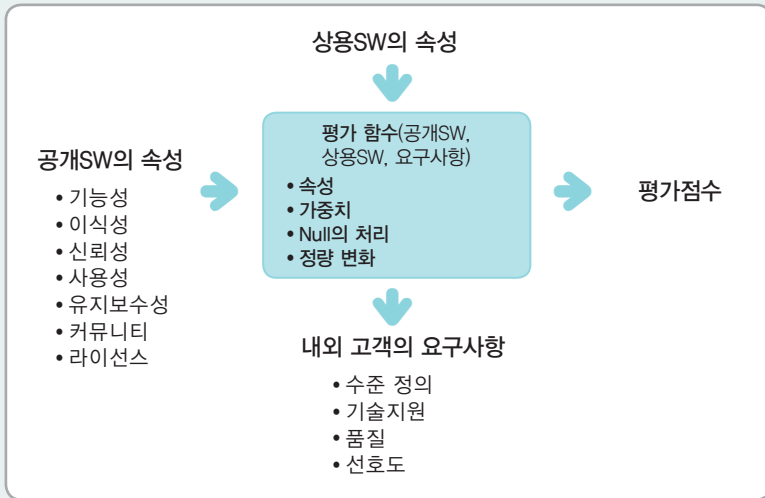
- 1 점 : 0.0 <= 지표 < 0.2
- 2 점 : 2.0 <= 지표 < 0.4
- 3 점 : 4.0 <= 지표 < 0.6
- 4 점 : 6.0 <= 지표 < 0.8
- 5 점 : 0.8 <= 지표 <= 1.0

5) 만일 MariaDB를 평가해야 하는 공개소프트웨어로 선정하였고 지원 가능한 서비스가 설치 툴을 제공하고 패치 업데이트가 가능하다면 5개 중 2개가 가능하므로 위의 등급에 따라서 지표는 40%가 되고 배점은 3점을 받게 된다.

위에서 예를 든 바와 같이 다른 속성에 대해서도 동일한 방법으로 지표를 선정하고 정량화시킬 수가 있다. 다만 설명의 편리성에 따라 지원성 평가 지표를 (제공하는 지원 서비스의 수 / 최대 지원 가능한 수)의 비율로 정의해 놓고 지표가 놓이는 값의 범위에 따라 등급을 정함으로써 정량적인 측정이 가능하게 되었다. 여기서는 지표의 범위가 0부터 1까지로 보고 0.2 범위로 구분하여 5등급으로 나누었지만 다양한 방법으로 점수를 계산할 수 있다.

(7) 평가

평가는 공개소프트웨어의 도입, 개발, 대체, 전환, 기술지원, 유지 보수, 컨설팅, 교육 등을 목적으로 성숙도와 적용성을 수치로 산출하는 활동이다. 이처럼 다양한 목적으로 공개소프트웨어를 적용하기 위해서는 도입, 대체, 전환의 문제가 없는지 사전에 평가하여야 한다. 이러한 평가의 결과는 숫자로 계산될 수 있어야 하며 이를 위해서 정량화가 가능한 지표를 정의하여야 한다. 이 정량화 작업은 정성적인 특성을 숫자로 변환시켜야하기 때문에 객관적으로 타당한 기준을 제시하기가 거의 불가능하다. 그래서 지금까지 발표된 수많은 공개소프트웨어 성숙도 평가 모델들은 정량화 방법을 구체적으로 공개하지 않고 주관적인 경험에 의존하고 있는 것이 사실이다. 비록 정량화 방법이 제시되었더라도 옳고 그른지를 검증하기란 결코 쉽지가 않다. 이처럼 평가 모델의 객관성과 신뢰성에 문제가 있는 것을 사실이지만 그렇다고 공개소프트웨어의 수준을 측정할 방법을 전혀 제시하지 못한다면 공개소프트웨어의 실용적 가치를 알릴 방법이 전혀 없으며 적용도 불투명하게 된다. 따라서 어느 정도의 오차를 감수하더라도 각 속성 별로 평가에 영향을 미치는 지표를 정량화시키고 채점할 수 있어야 한다. 이러한 원칙을 반영하여 평가 모델을 설계하면 다음 그림과 같이 표현할 수 있다.



[그림 23] 공개소프트웨어의 성숙도 모델의 설계

본 가이드에서는 공개소프트웨어 도입과 적용을 고려하고 있는 사용 주체에게 사전에 유용한 정보를 제공할 수 있도록 활용 가능한 평가 방법과 지표를 아래 표에서 보는 바와 같이 매우 구체적으로 제공하고자 한다. 그리고 각 속성에 대한 분석 방법은 이전 장에서 지원성을 가지고 설명한 예와 동일하다. 즉, 정량적 수치를 최하위 1점에서 최상위 5점까지 5개 단계로 구분하였다.

〈표 10〉 속성 별 정량화 공식과 적용 방법

속성군	속성	채점 방법	분석 방법
커뮤니티	나이 및 규모	변수 = {버전 번호, 연령} 지표 = 최종 버전 번호 x 나이 1 점: 0 <= 지표 < 12 2 점: 12 <= 지표 < 24 3 점: 24 <= 지표 < 72 4 점: 72 <= 지표 < 180 5 점: 180 <= 지표	지표는 최종 버전 번호와 월 단위의 커뮤니티 나이를 곱해서 산출함 버전 번호가 1.0 이상이고 커뮤니티 나이도 12개월 이상이 되어야 자생력이 있는 커뮤니티로 인정함 버전이 3.0 이상이고 연수가 5이상이면 최상위 수준으로 인정함
	주체	변수 = {후원 단체 유무} 1 점: 지원 없음 2 점: 하나의 중소기업 지원 3 점: 복수의 중소기업 지원 4 점: 하나의 대기업의 지원 5 점: 복수의 대기업의 지원	인력 및 자금에 대한 후원 단체의 유무로 측정함

속성군	속성	채점 방법	분석 방법
커뮤니티	접근성	변수 = {계시판, 포럼, 위키, 검색성, 인터넷} 지표 = 제공하는 접근 방법의 종류 / 전체 접근 방법의 종류 개수 1 점: 0.0 (≦ 지표 < 0.2) 2 점: 2.0 (≦ 지표 < 0.4) 3 점: 4.0 (≦ 지표 < 0.6) 4 점: 6.0 (≦ 지표 < 0.8) 5 점: 0.8 (≦ 지표 ≦ 1.0)	전체 접근 방법의 종류 개수 = 5 1. 게시판 운영 2. 포럼 운영 3. 위키 운영 4. 인터넷 검색 시 첫 페이지 출력 5. 인터넷 사이트에서 정보 제공 외부에서 커뮤니티로 연락하거나 관련 정보를 얻을 수 있는 용이성 OSS 커뮤니티에 대해 전문 정보를 제공하는 인터넷 사이트로는 openhub.net, wikipedia.org 등이 있음
	성숙성	변수 = {기간, 버전 출시, 관리 체제, 평가 방법, 위원회 운영} 지표 = 충족하는 성숙지표의 종류 / 전체 성숙 지표의 종류 개수 1 점: 0.0 (≦ 지표 < 0.2) 2 점: 2.0 (≦ 지표 < 0.4) 3 점: 4.0 (≦ 지표 < 0.6) 4 점: 6.0 (≦ 지표 < 0.8) 5 점: 0.8 (≦ 지표 ≦ 1.0)	전체 성숙 지표의 종류 개수 = 5 1. 최초 버전 출시 이후 3년 이상 지속적으로 신규 버전 출시 2. 최근 배포한 안정된 버전의 넘버가 1.0 이상 3. 관리 운영자(maintenance operator), 커미터(심의회), 개발자 등의 운영 체제 확립 4. 기여도 및 참여도에 따른 개발자의 등급 체제 확립 5. 이사회 운영 - 개인의 독단적 판단이 아닌 위원회에 의한 의사 결정 방식
신뢰성	기능 적합성	변수 = {표준기능, 기능 별 가중치} 기능 적합성 = SUM(표준 기능 × 표준 기능 가중치)	표준 기능은 OSS가 속하는 소분류 카테고리 안에서 표준적으로 제공하는 기능임 가중치는 평가자에 의해 결정됨
	가용성	변수 = {연간 가동률} 지표 = 연간 실행 시간 / 1년 1점: server - 99% 이하 application - 90% 이하 2점: server - 99.9% 이하 application - 94% 이하 3점: server - 99.99% 이하 application - 96% 이하 4점: server - 99.999% 이하 application - 98% 이하 5점: server - 100% application - 100%	연간 가동률은 제품의 추천 사양 환경에서 운영이 중단되지 않고 실행되는 수준임 일반적으로 1년 동안 초단위의 가용한 시간 비율이 사용됨
	회복성	변수 = {회복기간} 1점: 1일 < 회복기간 2점: 1시간 < 회복기간 (≦ 1일) 3점: 10분 < 회복기간 (≦ 1시간) 4점: 1분 < 회복기간 (≦ 10분) 5점: 0초 < 회복기간 (≦ 1분)	1일: 참을 수 없음 1시간: 참을 수 있으나 불만이 큼 10분: 불만 없이 참을 수 있음 1분: 지연을 느끼지 못함

속성군	속성	채점 방법	분석 방법
타제품 이식성	상용 대체 용이성	변수 = {대체기간} 1점: 12주 <= 대체기간 2점: 8주 <= 대체기간 < 12주 3점: 4주 <= 대체기간 < 8 주 4점: 2주 <= 대체기간 < 4 주 5점: 0주 <= 대체기간 < 2 주	CSS를 대체하는데 걸리는 시간 = 데이터 이전을 포함하여 테스트까지 완 료하는데 소요되는 주단위 기간임
	대체 후 가능성	변수 = {확장기능, 기능 별 가중치} SUM(확장 기능 X 확장 기능 가중치)	확장기능 = OSS가 속하는 소분류 카테고리 안에서 표준 이외에 제공하는 기능 가중치는 평가자에 의해 결정됨
	대체 방법론	변수 = {대체 방법론} 1점: 대체 방법론 없음 3점: 개념적 방법론만 있음 5점: 실용적 방법론도 있음	개념적 방법론: 유즈 케이스, 다이어그램, 흐름도, 문장 서술, 활동 단계(phase) 수준의 전환 방법론 실용적 방법론: Data type, parameter, procedure, API, 활동 단위(activity) 수준의 전환 방법론
라이선스 이슈	라이선스 충돌	변수 = {라이선스 충돌 개수} 1점: 충돌 존재 5점: 충돌 부재	충돌: 배포하는 제품과 그 내부에 사용 된 OSS 간의 라이선스 의무사항이 불 일치함
	면책 서비스	변수 = {면책 서비스 조건} 1점: 면책 서비스 불가 5점: 면책 서비스 제공	면책 서비스: 사용자가 법적인 문제에 연루되지 않도록 공급자가 모든 책임을 지는 서비스
	특허 침해성	변수 = {특허 침해 개수} 1점: 특허 침해 있음 5점: 특허 침해 없음	특허 소송 보복 조항이 있는 라이선스라 할지라도 제3자의 특허에 대해서는 침 해가 발생한 경우 보호받을 수 없음
고객 충실도	SLA 충실도	변수 = {기술 서비스 정책 충실도} 지표 = 제공하는 기술서비스의 종류 / 전체 기술 서비스의 종류 개수 1 점: 0.0 <= 지표 < 0.2 2 점: 2.0 <= 지표 < 0.4 3 점: 4.0 <= 지표 < 0.6 4 점: 6.0 <= 지표 < 0.8 5 점: 0.8 <= 지표 <= 1.0	전체 기술 서비스의 종류 개수 = 5 1. 기술지원 서비스의 가격 정책이 수립 되어 있음 2. 최초 지원 요청 후 응답 시간이 지정 되어 있음 3. 상시 패치 및 업데이트 서비스가 지 정되어 있음 4. 고객 창구가 개설되어 있음 5. 교육 또는 컨설팅 서비스가 제공되고 있음
	문제 해결 능력 만족도	변수 = {해결기간} 1 점: 2주 <= 해결기간 2 점: 1주 <= 해결기간 < 2 주 3 점: 2일 <= 해결기간 < 1 주 4 점: 1일 <= 해결기간 < 2 일 5 점: 해결기간 < 1 일	2주 이상은 참을 수 없는 기간으로 설정함 당일 접수된 문제를 당일 해결할 수 있 으면 최고 수준으로 설정함

속성군	속성	채점 방법	분석 방법
고객 충실도	기술보증	변수 = {보증 조건, 보상 조건} 1점: 제품 보증 및 보상 없음 3점: 제품 보증 제공 및 보상 없음 5점: 제품 보증 제공 및 보상 제시	OSS 자체는 보증(warranty)가 없으나 사업적인 목적으로 문제 발생 시 즉각적인 해결 서비스를 제공함 제품 자체의 하자로 인하여 사용자의 손해가 발생할 경우 보상을 약속할 수 있음
	선호도	변수 = {점유율, 레퍼런스 개수} 1 점: 점유율 = 0%, 레퍼런스 0개 2 점: 점유율 < 2%, 레퍼런스 < 2개 3 점: 점유율 < 5%, 레퍼런스 < 5개 4 점: 점유율 < 8%, 레퍼런스 < 8개 5 점: 점유율 > 10%, 레퍼런스 > 10개	점유율은 상용을 포함한 시장에서 copy 수 기준으로 산정함 레퍼런스는 상용 서비스를 제공하는 시스템으로 사용되는 사이트 개수임

다음으로는 실제 공개소프트웨어를 가지고 평가 방법을 설명하도록 한다. 이를 위해 웹 애플리케이션 서버라는 분류를 지정하고 이 분류에 속하는 공개소프트웨어의 성숙도와 적용성을 평가하는 방법을 설명하도록 한다.

1) 지정된 데이터베이스 분류에 속하는 상용소프트웨어와 공개소프트웨어를 선택한다.

〈표 11〉 분류 예시

분류	상용소프트웨어	공개소프트웨어
Web Application Server	JEUS WebLogic, Websphere	JBoss
		Tomcat
		Glassfish
		JOnAS
		Resin

- 위 테이블에서 열거한 속성에 대해서 정량화 지표에 따라 평가 점수를 계산한다.
- 속성의 비중에 따라서 평가 점수에 가중치를 부여한다.
- 기본 속성과 확장 속성에 대해서 가중치를 부여하면 다음과 같이 속성군 별로 평가 점수가 계산된다.

- 기본 속성군 점수 = Sum(기본 속성 점수 × 기본 속성 가중치)
- 확장 속성군 점수 = Sum(확장 속성 점수 × 확장 속성 가중치)

- 최종 평가 점수는 기본 속성과 확장 속성에 대해서 가중치를 반영하고 합산한 점수가 된다.
- 평가점수 = (기본 속성군 점수 × 기본 속성군 가중치)
+ (확장 속성군 점수 × 확장 속성군 가중치)

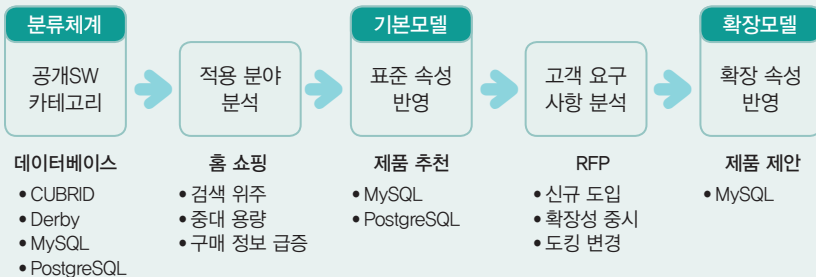
5) 동일한 분류에 속하는 공개소프트웨어 각각에 대해서 합산 점수를 계산하고 공개소프트웨어 간의 순위를 정한다.

〈표 12〉 평가 결과

분류	상용소프트웨어	공개소프트웨어	평가 점수	순위
Web Application Server	JEUS WebLogic, Websphere	JBoss	4.95	1
		Tomcat	4.32	2
		Glassfish	4.09	3
		JOnAS	2.99	4
		Resin	2.97	5

이 단계에서 계산한 최종 평가 점수는 동일한 분류에 대해서만 한정된 것이며 다른 분류에 속한 상용소프트웨어나 공개소프트웨어와는 무관하다. 즉, JBoss를 다른 분류에 속하는 OpenStack과 비교할 수 없는 것과 마찬가지이다. 이렇게 산출한 평가 결과는 다음과 같은 방법으로 활용할 수가 있다.

- 사용자(내부 사용자, 고객)의 공개소프트웨어 카테고리(사용 영역) 정의
- 사용 영역과 포함되는 카테고리의 공개소프트웨어를 표준 모델로 평가함
- 평가 결과에 따라 공개소프트웨어 후보군 선정
- 후보 공개소프트웨어와 사용자 선호 공개소프트웨어를 대상으로 확장 모델로 평가함
- 확장 모델의 결과에 따라 최종적으로 추천할 공개소프트웨어 선정



[그림 24] 공개소프트웨어 평가 결과의 활용

(8) 계약

계약은 공개소프트웨어의 사용 조건과 의무사항 이행을 약속하는 활동이다. 공개 소프트웨어는 상용 제품과 다르게 라이선스에 대한 비용 청구 조건이 없으며, 배포 받고 사용하는 순간부터 해당 라이선스의 의무사항 준수에 동의한 것으로 간주된다.

공개소프트웨어의 계약은 저작권자와 사용자 간에 체결되는 사용조건과 제약사항이므로 크게 권리와 의무사항으로 나누어진다. 공개소프트웨어의 권리로는 사용, 수정, 배포의 자유가 대표적이다. 또한 공개소프트웨어의 의무사항으로는 저작권 선언문구의 보존, 라이선스 충돌 제거, 고지의 의무, 소스코드 공개의 의무 등이 여기에 속한다. 상용 제품을 공급하는 회사와의 계약 사항과는 달리 공개소프트웨어는 가격을 지불할 필요가 없지만 보증 부인(No Warranty) 조건으로 사용해야 하는 특징이 있다. 이는 무료이기 때문에 제품의 품질에 대해서는 책임을 지지 않겠다는 의도가 아니라 무료로 공개소프트웨어를 배포하는 저작권자를 보호하고 공유와 활성화에 이바지 하려는 의도로 해석하여야 한다. 즉, 이타적인 취지에서 값으로 환산할 수 없는 소스코드를 무료로 배포하는 저작권자에게 그 공개소프트웨어로 인해서 발생하는 모든 문제에 대해서 책임을 묻는다면 더 이상 공유, 참여, 협업, 개방의 공개소프트웨어 정신이 실현될 수 없기 때문이다. 다만 품질, 기능, 성능, 보안 등 어떠한 부분에서든 문제가 발생하면 누구에게 책임을 묻기 보다는 서로 협동해서 문제를 해결할 수 있는 방안을 마련하기 위한 계약 조건의 하나로 이해할 수 있는 것이다. 공개소프트웨어 사용에 대한 전반적이고 구체적인 계약 조건은 각각의 라이선스에 상세히 기술되어 있다.

근본적으로 공개소프트웨어에는 보증(Warranty)이라는 계약 조건이 없기 때문에 공개소프트웨어를 적극 도입하거나 안심하고 사용하지 못하는 고충이 있다. 그러므로 이러한 고민을 해결해 주는 기술 지원 서비스를 제공하는 업체가 많이 생겨나게 되었다. 일반적으로 공개소프트웨어 서비스를 제공하는 업체는 SLA(Service Level Agreement)에 따라서 사용자와 계약 조건을 설정하게 된다. 이 계약은 서비스의 수준에 대한 약정으로써 하자 보수와 품질 보증을 중요시 한다. 즉, 사용자의 입장에서는 어떠한 공개소프트웨어를 적용하든 관여하지 않지만 사용자가 원하는 사양과 수준으로 운영되어야 한다는 조건이다. 이를 준수하기 위해서는 운영상 발생할 수 있는 버그, 에러, 장애 등에 대한 문제 해결과 고객 보호에 대해서 분명한 계약 조건을 명시할 수 있어야 한다.

예를 들어 장애가 발생하면 기한 이내에 최대한 즉각적으로 문제를 해결하며, 장애에 따른 고객 손실을 특정 금액 내(예: 계약금)에서 보상해 준다는 약속이 있어야 한다. 이러한 외부 서비스에 대한 기본적인 항목과 그 내용을 살펴보면 아래 표와 같다. 물론 이 분류에 속하지 않은 많고 다양한 서비스 항목들이 있을 수 있다. 이들 추가적인 항목들은 외부 서비스의 수준 향상이 첫 번째 이유가 되어야 하고 이런 분명한 근거 위에서 고급 서비스에 대한 차별적인 가격을 청구할 수 있는 가격 정책을 수립하여야 한다.

〈표 13〉 기술 지원 서비스의 내용의 예시

서비스항목	주요 서비스 내용
설치	목적 소프트웨어를 고객이 원하는 시스템 환경에 옮기고 정상적으로 작동되도록 하는 작업
패치 제공	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
최적화	시스템 또는 DBMS의 성능을 향상시키거나 저장장치를 효과적으로 사용하기 위해서 파일, 네트워크, 인덱스, 캐시, 버퍼에 관련된 파라미터를 변경하는 작업
튜닝	시스템 성능 향상을 위한 환경변수를 조정하는 작업
문제 해결	목적 소프트웨어 자체의 문제(오동작, 에러, 버그, 해킹) 또는 운용 환경상의 문제(연동, 설정변경) 등을 기술적으로 해결해 주는 작업
모니터링	설치된 소프트웨어의 실시간 운용 상황을 동적으로 관찰하여 통계적 자료를 제공하는 업무
온라인 지원	포탈이나 이메일을 통해 질문이나 지원 요청을 접수하여 지식 베이스 또는 전담 기술인력을 동원하여 접수된 요청사항을 온라인으로 지원해주는 업무
기술 지원	마이그레이션, 커스터마이제이션, 백업 등 공개소프트웨어 서비스에 관련된 사항을 지도해 주는 업무
개발자 지원	아키텍처링, 파라미터 구성, 성능 튜닝, 최적화 등의 개발 업무에 대해 조언하는 업무
보증	공급한 공개소프트웨어 제품의 소스코드에 아무런 법적인 문제가 없도록 지원하는 업무

한편 공개소프트웨어를 적극 도입하고 최대한 활용하기 위해서는 계약에 있어서 관련된 이해관계자들과의 권한과 책임을 명확히 설정해야 한다. 공개소프트웨어의 특성상 라이선스의 다양성으로 인해 향후 도입, 활용에 있어서 예상하지 못한 라이선스 위반으로 인한 저작권 분쟁, 소유권 및 사용권 분쟁 등이 발생할 수 있다. 따라서 분쟁의 예방 차원에서 공개소프트웨어를 활용한 개발 및 발주, 공급을 포함한 일련의 공급망 관리에 대한 구체적인 내용이 계약내용에 포함되어야 한다. 관리의 부재와 소홀로 인해 발생하는 피해에 대해 책임을 명확히 기재하는 것이 향후 피해와 실패 비용을 최소화시키는 지름길이다.

공개소프트웨어의 가격 측면에서 라이선스 자체가 무료이며 배포할 당시 매체 또는 전송에 필요한 최소한의 비용만 청구할 수 있도록 규정하고 있기 때문에 사실상 비용이 발생하지 않는다고 볼 수 있다. 이렇게 가격은 무료이지만 공개소프트웨어를 배포 받고 사용하는 순간 해당 라이선스의 조건에 동의하여 계약한 것으로 간주된다. 즉, 이렇게 자동적으로 계약이 체결되는 순간부터 공개소프트웨어 사용자는 해당 라이선스 전문에 기재된 모든 의무사항을 성실히 준수할 의무를 가지게 된다.

한편 공개소프트웨어를 기반으로 한 사업 수행에서는 계약 사항이 조금 달라진다. 상용 제품의 계약서에는 일반적으로 보증에 대한 조건이 명시되어 있다. 이 조건이 특정 기간 이내에 발생하는 장애나 하자에 대해서 책임을 지겠다는 내용이다. 책임 한도에 대해서는 무한 책임을 요구할 수는 없지만 통례상 판매가격의 범위 내에서 보상 또는 배상으로 책임을 지고 있다. 반면에 공개소프트웨어는 아무런 보증도 약속하지 않고 있다. 가격이 무료이고 사용에 제한이 없으니 모든 위험 부담을 사용자가 가져가야 한다는 의미로 해석할 수 있다. 따라서 공개소프트웨어 사업은 라이선스가 아니라 기술 서비스를 중심으로 계약이 이루어진다. 즉, 가격 청구는 서비스 계약으로만 가능하다. 그리고 라이선스 위반에 대한 위험을 의식하여 계약을 꺼려하는 고객을 위해서 계약서 안에 면책 조항을 추가하기도 한다. 최근 라이선스 위험뿐만 아니라 제3자 특허 부분에서도 사고가 발생하고 있는 상황이므로 특허 침해에 대한 내용도 계약서에 반영해야 할 필요가 있다.

공개소프트웨어의 기술 서비스에 필요한 시스템 관리 및 유지보수를 주문하는 발주사는 무엇보다 현재 시스템에 대한 공개소프트웨어 사용현황과 라이선스 호환성을 검토해야 하며 이러한 사실을 바탕으로 수행자에게 명확한 공개소프트웨어 사용범위 및 라이선스 제약 조건을 제시해야 하고 향후 내부적으로 사용된 공개소프트웨어에 대한 재배포 및 재사용 범위를 명확히 해야 한다. 계약서에는 이러한 요구사항을 바탕으로 수행자의 산출물 및 유지보수 계약완료 이후의 내부 자산화 된 소프트웨어와 공개소프트웨어에 대한 기술적 재사용에 대한 범위와 책임을 명확히 하는 것이 필요하다.

다음은 공급망 차원에서 수요자와 공급자 간의 계약 조건이 매우 중요하다. 실제로 공개소

프트웨어 기반의 기술 지원 서비스를 제공하는 업체가 모든 문제의 책임을 가져가는 것이 사실이다. 그러나 문제가 발생한 경우 공급망 사슬에 있어서 공개소프트웨어가 전달되는 순서의 역순으로 책임이 전가된다. 따라서 수요자와 공급자는 계약을 체결할 당시 향후 발생할 모든 문제의 책임 소재를 분명히 하여야 한다. 공급망 구조에서 샘플로 참조할 수 있는 기본적인 양식은 다음과 같다.

- **계약의 목적**

- 공개소프트웨어의 서비스(설치, 개선, 운영, 유지, 교육, 컨설팅, 라이선스 검증 등)를 공급하기 위한 권리와 의무를 명확히 한다.

- **저작권 인정 및 라이선스 준수**

- 계약의 이행에 있어서 공개소프트웨어의 저작권, 라이선스, 특히 관련 법규와 준수 사항을 명시한다.

- **계약효력 및 범위**

- 계약 체결일로부터 1년 동안 적용한다.
- 계약기간 동안 추가적으로 체결되는 주문서, 계약서, 합의서, 각서, 견적서, 약정서 등에 대한 유효성과 우선순위를 정한다.

- **용어의 정의**

- 계약 대상인 제품을 정의하고 설명한다.
- 기본적으로 주문서, 견적서, 영업일, 소프트웨어, 기술지원 서비스, 라이선스 등을 정의한다.

- **계약목적물**

- 일반적으로 주문서와 견적서에 명시된다.
- 주문서와 견적서간에 일치하지 않는 부분이 있을 경우 무엇을 우선 할지 명시한다.

- **주문**

- 견적서에 제안된 상품에 대해 지정된 납품일자 내에 납품되었는지 확인할 수 있는 방법을 명시하고 주문서의 송부 방법을 명시한다.

- **계약이행 보증**

- 계약의 이행보증이 필요하다면 그 절차, 방법, 조건, 기간 등을 명시한다.
- 계약 불이행에 따른 귀책사유로 인한 계약 해지 및 손해배상 조건을 명시하고 이행 상황을 점검하기 위하여 관련 자료를 요청할 수 있는지 명시한다.

- **지급이행 보증**

- 선금금을 지급하는 경우 선금금 이행보증보험증권에 대한 절차, 방법, 조건을 명시한다.
- 일반적으로 이행 보증금은 선금금(V.A.T.포함) 전체에 해당하는 금액이며 이행 보증 기간은 통상적으로 주문서 발행일로부터 주문서의 인수 희망일까지로 한다.

• **납품**

- 계약목적물의 납품 장소, 검수조건(납품, 설치, 시운전) 및 일정을 명시 한다
- 계약금액의 변경을 요구하거나 납품을 거부할 수 있는 조건을 명시하고, 납품 시 발생하는 비용의 지불 주체를 명시한다.

• **납기지체의 보상**

- 납기에 이의가 있는 경우 통보 및 재협의 방법을 명시한다.
- 납기를 지연한 경우에 지체일수에 따른 지체배상금의 지급 방법을 명시한다.
- 귀책사유로 고객에게 지불하여야 할 지체보상금이 많을 경우에는 그 차액에 대한 배상 조건을 명시한다.
- 천재지변, 불가항력, 외부원인 등 지체보상금의 면제 조건을 명시한다.

• **납품의 변경 및 중지**

- 고객의 납품변경 또는 중지 요청이 있을 때와 같이 납품 내용의 변경 또는 일시 중지 에 대한 조건을 명시한다.
- 납기 변경 및 중지예 따른 상호 협의 절차, 방법, 조건을 명시한다.

• **검수 및 인도**

- 납품완료 후 검수 절차, 방법, 조건을 명시한다.
- 일반적으로 상세한 검수조건은 주문서 상에 명시한다.
- 검사일인 물품의 인도, 설치, 환경 구성 등을 완료한 직후로 명시한다.
- 완납 이전에 기납 부분이 있을 시 대가의 전부 또는 일부를 지급받을 수 있는지 명시한다.
- 검수에 대하여 이상이 있는 경우 시정 요청, 제품 변경, 재검수 등의 절차, 방법, 조건 을 명시한다.

• **대금청구 및 지급**

- 검수가 완료된 경우, 세금계산서, 거래명세서, 하자이행 보증서 등 청구에 필요한 서류를 명시한다.
- 대금지급 시기, 방법, 대금지급기준을 명시한다.

• **하자이행 보증**

- A/S, 교환, 반품, 회수 등 하자보증의 항목을 명시한다.
- 하자의 무상 또는 유상 보증기간을 명시한다.
- 하자이행 보증보험증권과 금액의 설정 조건을 명시한다.
- 판매한 상품의 하자과 관련하여 보증 책임의 조건과 처리 방법, 범위 등을 명시한다.
- 인력 투입, 출장, 장비 동원 등 하자이행 비용의 지불 절차, 방법, 조건을 명시한다.
- 제품상의 하자, 보안상에 결함이 있는 경우 리콜(수리, 교환, 환불) 비용 부담의 절차, 방법, 조건을 명시한다.
- 상품의 교환, 반환, 환불, 대금 정산의 절차, 방법, 조건을 명시한다.

• **계약의 해지**

- 계약을 해지하는 사유를 명시한다.
- 통상적으로 해지 사유는 영업취소, 허가취소, 영업정지, 가압류, 가처분 등의 처분, 담합, 결탁, 뇌물제공, 파산, 화의, 회사정리, 해산결의, 합병, 기밀 누설, 계약불이행, 납품 불가, 허위자료 등이 있다.
- **권리 및 의무의 양도**
 - 통상적으로 계약 후 발생하는 권리 또는 의무는 양도 또는 승계 할 수 없는 조건으로 명시한다.
 - 검수를 마친 계약 목적물을 매각, 양도, 대여하거나 담보로 제공할 수 없는 조건으로 명시한다.
 - 계약에 의한 권리, 의무를 제3자에게 양도하는 경우 양도한 부분에 대한 대가의 지급 절차, 방법, 조건을 명시한다.
- **위험부담**
 - 계약 목적물의 검수완료까지 전부 또는 일부가 멸실, 훼손, 변질되었을 경우 대체품의 납품 또는 손해 배상의 절차, 방법, 조건 등을 명시한다.
- **손해보상**
 - 상대방이 손해를 입은 경우에는 귀책과 배상 책임을 명시한다.
 - 손해 보상을 위한 산업재해 보상보험 등 필요한 보험의 가입여부를 명시한다.
- **기술지원**
 - 기술지원에 필요한 자료제공 또는 기술교육 등 부가적인 서비스에 대해서 유·무상 조건을 명시한다.
- **지적재산권**
 - 계약기간 동안 작성, 개발한 각종 프로그램, 툴, 문서, 양식 등 부가적인 자료에 대한 지적재산권의 소유에 대해서 명시한다.
 - 지적재산권의 사용과 유·무상 가격 조건을 명시한다.
 - 계약 목적물의 사용으로 계약 이행 시 산출되는 결과물 일체의 지적 재산권(저작권 및 판권포함)의 소유자를 명시한다.
 - 지적재산권의 분쟁으로 발생하는 손해에 대한 비용과 책임에 대한 면책 또는 배상 절차, 방법, 조건을 명시한다.
- **비밀유지**
 - 기밀정보를 정의하고 예외 대상을 명시한다.
 - 기밀정보의 공개, 보호, 누설에 대해 명시한다.
 - 기밀정보의 유효기간을 명시한다.
- **분쟁의 해결**
 - 계약의 수행 중 분쟁이 발생하는 경우에는 해결 원칙과 방법을 명시한다.
 - 쌍방간에 해결이 이루어 지지 않는 경우의 최종적인 해결 방법을 명시한다.

(9) 설계

설계는 공개소프트웨어 사용자 및 개발자를 통해 도출된 요구 사항을 충족시킬 수 있도록 컴포넌트, 모듈, 함수 등 기능 구현 단위의 스펙을 정의하고 기능 구현 단위 간의 호출 관계, 파일 간의 의존 관계, 서비스 레이어, 공통 파라미터 등 소프트웨어 구조 및 절차를 작성하는 활동이다.

공개소프트웨어의 설계 작업은 상용 제품과 마찬가지로 요구분석 단계에서 파악한 요구를 구현 가능한 설계안으로 바꾸는 작업이다. 요구는 개념적이고 추상적이나 설계는 구체적이고 실질적이어야 한다. 특히, 공개소프트웨어는 자체적으로 확보하지 않고 외부의 커뮤니티를 통해서 획득하기 때문에 상세설계 단계에서는 서비스 컴포넌트의 어떤 영역에서 공개소프트웨어가 활용되는지 식별하고, 직접개발 영역과 외부획득 영역을 구분하여야 한다. 따라서 다음과 같은 부분에서 각별한 주의가 요구된다.

- 커뮤니티에 공개할 설계영역과 내부에서 관리할 설계영역의 구분
- 내부개발자에 의한 기능의 구현
- 내부 조직에 의한 기능테스트 및 시험
- 알려진 버그에 대한 수정

공개소프트웨어 서비스 제공을 위해서는 공개소프트웨어를 그대로 사용하지 않고 맞춤형으로 개작하게 된다. 이러한 경우에는 안정적인 외부 서비스를 제공하기 위하여 향후 버전 관리와 업데이트에 문제가 없는 범위에서 맞춤형 설계를 진행하여야 한다.

공개소프트웨어는 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 분기 과정 등이 비공개소프트웨어의 관리와 다르며 이러한 부분이 비즈니스에서 매우 중요한 의미를 가지게 된다. 그렇기 때문에 공개소프트웨어 프로젝트의 자원자들이 자사의 소프트웨어에 대한 쉬운 접근이 가능하도록 소프트웨어 아키텍처상의 결합도를 낮추고 기능들의 변화가 용이하도록, 관리하는 아키텍처의 오염 지표 관리가 필요하다.

1) 소프트웨어 아키텍처

시스템이 더욱 복잡해지면서 시스템 분할을 잘 정리하는 것이 관건이 되었다. 개발이 시작 되면 잘못된 분할을 수정하거나 서브시스템의 인터페이스를 바로잡는 것이 매우 어렵기 때문이다. 이런 문제의 중요성을 인식하여 소프트웨어 아키텍처 개념이 생겨났다. 아키텍처적의 오염 지표를 없애고, 기능들을 분할할 때 사용할 수 있는 방법은 다음과 같다.

- 의존성 분석 도구 DSM (Dependency Structure Matrix)
 - 아키텍처를 시각화 하는 방법 중 서로간의 의존성을 분석하는 도구로는 Dependency Structure Matrix가 존재한다.
- 불안정성(Instability)
 - 불안정성은 패키지의 변화 여부를 측정하는 지표이다. 여기서 Stability는 안정성 보다는 “부동, 고정”이라는 의미로 해석 해야 한다. 그래서 “Instability”는 변경할 수 있는 여력으로 해석할 수 있다.
- Abstractness
 - Abstractness는 패키지나 Code Element에 Interface, Abstract Class가 얼마나 포함되어 있는지를 나타내는 지표이다.

2) 사용자 인터페이스 설계

설계는 사용자가 시스템과 상호작용하기 위한 방법을 포함하고 있는 사용자 인터페이스, 시스템 입력, 시스템 출력을 설계하는 작업을 포함한다. 어떤 사용자 인터페이스 설계 요소들이 있고 이들을 언제 사용하는지, 또한 사용자 요구에 맞는 시스템을 설계하는데 도움이 되는 스토리 보드 작성이나 프로토타이핑과 같은 방법을 이용한다.

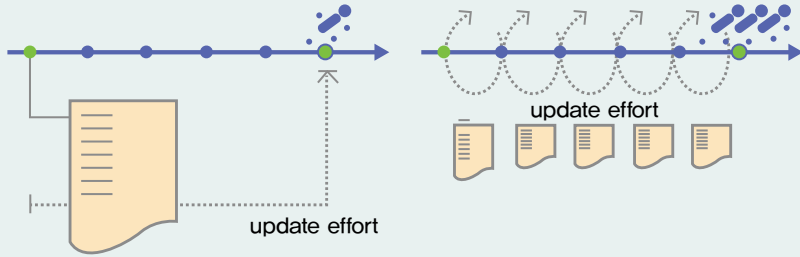
3) 데이터베이스 설계

자료를 저장할 데이터베이스에 대한 상세한 데이터 모델을 작성한다. 여기서는 저장할 자료를 결정하고 그들 사이의 관계를 찾아내는 개념적 스키마를 작성하는 작업이 필요하다. 여러 가지 정보 사이의 관계를 결정한 후에는 자료를 DBMS(Database Management System)에서 지원하는 정장 객체로 맵핑한다. 관계형 데이터베이스의 경우 저장 단위 객체는 테이블이다. 데이터베이스의 물리적인 설계 작업은 저장 매체에 있는 데이터베이스의 물리적인 형상, 자료요소, 자료 타입, 인덱스 선택 등에 대한 선택을 결정한다.

(10) 개발

공개소프트웨어의 개발 단계는 설계 단계에서 정의된 기능 구현 단위의 소스코드를 작성하고 각 구성 요소를 묶어서 하나의 통합 소프트웨어를 만드는 활동이다.

공개소프트웨어는 대부분 메인라인(mainline)을 바탕으로 개발 및 릴리즈를 운영하며, 메인라인을 기준으로 branch, fork등을 하여서 추가적인 개발을 한다. 이러한 공개소프트웨어의 메인라인을 기반으로 개발하는 것을 업스트림(upstream) 기반의 개발이라고 한다. 리눅스 커널을 비롯하여, 일반적인 SCM(Source Control Management)의 방법을 사용하고 있는 공개소프트웨어는 대부분 메인라인을 바탕으로 개발하고 있다.



[그림 25] 업스트림(upstream) 기반의 작업 패턴

외부 공개소프트웨어 커뮤니티에서 주도하는 기술이 요소기술이 되고 기업 고유의 소프트웨어가 기능적 차별화 또는 개선된 기능을 제공하는 경우라면, 해당 공개소프트웨어 커뮤니티에 기업 고유의 소프트웨어를 기여하는 방식으로 참여를 하는 것이 기술적, 전략적으로 훌륭한 선택이 될 수 있다. 즉, 기존 공개소프트웨어 커뮤니티의 기술적 기반, 활성화, 주요 개발자, 인지도 등 다양한 장점을 그대로 살리면서, 해당 공개소프트웨어의 장점을 훌륭히 부각할 수 있기 때문이다.

공개소프트웨어 개발 방법론은 폭포수 모델(Waterfall Model) 같은 전통적인 소프트웨어 개발 방법론은 적합하지 않다. 왜냐하면 이러한 전통적인 방식은 원칙적으로 전단계로 회귀하는 것이 허용되지 않기 때문이다. 대개 공개소프트웨어 소프트웨어 개발은 초기단계는 많지 않은 요구사항들을 가지고 빠르게 릴리즈 한 후에 자발적 참여자를 모으고 공론화 하는 과정을 거치면서 성장해 나간다. 따라서 빠른 프로토타이핑(prototyping) 후 점진적으로 반복 개발 방법이 효율적이다. 즉, 익스트림 프로그래밍(Extreme Programming) 같은 형태의 애자일 방법론(Agile method)이 적합하다.

공개소프트웨어 개발은 일반적으로 전 세계에 흩어진 개발자들이 온라인상에서 협업을 통해 개발을 진행한다. 따라서 개발에 필요한 도구들 또한 매우 중요하다. 물론, 상용소프트웨어의 경우와 유사한 경우도 있으나 공개소프트웨어 커뮤니티가 가지는 특수성으로 인한 환경 및 툴의 차이점은 존재한다. 대표적으로 의사소통 수단(Communication channels), 버전 관리(Version control system), 버그 추적(Bug trackers), 테스트 및 디버깅(Test & Debugging), 패키지 관리(Package Management) 등이 필요하다.

먼저, 의사소통 수단(Communication channels)이 필요하다. 일반적으로 가장 많이 활용되는 의사소통 수단은 이메일과 메일링리스트이다. 또한 실시간 통신을 위해서는 IRC(Internet Relay Chat) 같은 형태의 메시징을 이용한다.

공개소프트웨어 개발자들은 대부분 지역적으로 분산되어 있는 자발적 참여자들이기 때문에 작업 시간에 대한 편차나 소스코드 개발의 협업에 대한 관리를 위해 버전 관리 시스템 (Version Control System)이 반드시 필요하다. CVS(Concurrent Versions System)는 가장 유명한 공개소프트웨어 프로젝트에서 사용되는 소스코드 협업 툴이다. CVS는 여러 사람이 동시에 파일이나 소스코드에 접근하여 작업할 수 있는 환경을 제공해 준다. 최근에는 CVS를 대체하기 위하여 개발된 SVN(The Subversion revision control system)을 많이 사용하며, 리눅스 커널 관리에 이용되는 GIT, Python 프로그래밍 언어 관리에 사용되는 Mercurial 등도 많이 사용되고 있다.

(11) 패키징

패키징은 공개소프트웨어의 설치, 복제, 업데이트 등이 편리하도록 관련된 요소 프로그램을 모으고 구성에 필요한 보조 프로그램 또는 유틸리티를 추가하는 작업이다. 이 단계에서 직접 개발한 소스코드와 외부에서 획득한 공개소프트웨어를 결합하게 된다. 또한 이 과정에서 서비스 제공자는 성능향상 또는 안정성 등의 이유로 별도의 설치 파라미터 변경 또는 패치를 추가하게 된다.

공개소프트웨어는 소스코드를 공개해 주는 장점이 있지만 반면에 설치 및 업데이트 등의 사용 및 관리 편리성이 매우 약한 편이기 때문에 기술 지원 서비스의 품질을 위해서는 패키징이 매우 중요하다. 이는 패키징이 설치의 준비 과정이며 설치 이후의 운영과 유지에 필요한 요소를 제공하기 때문이다. 따라서 다양한 패키지 관리방법(YUM, APT, dpkg 등)을 통해서 공개소프트웨어가 손쉽게 설치, 업데이트, 삭제 등이 될 수 있도록 하여야 한다. 또한, 패키지 관리(Package Management)에는 소스(Source) 버전으로만 제공할 것인지, 바이너리(Binary) 버전을 제공할 것인지, 바이너리 버전을 제공한다면 순수 바이너리 혹은 패키지 관리 시스템(RPM, Red Hat Package Manager) 또는 APT(Advanced Packaging Tool)을 제공할 것인지에 대한 결정이 사전에 필요하다.

1) 플랫폼 레벨

바이너리 방식은 리눅스 배포판의 경우 dpkg, RPM, tgz, Pacman 등이 있으며, Open Solaris의 경우는 pkgadd, IPS를 사용한다. Cross-platform의 경우는 dpkg, IPS, OpenPKG 등이 있다.

(가) dpkg : Debian Linux에서 사용되는 고유의 패키징 시스템으로 .deb 형식을 사용한다. Cross-Platform을 지원한다.

(나) RPM : RPM(Redhat Package Manager)은 Red Hat에 의해 만들어 졌으며, 현재는 많은 다른 리눅스 배포판에서 사용하고 있다. RPM은 LSB(Linux Standard Base)의 패키징 형

식으로, Red Hat의 up2date, Mandriva의 urpmi, openSUSE의 ZYpp, 그리고, Yellow Dog Linux와 Fedore core의 YUM등의 기반으로 사용이 되고 있다.

(다) **tgz** : 표준 tar와 gzip을 이용한 tgz 패키지 생성 방식으로 Slackware Linux 및 파생 리눅스에서 사용되던 패키징 방식이다.

(라) **pacman** : Arch Linux, Deli Linux 등에서 사용하며 미리 컴파일 된 패키지를 tar를 이용하여 아카이브 형태로 압축하여 배포한다.

(마) **IPS(Image Packaging System)** : OpenSolaris와 Solaris에서 사용되며 pkg라고도 알려져 있다. Cross-Platform을 지원한다.

(바) **OpenPKG** : RPM Package Manager 기반의 Cross-platform 패키지 관리 시스템이다. Linux, BSD, Solaris와 같은 유닉스 기반 시스템에서 동작한다. 소스 컴파일 방식에는 portage/emerge, apt-build, ABS(Arch Build System)이 있으며, hybrid 방식의 ports가 있다.

(사) **ports** : FreeBSD Ports, 일반적으로 ports라고 알려져 있으며 소스나 바이너리를 가지고 Makefiles을 사용하여 설치하는 시스템이다. 기능이 유사한 시스템으로는 Mac OS X의 MacPorts, NetBSD의 pkgsrc 와 OpenBSD의 ports 등이 있다.

(아) **portage/emerge** : Gentoo Linux에서 사용되며, BSD Ports 시스템에서 파생되었다. ebuilds 라고 하는 스크립트를 사용하여 소프트웨어를 설치한다.

(자) **apt-build** : deb 소스 리파지토리에서 소프트웨어를 자동으로 컴파일하고 설치하는 기능을 제공하는 시스템으로, deb 패키지를 사용하는 리눅스 배포판에서 사용한다.

(차) **ABS(Arch Build System)** : 패키지 다운로드 및 의존성 검사를 자동으로 수행하며, 소스나 다른 바이너리 아카이브로부터 바이너리 패키지 빌드를 자동으로 수행한다. Arch Linux에서 사용한다.

2) 어플리케이션 레벨

어플리케이션 레벨의 패키징 방식은 프로그래밍 환경에 따라 각자 고유한 패키징 방식을 제공하고 있다.

(가) **Composer** : PHP를 위한 의존성 관리 도구이다.

(나) **PEAR** : PHP를 위한 프로그래밍 라이브러리이다.

(다) **CPAN** : Perl을 위한 프로그래밍 라이브러리 및 프로그램 관리자이다.

(라) **CRAN** : R을 위한 프로그래밍 라이브러리 및 프로그램 관리자이다.

(마) **Easyinstall** : Python eggs를 사용하는 Python을 위한 프로그래밍 라이브러리 및 프로그램 관리자이다.

- (사) PyPI : Python 패키지 관리자이다.
- (아) RubyGEMs : Ruby를 위한 리파지토리 및 패키지 관리자이다.
- (자) Ivy : Java용 패키지 관리자로 Ant 빌드 툴에 통합되었다.
- (차) Maven : Java를 위한 빌드 툴 및 패키지 관리자.

(12) 시험

시험은 요구 사항 반영 여부와 설계 단계에서 정의한 사양이 오류나 장애가 없이 정상적으로 동작하는지 공개소프트웨어의 기능과 성능 그리고 품질 등을 확인하고 규명하는 활동이다. 따라서 공개소프트웨어의 시험은 공개소프트웨어 제품이나 서비스의 품질에 대한 정보를 이해관계자에게 제공하기 위한 일련의 조사활동으로 볼 수 있다. 공개소프트웨어 시험은 공개소프트웨어에 대하여 설계나 구현 단계에서 정의된 요구사항들을 만족 하는지, 예상대로 동작되는지, 일관성 있게 실행이 되는지 그리고 이해관계당사자의 요구를 만족하는지를 확인하고 검증하는 절차로 정의할 수 있다.

공개소프트웨어 시험은 소프트웨어 개발 프로세스에서 어느 시점에 수행을 하느냐에 따라 다르다. 전통적으로 대부분 시험 프로세스는 코딩이 완료된 후에 수행을 하는 것이 일반적이나, 공개소프트웨어의 경우는 주로 애자일(Agile) 방식의 개발 프로세스를 이용하기 때문에 지속적으로 시험 프로세스가 진행이 되어야 한다. 이처럼, 시험의 방법론은 소프트웨어 개발 방법론에 따라 좌우된다.

또한, 이미 개발이 완료된 공개소프트웨어 시스템 전개의 경우 단위 시험 등 개발 과정 진행 중의 시험을 제외한 설치 혹은 설치 이전 환경 구성 단계에서부터의 시험 프로세스가 필요하다.

1) 시험 유형

시험 유형은 설치 테스트(Installation testing), 호환성 테스트(Compatibility testing), 스모크 테스트(Smoke testing), 회귀 테스트(Regressing testing), 기능/비기능 테스트(Functional vs Non-functional testing), 성능 테스트(Performance testing), 이용성 테스트(Usability testing), 국제화 & 지역화(Internationalization and localization) 등이 있다.

2) 시험 프로세스

전통적인 CMMI/Waterfall development 모델, 애자일(Agile or Extreme development model) 모델, Top-down & bottom-up 모델이 있다.

상용소프트웨어와 공개소프트웨어의 시험 절차나 방법론은 크게 다르지 않다. 다만, 시계열 순으로 최근의 공개소프트웨어 프로젝트에서 채택되고 있는 개발 방법론의 차이(Waterfall → Agile)로 인해 시험 방법이 다르다고 할 수 있겠다.

공개소프트웨어는 공유와 협업의 원칙으로 인해 개발 업무도 분산되어 있고 세분화 되어 있다. 따라서 매우 빈번하게 통합(Integration)이 발생하기 때문에 지속적인 시험관리 및 관리 자동화 툴이 반드시 필요하다. 대표적인 공개소프트웨어 기반의 시험관리 및 자동화 툴로는 FitNesse, Mozilla Testopia, Litmus 등이 있다.

(13) 배포

배포는 공개소프트웨어를 외부 사용자에게 전달해 주는 활동으로서 전달 매체(인터넷, 저장소, 문서), 배포물의 형태(소스코드, 실행 코드), 사용자 유형(고객, 불특정 개인·단체) 등에는 제한이 없다.

공개소프트웨어의 배포는 상용 제품과는 달리 배포 시 라이선스를 고지하고 배포해야 하는 의무사항의 준수 여부에 따라서 직접배포와 간접배포로 나누어진다. 이처럼 배포의 유형이 상용 제품과 다르기 때문에 달라지는 부분에 유의하여야 한다. 우선 공개소프트웨어의 배포에서는 소스코드의 전달을 감안하여야 한다. 그러나 상용 제품은 프로그램 안에 들어 있는 영업비밀, 핵심기술, 특허 등 지적 재산을 노출시키지 않아야 하기 때문에 육안으로 보기에 암호화된 파일과 동일한 실행 프로그램만 전달한다. 이렇게 보안된 형태의 배포는 기술의 유출이 없이 소프트웨어 사용에 대한 대가로 연 단위로 라이선스 요금을 징수하는 사업 모델에 있어서는 매우 중요한 요소이다.

또한 공개소프트웨어는 라이선스 종류에 따라 소스코드의 공개와 라이선스 고지의 의무가 주어진다. 즉 라이선스 조건에 따라 반드시 실행 파일에 상응(corresponding)하는 소스 코드를 제공할 의무가 있을 수 있다. 이는 공개소프트웨어를 배포 받는 자에게 배포하는 자가 실행 파일 뿐만 아니라 소스코드도 전달해야 하는 의무를 의미하는 것이다. 공개소프트웨어 라이선스는 전 세계에 현재까지 2,000여종이 넘게 공표되어 있고 OSI(Open Source Initiative)¹⁸⁾에서 인증한 라이선스 종류도 70여종이 넘음에 따라 해당 라이선스에서 요구하는 의무사항이 다양하다. GPL, LGPL, CPL, MPL 등 일부 공개소프트웨어 라이선스 중에서는 반드시 실행 파일에 상응하는 소스코드를 제공할 의무가 있을 수 있다. 즉, 공개소프트웨어를 배포 받는 자에게 배포하는 자가 실행 파일 뿐만 아니라 소스코드도 전달해야 하는 의무를 의미하는 것이다. 영리조직의 경우에 있어서는 조직의 영업비밀, 핵심기술, 특허 등 지적 재산을 노출시키지 않아야 하는 경우가 있기 때문에 공개소프트웨어를 직접 가져다 활용할 경우에는 배포 시에 발생하는 해당 라이선스 의무사항을 면밀히 검토하여 향후 저작권 분쟁, 특허분쟁, 지적재산의 노출 등의 위험을 회피해야 한다. 특히 조직의 독점 프로그램과 연결해서 상용 제품을 제작하는 경우에는 독점 프로그램의 소스코드도 같이 배포해야

18) OSI(Open Source Initiative) : 공개소프트웨어의 활성화 및 인증을 관장하기 위해 1998년 설립된 비영리 단체로 OSD(Open Source Definition)를 기준으로 인증을 진행한다.

하는 경우가 발생할 수 있기 때문에 배포 시에 공개소프트웨어의 포함 여부는 매우 중요하다. 소스코드 공개의 의무를 규정하고 있는 공개소프트웨어의 라이선스는 대부분 고지의 의무도 동시에 요구하고 있기 때문에 독점 개발된 상용 제품 보다는 배포 시 확인해야 하는 사항이 더욱 많게 된다.

일반적으로 공개소프트웨어는 이미 인터넷상의 커뮤니티 다운로드 웹페이지에 공개되어 있기 때문에 공개소프트웨어를 개작하지 않은 이상 소스코드를 별도로 배포할 의무는 없다. 그러나 소스코드를 다운 받을 수 있는 URL은 반드시 명기해야 한다.

마지막으로 공개소프트웨어가 포함된 프로그램을 배포할 경우에는 라이선스에 따라서 이름과 버전을 분명하게 표시해야 한다. 표시 방법으로 배포받는 자와 사용자가 쉽게 확인할 수 있도록 매뉴얼, 첫 실행화면, 메뉴 버튼 등을 사용할 수 있다. 라이선스의 이름과 버전만 가지고도 해당 선언문을 인터넷상에서 쉽게 검색하고 확인할 수 있지만 사용자의 편의를 위해서 배포되는 프로그램이 설치되는 기기에 저장하는 경우도 많이 있다.

(14) 설치

설치는 공급 계약된 공개소프트웨어를 최초 구동할 수 있도록 서비스 제공자가 직접 작업을 수행하거나 최초 구동에 필요한 소프트웨어 또는 관련 문서를 고객에게 제공하는 활동이다. 설치가 필요한 영역을 산업별로 살펴보면, 임베디드 소프트웨어 혹은 솔루션 산업에 있어서는 특별히 공개소프트웨어 사용과 관련하여 설치시에 특별한 검토사항은 없다. 다만 시스템 통합(System Integration, SI) 서비스 산업의 경우에는 공개소프트웨어의 특성상 설치 방법이 다양할 수 있기 때문에 공급 계약된 공개소프트웨어를 최초 구동할 수 있도록 서비스 제공자가 직접 작업을 수행하거나 최초 구동에 필요한 소프트웨어 또는 관련 문서를 고객에게 제공하여야 한다. 일반적으로 설치에 필요한 매뉴얼은 해당 커뮤니티에서 찾을 수 있지만 설치 내용이 복잡하거나 상세한 매뉴얼이 없는 경우에는 직접 설치하는데 큰 불편 사항이 있다. 물론 소스코드를 분석할 수 있는 개발자라면 직접 설치를 시도할 수도 있지만 파라미터와 구성에 따라 설치 후 작동과 성능이 크게 달라질 수 있으므로 최적 설치가 불가능할 수도 있다. 따라서 기술 지원 서비스를 전문적으로 수행하는 외부 벤더의 지원을 받기도 한다.

공개소프트웨어의 설치의 상용 제품과 별반 다를 바가 없다. 설치 중 가장 간단한 방법은 컴퓨터에 실행 파일을 복사하는 것이다. 그러나 공개소프트웨어가 복잡해지면서 단순한 복사 이상의 작업이 필요하게 되었다.

예를 들어 리눅스 환경에서 실행시키고자 한다면 어느 디렉터리에 실행 파일을 저장해야 하는지, 실행 파일을 구동시키는 시스템 명령어는 어떻게 지정해야 하는지, 현재의 시스템

환경에서 메모리, 하드디스크, 네트워크 등 성능에 영향을 주는 환경 변수에는 어떤 값들을 주어야 하는지 그리고 실행 파일 자체의 파라미터가 있다면 이를 어떻게 조정해야 하는지 설치에 관련되어 미리 준비되어야 하는 일들이 많이 생기게 되었다. 더구나, 특정 버전의 공개소프트웨어를 설치한 이후에도 커뮤니티는 지속적으로 버전을 업그레이드하기 때문에 최신 버전이 발표될 때마다 즉각적으로 업데이트할 필요가 있다. 이는 업그레이드 된 버전이 이전 버전의 문제점을 해결하고 성능과 기능면에서 향상되었기 때문에 안정적인 운영뿐만 아니라 효용성을 증대시켜주기 때문이다.

공개소프트웨어는 윈도우나 맥의 소프트웨어 설치와 다르게 다양한 설치방법을 제공한다. 특정 기업에서 제공하는 설치 방식이 아니라 사용자에게 소스코드와 다양한 설치법을 함께 제공하는 공개소프트웨어 특성으로 인하여 사용자들이 불편함을 호소하였으며 그로인해 공개소프트웨어 개발자들은 사용자 편의성 향상에 많은 노력을 해왔다. 현재는 다수의 공개소프트웨어들이 쉬운 설치 방법들을 지원하고 있다.

공개소프트웨어 설치방법은 크게 4가지로 분류 할 수 있다.

1) 소스코드 설치(컴파일)

컴파일은 가장 기본적인 설치 방법이다. 소프트웨어 개발자가 패키징을 해놓지 않았다면 제공되는 README 파일 또는 INSTALL 파일에 기재된 설치 방법을 보고 그대로 설치하는 유형이다. 대부분의 경우 GNU autotools를 이용해서 배포되기 때문에 Configure -> Make -> Make install 의 순서로 설치된다. 설치 후 소프트웨어 사용을 위한 다양한 환경 설정, 실행 파일의 구동 및 종료 등을 모두 직접 수정해야 하기 때문에 별도의 설치 지원 서비스를 제공하는 기업도 있다.

2) 바이너리 설치

자신의 환경에 적합한 바이너리 파일을 이용하는 설치방법이다. 소스 컴파일의 과정은 없지만 해당 바이너리 파일이 구동되기 위한 라이브러리 버전을 맞춰 줘야 하고, 설치 후의 환경설정 과정도 소스 컴파일 방식과 동일하다.

3) 패키지 매니저를 통한 패키지 설치

소스 설치나 바이너리 설치의 불편한 점 때문에 레드햇 계열이나, 데비안 계열의 리눅스 에서는 일찍이 패키지 매니저를 이용해서 어플리케이션들을 관리했다. 레드햇에서는 RPM, 데비안에서는 DPKG를 이용해서 어플리케이션을 설치하거나, 삭제 할 수 있다. 패키지 파일은 압축 파일로서 배포되는 소스 파일들 외에 설치 전 실행 스크립트, 설치 후 실행 스크립트 등을 가지고 있어서 사용자가 어플리케이션을 사용하기 위해 해야 하는 일들을 자동으로 해준다.

4) 원격 저장소를 이용하는 패키지 설치

최근의 알려진 공개소프트웨어들은 일반 사용자에게 가장 접근하기 힘들었던 점 중에 하나였던, 프로그램의 설치 및 제거를 개선하여 쉬운 설치방법을 제공하고 있다. 또한 사용자가 패키지 간에 발생 했던 의존성을 해결하기 위하여 일일이 패키지를 찾아다니지 않고, 명령어 한 줄로 간단하게 시스템에 어플리케이션을 설치할 수 있게 되었다. 레드햇에서는 YUM, 데비안에서는 APT-GET, 우분투에서는 APT-GET 뿐만 아니라 GUI 설치도구인 시냅틱이라는 패키지 관리자를 이용하여 손쉬운 설치 및 삭제를 제공한다.

(15) 운영

운영은 공개소프트웨어를 원활히 사용할 수 있도록 인프라, 하드웨어, 네트워크, 소프트웨어 등 제반 환경을 구성하고 설치된 공개소프트웨어의 목적 기능을 제공하는 활동이다. 자체적인 운영 능력이 있는 경우에는 비용 부담이 없이 공개소프트웨어를 사용할 수 있지만 용도의 중요성과 공개소프트웨어의 복잡성에 따라서는 외부 전문 업체의 지원이 필요한 경우가 있다.

운영은 공개소프트웨어를 구동시켜서 설계된 기능에 맞게 지속적으로 가동시키는 활동을 의미한다. 운영의 주요 요소에는 실행 명령어를 통한 기동, 더 이상 실행이 필요하지 않을 때의 정상 종료, 정상적인 운영 수준인지 판단하는 모니터링 및 로깅(logging), 비정상 종료 되었거나 비정상적인 수준이 아니라고 판단되었을 경우의 백업, 강제 종료, 복구, 재실행 등이 포함된다. 따라서 공개소프트웨어의 운영은 상용 제품과 특별히 다를 바가 없다.

운영 업무의 수행에 있어서는 외부 인력도 가능하지만 일반적으로 운영에 대한 책임은 직접 운영하고 있는 사용 주체가 지게 된다. 그래서 운영상의 문제로 인하여 정상적인 가동이 어려울 경우에 외부 업체와의 유지보수 계약을 체결해서 이러한 문제를 해결하고 기술적인 도움을 받게 되는 것이다.

이 단계에서는 공개소프트웨어 관리에 관련된 전문성과 기술지식을 보유한 운영 조직을 구성하고, 운영 조직은 공개소프트웨어 기술을 설계, 개발, 전환, 운영, 개선하기 위한 자원이 적절히 제공되고 효과적으로 훈련되는지 확인한다.

운영의 목적은 공개소프트웨어에 대한 관리적, 기능적 요구사항을 식별하여 조직의 비즈니스 프로세스를 효과적으로 지원하는 것으로 다음과 같다.

- 어플리케이션이 잘 설계되고 유연하고 비용 효과적이어야 함
- 요구된 기능이 가용한지 확인하여 비즈니스 성과를 달성하도록 해야함
- 조직의 적절한 기술력이 있는지 확인하여 어플리케이션을 항상 최적의 상태로 운영해야함

- 장애 발생 시 신속하게 진단하고 해결하도록 기술력을 사용해야함

한편, 공개소프트웨어를 직접 가져와서 개작 배포할 경우 일반적으로 임베디드 소프트웨어 혹은 솔루션산업의 경우에는 특별한 운영검토사항이 없다. 다만, S서비스 산업의 경우 계약 조건에 따라 운영을 대행할 수 있어서 사전에 체결한 계약 조건에 따라 고객이 원하는 수준으로 공개소프트웨어를 원활히 사용할 수 있도록 인프라, 하드웨어, 네트워크, 소프트웨어 등 제반 환경을 구성하고 설치된 공개소프트웨어의 목적 기능을 제공해야 한다. 이 경우 공개소프트웨어는 지속적으로 새로운 패치, 버전이 개발됨에 따라 지속적인 모니터링이 필요하고 이를 운영절차에 반영해야 하기 때문에 단순한 운영에서 확장된 유지보수의 개념이 포함될 수 있다. 따라서 운영에 따른 책임과 범위를 명확히 하여 운영계약에 반영하여야 한다.

(16) 유지보수

유지보수는 공개소프트웨어를 최고의 운영 상태로 구동시키기 위해서 버전 업데이트 및 패치, 버그 수정, 보안 취약점 개선, 설정 변경 등 보완과 개선을 지속하는 활동이다. 이 활동을 자체적으로 수행할 수도 있지만 운영 특성과 중요도에 따라 외부의 서비스를 유료로 이용하는 경우가 있다. 이 서비스를 제공하는 외부 업체는 일정 기간 동안 해당 공개소프트웨어의 최적 운영 상태를 보장하는 계약을 유료로 체결하게 된다.

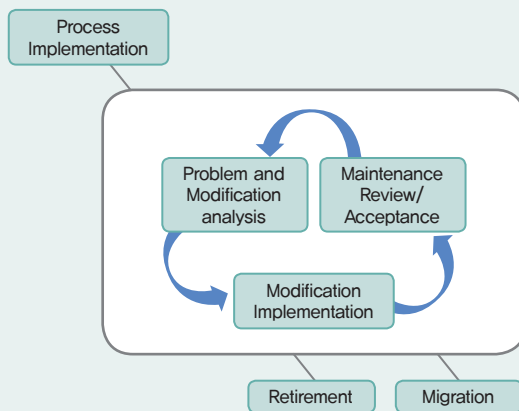
유지보수는 설치된 소프트웨어가 최상의 상태로 운영되도록 수행하는 일련의 활동이다. 이 활동에는 최상의 운영을 저해하는 제품 자체의 버그, 오류, 장애 등의 문제 해결과 패치 적용이 대표적이다.

상용 제품도 마찬가지로이지만 공개소프트웨어 또한 독립적으로 실행되지 않고 연결되는 소프트웨어와 연동하는 경우가 많으므로 연동 및 연계에 따른 문제 해결도 이 범위에 포함시킬 수 있다. 더욱이 문제 발생에 따른 오작동, 서비스 정지, 보안 자료 유출 등으로 인하여 심각한 피해가 발생할 수 있으므로 유지보수 계약을 통해서 안전장치를 확보하여야 한다. 이러한 부분에 있어서는 상용 제품일 경우 무조건 유지보수 계약을 체결하는 경우가 대부분이지만 공개소프트웨어에 있어서는 비용 절감을 고려해서 유지보수 서비스 없이 자체적으로 해결하는 경우도 있다.

그러나 핵심 업무지원 서비스에 공개소프트웨어가 적용되었으면 고객 대상 서비스 수준이 가장 중요하므로 전문 공개소프트웨어 기술업체를 통해 유지 보수 서비스를 받아야 한다. 보통은 이러한 유지보수 비용을 가산하더라도 전체적인 비용인 TCO가 크게 감소되기 때문에 공개소프트웨어를 도입하는 만큼 내부 자체적인 운영이 전제되었거나 비핵심 업무 시스템이 아니라면 유지보수 서비스를 받아야 한다.

공개소프트웨어에서는 특히 유지보수가 연간 서비스 계약(Subscription)을 통해서 지원된다. 즉, 유지보수 서비스를 제공하기 위해서 상시 대기하거나 운영해야 하는 인력들이 필요하기 때문에 유지보수 항목의 실행뿐만 아니라 준비 및 개선 업무에 필요한 부가적인 비용을 포함시켜 포괄적인 의미의 서비스를 제공하고 있는 것이다.

ISO/IEC의 유지보수 프로세스는 6개의 활동으로 구분되어 있다. 프로세스 실행활동은 프로세스에서 제품 인도 이전(Pre-delivery) 활동으로만 고려된다. 제품 인도 이전의 활동은 소프트웨어 라이프 사이클에서 설계나 개발 상태에서 수행된다. 제품 인도 이전 활동의 목적은 유지보수 프로세스에 대한 환경과 계획을 생성하기 위함이다. 다른 활동들은 제품 인도 이후(Post-delivery) 활동으로 고려되고, 제품 인도 이후에 수행된다. 이들의 목적은 잘 통제된 방식으로 수정을 계획하고 실행하기 위함이다. 추가로, 이러한 활동들은 소프트웨어 마이그레이션과 폐기를 지원한다.



[그림 26] ISO/IEC 14764 유지보수 프로세스

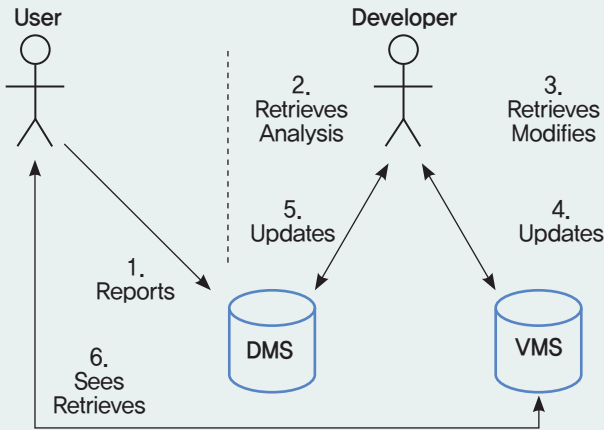
프로세스 수행 활동은 유지보수 계획과 절차를 개발하고 수정 요청에 대한 절차를 수립하고 환경구성관리 프로세스를 실행하는 업무를 처리한다. 프로세스 수행의 목적은 유지보수 프로세스의 유효관리를 위한 환경을 형식화 하는 것이다.

공개소프트웨어 유지보수 프로세스는 ISO/IEC 유지보수 프로세스 모델과 매우 유사하다. 하지만 차이점은 공개소프트웨어 유지보수 프로세스는 잘 설명된 마이그레이션과 폐기 활동을 가지고 있지 않다는 것과 설계 업무가 잘 정제되어 있지 않다는 것이다. 공개소프트웨어 개발자와 사용자는 지역적으로 분산되어 있다. 따라서 결함 보고서나 소스코드의 관리가

프로젝트 성공의 중요한 요소이다. 이러한 이유로 공개소프트웨어 프로젝트에서는 최소한 결함관리시스템과 버전관리시스템 두 가지를 사용해야 한다.

결함관리시스템의 목적은 결함보고서 관리를 위한 절차를 수립하기 위함이다. 결함관리시스템은 결함 보고와 수정 요청을 할당하고 보고하는 절차를 제공한다. 또한 결함보고서에서 만들어진 모든 변경사항을 저장하기 때문에 결함 제어 및 추적에 대한 유연한 환경을 제공한다. 버전관리시스템은 소스코드관리에 대한 환경을 수립한다.

공개소프트웨어 유지보수 프로세스의 활동은 아래와 같다.



[그림 27] 공개소프트웨어 유지보수 단계별 활동

공개소프트웨어 유지보수 활동들은 다음을 포함한다.

- 사용자는 결함을 찾아서 그것을 결함관리시스템에 보고한다.
- 사용자는 유사 결함이 있는지 확인해 본다.
- 개발자는 결함관리시스템에서 결함정보를 가지고 와서 결함의 존재를 보장하고 분석하여 결함의 누락된 속성을 완성한다.
- 실행단계에서 개발자는 버전관리시스템에서 소스코드를 가지고 와서 결함을 수정하거나 개선을 위해 소스코드를 수정한다.
- 추가로 개발자는 수정된 사항을 버전관리시스템에 등록한다.
- 수정사항은 버전관리시스템에서 주 버전의 소스코드에 허용되어 합치기 전에 검토를 해야 한다.

- 결함의 해결과 그 상태정보는 결함관리시스템에 갱신된다.
- 이후, 버전관리시스템에서 새로운 버전의 소프트웨어를 이용 가능하다.
- 그 다음 사용자는 수정된 버전을 가지고 올 수 있다.

한편, 유지보수가 적용되는 산업 영역별로 구분해서 살펴보면 공개소프트웨어를 직접 가져와서 개작 할 때 일반적으로 임베디드 소프트웨어 혹은 솔루션 산업의 경우에는 특별한 유지보수 검토사항이 없다. 다만, SI서비스 산업의 경우 유지보수 계약을 체결할 경우 서비스 제공자가 공개소프트웨어를 사용하는 고객에게 최적의 운영 상태를 보장하기 위해서 계약 기간 동안 버전 업데이트 및 패치, 버그 수정, 보안 취약점 개선, 설정 변경 등 보완과 개선을 지속하는 활동을 제공하여야 한다.

유지보수는 설치된 소프트웨어가 최상의 상태로 운영되도록 수행하는 일련의 활동이므로 최상의 운영을 저해하는 제품 자체의 버그, 오류, 장애 등의 문제 해결과 패치 적용이 대표적인 실행 항목이다. SI 서비스 산업의 경우 공개소프트웨어를 가져와 개작 사용 후 제공하였으면 해당 공개소프트웨어에 대해 상용 제품과 유사한 수준에서의 유지보수 제공이 필요하며, 공개소프트웨어는 독립적으로 실행되지 않고 연결되는 소프트웨어와 연동하는 경우가 많이 있으므로 연동 및 연계에 따른 문제 해결도 이 범위에 포함시킬 수 있다. 더욱이 문제 발생에 따른 오작동, 서비스 정지, 보안 자료 유출 등으로 인하여 심각한 피해가 발생할 수 있으므로 고객의 입장에서는 유지보수 계약을 통해서 안전장치를 확보하여야 하지만 공개소프트웨어는 코드 및 제반 기술이 공개되어 있는 만큼 비용 절감을 고려해서 유지보수 서비스 없이 자체적으로 해결하는 경우도 있다. 여기서 중요한 점은 공개소프트웨어를 개작 사용 후 고객에게 서비스 하는 경우 핵심업무에 적용된 서비스에 대해서는 SI서비스 제공자가 활용한 공개소프트웨어에 대해 전문 기술을 확보 하던가 전문 기술서비스 업체와의 컨소시엄 계약을 통해 유지보수 서비스를 제공하는 것이 바람직하다.

(17) 기술지원

기술지원은 서비스 제공자가 공개소프트웨어를 사용하는 고객에게 성능 개선, 장애 해결, 오류 수정, 추가 요구 사항 반영 등 공개 소프트웨어 기술 기반의 서비스를 제공하는 활동이다. 가입자 계약에 의해서 제공되는 운영 및 유지 보수 서비스뿐만 아니라 요청 기반의 단기 서비스도 포함 된다.

개발자가 모두 인접해있는 상용소프트웨어와는 다르게 공개소프트웨어 프로젝트는 주로 원격에서 이루어지는 특성을 가지고 있기 때문에 커뮤니케이션 방식이 다르고 이메일, 전화, 메신저등은 공개소프트웨어 프로젝트에서 공동작업을 하기에 적합하지 않기 때문에 버그나 이슈의 처리를 위한 도구가 필요하다.

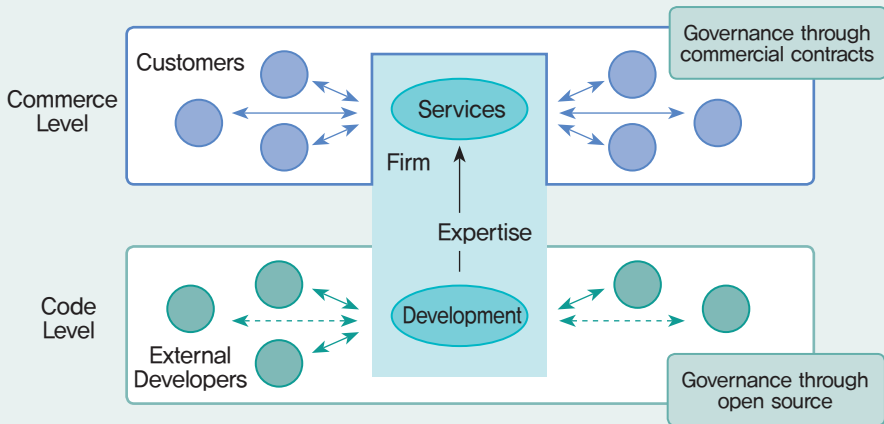
Bugzilla, Mantis, Trac, Jira 등의 이슈 트래커 또는 버그 트래커는 기업이 공개소프트웨어 전환 후 프로젝트의 관리과정에서 없어서는 안되는 중요한 도구이다. 전환한 소프트웨어의 문제점을 발견하고 해결하기 위한 과정을 시스템화 한 것으로 문제의 발견, 증상의 규격화 된 식별, 원인 규명, 재구현, 문제해결의 과정을 모두 관리할 수 있다.

기술지원이 필요한 영역을 산업별로 살펴보면, 공개소프트웨어를 직접 가져와서 개작 배포 할 때 일반적으로 임베디드 소프트웨어 혹은 솔루션산업의 경우에는 특별한 기술지원 검토 사항이 없다. 다만, SI 서비스 산업에 있어 유지보수 계약을 체결할 경우 제공된 시스템 및 솔루션에 있어서 공개소프트웨어를 활용하였으면 해당 공개소프트웨어에 대한 기술지원 서비스가 유지보수 계약의 일환으로 포함될 수 있다. 문제는 해당 공개소프트웨어는 커뮤니티에서 개발, 패치, 유지보수 되는 만큼 서비스 제공자가 모든 기술지원에 대해 유연히 대처하고 수행함에 있어서 한계가 있다. 따라서 전문 기술에 대한 역량 확보 혹은 전문기술업체와의 협력이 필요하다.

(18) 커뮤니티

커뮤니티 활동은 개발에 참여하여 사양을 정하고 소스코드를 기여하며 문제를 해결하는 것 뿐만 아니라, 재정적 지원, 교류 참여 등도 포함된다. 이처럼 커뮤니티에 참여하는 이유는 공개소프트웨어를 활용하여 외부서비스를 제공할 때 외부의 리소스를 적극적으로 활용하기 위해서이다. 이미 알려진 데로, 대표적인 외부 리소스인 공개소프트웨어 개발자 중에는 역량과 열정을 갖추고 있는 고급 인력들이 많다. 특히 리눅스 커널, GNOME, Apache HTTP Server 등 전 세계적으로 매우 인정받고 많이 사용되고 있는 공개소프트웨어 일수록 핵심 개발자들의 역량은 상상을 초월하는 경우가 많다.

일반적인 규모의 기업에서는 대규모의 소프트웨어 인력을 운영하기가 어렵기 때문에, 고도의 기술을 요구하는 프로젝트에 대해서 마음껏 인력을 투입하기 어렵다. 인력이 있다고 하더라도 원하는 기술을 달성하기 위한 고급 개발자는 더욱욱 찾기 어렵다. 따라서 공개소프트웨어 커뮤니티의 참여를 기반으로 적극적인 외부 리소스의 활용이 필요하며 이를 위해서는 어떻게 외부 개발자들과 소통하고 협업할 것인가, 어떻게 같이 성장할 것인가에 대해서 심각하게 고민할 필요가 있다.



[그림 28] 공개소프트웨어와 외부 리소스의 활용

공개소프트웨어 커뮤니티 참여자에는 개발자, 관리자, 사용자등이 있으며 개발자 간 소통 방법에는 커뮤니티 참여자의 유형에 따라 편리하게 사용할 수 있는 메일링 리스트, 포럼, 버그 리포트, 프로젝트 관련 문서, FAQ 등을 준비해야 한다. 이러한 대응 자료가 준비되고 커뮤니티 참여자들과 소통을 강화하여야 커뮤니티 조직이 구성되며, 버그 리포트에 대한 신속한 대응과 사용자 및 참여 개발자의 문제점을 안정적으로 해결함으로써 커뮤니티는 활성화되며 이렇게 형성된 커뮤니티가 제품의 문제점 해결 방안 및 요구사항을 발생하고 스스로 해결해 나간다.

소스코드 차원에서 공개소프트웨어 커뮤니티에 참여하는 유형을 나누어 보면 크게 단순 사용, 적극적 참여 그리고 전략적 사용으로 분류될 수 있다.

1) 단순 사용

- 공개소프트웨어에 대해 어떠한 개작도 없이, 있는 그대로 사용만 하는 방법
- 사용 버전의 선택은 자유로우며, 버전 반영에는 시간적 지연이 발생할 수 있음
- 어떠한 상황에서도 커뮤니티의 현재 뿐만 아니라 향후의 어떠한 요구 사항과 기능 정의 그리고 로드맵까지 그대로 수용해야 함
- 공개소프트웨어 커뮤니티를 전혀 기술적으로 선도할 수 없거나 공개소프트웨어 커뮤니티의 발전 방향이 참여자의 향후 계획과 일치하는 경우에만 적합함

2) 적극적 참여

- 현재 운영 중에 있는 공개소프트웨어 커뮤니티에 공개소프트웨어 커뮤니티를 주도적으로 리드하는 의사 결정자 또는 전문 개발자를 투입하여 커뮤니티에 전폭적으로 기여하

는 형태임

- 내부에 보유하고 있는 소프트웨어를 외부에 공개하고 주도적으로 이 커뮤니티를 운영하는 경우도 이 부류에 속함
- 적극적인 참여로 커뮤니티 운영과 공개소프트웨어의 확산이 잘되면, 사실상 업계 표준으로 시장을 선도하거나 독점적인 상용 제품을 견제하는데 매우 효과적임
- 주도적인 사용자에게 필요한 기능 및 사양 변경에 대한 우선순위를 높임으로서 외부 리소스를 무료로 개발하거나 테스트에 이용할 수 있음

3) 전략적 사용

- 분기한 공개소프트웨어 버전에 기초하여 맞춤형 개선을 통해 사용자 요구사항을 독자적으로 반영하고 자체 소프트웨어를 확보하는 데 목적이 있음
- 원조 커뮤니티의 개선 사항을 반드시 반영할 필요는 없음
- 기존 공개소프트웨어의 업그레이드된 버전을 반영하려면 최적화된 소스코드를 이식해야 하므로 엄청난 코딩과 테스트 작업이 필요함
- 한 번 분기된 프로젝트를 기존 커뮤니티로 합류하는 것은 거의 불가능함
- 분기는 맞춤형 개선 사항을 선호하는 고객이 계속 확보되거나, 아니면 내부적인 용도만으로 사용할 경우가 아니면 고려하지 않는 것이 현명함

소스코드 차원 이외에도 문서화, 운영, 커뮤니케이션 등 다양한 방법으로 공개소프트웨어 커뮤니티에 참여할 수 있다. 공개소프트웨어가 상용 제품에 비해서 매뉴얼, 샘플, 교육 자료 등 프로젝트를 이해하고 설명하는 보조 자료가 부족하기 때문에 문서화 작업이 커뮤니티 참여의 방법 중 하나가 될 수 있다. 공개소프트웨어 커뮤니티가 인터넷 상의 가상 조직으로 의사 결정, 조직구성, 정책수립 등 운영에 해당되는 일들에 참여할 기회가 많이 있다. 또한 커뮤니티의 특성상 집단 지식 및 지적 자산의 공유, 협업, 연락 등 커뮤니티 내의 커뮤니케이션에 적극 참여하여야 한다.

커뮤니티 활동이 필요한 영역을 산업별로 살펴보면, 공개소프트웨어를 직접 가져와서 복제 혹은 개작 배포할 경우 일반적으로 임베디드 소프트웨어 혹은 솔루션산업의 경우 뿐 아니라 SI 서비스 산업의 경우에도 내부정책에 따라 특별한 커뮤니티 활동에 대한 검토사항이 없을 수 있다. 다만, 기본적으로 조직에서 외부 공개소프트웨어를 가져와 배포할 경우 개발 현황, 품질, 보안취약점, 라이선스 등에 대한 동향과 검토 등을 포함한 모니터링이 필요함에 따라 해당 공개소프트웨어의 커뮤니티에 대한 모니터링 전략이 필요하며 필요시에 커뮤니티에 적극 참여하여 해당 기술 및 전략을 선도하는 커뮤니티 활성화전략이 필요할 수 있다. 현재의 소프트웨어 개발환경이 조직내부에서 대규모의 소프트웨어 인력을 운영하기가 어렵고 장기적인 투자와 고도의 기술을 요구하는 프로젝트에 대해서 충분한 인력을 투입하기 어려운 것이 사실이어서 가급적 공개소프트웨어 커뮤니티의 참여를 기반으로 적극적인 외

부 리소스 활용을 위해서는 어떻게 외부 개발자들과 소통하고 협업할 것인가, 어떻게 같이 성장할 것 인가에 대해서 심각하게 고민하고 운용할 필요가 있다.

(19) 컴플라이언스

컴플라이언스 실행 항목은 공개 소프트웨어가 피해포자에게 전달되는 과정에서 배포자가 라이선스 의무사항을 준수하는 활동으로서 공개 소프트웨어의 프로젝트 명칭, 원저작자 표기, 저작권 선언, 라이선스 명칭, 파일 경로, 결합 방법, 충돌 여부 등 준법에 관련된 정보를 제공한다.

공개소프트웨어는 소스코드를 공개하여 누구나 활용이 가능하게 함으로서 소프트웨어의 사용이 증가하는 반면 공개소프트웨어 라이선스에 대한 부족한 인식으로 법적 분쟁의 발생, 기업 이미지 하락 등의 부정적인 문제가 발생할 가능성이 있다. 공개소프트웨어도 저작권이 있으며 라이선스별로 사용과 배포 등에 관련된 다양한 의무사항을 요구하고 있으므로 공개 소프트웨어 전환을 선택함에 있어서 라이선스의 특징과 의무사항을 다음의 항목별로 면밀히 검토해야 한다.

1) 사용권 고지의 의무

- 누가 해당 소프트웨어를 개발하였는지 공지함
- 고객에게 어떤 공개소프트웨어를 사용하였는지 알림

2) 저작권 고지의 의무

- 소스코드상에 이미 표시되어 있는 저작권 관련 문구는 절대로 삭제하여서는 안됨

3) 소스코드 공개의 의무

- 일부 공개소프트웨어 라이선스는 공개소프트웨어로 개발한 결과물의 소스코드 공개를 의무화 하고 있음

4) 특허 포기의 의무

- 사용자가 해당 공개소프트웨어를 만든 당사자에게 특허 침해 소송을 제기할 수 없음

공개소프트웨어의 컴플라이언스 문제는 자칫 잘못하면 활성화에 따른 긍정적인 효과를 감쇄시킬 수도 있다. 따라서 투자비용, 투입 인력, 관리 방법 등 다양한 측면에서 적절한 방법으로 법적 위험의 예방과 사후 처리 활동을 수행하여야 한다. 우선 연구, 교육, 실험, 자체 운영 등 내부적으로 공개소프트웨어를 사용하고 외부로 배포하지 않으면 컴플라이언스 위험이 없음을 주지하여야 한다. AGPL(Affero General Public License)로 선언된 소프트웨어

가 네트워크 인터페이스를 통해서 다른 소프트웨어와 연동될 경우에는 상대방 소프트웨어의 소스코드도 공개해야 하는 의무가 있다. 이러한 특별한 경우를 제외하고는 외부 배포가 없다면 공개소프트웨어 라이선스의 의무사항을 준수할 필요가 없으므로 최대한 적극적으로 사용하여야 한다. 다만, GPL 계열, CPL, EPL, MPL, OSL 등 copyleft에 의해서 공개의 의무가 주어진 공개소프트웨어를 외부에 배포하는 경우에는 주의를 기울여야 한다.

라이선스 위반에 대한 효과적인 예방 방법은 교육, 프로세스, 사전 라이선스 식별이 대표적이다. 또한 공개소프트웨어 저작권 보호 단체나 라이선스 이슈가 있는 공개소프트웨어 커뮤니티와 대외적인 협력관계를 유지하는 것도 훌륭한 예방책이 될 수 있다. 한편 저작권자 또는 외부 단체가 저작권 위반을 발견하고 소송이나 권리를 주장한 경우에는 다음과 같은 사후 처리 방법으로 조치할 수 있다.

- 라이선스 충돌을 유발시키는 공개소프트웨어를 제거하거나 프로세스 분리와 같은 방법으로 충돌을 우회한다.
- 라이선스 충돌을 유발시키지 않는 공개소프트웨어로 대체한다.
- 공개소프트웨어와 연결되는 자체 개발 소프트웨어를 제거한다.
- 위에서 열거한 우회, 대체, 제거 등의 방법이 불가능할 경우에는 자체 개발 소프트웨어를 공개한다.
- 라이선스 규정에 따라 배포의 의무를 준수한다.

(20) 교육

교육은 서비스 제공자가 공개 소프트웨어를 사용하는 고객에게 공개 소프트웨어에 관련된 제반 기술, 노하우, 기술, 경험, 유tm케이스, 교훈, 샘플, 패턴 등을 현장 또는 온라인 환경에서 제공하는 활동이다.

교육은 크게 내부 교육 활동과 외부 교육 서비스로 구분할 수 있다. 내부 교육 활동은 자체적인 운영과 사용을 위해서 내부 인력을 대상으로 수행하게 된다. 반면에 외부 교육 서비스는 하나의 공개소프트웨어 사업 모델로서 교육 교재, 강의, 자격증 등을 통해서 수익을 창출하게 된다. 내부 교육 활동에 있어서는 기술적인 면에서 지식과 스킬을 내재화시키는 효과도 있지만 공개소프트웨어라는 이질적인 시스템 도입에 따른 변화관리의 효과도 매우 크다고 볼 수 있다. 모든 구성원들이 조직의 공개소프트웨어 정책을 정확히 인지하여 소프트웨어를 개발, 수정, 획득, 사용하는데 참여하고 공개소프트웨어 관련 조직의 지침들을 준수하는 것이 중요하다.

공개소프트웨어 정책을 수립해온 조직이 공개소프트웨어 교육 계획을 수립할 때 두 가지 그룹을 고려하는 것이 필요하다.

첫 번째 그룹은 현재 조직 내 구성원들로서 소프트웨어 개발에 직접 참여하는 개발자 및 관리자 그룹과 공개소프트웨어 정책 프로세스에 포함되는 법률 자문 그룹, 마케팅 및 세일즈 그룹 등이 포함될 수 있으며 이들 구성원 중 일부는 조직의 공개소프트웨어 정책을 수립하는 데 참여한 주도 그룹이 있을 수 있고 조직 내 공개소프트웨어 정책의 존재와 정책의 특정 요구사항들을 전혀 인지하고 못하고 있는 그룹이 있을 수 있다.

두 번째 그룹은 개발 공급망 관리에 포함되는 외부 구성원들로서 조직의 제품 및 솔루션 개발에 참여하는 개인 개발자, 외주협력업체, 서드파티 공급업체 등의 구성원이 포함되어야 하며 조직의 공개소프트웨어 정책 및 필요 요구사항들을 준수 하고 개발 프로세스에 반영할 수 있도록 인지시켜야 한다.

결론적으로, 조직은 두 단계의 교육계획을 분리 수립해야 하며 첫 번째 단계는 새로운 공개소프트웨어 정책을 현재 조직 내 구성원들의 필요역량에 부합되게 적절한 교육내용을 전달하는 것이며, 두 번째 단계는 정기적으로 필요시 되는 조직 외부 구성원들을 위한 교육계획을 수립해야 한다.

(21) 모니터링

모니터링은 협의의 의미에서 사용자가 사용하는 공개 소프트웨어의 상태를 주기적으로 점검하고 기록하는 활동과, 광의의 의미에서 향후 안정적인 운영과 유지 보수를 보장하기 위해서 해당 공개 소프트웨어 커뮤니티의 동향, 진행 현황 및 주요 이슈 등을 주기적으로 추적하고 검토하는 활동까지 포함한다.

협의의 모니터링 활동은 공개소프트웨어의 특성상 상용소프트웨어와 상이점이 적지 않다. 이 모니터링 활동에 입력이 되는 공개소프트웨어 사용에 대한 피드백은 상용소프트웨어에 대한 것보다 훨씬 중요하다. 일반적으로 공개소프트웨어는 개발자 중심으로 개발되었기 때문에 사용자 관점의 다양하고 자세한 요구사항을 수용하지 못하고 있다. 따라서 새롭게 설치되어 운영되고 있는 공개소프트웨어에 대해서 꾸준히 사용자의 의견을 수집하고 해당 공개소프트웨어 커뮤니티에 반영할 필요가 있다. 이렇게 수렴된 의견이 차기 버전의 요구사항이 되어 추가적인 개선사항이 도출되고 맞춤형 작업이 커뮤니티를 통해서 진행되도록 하여야 한다. 이는 완성품으로 배달된 상용소프트웨어와는 달리 사용자가 제품 특성에 맞추기 어려울 경우에는 소스코드 수준에서의 개선이 필요하기 때문이다.

공개소프트웨어를 사용자의 요구사항에 따라 자체적으로 변경할 경우에는 무엇보다도 빠르고 정확한 피드백의 수집이 중요하다. 이를 위해서는 공개소프트웨어 운영에 책임이 있는 주관 부서와 현장과의 연락 통로를 만들어야 한다. 이 연결 통로를 만드는 가장 편리한 방법 중에 하나는 “foss@” 또는 “공개소프트웨어@”와 같이 가상 이메일 주소를 개설하는 것이다. 물론 현장의 연락 창구 업무를 수행하는 담당자를 지정할 수도 있지만, 담당자가 변경되

는 상황도 있고 업무량이 많아 여러 명의 담당자가 필요할 수도 있으므로 가상 이메일을 공유하는 방법은 매우 효과적일 수 있다. 또한 이 연결 창구는 양방향 소통의 통로가 되어야 한다. 즉, 현장의 요구사항, 불만, 애로점, 제안 등을 수렴하고 즉각적인 응대의 수단으로 사용되어야 한다.

추가적으로 이 단계에서는 공개소프트웨어 라이프사이클 전 영역에서 개선 기회의 리뷰, 평가, 추천이 가능하도록 결과에 대한 리뷰 및 분석이 필요하다. 많은 공개소프트웨어 기술지원 서비스 업체들이 지속적인 개선을 수행하는 것이 필요하다는 것을 인지하고 있지만, 실질적으로 소프트웨어의 전체 생명주기 위에서 지속적인 관리가 미흡한 경우가 대부분이다. 이처럼 실효성이 미미한 이유는 명확한 목표 정의, 절차의 문서화, 분명한 역할과 책임 등을 먼저 수립하지 못하기 때문이다.

공개소프트웨어 거버넌스의 수준을 향상하기 위한 모니터링을 통해 지속적인 개선을 하기 위해서는 다음과 같은 단계를 적용할 수 있다.

- **계획** : 공개소프트웨어 거버넌스 활동의 설계 또는 수정
- **실행** : 계획의 구현 및 활동 요소의 관리
- **점검** : 프로세스와 정책 준수의 측정, 목적과의 비교, 보고서 작성
- **조치** : 개선을 위한 변경에 대한 계획 수립 및 구현

광의의 의미에서 볼 때 모니터링은 내부 사용자 또는 고객이 사용하는 공개 소프트웨어의 상태를 주기적으로 점검하고 기록함으로써 향후 안정적인 운영과 유지보수를 보장하고, 공개소프트웨어 커뮤니티의 동향, 진행 현황 및 주요 이슈 등을 주기적으로 추적하고 검토하는 활동을 포함한다. 일반적으로 공개소프트웨어는 개발자 중심으로 개발되었기 때문에 사용자 관점의 다양하고 자세한 요구사항을 수용하지 못하고 있다. 따라서 새롭게 설치되어 운영되고 있는 공개소프트웨어에 대해서 꾸준히 사용자의 의견을 수집할 필요가 있다. 이러한 의견에 따라서 추가적인 개선사항이 도출되고 맞춤형 작업이 진행되어야 한다. 모니터링 활동의 주체로서 내부사용자 즉, 공개소프트웨어를 직접 가져와서 개작하여 외부에 배포할 경우 활용된 공개소프트웨어가 조직의 제품 및 솔루션을 구성하는 주요 기술인만큼 제품 및 솔루션의 성능향상, 안정성, 지속적인 개선을 위해 해당 커뮤니티에 대한 면밀한 모니터링 활동이 필요하며, SI 서비스 산업의 경우 고객에게 공급한 시스템에 특정 공개소프트웨어가 사용되었을 경우에는 유지보수, 기술지원의 관점에서 해당 공개소프트웨어에 대한 지속적인 모니터링 활동이 필요하게 된다.



03장

적용가이드

1. 적용가이드 개요
2. 내부사용 관점
3. 외부배포 관점
4. 외부서비스 관점



1. 적용가이드 개요

본 가이드에서는 공개소프트웨어를 사용하는 다양한 관점에서 용이하게 활용될 수 있도록 사용자 관점을 내부사용, 외부배포, 외부서비스로 구분하였으며 각 관점은 다음과 같다.

- 내부사용 – 공개소프트웨어를 기업내부에서 연구, 참조, 재사용 하는 경우
- 외부배포 – 공개소프트웨어를 직접 판매, 배포하는 경우
- 외부서비스 – 공개소프트웨어를 활용하여 기업의 서비스를 제공하는 경우

2장에서 공개소프트웨어를 사용하는 다양한 사용자 관점에서 공개소프트웨어 거버넌스를 효과적으로 구축할 수 있는 프레임워크(Framework)를 제시하였는데, 이것은 완성품이 아니라 중간 산출물이다. 즉, 뼈대만 있고 살이 없기 때문에 사용자 입장에서는 본인이 소속된 조직의 특이한 성격과 현재 상황에 맞도록 구체화시키는 작업이 추가적으로 필요하다. 이렇게 함으로써 궁극적으로 공개소프트웨어의 이점을 최대한 활용하고 공개소프트웨어 컴플라이언스 이슈와 같이 잠재적인 위험을 최소화시킬 수 있는 거버넌스 체제가 완성될 수 있게 된다. 그러나 형태만 있는 경우 내용이 없으면 처음 거버넌스라는 개념을 접근하는 사용자에게는 이해하고 적용하기가 매우 어려울 수 있다. 그러므로 본 장에서는 적용 가이드를 통하여 구체적인 설명과 함께 양식, 패턴, 사례 등을 최대한 제시하려고 한다.

한편 이 거버넌스 프레임워크는 공개소프트웨어의 사용 방식에 따라 상이할 수밖에 없다. 이러한 상이점을 감안하여 공개소프트웨어의 위치에 따라 내부사용, 외부배포, 외부서비스와 같이 중복되지 않는 3가지 관점으로 구분하고 각각의 활용 케이스를 7가지로 분류하여 각 특성에 맞는 적용가이드를 제시하였다.

이를 위해서는 먼저 공개소프트웨어가 어떤 경로로 유입되고 순환되어 결국 외부로 배포되는지 전반적인 이동 경로를 파악하는 작업을 이해하여야 한다. 즉, 공개소프트웨어 활용 라이프 사이클을 파악하고 관련된 활동을 이해하여야 한다. 비록 이 라이프 사이클은 바라보는 관점에 따라서 다양한 방법으로 기술할 수 있으나 공인된 표준이나 최선의 해답이 없다. 다만 효용성 차원에서는 개발 생산성 향상, 개발자 역량 강화, TCO(Total Cost Ownership) 절감 등을 최대한 성취할 수 있도록 거버넌스 체제를 설계하고 적용하는 것이 최선이다. 이

렇게 함으로써 공개소프트웨어 거버넌스의 본래 목적인 공개소프트웨어 혜택의 극대화과 컴플라이언스 위험 없이 안전하게 사용할 수 있는 토대를 구축할 수 있다.

공개소프트웨어 활용 라이프 사이클은 거버넌스 구축에 있어서 특히 중요하다. 본 가이드에서는 공개소프트웨어 활용 라이프 사이클을 다음과 같이 정의하고 각 단계별 활동요소를 제시하였다.

① 정책수립의 단계

해당 조직의 공개소프트웨어 적용 순응도 수준을 측정하여 그에 따른 정책·전략 및 운영 계획을 수립한다.

② 획득 단계

공개소프트웨어를 조사하고 분석하여 적용을 위한 타당성을 평가한다.

③ 적용 단계

공개소프트웨어를 조직에 적용하기 위한 활동요소 및 관리내용을 제시한다.

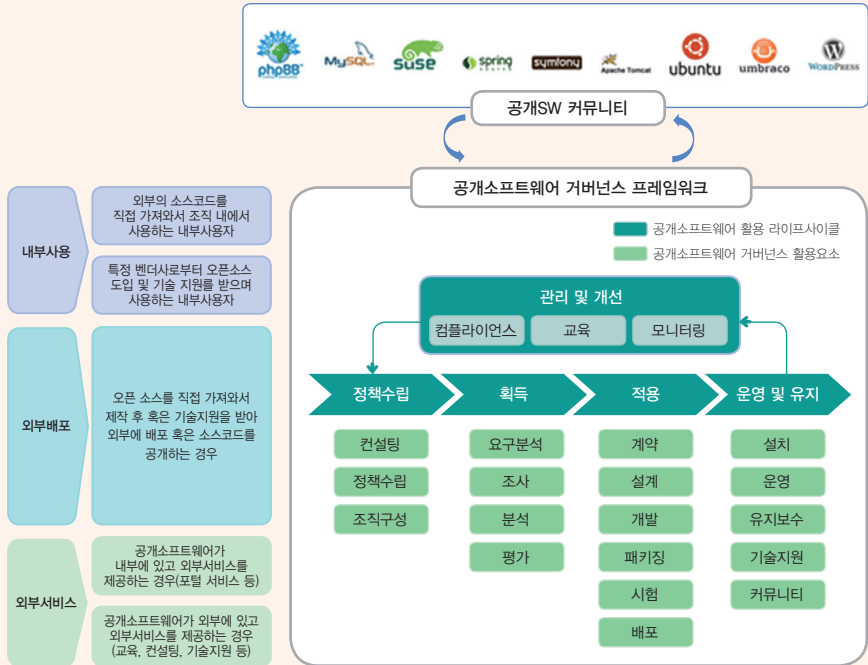
④ 운영 및 유지 단계

공개소프트웨어의 설치 및 운영을 위한 유지보수를 포함하고 있으며, 특히 공개소프트웨어 개발의 특징이라고 할 수 있는 커뮤니티 관련 사항이 포함되어 있다.

⑤ 관리 및 개선 단계

공개소프트웨어 활용 라이프 사이클 전반에 걸쳐 활동결과를 평가하고 성과를 측정하여 조직의 공개소프트웨어 거버넌스를 지속적으로 개선할 수 있도록 제시한다.

이상과 같은 내용들은 [그림 29]에서 종합적으로 표현하고 있다.

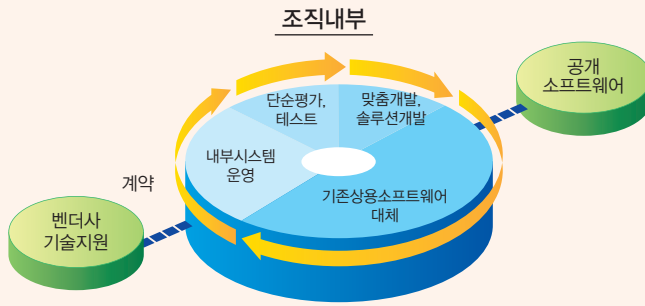


[그림 29] 공개소프트웨어 거버넌스 체계

2. 내부사용 관점

내부사용이란 공개소프트웨어를 외부에 배포하지 않고 단순 평가, 테스트, 패키징, 맞춤 개발, 기존 상용 소프트웨어 대체, 자체 솔루션 개발, 내부 시스템 운영 등의 다양한 방법으로 사용주체의 내부에서만 활용하는 경우이다. 내부사용의 경우에는 외부의 소스코드를 직접 가져와서 조직 내에서 사용하는 관점과 특정 벤더사로부터 공개소프트웨어 도입 및 기술지원을 받으며 사용하는 관점으로 나누어진다.

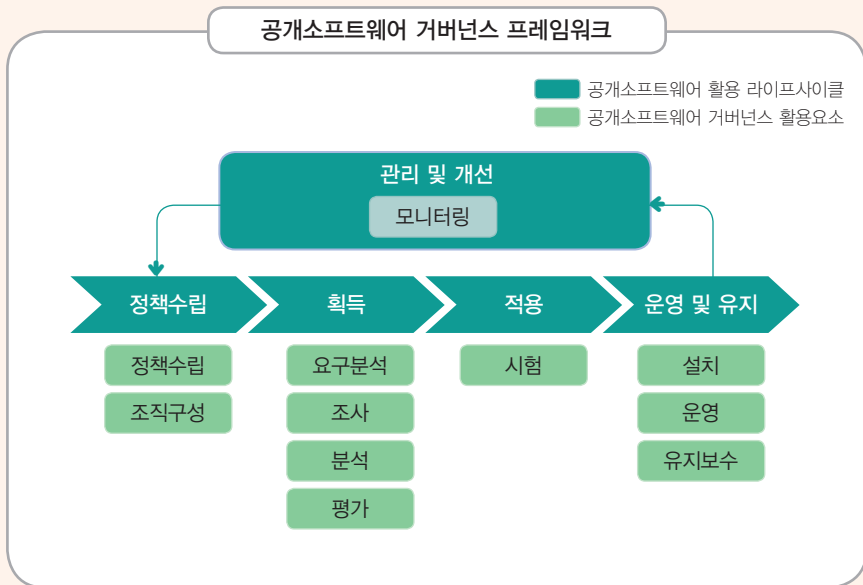
첫 번째 관점에서는 공개소프트웨어 커뮤니티로부터 소스코드를 직접 가져와 외부의 지원이나 간섭이 없이 자체적으로 설치하고 운영한다. 반면에 두 번째 관점에서는 자체적인 기술과 역량이 부족하여 외부 전문 업체의 도움이 필요한 경우이며, 쌍방간의 계약 조건에 따라 차별화된 공개소프트웨어 관련 서비스가 제공된다. 각각의 관점에 해당되는 실행 항목을 살펴보면 다음과 같다.



[그림 30] 내부사용 관점

(1) 외부의 공개소프트웨어를 직접 가져와서 조직 내에서 사용하는 경우

외부의 공개소프트웨어를 직접 획득하여 조직 내에서 사용하기 위해서 필요한 공개소프트웨어의 라이프 사이클 단계별 거버넌스 활동 요소를 정의하면 다음 그림과 같이 전개될 수 있다.



[그림 31] 내부사용 관점 Case 1

1) 정책수립

공개소프트웨어를 내부적으로 사용하는 경우에는 저작권 침해 및 컴플라이언스 위험이 비교적 미미하다. 따라서 여기에서의 정책 수립은 공개소프트웨어의 활용 가치를 극대화시키는 방향으로 초점이 맞추어져야 한다. 이를 위해서는 학습을 통해 개발인력의 역량을 제고하고 재사용을 통해 품질과 생산성을 향상시키며 원가와 비용을 절약시켜야 한다. 즉, 시스템 개선 및 고도화 단계에서 공개소프트웨어를 우선적으로 고려하며 무엇보다도 다양한 방법으로 자유롭게 사용할 수 있는 풍토가 조성되도록 프로세스, 협업 체계, 조직 구성 등 기반 조성과 관련한 개발 환경을 정비해 나가야 한다.

2) 조직구성

공개소프트웨어를 내부적으로 사용하고 배포가 없는 경우에는 공개소프트웨어를 최대한 활용할 수 있는 조직 체제를 구축하여야 한다. 즉, 활성화 측면에서 연구개발, 신규 사업 발굴, 품질관리 등 활성화에 관련된 조직 간에 협력과 소통이 원활하도록 여건을 만들어 주어야 한다.

3) 요구분석

여기서의 요구분석은 내부 사용자 및 개발자의 고민과 요구를 기반으로 기능, 성능, 연동, 호환, 보안 등의 관점에서 필요한 사항을 분류하고 해석하는 활동이다. 공개소프트웨어는 상용 제품과는 달리 일반 사용자의 요구 사항이 제대로 반영되지 않는다. 도리어 공개소프트웨어 커뮤니티 내에서 유능한 개발자 중심으로 요구사항이 수집되고 반영되어 있다. 따라서 사용 주체의 요구사항은 공개소프트웨어 커뮤니티와 상이할 수도 있게 된다. 만일 사용 주체가 자체적으로 사용하고자 하는 공개소프트웨어 커뮤니티의 요구사항을 반영할 만큼 막강한 영향력을 행사할 수 있는 수준이 아니라면 현재 버전에서 제공되는 공개소프트웨어의 기능과 사양에 맞추어 사용하여야 한다. 이러한 경우에는 요구사항을 반영할 수 없는 입장이므로 요구사항을 수집해서 분석할 필요도 없게 된다. 결론적으로 내부에 공개소프트웨어를 자체적으로 설치하여 사용하는 상황이고 해당 커뮤니티를 선도하는 리더가 아니라면 되도록 현재의 공개소프트웨어 버전을 그대로 수용하여야 한다.

4) 조사

여기서의 조사는 사용 주체가 자체적으로 공개소프트웨어의 속성과 커뮤니티 정보를 취합하는 활동이다. 이 활동을 통해서 내부적으로 적용할 수 있는 공개소프트웨어에는 어떤 종류가 있는지 먼저 조사하여야 한다. 또한 이러한 조사 활동을 체계적으로 수행하기 위해서는 공신력 있는 기관이 작성한 분류 체계를 참조하여야 한다. 또한 공개소프트웨어의 사용 목적이 상용 제품의 대체 또는 연동에 있다면 상용 제품과의 비교를 통해 공개소프트웨어 적용성을 평가하여야 한다. 만일 공개소프트웨어의 사용 목적이 자체 솔루션의 개발과 출시

에 있다면 라이선스 차원에서 컴플라이언스 이슈가 없는지도 먼저 알아 보아야 한다.

5) 분석

여기서의 분석은 공개소프트웨어의 성숙도와 적용성을 평가하는데 필요한 속성과 척도를 자체적으로 정의하고 주관적인 방법으로 평가 항목을 정량화시켜 기능성, 이식성, 신뢰성, 사용성, 유지보수성, 커뮤니티 존속성 등 평가에 필요한 데이터를 취합하고 정리하는 활동이다. 공개소프트웨어의 성숙도와 적용성을 평가하는데 필요한 속성을 정의하고 이 속성을 정량화시키는 과정에서 반드시 자체적인 척도와 기준을 먼저 설정하는 것이 중요하다. 이 과정에서 공개소프트웨어 자체적인 특성을 파악하려고 한다면 기본속성을 분석하는 것만으로도 충분하지만 사용 제품과의 비교가 필요하다면 확장속성까지도 고려하여야 한다. 분석은 일반적으로 평가를 전제로 하기 때문에 객관적이고 공정한 지표가 필요하다. 또한 지표들은 두 정량적으로 산출되어야 한다. 그렇지 않으면 평가 단계에서 비교하고자 하는 공개소프트웨어 간의 순의 또는 수준을 알 수 없기 때문이다. 그러나 이 과정에서는 정량적인 지표를 주관적으로 선정하고 정의하여야 한다. 이렇게 함으로써 사용을 목적으로 하고 있는 공개소프트웨어에 대해 고유한 지표를 확보할 수 있게 된다. 따라서 해당 공개소프트웨어에 대해서 외부에서 발표한 결과와 상이한 결과를 얻게 되는 것은 당연하며 이에 대한 자체적인 확신을 가지고 있어야 한다.

6) 평가

여기서의 평가는 공개소프트웨어의 내부 사용 목적에 따라서 성숙도와 적용성을 수치로 산출하는 활동이다. 평가 결과를 수자로 산출하기 위해서 분석 단계의 모든 속성을 주관적인 지표로 정량화시켜야 한다. 따라서 모든 사용 주체에게 공정한 지표가 아니며 자체적인 용도로만 사용할 지표가 설정되는 것이다. 이는 곧 평가를 수행하는 사용 주체가 독자적인 기준과 판단으로 평가 체계를 수립해야 한다는 것을 의미한다. 이렇게 얻은 최종 평가 점수는 동일한 분류에 속하는 공개소프트웨어에 한정해서 사용하여야 한다. 또한 평가 대상과는 다른 분류에 속한 상용 비공개 제품이나 공개소프트웨어와는 무관하게 취급하여야 한다.

7) 시험

여기서의 시험은 개발된 공개소프트웨어에 모든 요구 사항이 반영되었는지의 여부와 설계 단계에서 정의한 사양이 오류나 장애가 없이 정상적으로 동작하는지 기능과 성능 그리고 품질 차원에서 확인하고 원인을 규명하는 활동이다. 공개소프트웨어를 시험하는 항목과 방법은 상용 제품과 크게 차이가 나지 않는다. 다만 시험하는 대상이 자체적으로 개작한 부분과 인터페이스 변경사항에 한정될 뿐이다. 비록 시험 대상이 외부로 배포되지 않는 상황일 지라도 시험을 수행한 결과 오류가 발견되면 이를 즉각 해당 커뮤니티에 버그로 보고해야 하는 점은 상용 제품의 시험과 상이한 부분이다. 일반적으로 최근에 보고된 버그를 조치하

는 프로그램이 패치의 형태로 계속 배포되기 때문에 해당 커뮤니티의 버전 갱신 여부를 수시로 확인하여 항상 최신 버전으로 동기화시켜야 한다.

8) 설치

여기서의 설치는 일반적인 소프트웨어의 설치와 다른 점이 없다. 설치 과정에서 실행에 필요한 오브젝트 파일, 헤더 파일, 라이브러리 파일, 설정 파일 등을 컴퓨터에 복제하고 운영에 필요한 파라미터 값을 지정하게 된다. 더구나 공개소프트웨어를 외부로 배포하지 않고 내부 용도로만 사용하고 있다면 공개소프트웨어를 지나치게 상업적으로 이용하려는 이기적인 업체와 같이 공개소프트웨어가 설치되는 하드웨어의 기종에 따라서 실행이 안 되도록 제한하는 Tivolisim에 대한 우려도 없다. 그러나 향후 외부 배포에 대한 가능성을 배제하지 않는다면 모든 사용자가 특정 제품에 종속되지 않도록 공개소프트웨어의 철학과 취지에 맞도록 모든 환경에서 설치될 수 있는 환경을 구축하여야 한다.

9) 운영

여기서의 운영은 일반 소프트웨어의 운영과 동일하다. 즉, 공개소프트웨어가 설치된 이후 구동, 정지, 로깅, 모니터링 등 일련의 컴퓨터 조작을 통하여 목적인 기능 또는 서비스를 지속적으로 가동시키게 된다. 공개소프트웨어를 외부로 배포하지 않고 자체적으로 사용하는 상황에서는 운영을 책임지는 내부 인력을 배치하여 운영에 대한 모든 책임을 가져야 한다. 따라서 시스템 장애나 외부 공격과 같은 문제가 발생할지라도 충분히 대처할 수 있는 기술력을 먼저 확보하고 운영하여야 한다. 그러나 모든 시스템적인 문제를 사전에 예측할 수도 없고 처음 접해 본 문제를 능숙하게 해결할 수 없기 때문에 비교적 운영이 쉬운 시스템부터 단계적으로 대상 시스템의 수준을 높여 가는 전략이 필요하다. 여기서 쉬운 시스템이라 오랜 시간동안 많은 사이트에서 비교적 안정적인 평가를 받고 있으며 문제 해결 속도가 빨라서 버그가 보고되면 즉각적으로 패치가 나오는 공개소프트웨어를 말한다. 또한 데이터의 처리량, 동시 사용자, 온라인 트랜잭션의 수 등의 성능에 관련된 파라미터가 짧은 시간에 급격히 증가하는 시스템은 피하여야 한다.

10) 유지보수

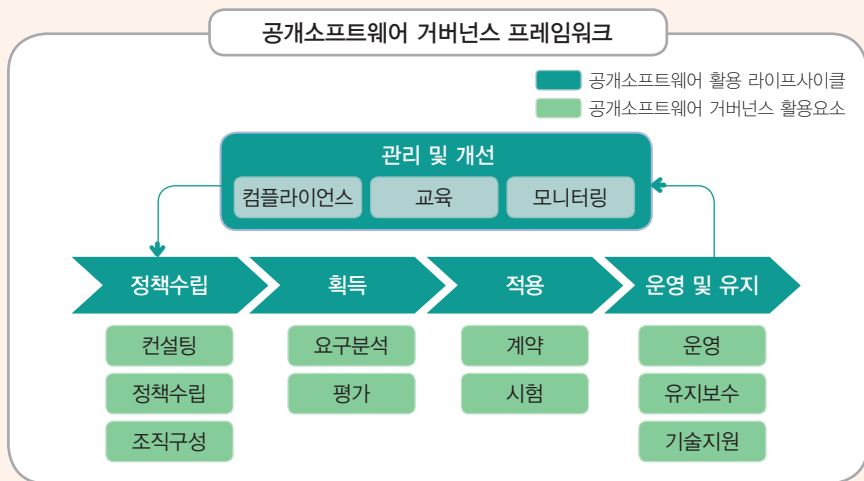
여기서의 유지보수는 활동 내용 측면에서 상용 소프트웨어와 상황과 크게 다르지 않다. 다만 이 경우에는 외부의 기술 지원이 없이 공개소프트웨어를 내부적으로 사용해야 하므로 자체적인 해결 활동이 반드시 포함되어야 하며 자체적인 개발 능력도 필요하다. 유지보수 단계에서 아무리 작은 규모로 공개소프트웨어를 자체적으로 개발하게 될지라도 이는 공개소프트웨어의 개작에 해당되기 때문에 커뮤니티 버전과는 다른 새로운 버전을 파생시키게 된다. 그러므로 개작을 시작하기 이전에 커뮤니티 버전과 일치시킬지 독립적으로 계속 나갈지 먼저 결정해야 한다. 공개소프트웨어의 특성상 사용자의 편의성과 특성에 맞게 제작되

지 않고 개발자 중심으로 사양이 설계되는 경우가 많기 때문에 장기간 사용하다 보면 사용자의 불편사항 또는 추가 요구사항이 발생하기 마련이며 이들을 수용하기 위해서는 맞춤형 개발이 불가피하기 때문에 자체 개작에 대한 요구가 생기게 되기 마련이다. 그러나 운영하고 있는 공개소프트웨어를 가능한 새로운 버전에 추가 요구사항이 반영되기 전까지 커뮤니티 버전과 동일하게 유지하여야 하며 맞춤형 개발을 지양하여야 한다. 결국 가장 자연스러운 방법은 공개소프트웨어를 잘 활용하는 선진 ICT 기업과 같이 프로젝트에 적극 기여하고 리드하면서 추가적으로 발생하는 요구사항을 차기 버전에 반영시키면서 자체적인 필요를 충족시키는 전략이라고 볼 수 있다.

11) 모니터링

여기서의 모니터링은 자체적으로 설치한 공개소프트웨어가 정상적으로 동작하는지 그 자체의 운영 상태를 지속적으로 점검하는 활동이다. 또한 동일한 공개소프트웨어를 설치해서 운영하고 있는 사이트가 외부에 있다면 다양한 조사와 연락 채널을 통해 사용 사례에 대한 정보를 취합하여야 한다. 일반적으로 공개소프트웨어를 내부용도로 자체 운영하고 있는 상황에서는 엄격한 조건이 적용되지 않는 경우가 많기 때문에 비교적 어느 정도 모니터링의 수준이 완화되어 있는 경우가 많다.

(2) 특정 벤더사를 통해 공개소프트웨어 도입 및 기술지원을 받으며 사용하는 경우



[그림 32] 내부사용 관점 Case 2

내부적으로 공개소프트웨어를 가지고 있으나 공개소프트웨어의 설치와 운영을 외부의 기술 지원 서비스 업체를 통해서 진행하는 경우이다. 이 단계에서 공개소프트웨어의 거버넌스 활동 요소를 라이프 사이클 단계별로 정의하면 다음 그림과 같이 전개할 수 있다.

1) 컨설팅

컨설팅은 서비스 제공자가 공개소프트웨어를 원활히 도입, 활용, 관리할 수 있도록 조사, 진단, 분석, 개선 방안, 전략, 로드맵, 방법론 등을 고객에게 서비스로 제공하는 활동으로 자문 서비스와 산출물 제출이 포함된다.

내부사용 관점에서는 외부의 도움으로 내부의 문제를 발견하고 해결책을 찾아야 하므로 최대한 정확하게 문제를 진단할 수 있도록 자료를 공개하고 지원을 아끼지 않아야 한다. 이를 위해서는 보안 사항에 대한 접근도 불가피하므로 비밀유지계약(NDA, Non-Disclosure Agreement)을 체결하여 상호 협력에 장애요소가 없도록 사전에 준비해야 한다.

2) 정책수립

공개소프트웨어의 설치와 운영을 외부 기술지원 서비스 업체에게 맡긴 상황에서는 보안 이슈 등 특별한 사항을 제외하고는 새로운 정책을 수립할 필요가 없다. 즉 외부 업체가 사용 주체의 데이터를 접근해서 처리할 수 있는 환경이므로 내부 기밀 정보가 유출되거나 변경되지 않도록 보안 정책을 수립하여야 한다.

3) 조직구성

공개소프트웨어를 내부적으로 사용하고 배포가 없는 경우에는 공개소프트웨어를 최대한 활용할 수 있는 조직 체제를 구축하여야 한다. 즉, 활성화 측면에서 연구개발, 신규 사업 발굴, 품질관리 등 활성화에 관련된 조직 간에 협력과 소통이 원활하도록 여건을 만들어 주어야 한다.

4) 요구분석

여기서의 요구분석은 외부 기술지원 서비스 업체가 사용 주체의 내부 사용자 및 개발자의 고민과 요구를 기반으로 기능, 성능, 연동, 호환, 보안 등의 관점에서 필요한 사항을 분류하고 해석하는 활동이다. 따라서 사용 주체는 목적에 맞는 최적의 공개소프트웨어를 선택하기 위해서 최대한 정확하고 자세한 요구 사항을 외부 업체에게 제공하여야 한다.

5) 평가

여기서의 평가는 공개소프트웨어의 내부 사용 목적에 따라서 성숙도와 적용성을 수치로 산출하는 활동으로써 자체적으로 수행할 수도 있고 외부 기술지원 서비스 업체에게 일임할

수도 있다. 특히 기술지원 서비스를 비교하고 최적의 업체를 선정하기 위해서 평가의 대상을 공개소프트웨어가 아닌 외부 업체로 대체할 수 있다.

6) 계약

여기에서의 계약은 사용 주체와 외부 기술지원 서비스 업체 간에 공급되는 공개소프트웨어의 사용 조건과 의무 사항에 대한 이행 및 책임을 약정하는 활동이다. 이 계약은 공개소프트웨어의 저작권자와 사용자 사이에 체결되며 약정과는 상이하다. 즉 모든 공개소프트웨어는 보증 부인(no warranty) 조건으로 무료 배포되지만 외부 업체는 보증 조건으로 유료 서비스를 제공하기 때문이다. 따라서 사용 주체는 계약서 상에 독소 조항이 없는지 미리 확인하고 만일의 피해에 대처하여야 한다. 특히 공개소프트웨어는 상용 제품에 비해서 가격이 훨씬 저렴하기 때문에 보상과 배상의 한도와 SLA(Service Level Agreement)를 명확히 정해야 한다.

한편 컴플라이언스 부분에 있어서는 외부 배포가 없는 상황이므로 라이선스 위반에 대한 위험은 극히 미미하나 AGPL에 대한 이슈가 있는지 외부 업체를 통해서 확인하여야 한다.

7) 시험

여기서의 시험은 외부 기술지원 서비스 업체가 개발된 공개소프트웨어에 모든 요구 사항이 반영되었는지의 여부와 설계 단계에서 정의한 사양이 오류나 장애가 없이 정상적으로 동작하는지 기능과 성능 그리고 품질 차원에서 확인하고 원인을 규명하는 활동이다. 일반적으로 설치된 공개소프트웨어의 시험은 외부 업체가 책임지고 진행하기 때문에 사용 주체는 검수 단계에서 정상적인 운영 상태만 확인하게 된다.

8) 운영

여기서의 운영은 외부 기술지원 서비스 업체가 일반 소프트웨어의 운영과 동일한 방법으로 책임을 지고 수행하게 된다. 따라서 사용 주체는 운영 결과에 대해서만 외부 업체로부터 보고를 받으면 된다.

9) 유지보수

여기서의 유지보수는 외부 기술지원 서비스 업체가 설치된 공개소프트웨어의 최적 최상의 운영에 책임을 지고 수행하는 활동이다. 또한 내용 측면에서 상용 소프트웨어와 크게 다르지 않다. 따라서 사용 주체는 유지보수 활동에 대해서는 외부 업체에게 일임해야 한다. 그러나 공개소프트웨어를 운영하고 있는 도중에 불편 사항, 개선 사항, 오류 사항이 발생할 수 있다. 이 경우에는 사용 주체가 지나치게 요구사항 반영을 요구하게 되면 외부 업체의 맞춤형 개발을 유도하게 되어 커뮤니티 버전과는 다른 별도의 버전을 만들게 된다. 따라서 차기 버전으로 업데이트 되기 전까지는 최대한 커뮤니티 버전과 동일하게 유지될 수 있도록 요

구사항 관철을 자제하여야 하며 파라미터 변경 수준으로 유지하여야 한다.

10) 기술지원

이 경우 기술지원을 외부에서 받기 위해서 사용하는 공개소프트웨어에 대한 성능 개선, 장애 해결, 오류 수정, 추가 요구 사항 반영 등 공개 소프트웨어 기술 기반의 서비스를 제공하는지 확인하고, 가입자 계약에 의해서 제공되는 운영 및 유지 보수 서비스뿐만 아니라 요청 기반의 단기 서비스도 포함하여 계약을 체결해야 한다.

또한 개발자가 모두 인접해있는 비공개소프트웨어와는 다르게 공개소프트웨어 프로젝트는 주로 원격에서 이루어지는 특성을 가지고 있기 때문에 커뮤니케이션 방식이 다르고 이메일, 전화, 메신저등은 공개소프트웨어 프로젝트에서 공동작업을 하기에 적합하지 않기 때문에 버그나 이슈의 처리를 위한 도구가 필요하다.

11) 컴플라이언스

이 경우에는 공개소프트웨어를 외부로 배포하지도 않고 내부적인 용도로만 사용하기 때문에 카피레프트에 따른 컴플라이언스 이슈가 없다. 혹시라도 외부 업체가 설치한 공개소프트웨어로 인해 컴플라이언스 문제가 발생할 경우를 대비해서 계약 조건으로 보상, 배상, 사후 처리에 대한 이행 사항을 명시하여야 한다.

12) 교육

여기에서는 공개소프트웨어를 내부적인 용도로 도입하고 자체적으로 개발 및 운영하는 상황이므로 전문적인 지식, 기술, 경험을 보급 확산시키는 교육이 매우 중요하다. 이를 위해서 내부 인력을 교육 프로그램 개발자 또는 강사로 활용시키는 방법은 원가 절감 차원에서 효과적이다. 비록 설치, 개발, 운영, 유지 차원에서는 외부의 지원을 받지 않고 있더라도 내부 인력의 기술 전문성과 기술 역량 제고를 위해서는 외부의 교육 전문 업체를 사용하는 것이 바람직하다. 또한 공개소프트웨어가 외부에 배포되지 않는 상황이기 때문에 컴플라이언스의 위험을 예방하는 교육은 최소화시켜야 한다.

13) 모니터링

여기서의 모니터링은 자체적으로 설치한 공개소프트웨어가 정상적으로 동작하는지 그 자체의 운영 상태를 지속적으로 점검하는 활동이다. 또한 동일한 공개소프트웨어를 설치해서 운영하고 있는 사이트가 외부에 있다면 다양한 조사와 연락 채널을 통해 사용 사례에 대한 정보를 취합하여야 한다. 일반적으로 공개소프트웨어를 내부용도로 자체 운영하고 있는 상황에서는 엄격한 조건이 적용되지 않는 경우가 많기 때문에 비교적 어느 정도 모니터링의 수준이 완화되어 있는 경우가 많다.

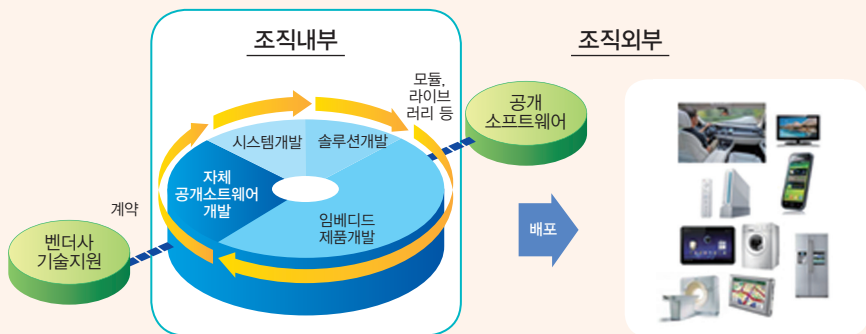
3. 외부배포 관점

공개소프트웨어를 활용함에 있어서는 내부적으로 조직의 개발자들이 외부에 있는 커뮤니티로부터 공개소프트웨어를 직접 가져와서 복제 및 개작 후 외부에 배포하는 경우, 특정 기술지원 서비스 업체로부터 기술지원을 받아 외부에 공개소프트웨어를 배포하는 경우, 내부에서 직접 개발한 소프트웨어를 공개소프트웨어 형태로 외부에 배포하는 경우로 나누어 질 수 있다.

공개소프트웨어를 복제 및 개작 후 외부에 배포하는 경우 공개소프트웨어의 활용 단위별로 크게 완제품수준의 공개소프트웨어를 가져와서 복제 혹은 일부 개작 후 외부에 배포하는 경우, 특정 모듈이나 라이브러리 소스코드를 가져와서 복제 혹은 개작 후 외부에 배포하는 경우, 파일단위에서 소스코드를 복제 혹은 개작 후 외부에 배포하는 경우로 세분화시킬 수 있다.

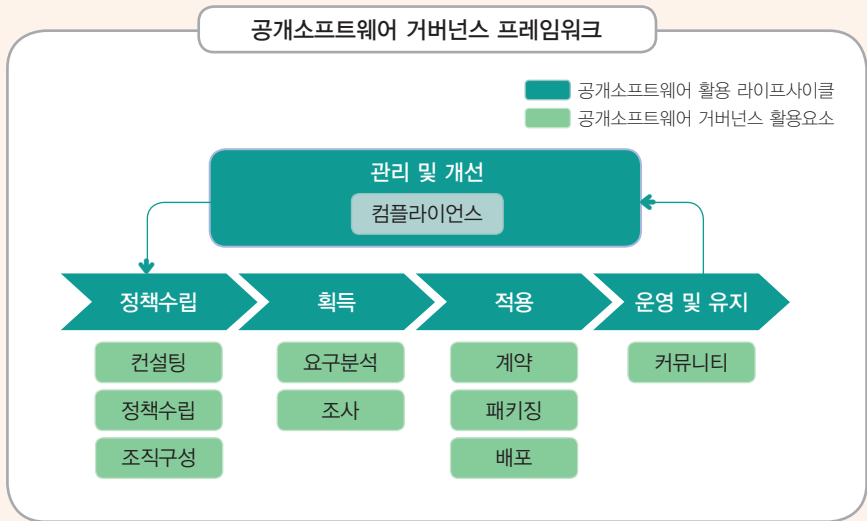
그러나 공개소프트웨어를 활용함에 있어서는 조직의 특성에 따라 다양한 방법을 적용할 수 있기 때문에, 배포에 있어서도 소스코드 공개 없이 오브젝트 코드만 배포하는 경우와 소스코드의 일부만 공개하는 경우 등 제품, 솔루션, 서비스 특성에 따라 다양한 배포정책을 운영할 수 있다. 따라서 본 가이드라인에서는 공개소프트웨어를 외부에 배포하는 경우를 구분하여 기술함을 전제로 한다.

공개소프트웨어를 활용하여 외부에 배포하는 경우에 필요한 거버넌스 활동은 관리 및 개선 영역으로 컴플라이언스와 교육, 모니터링은 공통으로 필요한 활동요소이며, 정책수립, 획득, 적용, 운영 및 유지의 라이프사이클 별 개별 활동이 필요하다. 사업별로 보면 임베디드 소프트웨어 개발 사업, 패키지 및 개별 솔루션 개발 사업, SI(System Integration) 사업에 적용될 수 있다.

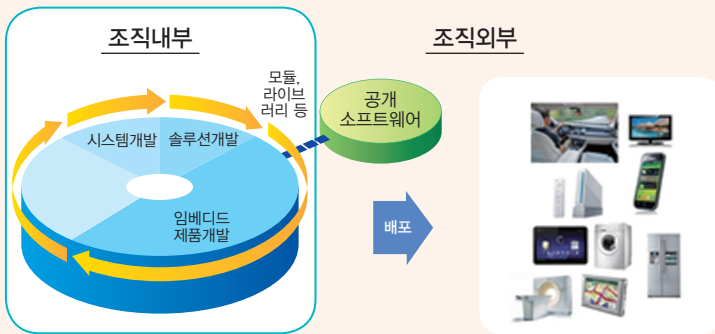


[그림 33] 외부배포 관점

(1) 공개소프트웨어를 직접 가져와서 개작 후 외부에 배포하는 경우



[그림 34] 외부배포 관점 Case 1



[그림 35] 공개소프트웨어를 직접 가져와서 개작 후 외부에 배포하는 경우

1) 컨설팅

공개소프트웨어를 가져다 외부에 배포하는 조직에 있어서는 조직수준 진단, 사업 및 개발 전략 수립, 정책수립, 공개소프트웨어의 조사, 선택, 검토, 승인 등에 대한 프로세스 구축 등이 필요하게 된다. 조직 내부에 전문 검토위원회 및 관리부서가 존재할 경우 자체적으로 이

와 같은 업무 수행이 가능할 수 있지만, 그렇지 못할 경우에는 외부 전문 컨설팅 기관의 도움이 필요할 수 있다. 다만, 규모가 크지 않은 조직이거나 개발제품 및 솔루션이 제한적인 경우 이러한 컨설팅 요구사항이 모두 필요한 것은 아니다. 예를 들어, 단일 임베디드 소프트웨어 제품을 개발, 판매하는 경우 해당 제품에 활용되는 특정 공개소프트웨어의 내역확인 관리만으로 충분할 수도 있다. 그렇지만 여러 개발부서를 통해 다양한 제품을 개발하는 임베디드 소프트웨어 조직이나 대형 혹은 다양한 솔루션을 개발하는 조직, SI 조직의 경우 컨설팅 요구사항에 포함된 모든 업무 활동이 필요하게 되며 컨설팅 요구사항은 조직의 특성을 반영하여 도출하여야 하지만 일반적으로 필요시 되는 컨설팅 요구사항은 다음과 같다.

- (가) 제품 개발 및 개발 경로, 사업모델 등의 파악을 통해 공개소프트웨어 의존성 및 향후 활용정도를 파악하기 위한 조직수준 평가 및 진단
- (나) 향후 제품 및 기술개발 로드맵을 설계함에 있어서 공개소프트웨어 기술 활용 및 개발 전략 구축을 위한 공개소프트웨어 기술센싱 및 기술전략 구축 컨설팅
- (다) 공개소프트웨어 조사, 선택, 검토, 승인 등의 관리를 위한 거버넌스 구축 컨설팅
- (라) 공개소프트웨어 사용현황 파악 및 자산관리를 위한 라이선스 관리 및 검증 컨설팅

배포 대상이 커뮤니티일 경우에는 컨설팅의 내용이 달라진다. 커뮤니티 마다 문화, 미션, 적용 분야, 조직의 운영 체제, 성숙도, 영속성 등 다양한 속성이 있다. 컨설팅은 이러한 속성에 따라서 어떤 커뮤니티가 사용 주체에게 적합한지 가이드 또는 의견을 줄 수 있어야 한다. 또한 선정된 커뮤니티를 어떤 방법으로 참여해야 하는지, 참여의 정도는 얼마나 적극적이어야 하는지, 참여 인력의 구성이나 규모는 어떻게 가져가야 하는지, 내부 지적재산권의 보호와 기여는 어떻게 가져 가야하는지 등 적용성 관점에서 컨설팅이 필요하다

2) 정책수립

조직의 개발자들이 조직 외부에 존재하는 공개소프트웨어를 개작해서 외부에 배포하는 경우에는 공개소프트웨어를 효과적으로 활용하여 개발생산성을 극대화시킬 뿐만 아니라 저작권 침해 및 컴플라이언스 위험을 극소화시키는 지침이 정책에 명확하게 반영되어 있어야 한다. 이러한 정책개발 및 운영에 있어서는 조직 내 개발 제품 및 솔루션, 활용되는 제반 기반기술에 대한 현황 및 향후 개발 로드맵을 바탕으로 현재 및 미래에 공개소프트웨어에 대한 활용도 및 의존도를 파악하기 위한 수준평가 및 진단을 통해 정책 내용과 적용범위가 검토되어야 한다. 조직수준 평가 및 진단의 목적은 다양한 조직의 사업전략과 범위, 개발 제품 및 솔루션에 부합된 적절한 정책수립 및 운영에 있다.

배포의 대상이 고객이 아니라 공개소프트웨어 커뮤니티일 경우에는 참여 및 기여 수준에 따른 정책을 수립하여야 한다. 즉 참여 및 기여가 가능한 커뮤니티의 선정과 선정된 커뮤니티를 통한 활동 보장 및 후원에 대한 정책이 필요하다. 또한 단체의 구성원이 커뮤니티에

소스코드를 배포하는 상황이므로 지적자산의 유출과 보안의 관점도 고려가 되어야 한다. 특히 소속 단체의 이익과 정책에 위배되는 행동을 한 경우에 어떻게 조치할 것인지에 대한 방안도 수립되어야 한다.

3) 조직구성

공개소프트웨어를 직접 가져다 사용하는 조직의 경우 공개소프트웨어 활용의 목적과 조직의 사업관점에서 공개소프트웨어의 의존성을 파악하여 조직의 수준에 부합된 조직을 구성, 운영하여야 하며, 조직수준과 사업전략 관점에서 조직은 여러 형태의 공개소프트웨어 관리 조직을 구성, 운영할 수 있다. 현재 모든 산업에서 공개소프트웨어 사용이 급증되고 있고 공개소프트웨어 활용 없이 소프트웨어를 개발하는 것 자체가 대단히 비효율적인 활동임을 감안할 때 공개소프트웨어 사용을 엄격히 금지하는 조직은 현재 개발환경에서 실행하기 어렵다고 볼 수 있고, 공개소프트웨어 사용을 방관하는 조직은 사업투명도의 회손, 사업 리스크 증가에 대한 경영진의 책임, 잠재적 주주 및 고객 소송 등과 같이 관리되지 않은 공개소프트웨어로 인한 개발기간 연장, 지적자산의 회손 등 수 많은 부정적 결과를 초래할 수 있다. 따라서 개발 프로젝트의 수명 주기에 따라 중요한 포인트에 공개소프트웨어 관리와 통제를 수반한 조직활동이 필요하다. 일반적인 활동요소에서도 언급되었듯이 중요한 포인트는 크게 6가지로 분류 될 수 있으며 각 포인트의 주요 활동은 어떤 공개소프트웨어가 어떻게 결합되어 있고, 해당 공개소프트웨어는 내부 정책에 따라 충분히 검토되었는지, 어떠한 컴플라이언스 이슈가 발생되고 준수해야 하는 의무사항은 어떤 것 인지를 확인하는 것이다. 주요 포인트는 프로젝트 개발 라이프사이클에 따라 주기는 다양할 수 있지만 공개소프트웨어가 처음으로 프로젝트에 첨부될 때, 내부 개발된 컴포넌트가 만들어지거나 수정 되었을 때, 모든 빌드 시에, 출시단계에서, 내부 컴포넌트를 공개소프트웨어 커뮤니티에 공헌하는 것을 고려할 때 혹은 다른 서드파티에 소유권을 전환할 때, 소프트웨어 컴포넌트에 있어서 중요한 소유권 부분을 획득하기 전에 반드시 확인해야 한다.

(가) 공개소프트웨어 거버넌스 조직 모델

임베디드 소프트웨어 산업, 혹은 솔루션 개발사업, SI 사업의 경우 해당 조직의 규모에 따라 전사 조직과 프로젝트 조직으로 구분되어 질 수 있다. 일반적으로 공개소프트웨어를 직접 가져다가 복제 혹은 개작 후 배포하는 조직의 경우 다음과 같은 조직구성이 필요하다.

- **공개소프트웨어 검토위원회(Open Source Review Board)** : 대규모 조직의 경우 공개소프트웨어 검토위원회를 구성하여 전사적 의사결정을 할 수 있지만 반드시 필요한 조직은 아니다. 해당 조직의 특성에 따라 전담 관리조직 혹은 전담 전문가 그룹을 통해 운영할 수 있다.
- **법무** : 공개소프트웨어가 다양한 라이선스로 구성되어 있고 결합 형태에 따라 법적 의

무사항이 다양한 만큼 사용된 공개소프트웨어 라이선스에 대한 법적 자문 및 정책반영과 운영은 반드시 필요하다. 다만, 사전에 정의된 라이선스 정책과 결합사용에 따른 기술적 검토를 전사 가이드라인에 반영하고 지속적으로 업데이트 지원한다면 법무 조직이 상시 운영하는 조직구성에 포함될 필요는 없고 필요시에 자문을 지원하는 정도로 구성, 운영될 수 있다. 소규모 조직의 경우에는 공개소프트웨어 역량프라지를 통한 자문요청도 가능하며 외부 법무법인을 활용할 수도 있다. 법무조직의 주요 활동 중에는 공급망 혹은 고객과의 계약 시에 공개소프트웨어 사용으로 인해 발생할 수 있는 여러 책임과 권한 등에 대한 계약검토도 반드시 포함되어야 한다.

- **개발** : 개발 조직은 공개소프트웨어를 결합, 활용하는 주체가 되는 조직으로 반드시 공개소프트웨어 관리 조직으로 구성되어 운영상에 반영하여야 한다. 특히 개발조직은 공개소프트웨어의 개작형태 및 결합형태, 사용 공개소프트웨어의 기술적 기능, 역할 등에 대해 법무 및 관리조직과 유연하게 커뮤니케이션 할 수 있도록 상세내역을 관리, 운영하여야 한다. 또한 이러한 관리활동은 별도의 프로세스로 운영하기보다 기존의 소프트웨어 엔지니어링 프로세스와 통합 운영하는 것이 바람직하다. 소규모 조직의 경우 때로는 과제 기획단계, 혹은 개발단계에서 상기 언급한 중요 포인트마다 사용된 공개소프트웨어 체크리스트 작성으로 활동이 충분할 수 있다.
- **영업 및 마케팅** : 사업모델 혹은 조직 수준에 따라 영업 및 마케팅 부서가 공개소프트웨어 관리 조직으로 활동할 수 있다. 영업은 주로 공급망 계약 혹은 고객과의 계약 시에 공개소프트웨어 사용으로 인해 발생하는 여러 모니터링 문제, 보안문제, 컴플라이언스 문제 등에 대해 사전 검토해야 하며 마케팅의 경우 제품 및 서비스 홍보 등에 있어서 준수 의무사항에 대한 사전검토 및 이행이 필요할 수 있다.

(나) 공개소프트웨어 거버넌스 조직 운영

일반적인 활동요소에서도 언급되었듯이 조직운영에 있어서는 조직의 수준과 규모에 따라 다음과 같이 다양한 형태의 운영이 가능하다.

- **전담부서** : 다양한 제품 및 솔루션, 여러 고객사에 대한 SI 서비스를 제공하는 대규모 조직의 경우 개발조직 내에 광범위하게 다양한 공개소프트웨어를 가져와 복제 혹은 개작 형태로 결합 사용됨에 따라 성과측정, 라이선스 관리 및 제반 공개소프트웨어 기반 기술에 대한 전사적 차원의 모니터링 및 통제가 필요함에 따라 전사적인 공개소프트웨어 관리를 전담하는 조직구성이 필요하다.
- **겸임부서** : 소규모 조직 혹은 대규모 조직이더라도 공개소프트웨어의 의존도가 상당히 떨어지는 경우 혹은 전담부서의 조직구성이 현실적으로 어려운 경우에는 기존 조직에 있어서 가장 공개소프트웨어 관리 업무에 적절한 조직에 해당 업무를 할당하여 관리하는 것이 필요하다. 겸임부서의 경우 기존 개발부서의 기획업무를 담당하고 있는 개발

기획, 품질관리, 품질보증, 지적재산, 해당 개발부서 혹은 지원부서등이 이러한 검임부서로서 권장되고 있다.

- **전담 혹은 검임인력** : 전담부서 혹은 검임부서의 조직구성이 어려울 경우 전담 혹은 검임인력을 선정, 운영함으로써 공개소프트웨어와 관련된 개발, 법무, 해당 커뮤니티 및 기술지원 업체 등과의 의사소통 및 협업을 지원할 수 있다.

공개소프트웨어를 외부에서 가져다 활용하는 경우 모든 조직에서 검토해야 할 조직수준 평가 및 진단의 범위는 다음과 같다.

- ① 서드파티와 공개소프트웨어 발견
- ② 서드파티/공개소프트웨어 컴포넌트 검토와 선택
- ③ 공급망관리/공개소프트웨어 결합
- ④ 공개소프트웨어 코드관리
- ⑤ 공개소프트웨어 유지보수와 지원
- ⑥ 라이선스 준법성 프로그램
- ⑦ 공개소프트웨어 커뮤니티 상호작용
- ⑧ 의사결정자 검토

외부 공개소프트웨어를 가져와 개작하여 배포하는 임베디드 소프트웨어, 솔루션, SI 사업의 경우 공개소프트웨어 정책은 조직 수준 평가 및 진단결과를 바탕으로 사업 및 개발전략에 부합되게 공개소프트웨어 사용, 소스코드 공개, 라이선스 검증, 회사 외부 커뮤니티 활동, 커뮤니티 생성 및 운영, 공개소프트웨어 라이프 사이클 또는 프로세스, 전사 조직의 역할과 책임의 정의, 공개소프트웨어 평가, 공개소프트웨어 교육, 주요 라이선스별 적용 방안, 조직별 공개소프트웨어 활용 방안, 공급업체의 공개소프트웨어 라이선스 의무 수행 등으로 세분화 될 수 있으며 사업별 적용 범위는 다음과 같다.

(다) 임베디드 소프트웨어 제품 및 솔루션을 개발 판매하는 조직의 경우

- 공통활동 요소에서 기술된 공개소프트웨어 사용 시 조직이 준수해야 하는 모든 준법성 요구사항에 대한 정책이 필요하며, 내부 조직 뿐 아니라 공급망에 포함되는 모든 협력업체들에게도 해당 준법성 정책을 적용해야 한다.
- 공통활동 요소에서 기술된 공개소프트웨어 사용 혹은 활동을 위한 승인절차를 위한 정책이 필요하며, 내부 조직 뿐 아니라 공급망에 포함되는 모든 협력업체들에게도 승인정책을 적용해야 한다.
- 공개소프트웨어 관리를 위한 조직자원은 여러 개발부서가 있는 대형 조직에서는 개발 부서간의 의견 조율과 의사결정을 위한 공개소프트웨어 검토위원회가 필요할 수 있지만, 소규모 조직에서는 공개소프트웨어 관리부서 혹은 전문 담당자를 선정하여 공개소

소프트웨어 사용과 승인, 컴플라이언스 이슈를 포함한 제반 업무를 수행할 수 있다.

(라) SI 조직의 경우

- 공통활동 요소에서 기술된 공개소프트웨어 사용 시 조직이 준수해야 하는 모든 준법성 요구사항에 대한 정책이 필요하며, 내부 조직 뿐 아니라 컨소시엄에 참여하는 모든 협력업체들에게도 해당 준법성 정책을 적용해야 한다. 특히 SI의 경우는 고객의 요구사항이 매우 중요한 만큼, 준법성 정책은 개별 프로젝트별로 고객의 요구사항을 파악하여 반영하여야 한다.
- 공통활동 요소에서 기술된 공개소프트웨어 사용 혹은 활동을 위한 승인절차를 위한 정책이 필요하며, 내부 조직 뿐 아니라 컨소시엄에 참여하는 모든 협력업체들에게도 해당 준법성 정책을 적용해야 한다. 계약 및 프로젝트 설계단계에서 고객의 요구사항을 파악하여야 하며 개발진행 중에도 설계단계에서 검토된 공개소프트웨어 이외의 사용에 대해서도 반드시 고객의 승인을 득 해야 한다.
- SI 조직에 있어서 공개소프트웨어 관리를 위한 조직자원은 두 가지 형태로 구분될 수 있다. 연구개발 대상이 되는 기술, 솔루션개발에 있어서는 개발 부서간의 의견 조율과 의사결정을 위한 공개소프트웨어 검토위원회가 필요할 수 있으며 이 또한 소규모 조직에서는 공개소프트웨어 관리부서 혹은 전문 담당자를 선정하여 공개소프트웨어 사용과 승인, 컴플라이언스 이슈를 포함한 제반 업무를 수행할 수 있다. 특정 고객사를 위한 프로젝트의 경우에는 해당 PM이 제반 업무를 수행, 감독하게 되며 전사 공개소프트웨어 검토위원회 및 관리부서 혹은 전문 담당자와의 유기적 협업 및 관리 프로세스 적용이 필요하다.

배포 대상이 커뮤니티인 경우에는 특별한 조직 체계를 구성하여야 한다. 일반적으로 커뮤니티로 배포되는 소스코드나 커뮤니티에 참여와 소통에 필요한 정보 교환은 인터넷을 통한 경우가 많기 때문에 모니터링이나 통제가 쉽지 않다. 그러므로 가급적 모니터링과 통제의 차원에서 커뮤니티, 참여자, 소스코드, 지원 비용 등을 중점적으로 관리하는 조직이 필요하다.

4) 요구분석

공개소프트웨어를 직접 가져다 개작해서 배포할 경우에도 기능 및 성능, 연동, 호환, 보안 등의 다양한 부분에서 요구사항을 수집하고 분석할 수 있다. 다만 요구 사항을 충족시킬 수 있는 공개소프트웨어는 자체적인 요구사항에 따라 개발되기 때문에 발전 방향 및 사양에 대한 지속적인 모니터링이 필요하다. 특히 SI의 경우에는 고객의 요구사항이 다양하고 변경 사항이 많기 때문에 세세한 비교가 필요하다.

공개소프트웨어의 경우 주요 커뮤니티에서 생성, 관리, 발전됨에 따라 해당 커뮤니티의 성숙도, 안정성, 발전 가능성, 주요 커미터의 정책과 구성 조직의 사업전략 등이 공개소프트웨어

어를 활용한 요구분석에 있어서 매우 중요시 될 수 있으며, SI의 경우에는 고객이 원하지 않는 공개소프트웨어를 사용함에 따른 분쟁을 최소화하기 위해 요구분석 단계에서 고개의 소리를 적극적으로 반영하여야 한다. 요구분석 단계에서 공개소프트웨어 사용에 대한 절차와 명세서를 상호합의하에 작성하는 것도 향후 분쟁을 최소화 하는 방법이 될 수 있다.

5) 조사

공개소프트웨어를 복제 혹은 개작하여 외부에 배포할 경우 공급망의 연결 구조와 수신처의 특성을 조사하여야 한다. 상업적인 공급 경로에 따라 중간 배포자 또는 최종 배포자가 될 수 있다. 중간 배포자의 경우에는 수신자가 대부분 단체가 되기 때문에 그 단체가 재배포를 하게 되면 문제가 없는지, 라이선스에 대한 고지의 의무는 있는지, 연동 방식과 모듈 간의 결합방식 등 라이선스 판별에 필요한 정보를 요구하고 있는지 등등 공급과 수급 사이에서의 제약 사항들을 조사하여야 한다. 반면 최종 배포자일 경우에는 배포 받는 자가 공개소프트웨어 커뮤니티이거나 개인이 된다. 커뮤니티가 배포의 대상이 되는 경우에는 재무적인 이해관계가 없기 때문에 제약 사항에 대한 조사 내용이 그리 많지 않다. 그러나 공급 사슬에서 고객에게 최종적으로 배포하는 경우라면 책임과 위험 측면에서 매우 심각한 입장이 된다. 이는 공개소프트웨어의 저작권 침해와 라이선스 위반에 대한 모든 책임은 최종 배포자에게 있기 때문이다. 따라서 공개소프트웨어의 배포 이전에 계약 사항, 판매 조건, 공급 방식 등 차후 문제를 일으킬 소지가 있는 부분을 사전에 파악하여야 한다.

6) 계약

공급 사슬에서 배포 행위가 납품에 해당되는 경우가 있고, 납품이 계약 이행에 중요한 사항이므로 공개소프트웨어를 개작하여 외부로 배포하는 단계에서는 배포에 관련된 계약 사항의 확인과 적법한 이행이 매우 중요하다. 특히, 임베디드 소프트웨어와 솔루션 사업의 경우에는 최종 사용자에게 제품을 판매 및 출시함에 있어서 관련되는 모든 공급망이 계약범위에 포함되며, SI 서비스 사업의 경우에는 계약 범위가 더욱 넓어져서 최종고객 뿐 아니라 최종 납품되는 프로젝트에 참여하는 협력업체까지 포함될 수 있다. 특히, SI 서비스 사업의 경우 외부 공개소프트웨어를 활용하여 고객에게 서비스하는 만큼, 다음 서비스 항목에 대한 내용이 고객과 합의되어 계약에 명시되어야 한다.

임베디드 소프트웨어 및 솔루션 사업의 경우 제품 및 솔루션에 포함되는 모든 공개소프트웨어의 품질, 보안취약점, 라이선스 문제 등은 최종 제품을 출시하는 공급사에 책임이 있는 만큼 조직의 내부 개발자 뿐 아니라 제품개발에 포함되는 외부개발자, 협력업체 등도 사용되는 공개소프트웨어에 대해 책임을 져야 하며 이러한 책임의 범위를 계약에 명시하여야 한다.

〈표 14〉 SI 서비스 사업의 주요 서비스 내용

서비스항목	주요 서비스 내용
설치	목적 소프트웨어를 고객이 원하는 시스템 환경에 옮기고 정상적으로 작동되도록 하는 작업
패치 제공	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
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
문제 해결	목적 소프트웨어 자체의 문제(오동작, 에러, 버그, 해킹) 또는 운용 환경상의 문제(연동, 컨피그레이션) 등을 기술적으로 해결해 주는 작업
모니터링	설치된 소프트웨어의 실시간 운용 상황을 동적으로 관찰하여 통계적 자료를 제공하는 업무
온라인 지원	포탈이나 이메일을 통해 질문이나 지원 요청을 접수하여 지식 베이스 또는 전담 기술인력을 동원하여 접수된 요청사항을 온라인으로 지원해주는 업무
기술 자문	마이그레이션, 커스터마이징, 백업 등 공개소프트웨어 서비스에 관련된 사항을 지도해주는 업무
개발자 지원	아키텍처링, 파라미터 구성, 성능 튜닝, 최적화 등의 개발 업무에 대해 조언하는 업무
보증	공급한 공개소프트웨어 제품의 소스코드에 아무런 법적인 문제가 없도록 지원하는 업무

공개소프트웨어의 특성상 다양한 저작권자 및 라이선스로 인해 향후 도입, 활용에 있어서 예상하지 못한 라이선스 위반으로 인한 저작권 분쟁, 소유권 및 사용권 분쟁 등이 발생할 수 있음에 따라 분쟁 예방차원에서 공개소프트웨어를 활용한 개발 및 발주, 공급을 포함한 일련의 공급망 관리에 대한 구체적인 내용이 계약내용에 포함되어야 하며 관리부재 및 소홀로 인해 발생된 피해에 대해 명확한 책임을 명시하는 것이 향후 피해를 최소화 하고 문제를 해결함에 있어서 필수조건이 될 수 있다. 특히, 공개소프트웨어 라이선스 자체에는 아무런 보증도 약속하지 않고 있음에 따라 이를 사용하여 고객에게 제품을 공급 혹은 서비스하는 조직은 계약을 꺼려하는 고객을 위해서 계약서 안에 면책 조항을 추가하거나 추가의 보증약속을 추가할 수 있다. 또한 근래에는 공개소프트웨어의 라이선스 위험뿐만 아니라 제3자 특허 부분에서도 소송 및 이슈가 발생되고 있는 상황이므로 특허 침해에 대한 내용도 계약서에 반영해야 할 필요가 있다.

임베디드 소프트웨어 혹은 솔루션 사업의 경우 계약단계에서 제품 개발에 참여하는 협력업체와의 계약 시에 사용된 공개소프트웨어에 대한 내역서 혹은 체크리스트를 요청하는 것이 명시되어야 하고 그 예는 다음과 같은 내용을 반영하여야 한다.

〈표 15〉 공개소프트웨어 구성 내역서의 예

공개소프트웨어 컴포넌트	결합형태	라이선스	기 능
GnuPG	라이브러리	GPL 3.0	데이터 암호화
배포정책	공개, 일부공개(공개 라이선스), 비공개		

SI 서비스 사업의 경우에는 완제품 수준의 공개소프트웨어 사용이 가능함에 따라 보다 구체화된 계약검토가 필요하며 다음과 같은 내용을 계약에 반영하여야 한다.

먼저 SI 프로젝트의 경우 프로젝트 발주, 수행, 검증관리 지침이 세부항목으로 검토 될 수 있다. 배포를 받는 입장에서 공개소프트웨어 구성 내역서 보다 더 구체적으로 소스코드 수준에서 공개소프트웨어 사용 내역을 요청할 수 있다. 이럴 경우에는 다음 양식을 참조하여 소스코드 사용명세서를 제출하며 된다.

(가) 계약조건 정의

발주자는 계약기간 및 사업종료 후에 수행자의 부적절한 공개소프트웨어 사용을 예방하고 올바른 지적재산권 행사를 위해 부적절한 공개소프트웨어 사용으로 인한 수행자의 법적 책임사항과 범위를 명확히 해야 한다.

(나) 수행자 선정

발주자는 수행자의 제안서, 능력, 공개소프트웨어 준법개발 여부, 그리고 과제의 특성에 따라 고려될 필요가 있는 요소들에 대한 평가에 기초하여 수행자를 선정한다.

(다) 계약 협상 및 체결

발주자는 수행자 및 수행자의 하도급자 등을 포함하여 발주할 모든 산출물과 관련 있는 공개소프트웨어 사용에 따른 법적 무결성의 명시와 법적 책임과 권한을 명시해야 한다.

(라) 사업계획서 검토 및 승인

발주자는 공급자의 수행계획서 상에 공개소프트웨어사용에 따른 합법적인 산출물을 보장하며 소프트웨어나 서비스를 개발, 관리함에 있어 공개소프트웨어 관리 프로세스의 적절성과 승인절차를 점검할 수 있도록 작성되었는지 검토한다.

(마) 수행자 계약이행 점검

계약서에 명시한 공개소프트웨어 사용에 따른 무결성을 검증하기 위하여 수행자의 작업을 평가한다.

□ 과제수행관리 지침 안**(가) 사업계획서 작성**

수행자는 수행계획서 상에 공개소프트웨어 사용에 따른 합법적인 산출물을 보장하며 소프트웨어나 서비스를 개발, 관리함에 있어 공개소프트웨어 사용유무와 적절한 관리 프로세스 및 승인절차를 통한 검증방안을 제시해야 한다.

(나) 개발계획 작성

개발계획에는 과제 기획단계에서 검토된 공개소프트웨어 사용계획서가 포함되어야 한다. 또한, 소프트웨어 구성요소의 공개소프트웨어 사용을 식별, 추적하고 적절한 공개소프트웨어 사용을 위한 관리방안이 포함되어야 한다.

(다) 소프트웨어 요구사항 정의

수행자는 소프트웨어 구성요소의 공개소프트웨어 코드 분류, 사용내역, 공개소프트웨어 라이선스 의무사항 준수 여부를 포함한 소스코드 상세명세서를 정의하고 문서화해야 한다.

(라) 소프트웨어 요구사항 검토 및 평가

수행자는 소프트웨어 구성요소의 공개소프트웨어 추적과 식별을 통해 소프트웨어 요구사항의 준법성을 검토 및 평가해야 한다.

(마) 인터페이스 설계

수행자의 인터페이스 설계기술서는 공개소프트웨어와 상용소프트웨어 외부 개체 및 공개소프트웨어와 상용소프트웨어 구성요소의 인터페이스 특성 및 결합형태 기술을 포함해야 한다.

(바) 소프트웨어 통합시험 요구사항 정의 및 계획 작성

수행자는 소프트웨어 통합 및 시험에 있어서 소프트웨어 개발 프로세스 상에 공개소프트웨어 식별 및 점검을 위한 적절한 일정과 방법을 결정해야 한다.

(사) 소프트웨어 단위시험 요구사항 정의 및 계획 작성

수행자는 소프트웨어 단위시험 계획을 작성함에 있어서 소프트웨어 상세설계 활동의 일부로서 공개소프트웨어 식별 및 관리를 위한 단위시험 요구사항을 포함해야 한다.

(아) 소프트웨어 상세설계/시험 검토 및 평가

합동검토 수행에는 공개소프트웨어 관리 검토를 계획하고 참여하는 것을 포함한다.

(자) 소프트웨어 통합시험 요구사항 정의 및 계획갱신

수행자는 소프트웨어 통합시험을 위한 계획을 작성 또는 수정함에 있어서 공개소프트웨어 식별 및 분석일정을 포함해야 한다.

(차) 소프트웨어 코드 및 단위시험 결과 검토

수행자는 소프트웨어 구성요소 중 적절한 공개소프트웨어 사용을 평가할 수 있는 기준을 준비하여 평가해야 하며, 평가 기준에는 공개소프트웨어 성숙도 모델, 공개소프트웨어 사용 내역, 라이선스 의무사항 준수여부 등을 고려해야 한다.

(카) 소프트웨어 자격시험 실시

수행자는 적절한 공개소프트웨어 사용을 평가할 수 있는 기준에 따라 소프트웨어 구성요소의 공개소프트웨어 식별 및 평가 결과를 문서화해야 한다.

(타) 소프트웨어 산출물 검토

수행자는 소프트웨어를 검토함에 있어서 소프트웨어 구성요소의 공개소프트웨어 준법성 여부를 검토해야 한다.

(파) 소프트웨어 검수

소프트웨어의 검수에는 공개소프트웨어를 포함한 소스코드 상세명세서를 포함해야 한다.

□ 과제검증 지침 안

(가) 검증준비

검증 프로세스에는 공개소프트웨어의 준법 사용에 대한 검증을 포함한다. 공개소프트웨어 사용에 대한 검증범위는 공개소프트웨어 사용 형태 및 정도, 공개소프트웨어 사용 대상 파일, 라이선스, 라이선스 의무사항 준수여부와 미준수 사항을 고려해야 한다.

(나) 검증수행

검증은 공개소프트웨어 사용에 대한 이상 유무, 공개소프트웨어 지적재산권 및 라이선스 준수여부, 사용정도 및 사용내역 등을 명시해야 한다. 또한 사업계획서에 명시된 검증주기와 보고 일정을 검토하여 적절한 관리 담당자를 선정하고 교육 및 검증이행을 실시하여야 한다.

(다) 요구사항 검증

요구사항 검증은 과제 배포 정책, 식별된 공개소프트웨어의 사용 유무, 공개소프트웨어 지적재산권 및 라이선스 준수여부, 사용정도 및 사용내역에 대한 문서화된 기록을 포함한다.

(라) 코드 검증

코드 검증은 공개소프트웨어 사용 코드와 관련 라이선스 및 지적재산권 준수여부를 포함하여 식별된 사용내역을 기록한다.

이상과 같은 공개소프트웨어 관련 발주, 수행, 검수 프로세스 혹은 지침을 수립한 후 실질적으로 내부사용을 목적으로 공개소프트웨어를 활용한 계약 시에 검토해야 할 계약유형은 다음과 같이 구분될 수 있다.

- 특정 벤더사로부터 공개소프트웨어 솔루션 및 스택 형태의 완제품을 발주/공급하는 계약일 경우
- 특정 협력업체로부터 공개소프트웨어를 활용한 솔루션을 개발 발주/공급하는 계약의 경우
- 특정 협력업체로부터 공개소프트웨어를 활용한 시스템을 개발 발주/공급하는 계약의 경우
- 특정 협력업체로부터 공개소프트웨어가 포함된 시스템관리 및 유지보수를 발주/공급하는 계약 경우

□ 특정 벤더사로부터 공개소프트웨어 솔루션 및 스택 형태의 완제품을 발주/공급하는 계약의 경우

(가) 발주사 검토사항

공개소프트웨어 솔루션이나 스택형태의 완제품을 발주하는 경우에는 기존 상용소프트웨어와 달리 지속적인 모니터링과 유지보수가 필요함에 따라 이에 대한 적절성 여부를 판단하여야 한다. 계약 전에 이상의 검토사항 중 기술 수요조사, 검수전략정의, 수행자선정을 검토한 후 계약이행 점검을 선행 검토한 후 과제수행관리 지침에 따라 해당 수행자가 적절히 공개소프트웨어의 모니터링과 유지보수를 시행할 수 있도록 계약서에는 산출물로서 소프트웨어 코드 및 단위시험 결과, 소프트웨어 자격시험, 소프트웨어 검수, 검증 보고서 등을 포함하여야 한다.

(나) 공급사 검토사항

수행자는 발주사의 기술수요조사 및 검수전략정의, 해당 산출물 요구사항이 적절한 지에 대한 검토를 통해 발주사 기술수요조사 내용이 사실과 부합되는 지에 대한 검토를 수행해야 하며 계약서에는 사실과 다른 공개소프트웨어 기술수요조사 내용이 반영되지 않았는지에 대한 확인이 필요하고 상호 정확한 사실 확인을 통해 향후 발생할 수 있는 컴플라이언스, 보안취약점, 품질상의 결함 등에 대한 책임소재를 명확히 해야 한다.

□ 특정 협력업체로부터 공개소프트웨어를 활용한 솔루션을 개발 발주/공급하는 계약의 경우

(가) 발주사 검토사항

공개소프트웨어를 활용한 솔루션을 개발 발주하는 경우에는 기술수요조사, 과제기획서 작성, 검수전략 정의, 제안요청서 준비, 제안서평가, 사업계획서 검토, 수행자 계약이행 점검 등 대부분의 발주관리 지침을 계약 전에 검토해야 하며 수행자가 발주자 라이선스, 보안, 기술 활용 전략 등과 부합되지 않은 공개소프트웨어 컴포넌트를 사용하지 않도록 계약서에 명시해야 한다. 또한 적절한 과제 수행관리를 위해 개발 산출물에는 사업계획서, 개발계획, 소프트웨어 요구사항 정의 등 대부분의 과제수행지침을 명시해야 하며 과제검증을 필수요구사항으로 제시해야 한다.

(나) 공급사 검토사항

공개소프트웨어를 활용한 솔루션 개발 수행자는 발주자 요구사항 및 라이선스, 보안, 기술 활용 전략 등을 사전에 합의하여 과제수행 결과 사전 요구사항과 다른 사실을 바탕으로 분쟁이 발생되지 않도록 책임소재를 명확히 명시해야 한다.

〈표 16〉 소스코드 사용명세서의 예

□ 과제명							
구 분		내 역					
소프트웨어 구성	자체개발	공개 소프트웨어	기타	코드명세	비 고		
■ UI	%	%	%	*첨부			
■ 어플리케이션	%	%	%	*첨부			
■ OS	%	%	%	*첨부			
■ 기타	%	%	%	*첨부			
	%	%	%	*첨부			
계	%	%	%	*첨부			
공개소프트웨어 사용 내역	파일명	사용정도	결합 형태	컴포넌트 명	라이선스 명	공개 여부 (O, X)	라이선스 준수여부 (O, X)
■ UI		%					
■ 어플리케이션		%					
■ OS		%					
■ 기타		%					
계							
라이선스 준수사항							
■ 라이선스 명	준수사항					비 고	
Apache 2.0	라이선스 및 Copyright 고지						

※ 본___은 국내 저작권 법 및 국제저작권협약을 준수하며 이를 상기 소스코드 사용명세서로 증명함

□ 특정 협력업체로부터 공개소프트웨어를 활용한 시스템은 개발 발주/공급하는 계약의 경우

(가) 발주사 검토사항

공개소프트웨어를 활용한 시스템 개발을 발주하는 경우 발주사는 더욱 세밀한 발주 및 수행관리, 검증관리가 필요하다. 모든 발주관리, 수행관리, 검수관리 지침을 검토 후 산출물로 요구해야 하며, 해당 시스템에 발주사가 원하지 않는 공개소프트웨어 라이선스를 포함한 코드가 유입되지 않도록 명확한 라이선스 정책을 제시해야 한다. 또한 부가적으로 수행자가 활용한 공개소프트웨어가 발주사의 소유권을 가진 소프트웨어와 결합하여 향후 시스템 유지보수 업체 변경 시에 내부자산이 유출될 수 있는 위험을 예방하기 위해 수행자의 인터페이스 설계, 상세설계 등에 대한 자세한 산출물을 제시해야 한다.

(나) 공급사 검토사항

공개소프트웨어를 활용한 시스템을 개발 공급하는 수행자는 과제수행관리 지침 및 지침을 모두 수행해야 하며, 발주자의 사전 요구사항 및 라이선스 정책 등과 다른 사실을 바탕으로 분쟁이 발생되지 않도록 책임소재를 명확히 명시해야 한다.

□ 특정 협력업체로부터 공개소프트웨어가 포함된 시스템관리 및 유지보수를 발주/공급하는 계약 경우

(가) 발주사 검토사항

공개소프트웨어가 포함된 시스템 관리 및 유지보수를 발주하는 발주사는 무엇보다 현재 시스템에 대한 공개소프트웨어 사용현황과 라이선스 호환성을 검토해야 하며 이러한 사실을 바탕으로 수행자에게 명확한 공개소프트웨어 사용범위 및 라이선스 범위를 제시해야 하고 향후 내부적으로 사용된 공개소프트웨어에 대한 재배포 및 재사용 범위를 명확히 해야 한다. 계약서에는 이러한 요구사항을 바탕으로 수행자의 산출물 및 유지보수 계약완료 이후의 내부 자산화 된 소프트웨어와 공개소프트웨어에 대한 기술적 재사용에 대한 범위와 책임을 명확히 하는 것이 필요하다.

(나) 공급사 검토사항

공개소프트웨어가 포함된 시스템의 관리 및 유지보수를 수행하는 수행자는 기본적으로 발주사의 자산화 된 코드가 상용이던 공개이던 직접적 재사용을 위해서는 발주사의 동의를 취득해야 한다. 공개소프트웨어의 특성상 제약조건 없이 재사용이 가능하지만 발주사의 요구사항 및 계약조건에 따라 향후 재사용에 있어서 분쟁 가능성이 내재되어 있는 만큼 과제 수행 중에 획득된 기술적 노하우에 대해 재사용 범위를 명확히 제시해야 한다. 이러한 재사

용에 대한 범위를 명확히 함으로써 향후 공개소프트웨어를 활용한 분쟁을 예방할 수 있다.

7) 패키징

외부 배포 단계에서 패키징은 필수 실행항목이 아니며 배포하는 소프트웨어의 구성 요소와 복잡성에 따라 패키징 여부를 결정하면 된다. 패치나 플러그인과 같이 단편적인 용도로 사용되는 경우에는 패키징이 불필요하다. 반면 배포하는 소프트웨어의 구성 요소가 많고 설치 과정이 복잡하며 패키징을 반드시 수행하여 설치상의 오류나 실수를 방지하여야 한다. 만일 배포 대상이 공개소프트웨어 커뮤니티일 경우에는 소스코드를 웹 사이트에 등록하는 과정이 배포이므로 패키징이 불필요하다.

8) 배포

공개소프트웨어의 배포는 첫째, 외부에 있는 공개소프트웨어를 전혀 변경하지 않고 그대로 외부로 전달하는 경우와 둘째, 외부에서 가져온 이후에 개작하여 외부로 전달하는 경우, 셋째 소스코드의 변경여부에 상관없이 상용 소프트웨어 또는 독점 소프트웨어와 연동되는 방식으로 제작하여 한꺼번에 외부로 전달하는 경우 등 크게 세 가지로 구분할 수 있다.

한편, 산업 유형 별로 배포 방식을 살펴보면 임베디드 소프트웨어 산업의 경우에는 특정 기기 및 장비가 배포 매체가 된다. 솔루션 산업의 경우에는 일반적인 데이터 저장 매체를 통해서 배포되며, SI 서비스 산업의 경우에는 고객이 접근하고 사용하는 시스템에 공개소프트웨어가 설치되므로 시스템 자체가 배포 매체가 된다. 일반적으로 공개소프트웨어의 라이선스 위반 문제는 언제나 배포 시점에서 발생하게 되므로 배포되는 소프트웨어의 결합 방식에 유의하여야 한다. 상기의 세 가지 배포 유형에서 첫 번째는 하나의 공개소프트웨어를 있는 그대로 외부에 보내는 경우이므로 컴플라이언스 이슈가 전혀 없다. 그러나 공개소프트웨어를 개작하거나, 소스코드 차원의 개작이 아니라도 라이브러리 호출과 같은 방식으로 연동되는 경우라면 라이선스 종류에 따라 컴플라이언스 이슈가 발생하게 되므로 두 번째와 세 번째 배포 유형은 특히 조심하여야 한다.

예를 들어 Affero GPL의 경우에는 네트워크 인터페이스를 통하여 연결되더라도 GPL 라이선스와 동일한 의무사항을 준수해야 하므로 비록 개작이 없더라도 컴플라이언스 문제가 생길 수 있다.

9) 커뮤니티

소스코드를 커뮤니티에 등재하는 활동도 배포에 해당된다. 개작된 공개소프트웨어를 커뮤니티로 배포하기 위해서는 커밋이라고 하는 특별한 단계를 거쳐야 한다. 따라서 커뮤니티에 배포하는 활동은 계약 조건이나 규제가 없어서 매우 자유롭지만 심사위원에 해당되는 커미터(committer)의 승인이 필요하므로 배포 활동 중에서 가장 어려운 일이라 할 수 있다.

10) 컴플라이언스

공개 소프트웨어 컴플라이언스는 배포자가 피배포자에게 공개소프트웨어가 활용된 제품 및 솔루션, 서비스를 전달하는 과정에서 배포자가 라이선스 의무사항을 준수하는 활동으로서 공개 소프트웨어의 프로젝트 명칭, 원저작자, 저작권 선언, 라이선스 명칭, 파일 경로, 결합 방법, 충돌 여부 등 준법에 관련된 정보를 제공해야 한다. 특히, 공개소프트웨어를 직접 가져와서 개작 후 외부에 배포하는 경우 해당 공개소프트웨어로 인해 발생하는 모든 컴플라이언스 이슈에 대한 책임은 최종 배포자에게 있고 소프트웨어 개발에 참여하는 모든 공급망에 포함되는 조직에게 2차 책임이 있다. 따라서, 조직 내부의 컴플라이언스 관리도 중요하지만 적절한 공급망 관리 프로세스를 통해 공급망에 참여하는 모든 조직에 대한 컴플라이언스 관리를 체계화하여 모니터링 하는 것이 필요하다.

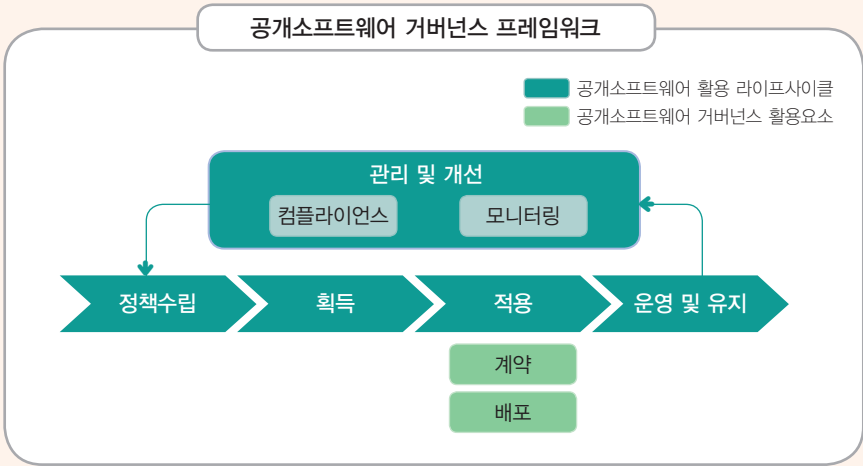
공개소프트웨어는 소스코드가 공개되어 있기 때문에 누구나 활용이 가능하여 소프트웨어의 사용에 여러 가지 혜택이 있는 반면, 저작권으로 보호받기 때문에 라이선스 부분에 있어서 법적분쟁의 발생, 기업 이미지 하락 등의 문제 발생 가능성이 있다. 이처럼 저작권이 있는 공개소프트웨어는 라이선스별로 사용과 배포 등에 관련된 다양한 의무사항을 요구하고 있으므로 공개소프트웨어를 개작하여 배포할 때에는 조사 및 분석단계에서부터 배포에 이르기 까지 라이선스의 특징과 의무사항을 면밀히 검토하여 내부 정책을 준수하며 사용할 수 있도록 통제하여야 한다. 라이선스 별로 주요 의무사항이 다양하지만 주요 의무사항으로는 사용권 고지의 의무, 저작권 고지의 의무, 소스코드 공개의 의무, 특허포기의 의무 등이 쟁점 사항이 될 수 있다.

(2) 특정 벤더사를 통해 공개소프트웨어 도입 및 기술지원을 받아 외부에 배포하는 경우

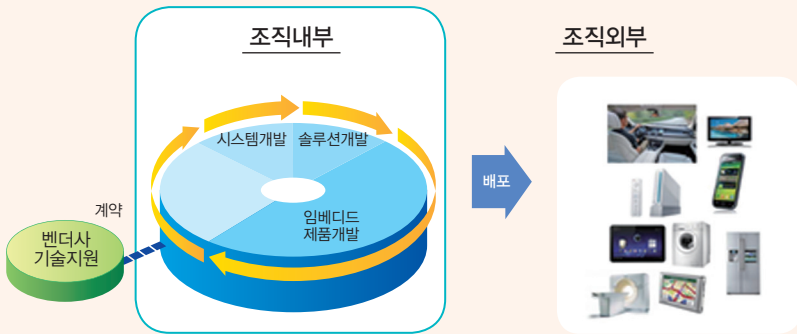
공개소프트웨어의 배포 주체가 공개소프트웨어를 외부에 배포할 때 특정 벤더사의 지원을 받는 경우는 대개 SI 업체가 전문적인 공개소프트웨어 기술지원 업체를 통해서 소프트웨어를 납품하거나, 사용 주체와 배포 주체 간의 공급 계약에 의해서 지리적으로 떨어져 있는 사용 주체의 지점이나 계열사 등을 대상으로 공개소프트웨어의 배포를 지원해 주는 경우이다. 따라서 이 경우에 관련된 실행 항목은 무척 제한적이어서 계약, 배포, 컴플라이언스, 모니터링 정도만이 관련된다. 특히 커뮤니티가 실행 항목에 들지 않는 이유는 소스코드를 공개소프트웨어 커뮤니티의 프로젝트에 등재하는 일은 너무 단순하기 때문 구태여 벤더사의 기술지원 서비스를 받을 필요가 없기 때문이다.

1) 계약

배포는 공급의 주요한 이행 항목이고 계약은 공급에 대한 조건을 명시하게 되므로 공급계약



[그림 36] 외부배포 관점 Case 2



[그림 37] 특정 벤더사를 통해 공개소프트웨어 도입 및 기술 지원을 받아 외부에 배포하는 경우

사항에 따라 배포를 이행하여야 함은 당연하다. 즉, 배포의 일시, 장소, 전달 매체, 불이행에 대한 책임과 보상 절차, 배포에 필요한 비용 처리 절차 및 조건 등을 사전에 살펴보고 배포를 이행하여야 한다. 특히 공개소프트웨어는 소스코드와 실행 파일의 배포 뿐 아니라 고지의 의무가 있으므로 배포되는 공개소프트웨어에 관련된 정보를 제공하여야 한다. 일반적으로 이러한 공급 제품에 대한 정보를 BoM(Bill of Material) 형식으로 전달하고 있으며 이 양식은 공개소프트웨어의 명세서로 간주할 수 있다. 아직까지는 이 명세서에 대한 표준이 정의되어 있지 않기 때문에 필수적으로 기재하여야 하는 항목과 순서는 배포하는 주체와 배포 받는 주체

간에 지정하여도 전혀 문제가 되지 않는다. 다만 리눅스 파운데이션에서 활발하게 정립하고 있는 SPDX(Software Package Data Exchange)라고 하는 상용 표준 체계를 참조할 수 있다. 그러므로 현재 수준에서는 배포되는 파일 단위로 저작 정보(저작자 이름, 소속, 연락처 및 저작 일자)와 라이선스 정보(이름, 버전)를 제공하는 것이 기본이다. 또한 배포되는 소프트웨어를 구성하는 라이선스들 간의 충돌 여부도 명시하여야 한다. 만일 카피레프트를 따르는 공개 소프트웨어를 사용했기 때문에 공개의 의무가 발생했지만 실행 파일만 배포하는 상황이라면 소스코드공개 또는 코드공개에 대한 서면의사제공(written offer)으로 대체할 수 있다. 공개소프트웨어의 BoM과 마찬가지로 코드공개에 대한 서면의사제공의 표준이 결정되어 있지는 않지만 약식의 공급 계약서 양식으로 작성되어야 한다. 즉, 누가 누구에게 어떤 소스코드를 어떠한 방법으로 언제까지 어떻게 배포할 것인지 명시되어야 한다. 또한, 이러한 배포 약속이 이행되지 않을 경우에는 책임과 후처리 방안도 명시되어야 한다.

2) 배포

벤더사의 지원으로 공개소프트웨어를 배포할 경우에도 상용소프트웨어의 공급 절차와 전달 방법과 크게 상이하지 않다. 다만 외부의 도움이 없이 자체적으로 배포하는 상황보다는 훨씬 편리하고 안전한 배포 서비스를 받게 된다. 이 경우의 배포는 공급 절차 중 하나이므로, 배포를 받는 자는 상용 제품과 동일한 방법으로 검수를 수행하여야 한다.

3) 컴플라이언스

모든 외형적인 컴플라이언스 문제는 배포 단계에서 발생한다. 이는 최종 배포주체가 모든 책임을 지는 구조이기 때문이다. 비록 최종적으로 공개소프트웨어를 배포하는 주체는 배포를 대행하는 서비스를 받았을 지라도 컴플라이언스 위반에 대한 책임을 회피할 수가 없다. 그러므로 공개소프트웨어의 배포 주체는 배포 대행자에게 컴플라이언스에 관련된 모든 정보를 요구해야 하며, 추후 사고가 발생하면 사후 처리를 어떻게 진행할 것인지 계약 조건으로 명시해 놓아야 한다.

4) 모니터링

배포에 관련되어서는 협의가 아닌 광의의 모니터링이 해당된다. 즉 배포되는 공개소프트웨어의 실제적인 운영 상태에 대한 정보보다는 배포가 수행되는 상황에 대한 정보가 필요하기 때문이다. 즉, 배포가 단번에 완료되지 않고 순차적 또는 비정기적으로 진행되고 있다면 각각의 상황에 대해서 배포의 대상, 조건, 방법, 규제 등에 대해서 현황을 파악할 수 있는 정보를 지속적으로 수집해 나가야 한다.

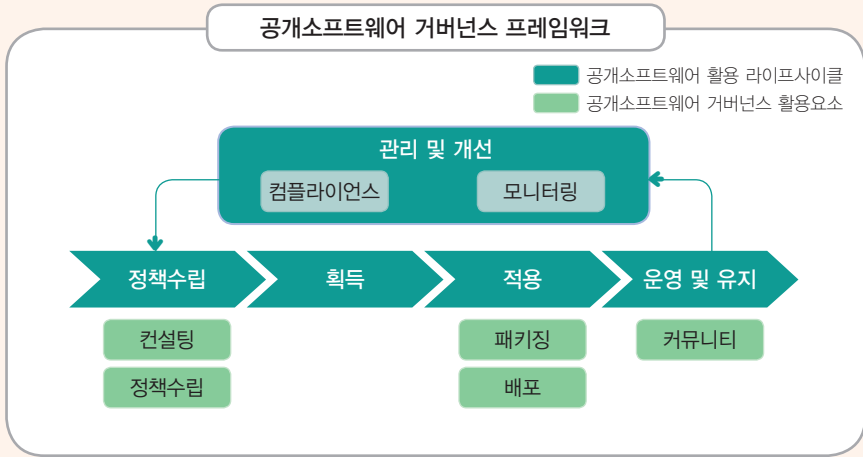
(3) 직접 개발한 소프트웨어를 공개소프트웨어로 외부에 배포하는 경우

이 경우는 공개소프트웨어의 배포 주체가 외부의 공개소프트웨어를 재사용하여 내부적으로 독점 소프트웨어를 개발한 이후 특정 공개소프트웨어 라이선스를 붙여서 해당 소프트웨어의 소스코드를 외부로 공개하는 상황이다. 이런 상황에서는 내부의 지적재산이 자연스럽게 유출된다. 따라서 이러한 손실을 감수하고 지적자산을 외부로 공개하는 이유가 분명하여야 한다. 숨어 있는 이유는 많이 있겠지만 ① 실패한 소프트웨어 프로젝트의 탈출 전략, ② 진입 장벽을 높이고 차별적인 기술력을 확보한 이후에 공개소프트웨어 프로젝트를 진행, ③ 엔터프라이즈 버전을 보유한 상태에서 상업적인 저변 확대를 위해 커뮤니티 버전을 공개하는 경우가 대부분이다.

첫 번째 경우에는 죽어가는 프로젝트를 살리기 위해서 모든 방법을 다 써 보았지만 대안이 없기 때문에 소스코드를 공개해서 외부와 협업하고 프로젝트 영속성을 확보하자는 전략이다. 그러나 소스코드를 공개하더라도 외부의 공개소프트웨어 개발자들은 관심이 없거나 협업에 대한 의지가 부족할 수 있기 때문에 어차피 성공할 수 없는 프로젝트이기 때문에 공개하겠다는 계획은 실패할 가능성이 매우 크다. 두 번째 경우는 첫 번째보다는 프로그램의 완성도가 높고 매력도가 있기 때문에 공개 이후에 배포 주체는 프로젝트를 선도할 가능성이 크다. 다만 공개소프트웨어 커뮤니티는 협업에 의해서 개발이 진행되고 집단적인 지적 자원을 공유하는 체제이므로 최초 활성화에 주력하여야 한다. 그렇지 않으면 참여는 없고 자체적으로만 커뮤니티를 운영하여야 하는 위험이 있기 때문이다. 세 번째 경우는 공개하는 소프트웨어의 완성도와 제품적인 효용성이 매우 크기 때문에 사업적인 전략으로 소스코드를 공개하게 된다. 이때 배포 주체는 공개소프트웨어 프로젝트의 소스코드를 대부분 소유하고 있으며 차별적인 기술 서비스와 고품질의 공개소프트웨어를 근간으로 하는 사업 모델을 표방한다. EnterpriseDB 와 MySQL 등이 이 분야의 대표적인 공개소프트웨어이다.

1) 컨설팅

자체적으로 개발한 소프트웨어를 공개소프트웨어로 배포하기 이전에 공익적, 사업적 관점에서 최선의 전략 수립과 이행 방법을 얻기 위해서 컨설팅이 필요할 수 있다. 이 컨설팅은 객관적인 시각에서 원인과 결과를 분석해 주기 때문에 정책수립이 대표적인 요소에 포함된다. 그렇기 때문에 독점 소프트웨어의 배포와 효과를 경험해 본 컨설턴트가 다양한 참조 사례를 가지고 컨설팅을 수행하여야 한다. 그러나 컨설팅이 도리어 독점 소프트웨어뿐만 아니라 향후 개발 로드 맵이나 내부 보안 정보의 유출의 계기가 될 수도 있으므로 상호 계약 조건에 유의하여야 한다.



[그림 38] 외부배포 관점 Case 3



[그림 39] 직접 개발한 소프트웨어를 공개소프트웨어로 외부에 배포하는 경우

2) 정책수립

자체적으로 개발한 소프트웨어를 공개할 경우에는 향후의 파급효과 차원에서 신중한 정책 수립이 필요하다. 이는 개발을 수행한 부서의 진로뿐만 아니라 배포 주체의 이미지, 사업적 영향도, 지적 자산의 유출 등 다양한 관점에서의 정책과 연관되기 때문이다. 소스코드는 아이디어의 표현이고 그 안의 아키텍처와 알고리즘 자체가 영업비밀이고 특허가 될 수 있기 때문에 고유한 자산이 외부로 유출되어 예기치 않은 피해가 발생되지 않도록 각별히 유의하여야 한다. 즉 소스코드의 공개와 커뮤니티 생태계의 근간이 되는 자유, 개방, 공유, 참여 철학에 지나치게 편중된 정책을 수립하게 되면 과거의 Netscape사의 전철을 밟을 수도 있

으므로 주의하여야 한다. 반대로 개방과 협업의 물결이 일고 있는 시대에 지나치게 배타적이고 폐쇄적인 정책으로 일관하게 되면 개방형 스마트폰 시장에서 몰락하게 된 Nokia의 전철을 밟게 될 수 있다. 따라서 배포 주체가 최초로 소스코드를 공개하게 되는 상황이라면 내부적으로 또한 외부적으로 어떠한 영향이 발생할 것인지 분명하게 예측해서 연구, 개발, 마케팅, 영업, 수주 차원에서 불이익이 발생하지 않도록 정책 수립에 유의하여야 한다.

3) 패키징

자체적으로 개발한 소프트웨어를 공개소프트웨어로 외부에 배포할 경우에 패키징은 선택적인 실행항목이다. 일반적으로 공개소프트웨어로 변경되는 소프트웨어에 대해서는 설치의 편리성을 위해서 패키지를 만들어야 하는 의무는 없으므로 공개소프트웨어 커뮤니티에서 자발적으로 패키징이 진행되고 있다.

4) 배포

자체적으로 개발한 소프트웨어를 공개소프트웨어로 외부에 배포할 경우에 대부분 배포 주체의 홈페이지 또는 공개소프트웨어 커뮤니티를 통해서 배포된다. 전자의 경우에는 향후 추가적인 개발이 전제되지 않고 배포 주체의 이미지 개선과 홍보 차원에서 소스코드를 공개하게 된다. 후자의 경우에는 소스코드를 공개할 커뮤니티를 먼저 만들어 놓고 협업을 통해서 외부의 지적 자원을 활용할 목적으로 소스코드를 공개하게 된다.

5) 커뮤니티

자체적으로 개발한 소프트웨어를 공개소프트웨어로 외부에 배포할 경우에 새로운 커뮤니티를 만들거나 기존의 커뮤니티를 통해서 배포할 수 있다. 이렇게 커뮤니티를 통해서 배포가 이뤄질 경우에는 배포를 받는 주체가 직접 소스코드를 가져가는 형태가 된다.

6) 컴플라이언스

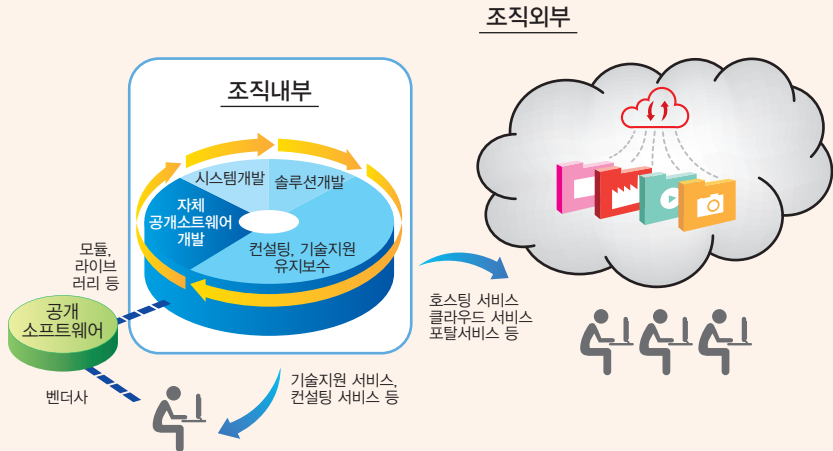
자체적으로 개발한 소프트웨어를 공개소프트웨어로 외부에 배포할 경우에는 자체 개발 부분의 라이선스를 먼저 지정하여야 한다. 라이선스가 결정된 다음에는 공개소프트웨어 커뮤니티에서 지정하는 방식으로 저작권 선언문을 작성하고 라이선스 전문을 파일 단위로 입력하여야 한다. 아직까지는 저작권 선언문구의 항목, 표현 양식, 순서 등에 대한 표준은 제정되어 있지는 않지만 향후 저작권 침해와 분쟁의 소지가 없도록 자세하고 명확하게 구성하여야 한다. 이러한 문서 작업이 완료된 다음에는 배포되는 소프트웨어에 포함된 라이선스 사이에 충돌이 없는지 확인하여야 한다. 또한 재배포되는 공개소프트웨어에 대해서 반복적으로 라이선스를 분석해서 재원과 인력을 낭비하는 일이 없도록 라이선스를 분석한 이후에 이 결과를 공개소프트웨어 파일별로 관리하는 체제를 구축하여야 한다.

7) 모니터링

자체적으로 개발한 소프트웨어를 공개소프트웨어로 외부에 배포할 경우에 배포 방법에 따라서 모니터링의 내용이 달라진다. 배포 주체의 웹 사이트에서 공개소프트웨어가 배포되고 있다면 방문자수, 다운로드 수, 사용 사례 등을 중심으로 모니터링이 진행된다. 이 결과에 따라서 배포 환경을 유지하거나 개선할 지에 대한 결정이 이루어질 수 있다. 한편 배포가 커뮤니티를 통해서 진행된다면 커뮤니티 자체가 모니터링의 대상이 된다. 특히 이 경우에는 커뮤니티의 성숙도, 영향도, 영속성 차원에서 정보를 모을 수 있는 모니터링이 진행되어야 한다.

4. 외부서비스 관점

외부서비스 관점이란 공개소프트웨어를 활용하여 외부 고객을 대상으로 다양한 유형의 IT 서비스를 제공하는 사용자를 의미한다. 공개소프트웨어의 위치가 조직의 내부에 있는지 외부에 있는지에 따라서 다음과 같은 경우로 구분할 수 있다.

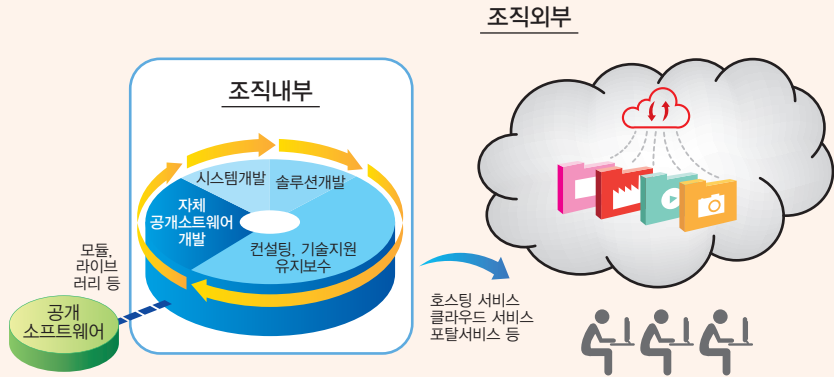


[그림 40] 외부서비스 관점

(1) 공개소프트웨어가 내부에 있고 외부에 서비스를 제공하는 경우

사용자가 공개소프트웨어를 직접 개발하는 생산자가 아니지만 사용자의 조직 내부에 공개 소프트웨어를 획득하고 이를 활용하여 외부고객에게 다양한 서비스를 제공하는 비즈니스

모델의 경우를 의미한다. 기업의 비즈니스 모델이 공개소프트웨어를 활용하는 모든 경우가 해당되기 때문에 금융, 포털, 호스팅, 퍼스널 클라우드, 게임 등 공개소프트웨어와 직접 연관이 없는 비즈니스도 모두 여기에 해당된다.



[그림 41] 공개소프트웨어가 내부에 있고 외부에 서비스를 제공하는 경우

1) 호스팅 서비스

공개소프트웨어를 사용하여 고객에게 판매할 수 있는 호스팅 서비스를 제공

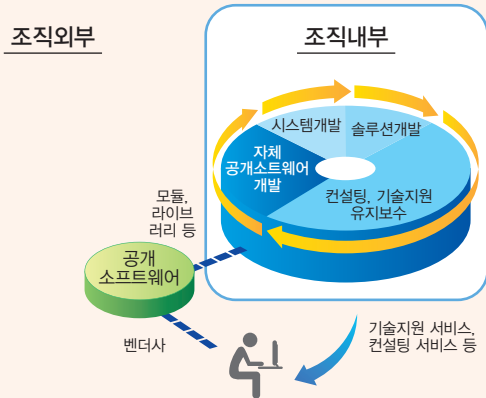
2) 광고 모델

공급 업체가 제품을 구축 할 때 공개소프트웨어를 사용하여 개발한 후 서비스(클라우드)로 제공하며, 해당 서비스의 사용자에게 광고를 게재

이 경우 사용자는 외부의 공개소프트웨어를 획득하여 조직 내에서 요구하는 기능을 수행할 수 있도록 개작하고, 조직의 핵심 비즈니스를 공개소프트웨어 기반에서 제공하게 되기 때문에 정책수립, 획득, 적용, 운영 및 유지, 관리 및 개선의 모든 단계에서 다양한 거버넌스 활동요소를 고려해야 하며, [그림 41]의 공개소프트웨어 거버넌스 프레임워크를 참고하여 공개소프트웨어를 사용할 수 있다.

(2) 공개소프트웨어가 외부에 있고 외부에 서비스를 제공하는 경우

서비스공급자가 기업 외부에 있는 공개소프트웨어를 활용하여 외부고객에게 다양한 서비스를 제공하는 비즈니스 모델의 경우를 의미한다. 공개소프트웨어를 대상으로 하는 IT 컨설팅, 비즈니스 컨설팅, 라이선스 컨설팅, 기술지원, 유지보수, 교육, 공개소프트웨어 인프라 운영 아웃소싱 등의 비즈니스가 여기에 해당된다.



[그림 42] 소스코드가 외부에 있고 외부에 서비스를 제공하는 경우

1) 구독 모델

사용자가 소프트웨어를 무료로 다운로드 받고 제한 없이 사용할 수 있는 모델

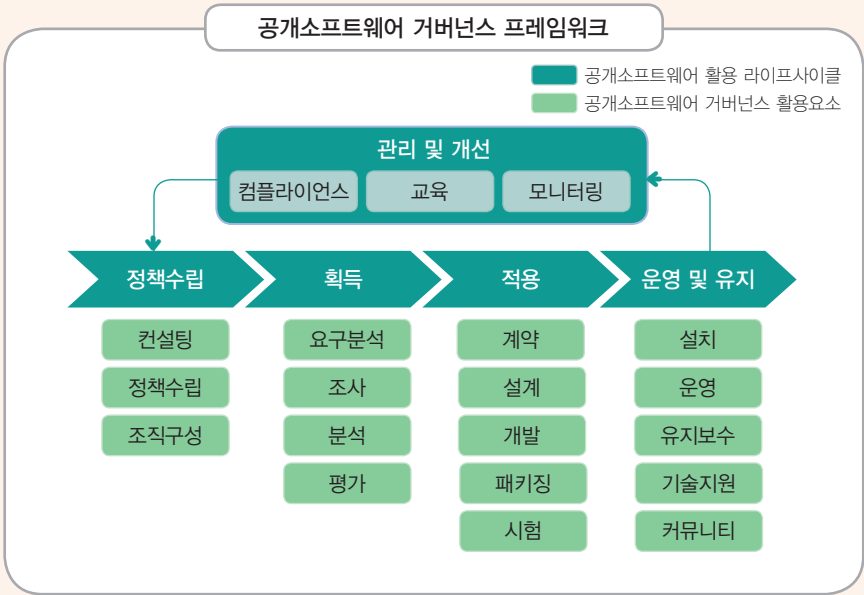
- 사용자가 수동으로 소프트웨어 업데이트를 확인해서 설치하고 무료로 다시 기술적인 문제에 대한 답변을 토론 포럼을 통해 해결할 수 있다.
- 사용자는 특정 문제를 도와 줄 컨설턴트와 계약자를 고용 할 수 있으며, 기업은 업데이트 및 지원의 수준을 가입제품 기준으로 제공하는 방식

2) 컨설팅 모델

공급 업체가 직접 공개소프트웨어를 제공하지는 않지만 고객이 공개소프트웨어와 관련된 전략적 의사 결정과 투자를 하는데 도움이 되어주고, 공급 업체는 컨설팅 비용을 고객에게 청구하는 방식.

이 경우 공개소프트웨어의 기업내부 적용과정이 없기 때문에 다른 유형의 사용에 비하여 비교적 거버넌스 활동요소가 적지만, 유지보수 서비스를 제공하는 경우에는 적용단계의 모든 거버넌스 활동요소(설계, 개발, 패키징, 시험, 배포, 설치)를 고려해야 한다.

이런 비즈니스 모델에서는 [그림 43]과 같은 공개소프트웨어 거버넌스 프레임워크를 참조할 수 있다. 외부서비스 사용자 관점에서 공개소프트웨어 거버넌스를 위한 주요 활동요소는 다음과 같다.



[그림 43] 외부서비스 관점 Case 1

1) 컨설팅

외부의 공개소프트웨어를 획득하여 고객에게 서비스를 제공하고자 하는 사용자가 공개소프트웨어에 대한 전문성이 없는 경우에는 외부의 전문가로부터 컨설팅을 받을 수 있다. 공개소프트웨어 컨설팅은

- 사업 환경 · 전략 확인
- 공개소프트웨어 현황분석
- 공개소프트웨어 도입원칙 수립
- 추진과제 정의
- 이행계획수립의 단계로 수행되며, 컨설팅을 통하여 전략과 요구사항, 공개소프트웨어 적용 고려 사항, 정보시스템을 효과적으로 지원할 수 있는 정보를 파악하고 시스템의 기본 방향과 체계를 수립하며 시스템 구축을 위한 개념모델과 공개소프트웨어 기반의 스택을 결정하고 구체적 실행계획을 제공하게 된다.

사용자는 공개소프트웨어 사용을 위하여 다음과 같은 위험에 대비하여 원칙을 수립하는 것이 중요하다.

- 공개소프트웨어 라이선스
- 기술지원의 가용성
- 내부 전담 자원의 확보
- 기술지원 서비스의 유연성
- 공개소프트웨어의 신뢰성
- 공개소프트웨어 프로젝트의 완성도 및 수명

사용자가 공개소프트웨어를 내부에 보유하지 않고 서비스를 제공하는 경우에는 컨설팅의 의미가 외부 전문가로부터 받는 것이 아니라 컨설팅을 제공하는 경우로 해석해야 한다.

외부서비스로 제공하는 공개소프트웨어 컨설팅은 다음과 같은 절차로 수행하게 되며, 각 절차의 세부적인 내용은 본 가이드 전체 활동요소 중 컨설팅 부분을 참고하기 바란다.

- 사업 환경/전략 확인
- 공개소프트웨어 현황분석
- 공개소프트웨어 도입원칙 수립
- 공개소프트웨어 추진과제 정의
- 공개소프트웨어 도입계획 수립

2) 정책수립

외부의 공개소프트웨어를 활용하여 고객에게 양질의 서비스를 제공하기 위해서는 반드시 정책수립 단계를 통해 조직의 사업전략에 적합한 정책을 정의하고 효율적 정책의 반영을 위한 조직 내 자원의 역할과 책임을 정의해야 한다.

공개소프트웨어 활용정책을 수립하기 위해서는 조직 및 구성원들이 공개소프트웨어를 사용할 때 준수해야 하는 준법성 요구사항들을 인지하게 하고, 모든 구성원들이 적합한 승인 절차를 통하여 공개소프트웨어를 활용할 수 있도록 준비해야 하며, 조직 전반에 어떻게 공개소프트웨어가 관리되어 질 것인가를 정의한 문서화된 공개소프트웨어 정책을 배포해야 한다.

외부의 공개소프트웨어를 활용하여 고객에게 서비스를 제공하는 경우에는 다음과 같은 정책이 반드시 필요하다.

- 조직 내 공개소프트웨어의 사용 범위
- 획득한 공개소프트웨어에 대한 평가방안
- 획득한 공개소프트웨어 또는 개작된 소프트웨어의 라이선스 검증절차
- 획득한 공개소프트웨어 프로젝트의 커뮤니티 참여 및 모니터링 방안

- 조직 내 공개소프트웨어가 사용되는 라이프사이클을 관리하기 위한 절차
- 공개소프트웨어 관련 기술지원 및 교육 방법

사용자가 공개소프트웨어를 내부에 보유하지 않고 외부 고객에게 서비스를 제공하는 경우에는 조직 내 공개소프트웨어의 사용이 없기 때문에 별도의 관련정책이 필요하지 않다.

3) 조직구성

외부의 공개소프트웨어를 활용하여 고객에게 서비스를 하는 사용자는 준비된 공개소프트웨어 활용정책의 운영을 전담하는 별도의 자원을 배정해야 한다. 사용자의 환경에 따라 전담 부서 또는 겸임부서를 운영하거나 별도의 전담인력 확보가 어려운 경우 겸임인력으로도 구성할 수 있다.

외부 서비스를 하는 사용자는 공개소프트웨어 활용정책을 전담하는 자원에게 다음과 같은 역할을 부여할 수 있다.

- 전사 공개소프트웨어 정책 수립 및 업데이트
- 전사 업무프로세스 및 개발방법론 개선
- 교육 강의 및 실습 지원
- 공개소프트웨어 SW 라이선스 검증 지원
- 공개소프트웨어 전담 창구 운영
- 공개소프트웨어 라이선스 검증을 위한 현장 지원

사용자가 공개소프트웨어를 내부에 보유하지 않고 서비스를 제공하는 경우에는 별도의 전담인력 또는 전담부서를 배정할 필요가 없고, 일반적인 기업의 조직 설계 및 운영절차를 적용하면 된다.

4) 요구분석

외부고객을 대상으로 서비스를 제공하기 위해서는 먼저 조직 내부와 외부의 이해관계자를 대상으로 인터뷰 또는 설문 등을 통해 서비스 요구사항을 수집하고 수집된 요구사항을 기반으로 요구분석을 수행하여 최종적으로 요구사항을 정의하게 된다.

서비스를 위하여 외부의 공개소프트웨어를 활용하기 위해서는 다음과 같은 항목에 대한 분석을 토대로 요구를 분석하는 과정이 필요하다.

- 서비스를 제공하는 시스템의 필수기능 파악
- 서비스를 제공하는 시스템의 특성 파악

- 시스템의 관리자 기술수준
- 시스템 운영 및 관리 방안
- 기타 시스템 제약사항

5) 조사

외부 서비스를 제공하기 위한 시스템의 요구사항을 만족하는 외부의 공개소프트웨어를 조사하기 위하여 사용자는 먼저 어떠한 항목을 조사할 것인지 선택해야 한다. 공개소프트웨어는 사용자의 서비스 유형 또는 사용자 조직의 운영환경에 따라서 공개소프트웨어 속성의 중요도가 각각 다르기 때문에, 어떤 항목을 조사할 것인지 정의하고, 각 항목의 가중치를 부여하여 항목별 중요도를 사용자에게 적합하게 조정하는 것이 필요하다.

자신의 서비스에 적합한 공개소프트웨어를 선정하기 위해서 사용자는 조사할 항목과 중요도를 식별한 후 SourceForge, GitHub, OpenHUB 등 다양한 채널의 공개소프트웨어 저장소를 활용하여 최초 등록일, 소스코드의 증가 속도, 참여 기업, 다운로드 횟수, 레퍼런스 개수, 핵심 개발자 및 커미터 등 사용자에게 필요한 공개소프트웨어 속성을 조사해야 한다.

외부서비스에 적합한 공개소프트웨어를 조사하는 단계는 다음과 같이 수행할 수 있다.

〈표 17〉 조사 활동의 단계와 내용

단계	활동 내용
Seeking	초기에 급성장하는 공개소프트웨어를 관찰하고 미래 사업에 핵심적인 기술 후보를 발견한다.
Sensing	기술 흐름과 고객 니즈에 따른 기술 적정성과 시장성을 파악한다.
Seeding	사업 가능성이 있는 기술을 비즈니스 모델과 연결시키고 구체화 시킨다.
Sourcing	필요한 인력을 확보하고 커뮤니티를 통해 기술을 내재화 시킨다.

서비스에 적합한 공개소프트웨어를 직접조사하기 어려운 경우 공개소프트웨어 역량프라자에서 제공하는 공개소프트웨어 목록(http://www.oss.kr/oss_repository12), 공개소프트웨어 테스트 결과자료(http://www.oss.kr/oss_repository6), 유망 공개소프트웨어 프로젝트 목록(http://www.oss.kr/oss_repository12/103649)을 참고할 수도 있다.

〈표 18〉 속성 별 정량화 공식과 적용 방법

속성군	속성	채점 방법	평가 방법
커뮤니티	나이 및 규모	변수 = {버전 번호, 연령} 지표 = 최종 버전 번호 x 나이 1 점: 0 <= 지표 < 12 2 점: 12 <= 지표 < 24 3 점: 24 <= 지표 < 72 4 점: 72 <= 지표 < 180 5 점: 180 <= 지표	지표는 최종 버전 번호와 월 단위의 커뮤니티 나이를 곱해서 산출함 버전 번호가 1.0 이상이고 커뮤니티 나이도 12개월 이상이 되어야 자생력이 있는 커뮤니티로 인정함 버전이 3.0 이상이고 연수가 50상이면 최상위 수준으로 인정함
	주체	변수 = {후원 단체 유무} 1 점: 지원 없음 2 점: 하나의 중소기업 지원 3 점: 복수의 중소기업 지원 4 점: 하나의 대기업의 지원 5 점: 복수의 대기업의 지원	인력 및 자금에 대한 후원 단체의 유무로 측정함
	접근성	변수 = {게시판, 포럼, 위키, 검색, 인터넷} 지표 = 제공하는 접근 방법의 종류 / 전체 접근 방법의 종류 개수 1 점: 0.0 <= 지표 < 0.2 2 점: 2.0 <= 지표 < 0.4 3 점: 4.0 <= 지표 < 0.6 4 점: 6.0 <= 지표 < 0.8 5 점: 0.8 <= 지표 <= 1.0	전체 접근 방법의 종류 개수 = 5 1. 게시판 운영 2. 포럼 운영 3. 위키 운영 4. 인터넷 검색 시 첫 페이지 출력 5. 인터넷 사이트에서 정보 제공 외부에서 커뮤니티로 연락하거나 관련 정보를 얻을 수 있는 용이성 OSS 커뮤니티에 대해 전문 정보를 제공하는 인터넷 사이트로는 openhub.net, wikipedia.org 등이 있음
	성숙성	변수 = {기간, 버전 출시, 관리 체제, 평가 방법, 위원회 운영} 지표 = 충족하는 성숙지표의 종류 / 전체 성숙 지표의 종류 개수 1 점: 0.0 <= 지표 < 0.2 2 점: 2.0 <= 지표 < 0.4 3 점: 4.0 <= 지표 < 0.6 4 점: 6.0 <= 지표 < 0.8 5 점: 0.8 <= 지표 <= 1.0	전체 성숙 지표의 종류 개수 = 5 1. 최초 버전 출시 이후 3년 이상 지속적으로 신규 버전 출시 2. 최근 배포한 안정된 버전의 넘버가 1.0 이상 3. 관리 운영자(maintenance operator), 커미터(심의자), 개발자 등의 운영 체제 확립 4. 기여도 및 참여도에 따른 개발자의 등급 체제 확립 5. 이사회 운영 - 개인의 독단적 판단이 아닌 위원회에 의한 의사 결정 방식

6) 분석

서비스에 활용하고자 하는 공개소프트웨어를 대상으로 유용성을 평가하기 위하여 다양한 공개소프트웨어 속성을 분석할 수 있다. 공개소프트웨어는 커뮤니티에 의해서 생성되는 특징을 가지고 있기 때문에 일반적인 소프트웨어의 기능적 분석만 하는 것이 아니라, 공개소프트웨어 커뮤니티에 대한 활동성, 로드맵, 영속성 등의 요소를 포함하여 분석해야 한다.

본 가이드의 전체 활동요소에서 보편속성에 대한 예시로 공개소프트웨어의 기능성, 신뢰성, 사용성, 유지보수성, 커뮤니티 등 분석에 필수적인 속성을 제시하고 있으므로 이를 기반으로 사용자의 서비스에 적합한 공개소프트웨어를 분석할 수 있다.

7) 평가

사용자는 서비스에 적합한 공개소프트웨어가 무엇인지를 선정하기 위하여 서비스를 위해서 어떤 공개소프트웨어 속성이 필요한지 선정하고 이 평가항목을 기준으로 비교 가능한 정량적 평가결과를 도출해야 한다.

본 가이드의 전체 활동요소 중 평가활동에서는 공개소프트웨어 속성 별 평가방법을 제시하고 있으며 이를 활용하면 공개소프트웨어의 각 속성을 5점 척도로 정량화하여 비교가 가능하다.

외부서비스에 사용되는 공개소프트웨어의 커뮤니티라는 속성의 예를 들면 커뮤니티의 규모나 설립시기, 커뮤니티의 운영주체, 커뮤니티의 접근성, 커뮤니티의 성숙성 등을 다음과 같이 평가하여 정량적 비교가 가능하다.

8) 계약

외부서비스를 위하여 공개소프트웨어를 활용하는 사용자는 공개소프트웨어를 사용하면서 자연스럽게 발생하는 저작권자와 사용자 간에 체결되는 계약과, 외부의 기술지원 서비스를 제공하는 기업과의 계약에 대하여 고려해야 한다.

공개소프트웨어의 획득은 별도의 비용지불이 없어도 서비스에 사용할 수 있지만 공개소프트웨어를 사용하기 위한 의무사항이 있음을 유의해야 한다. 따라서 해당 공개소프트웨어가 요구하는 의무사항이 사용자가 제공하고자 하는 서비스에 적합한지 여부를 검토해야 한다.

그리고 사용자 내부적으로 공개소프트웨어에 대한 기술적 대응이 어려운 경우, 외부의 기술지원 전문기업과 계약을 통해 공개소프트웨어를 서비스에 적용할 수 있는데 이 경우에는

사용자가 원하는 공개소프트웨어의 기술지원서비스 수준을 명확하게 약정해야 한다. 공개 소프트웨어를 기반으로 서비스를 운영하는 도중에 발생할 수 있는 버그, 에러, 장애 등에 대한 문제가 발생하면 장애를 해결하는 최대기간을 명시하고 이를 위반하면 손실에 대한 보상규정을 계약상에 포함해야 한다.

공개소프트웨어 기술지원 서비스 계약에서 포함할 수 있는 다양한 서비스 항목은 본 가이드의 전체 활동요소 중 계약부분을 참고할 수 있다.

9) 설계

공개소프트웨어는 외부에서 획득되는 경우이기 때문에, 서비스 컴포넌트의 어떤 영역에서 공개소프트웨어가 활용되는지 식별하고, 직접개발 영역과 외부획득 영역을 구분하여 상세 설계를 수행해야 한다.

대부분의 외부서비스 사용자는 서비스 제공을 위하여 공개소프트웨어를 그대로 사용하지 않고 맞춤형 개작을 하게 되는데 안정적인 외부 서비스를 제공하기 위해서는 향후 버전 관리와 업데이트에 문제가 없는 범위에서 맞춤형 설계를 진행하여야 한다.

공개소프트웨어는 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 분기 과정 등이 비공개소프트웨어의 관리와 다르게 비즈니스에서 매우 중요한 의미를 가지게 된다. 때문에 공개소프트웨어 프로젝트의 자원자들이 자사의 소프트웨어에 대한 쉬운 접근이 가능하도록 서비스 컴포넌트들의 결합도를 낮추고 향후 기능들의 수정이 용이하도록 아키텍처에 대한 관리가 필요하다.

10) 개발

공개소프트웨어는 커뮤니티에서 주도하고 있기 때문에, 사용자 스스로 공개소프트웨어를 개발하는 것이 불가능하다. 사용자가 서비스에 필요한 공개소프트웨어를 개발하기 위해서는 공개소프트웨어 커뮤니티에 참여해야 한다.

또한 외부 공개소프트웨어 커뮤니티에서 주도하는 기술이 요소기술이 되고 기업 고유의 소프트웨어가 기능적 차별화 또는 개선된 기능을 제공하는 경우라면, 해당 공개소프트웨어 커뮤니티에 기업 고유의 소프트웨어를 기여하는 방식으로 참여를 하는 것이 기술적, 전략적으로 훌륭한 선택이 될 수 있다. 즉, 기존 공개소프트웨어 커뮤니티의 기술적 기반, 활성화, 주요 개발자, 인지도 등 다양한 장점을 그대로 살리면서, 해당 공개소프트웨어의 장점을 훌륭히 부각할 수 있기 때문이다.

공개소프트웨어는 대부분 메인라인(mainline)을 바탕으로 개발 및 릴리즈를 운영하며, 메인라인을 기준으로 branch, fork등을 하여서 추가적인 개발을 한다. 이러한 공개소프트웨어의 메인라인을 기반으로 개발하는 것을 업스트림(upstream) 기반의 개발이라고 한다. 리눅스 커널을 비롯하여, 일반적인 SCM(Source Control Management)의 방법을 사용하고 있는 공개소프트웨어는 대부분 메인라인을 바탕으로 개발하고 있다.

사용자가 공개소프트웨어를 내부에 보유하지 않고 서비스를 제공하는 경우에는 직접 공개소프트웨어를 개발하지 않기 때문에 별도의 활동이 필요하지 않다.

11) 패키징

사용자가 직접 개발한 소스코드와 외부에서 획득한 공개소프트웨어의 결합과정에서 서비스 제공자는 성능향상 또는 안정성 등의 이유로 별도의 설치 파라미터 변경 또는 패치를 추가하게 된다.

공개소프트웨어는 소스코드를 공개해 주는 장점이 있지만 반면에 설치 및 업데이트 등의 사용 및 관리 편리성이 매우 약한 편이기 때문에 향후 서비스의 빠른 업데이트를 위해서는 패키징을 잘 하는 것이 필요하다.

공개소프트웨어는 소프트웨어의 설치, 업데이트, 삭제 등을 손쉽게 할 수 있도록 다양한 패키지 관리방법(YUM, APT, dpkg 등)을 제공하고 있으므로 이를 활용하면 사용자의 서비스에 적합한 패키징이 가능하다.

12) 시험

공개소프트웨어를 활용한 서비스를 제공하기 위해서는 반드시 설계나 구현 단계에서 정의된 요구사항들을 만족 하는지, 예상한대로 동작되는지, 일관성 있게 실행이 되는지 그리고 이해관계당사자의 요구를 만족하는지를 확인하고 검증해야 한다.

이 활동은 공개소프트웨어를 위한 별도의 시험이 아니라 외부서비스를 위하여 시험하는 것이기 때문에 일반적인 소프트웨어 공학의 시험방법 및 절차를 적용할 수 있다.

내부에서 개발하는 소프트웨어의 경우 대부분 시험 프로세스는 코딩이 완료된 후에 수행을 하는 것이 일반적이나, 외부에서 공개소프트웨어를 획득하게 되는 경우는 단 시간에 서비스에 적용하기 위하여 언제든지 시험을 수행할 수 있는 준비가 되어야 한다. 테스트 자동화에 대한 보다 상세한 내용은 본 가이드에서 제공하기에는 방대한 내용이므로 별도의 자료를 참고하기 바란다.

13) 설치

공개소프트웨어는 윈도우나 맥의 소프트웨어 설치와 다르게 다양한 설치방법을 제공한다. 특정 기업에서 제공하는 설치 방식이 아니라 사용자에게 소스코드와 다양한 설치법을 함께 제공하는 공개소프트웨어 특성으로 인하여 사용자들이 불편함을 호소하였으며 그로인해 공개소프트웨어 개발자들은 사용자 편의성 향상에 많은 노력을 해왔으며 현재는 다수의 공개소프트웨어들이 쉬운 설치 방법들을 지원하고 있다.

외부서비스를 위하여 외부에서 가져온 공개소프트웨어를 설치하는 과정은 다음과 같은 방법을 사용할 수 있다. 각 방법에 대한 세부적인 설명은 본 가이드의 전체 활동요소 중 설치 부분을 참고하기 바란다.

- 소스코드 설치(컴파일)
- 바이너리 설치
- 패키지 매니저를 통한 패키지 설치
- 원격 저장소를 이용한 패키지 설치

14) 운영

외부서비스를 위한 공개소프트웨어의 운영은 상용 제품과 특별히 다를 바가 없다. 운영 업무의 수행에 있어서는 외부 인력도 가능하지만 일반적으로 운영에 대한 책임은 직접 운영하고 있는 사용 주체가 지게 된다. 그래서 운영상의 문제로 인하여 정상적인 가동이 어려울 경우에 외부 업체와의 유지보수 계약을 체결해서 이러한 문제를 해결하고 기술적인 도움을 받게 되는 것이다.

공개소프트웨어 관리에 관련된 전문성과 기술지식을 보유한 운영 조직을 구성하고, 운영 조직은 공개소프트웨어 기술을 설계, 개발, 전환, 운영, 개선하기 위한 자원이 적절히 제공되고 효과적으로 훈련되는지 확인한다.

사용자가 공개소프트웨어를 내부에 보유하지 않고 고객의 시스템을 대상으로 운영을 위임 받아 서비스로 제공하는 경우에는 공개소프트웨어 시스템의 안정적인 운영을 위한 별도의 서비스 카탈로그를 구비하여 고객의 시스템을 운영해야 한다.

15) 유지보수

외부의 공개소프트웨어를 활용하여 외부서비스를 제공하는 경우의 유지보수 활동은 일반적인 시스템 유지보수 활동과 다르지 않다. 고객에게 제공하는 서비스를 최적의 운영 상태로 유지하기 위해서는 모니터링 및 업데이트, 패치 및 버그 수정, 보안 취약점 개선, 환경설정 변경 등의 활동을 수행해야 한다.

사용자는 내부에서 유지보수를 수행할 수도 있고 외부의 전문기업과 계약을 통해 유지관리를 위임할 수도 있는데 외부의 전문기업과 유지관리계약을 체결하는 경우 내부에서 유지보수를 수행하는 경우보다 안정적인 서비스를 제공할 수 있다.

핵심서비스에 공개소프트웨어가 적용되었으면 고객 대상 서비스 수준이 가장 중요하므로 전문 공개소프트웨어 기술업체를 통해 유지 보수 서비스를 받는 것이 좋다.

사용자가 공개소프트웨어를 내부에 보유하지 않고 외부 고객에게 유지보수 서비스를 제공하는 경우 제공되는 유지보수 서비스의 수준을 제시하고 계약을 통하여 서비스 수준에 따른 서비스 비용체계 및 서비스 수준유지 실패에 대한 보상방안을 사전 협의하고 준수해야 한다.

16) 기술지원

외부서비스를 제공하는 사용자는 외부고객의 서비스 만족도가 중요하기 때문에 내부적으로 공개소프트웨어에 대한 기술지원을 수행하는 것보다는 별도의 전문기업과 기술지원에 대한 계약을 체결하는 것이 좋다.

공개소프트웨어에 대한 기술지원기업의 목록은 공개소프트웨어 역량프라자에서 제공하고 있으므로 이를 참고하여, 자신의 서비스에 사용되는 공개소프트웨어 전문기업을 선택하여 기술지원 서비스를 체결하여 안정적인 외부서비스를 제공하는 것이 중요하다.

사용자가 공개소프트웨어를 내부에 보유하지 않고 외부 고객에게 기술지원 서비스를 제공하는 경우도 제공되는 기술지원 서비스의 수준을 제시하고 계약을 통하여 서비스 수준에 따른 비용체계 및 각종 제약사항을 사전에 협의해야 한다.

17) 커뮤니티

공개소프트웨어를 활용하여 외부서비스를 제공할 때 가장 먼저 생각할 수 있는 큰 목적중의 하나는 바로 외부 리소스의 적극적인 활용이다.

우수한 공개소프트웨어 개발자의 역량을 자신의 서비스에 활용하기 위해서는 공개소프트웨어 커뮤니티의 참여를 기반으로 어떻게 외부 개발자들과 소통하고 협업할 것인가, 어떻게 같이 성장할 것인가에 대해서 심각하게 고민하고 운용할 필요가 있다. 사용자는 해당 공개소프트웨어 커뮤니티에 참여하여 자신의 서비스에 사용되는 공개소프트웨어의 향후 로드맵, 활동성 등을 토대로 자신의 서비스의 미래방향을 결정할 수 있다

사용자가 공개소프트웨어를 내부에 보유하지 않고 외부 고객에게 서비스를 제공하는 경우에도 제공하는 서비스의 품질을 유지하기 위해서 공개소프트웨어 커뮤니티와의 소통은 중요하다.

반드시 개발자가 아니라도 커뮤니티에 참여하는 참여자는 개발자, 관리자, 사용자등 다양하기 때문에 사용자는 해당 공개소프트웨어 커뮤니티에 참여하여 지속적인 소통을 하는 것이 좋다.

18) 컴플라이언스

외부서비스에 사용되는 공개소프트웨어는 라이선스 의무사항이 존재하기 때문에 사용자가 공개소프트웨어에 라이선스에 대한 인식 없이 외부서비스에 사용하게 되면 법적 분쟁의 발생, 기업의 이미지 하락 등의 문제가 발생할 수 있다.

공개소프트웨어도 저작권이 있으며 라이선스별로 사용과 배포 등에 관련된 다양한 의무사항을 요구하고 있으므로 공개소프트웨어 전환을 선택함에 있어서 라이선스의 특징과 의무사항을 다음의 항목별로 면밀히 검토해야 한다.

- **사용권 고지의 의무**
 - 누가 해당 소프트웨어를 개발하였는지 공지함
 - 고객에게 어떤 공개소프트웨어를 사용하였는지 알림
- **저작권 고지의 의무**
 - 소스코드 상에 이미 표시되어 있는 저작권 관련 문구는 절대로 삭제 하여서는 안 됨
- **소스코드 공개의 의무**
 - 일부 공개소프트웨어 라이선스는 공개소프트웨어로 개발한 결과물의 소스코드 공개를 의무화 하고 있음
- **특허 포기 의무**
 - 사용자가 해당 공개소프트웨어를 만든 당사자에게 특허 침해 소송을 제기할 수 없음

보다 자세한 내용은 공개소프트웨어 역량프라자에서 발간한 “공개소프트웨어 라이선스 가이드”를 참고하기 바란다.

19) 교육

외부서비스를 위하여 공개소프트웨어를 활용하는 사용자는 사용자 조직 내부에서 소프트웨어 개발에 직접 참여하는 개발자 및 관리자 그룹과 공개소프트웨어 정책 프로세스에 포함되는 법률 자문 그룹, 마케팅 및 세일즈 그룹 등을 대상으로 교육을 수행할 수 있다.

교육을 통하여 소프트웨어 개발자들에게는 기술적인 면에서 지식과 스킬을 내재화시키는 효과가 있으며, 관리자들에게는 공개소프트웨어라는 이질적인 시스템 도입에 따른 변화관리의 효과를 기대할 수 있다.

사용자가 공개소프트웨어를 내부에 보유하지 않고 외부 고객에게 교육 서비스를 제공하는


경우에는 교육 교재, 강의, 자격증 등을 통해서 수익을 창출하게 된다. 이 경우 해당 공개소프트웨어 커뮤니티에 참여하여 최신의 공개소프트웨어 동향을 반영한 교육이 중요하다.

20) 모니터링

외부의 공개소프트웨어를 외부서비스에 사용하는 사용자는 향후 안정적인 운영과 유지 보수를 보장하기 위해서 해당 공개소프트웨어 커뮤니티의 동향, 진행 현황 및 주요 이슈 등을 주기적으로 추적하고 검토하는 활동이 필요하다.

사용자가 공개소프트웨어를 내부에 보유하지 않고 외부 고객에게 서비스를 제공하는 경우에도 마찬가지로 주기적인 공개소프트웨어 커뮤니티의 활동을 추적하고 검토해야 한다.

이때 공개소프트웨어의 모니터링을 위하여 가장 좋은 방법은 해당 공개소프트웨어 커뮤니티에 참여하여 활동하는 것이다. 대부분의 공개소프트웨어 커뮤니티는 메일링 리스트, 포럼 등을 통하여 소통하기 때문에 커뮤니티에 참여하게 되면 관련정보를 쉽게 확인할 수 있다.



04장



결론

제4장 결론



Open Source Software Governance Framework
and Its Applying Guide

공개소프트웨어는 시스템 개발에 있어 비용 효율성 측면 뿐 아니라 시스템 간 상호 운용성 확보 및 안전성 측면에서도 상용소프트웨어와 비교할 수 있을 정도로 성장하였다. 조직의 관점에서 이러한 공개소프트웨어를 사용하는 것은 이제는 대세라고 볼 수 있으며 국내에서의 공개소프트웨어 사용·적용 시장도 지속적으로 성장하고 있어 그에 따른 국내기업의 요구도 다양하게 증가하고 있다.

본 가이드에서는 공개소프트웨어를 안전하고 효과적으로 사용하기 위해 필요한 내용을 프레임워크로 구성하였다. 따라서 공개소프트웨어를 안전하게 사용·적용 및 배포하기 위해 필요한 사항을 다양한 관점에서 활용할 수 있도록 제시한 공개소프트웨어의 거버넌스 프레임워크는 많은 수요가 있으리라 생각한다.

공개소프트웨어를 활용하는 기업의 비즈니스 유형이 다양하게 존재하기 때문에 본 가이드에서는 내부사용, 외부배포, 외부서비스로 구분하여 제공하였지만, 본 가이드에서 제시하는 거버넌스 활동요소의 내용이 공개소프트웨어를 활용하고자 하는 기업에게 실질적인 도움이 되기 위해서는 많은 사례를 중심으로 활용 케이스를 확보하려는 노력이 필요하다. 공개소프트웨어를 배포하는 회사 및 활용하는 회사에서 안전하게 사용할 수 있도록 공개소프트웨어 테스트 및 인증에 대한 가이드가 각 단계에 세부적으로 기술될 수 있으며, 기업 간 공개소프트웨어 라이선스 정보를 주고받을 수 있는 표준화 등이 포함될 수 있도록 각각의 거버넌스 활동요소 내용들에 대한 세분화 및 확장이 필요하다.

소프트웨어 개발업체 및 공개소프트웨어 기술지원 업체, 공공기관의 시스템 담당자 및 공개소프트웨어를 활용하여 소프트웨어를 개발한 업체 등에서 상기 거버넌스 프레임워크를 사용하여 공개소프트웨어 거버넌스 체계를 구축하고 운영할 수 있다.

본 가이드가 국내 기업들의 공개소프트웨어 거버넌스 체계 구축에 기여하고 나아가 글로벌 경쟁력 강화의 발판이 될 수 있기를 기대한다.

첨부 자료

1. 공개소프트웨어 분류체계

메인카테고리	서브 카테고리(레벨1)	서브 카테고리(레벨2)
서비스 접근 및 전달	접근 채널 (Access Channel)	데스크톱 환경 웹브라우저 이미지 툴 원격 제어 멀티미디어
	협업 (Collaboration)	메일 오피스 메신저 재버 웹 미팅
	서비스 전송 (Service Transport)	SMTP POP/IMAP DNS DHCP LDAP FTP
컴포넌트	프로그래밍 (Programming)	Script Execution Environment
	패키지 (Package)	CMS CRM SNS EC ERP Wiki Forum/BBS Online Learning Groupware Portal ECM Survey System GIS ILMS
	데이터 관리 (Data Management)	DWH ETL OLAP

메인카테고리	서브 카테고리(레벨1)	서브 카테고리(레벨2)
		KDD Full-Text Search Digital Repositories
	데이터 교환 (Data Exchange)	Archiver
	데이터 표현 (Data Presentation)	Reporting
보안	응용 보안 (Application Security)	Anti-Virus Contents Filtering Anti-Spam Mail
	시스템/네트워크보안 (System/Network Security)	Firewall IDS Packet Analysis
	Encryption/Decryption	Encryption
	Certification	SSO Access Control
인터페이스 및 통합	인터페이스(Interface)	IP-PBX
	통합 (Integration)	BPM ESB Web Service SOA Cooperation with Windows
플랫폼 및 기반구조	데이터베이스 (Database)	DBMS DBMS Management Tool DBMS Development Tool
	운영체제 (Operating system)	Linux BSD UNIX 기타
	서비스 전송 서버 (Service Delivery Server)	Web Proxy Java Platform, Enterprise Edition Server
	시스템 관리 (System Management)	Integrated Monitoring 소프트웨어 Monitoring Network Monitoring Log Monitoring BackUp

메인카테고리	서브 카테고리(레벨1)	서브 카테고리(레벨2)
		Log Management Configuration Management Management Tool Service Desk
	소프트웨어공학 (소프트웨어 Engineering)	Ajax RIA (Framework)Java Platform, Enterprise Edition O/R Mapping Ruby Framework IDE Diagnostic Tools BTS RCS Forge Unit Test Test Coverage Integration Test Load Test Performance Analyzer Other Java Platform, Enterprise Edition Server
	기반구조 (Infrastructure)	Virtualization Infrastructure Virtualization Management Tool Cloud Infrastructure Cloud Management Tool Cloud API BI Distributed File System Distributed Memory Caching System Cluster Messaging Layer Replication FileSystem

2. 속성 별 정량화 공식과 적용 방법

속성	변환 공식	적용 방법
이해성	변수 = {제품 기술 자료} 지표 = 제공하는 기술 문서 종류 / 전체 기술 문서의 종류 1 점: 0.0 (≦ 지표 < 0.2) 2 점: 2.0 (≦ 지표 < 0.4) 3 점: 4.0 (≦ 지표 < 0.6) 4 점: 6.0 (≦ 지표 < 0.8) 5 점: 0.8 (≦ 지표 ≦ 1.0)	전체 기술 문서의 종류 = 5 1. 매뉴얼: 사용자, 기술자, 관리자 2. 데이터 시트: 테스트 결과 3. 교육 자료: 이론, 실습 4. 기술 보고서: 논문, 백서, 제안서 5. 가이드: 프로그램 샘플, 패턴
학습성	변수 = {학습 지원 서비스} 지표 = 제공하는 기술 서비스의 종류 / 전체 기술 서비스의 종류 1 점: 0.0 (≦ 지표 < 0.2) 2 점: 2.0 (≦ 지표 < 0.4) 3 점: 4.0 (≦ 지표 < 0.6) 4 점: 6.0 (≦ 지표 < 0.8) 5 점: 0.8 (≦ 지표 ≦ 1.0)	전체 기술 서비스의 종류 = 5 1. 컨설팅 서비스 2. 교육 서비스 3. 인증서 발급 서비스 4. 콜 센터 지원 서비스 5. 원격 지원 서비스
운용성	변수 = {운용 지원 유틸리티} 지표 = 제공하는 유틸리티의 종류 / 전체 유틸리티의 종류 1 점: 0.0 (≦ 지표 < 0.2) 2 점: 2.0 (≦ 지표 < 0.4) 3 점: 4.0 (≦ 지표 < 0.6) 4 점: 6.0 (≦ 지표 < 0.8) 5 점: 0.8 (≦ 지표 ≦ 1.0)	전체 유틸리티의 종류 = 5 1. 웹 기반 관리 콘솔 2. 서버 설정 3. 디플로이 4. 환경 설정 5. 모니터링
분석성	변수 = {이메일, 게시판, 이슈 트래킹, FAQ, Q&A 위키, 블로그} 지표 = 제공하는 분석 지원 종류 / 전체 분석 지원 종류의 수 1 점: 0.0 (≦ 지표 < 0.2) 2 점: 2.0 (≦ 지표 < 0.4) 3 점: 4.0 (≦ 지표 < 0.6) 4 점: 6.0 (≦ 지표 < 0.8) 5 점: 0.8 (≦ 지표 ≦ 1.0)	전체 분석 지원 종류의 수 = 5 1. 이-메일 연락 2. 게시판 운영 3. 이슈 트래킹 지원 4. FAQ 또는 Q&A 제공 5. 위키 또는 블로그 운영
전문기술	변수 = {지원 방식} 1 점: 기술 지원 불가 2 점: 커뮤니티 기술 지원 3 점: 해외 업체가 직접 지원 4 점: 국내 업체를 통한 간접 지원 5 점: 국내 업체의 직접 지원	상위 점수는 하위 점수의 배점기준을 충족 시킬 수 있음 4점: 국내업체는 지사 형태로 해외업체의 기술 지원을 연결함
시험성	변수 = {버전 종류, 테스트} 1 점: 품질 측정 불가 2 점: 안정화된 버전이 1.0 미만임 3 점: 안정화된 버전만 배포 4 점: 안정화된 버전 이전에 1개 이상의 테스트 버전 배포 5 점: 안정화된 테스트 수준에 따라 복수개의 다양한 버전 배포	릴리즈 버전이 1.0 미만인 경우에는 일반적 으로 상용 적용이 어려움 테스트 버전으로는 Alpha, Beta, Gamma 등이 있음

속성	변환 공식	적용 방법
관리체계	변수 = {로드맵, 커미터, 프로세스, 테스트, 운영자} 지표 = 진행하는 관리활동 종류의 수 / 전체 관리활동 종류의 수 1 점: 0.0 (= 지표 < 0.2) 2 점: 2.0 (= 지표 < 0.4) 3 점: 4.0 (= 지표 < 0.6) 4 점: 6.0 (= 지표 < 0.8) 5 점: 0.8 (= 지표 = 1)	전체 관리활동 종류의 수 = 5 1. 로드맵 발표 2. 복수 커미터 활동 (단독이 아님) 3. 심사 프로세스 운영 4. 품질 테스트 수행 5. 복수 유지 운영자(maintenance operator) 활동
대체성	변수 = {대체 기간} 1 점: 4주일 (= 대체 기간 2 점: 3주일 (= 대체 기간 < 4주일 3 점: 2주일 (= 대체 기간 < 3주일 4 점: 1주일 (= 대체 기간 < 2주일 5 점: 0주일 (= 대체 기간 < 1주일)	오픈 소스에서 오픈 소스로 전환하는데 필요한 시간으로 측정
대체 후 기능성	변수 = {표준 기능} 지표 = 제공 표준 기능 / 표준 기능 수 1 점: 지표 <= 60% 2 점: 60% < 지표 <= 70% 3 점: 70% < 지표 <= 80% 4 점: 80% < 지표 <= 90% 5 점: 90% < 지표 <= 100%	대체 전 오픈 소스의 표준 기능을 수행할 수 있는 수준으로 측정
설치성	변수 = {설치 파라미터} 1 점: 이기종 플랫폼으로 설치 불가 2 점: 일부 리눅스 기반 플랫폼 3 점: 모든 리눅스 기반 플랫폼 4 점: 모든 리눅스 및 2개 이하 상용 OS 기반 플랫폼 5 점: 모든 리눅스 및 대부분의 상용 플랫폼	플랫폼 전환의 용이성 OS 변경에 따른 환경 변수 조작에 대한 용이성 상용 플랫폼의 운영 체제로는 Unix, Windows, Mac OS 등이 있음
지원성	변수 = {툴, 패치, 가이드, 모니터, 알람, 콘솔} 지표 = 제공하는 지원 종류의 수 / 전체 지원 종류의 수 1 점: 0.0 (= 지표 < 0.2) 2 점: 2.0 (= 지표 < 0.4) 3 점: 4.0 (= 지표 < 0.6) 4 점: 6.0 (= 지표 < 0.8) 5 점: 0.8 (= 지표 <= 1.0)	전체 지원 종류의 수 = 5 1. 별도의 설치 툴 제공 2. 패치 및 가이드의 다운로드 제공 3. 제품 모니터링 툴 제공 4. 장애 알람 제공 (SMS, Push e-Mail) 5. 웹 콘솔 제공
상호 운용성	변수 = {호환성} 1 점: 단일 리눅스에서만 이식 가능 2 점: 일부 리눅스에서 이식 가능 3 점: 모든 리눅스에서 이식 가능 4 점: 모든 리눅스 및 2개 이하 상용 OS에서 이식 가능 5 점: 모든 리눅스 및 대부분의 상용 OS에서 이식 가능	설치성과는 달리 인터페이스가 있는 프로그램과의 연동성 및 최적의 운용 상태 유지를 위한 파라미터 설정 용이성 등을 측정함

속성	변환 공식	적용 방법
나이 및 규모	변수 = {버전 번호, 연령} 지표 = 최종 버전 번호 x 나이 1 점: 0 (<= 지표 < 12) 2 점: 12 (<= 지표 < 24) 3 점: 24 (<= 지표 < 72) 4 점: 72 (<= 지표 < 180) 5 점: 180 (<= 지표)	지표는 최종 버전 번호와 월 단위의 커뮤니티 나이를 곱해서 산출함 버전 번호가 1.0 이상이고 커뮤니티 나이도 12개월 이상이 되어야 자생력이 있는 커뮤니티로 인정함 버전이 3.0 이상이고 연수가 50이상이면 최상위 수준으로 인정함
주체	변수 = {후원 단체 유형} 1 점: 지원 없음 2 점: 하나의 중소기업 지원 3 점: 복수의 중소기업 지원 4 점: 하나의 대기업의 지원 5 점: 복수의 대기업의 지원	인력 및 자금을 대한 후원 단체의 유무로 측정함
접근성	변수 = {게시판, 포럼, 위키, 검색성, 인터넷} 지표 = 제공하는 접근 방법의 종류 / 전체 접근 방법의 종류 개수 1 점: 0.0 (<= 지표 < 0.2) 2 점: 2.0 (<= 지표 < 0.4) 3 점: 4.0 (<= 지표 < 0.6) 4 점: 6.0 (<= 지표 < 0.8) 5 점: 0.8 (<= 지표 <= 1.0)	전체 접근 방법의 종류 개수 = 5 1. 게시판 운영 2. 포럼 운영 3. 위키 운영 4. 인터넷 검색 시 첫 페이지 출력 5. 인터넷 사이트에서 정보 제공 외부에서 커뮤니티로 연락하거나 관련 정보를 얻을 수 있는 용이성 오픈 소스 커뮤니티에 대해 전문 정보를 제공하는 인터넷 사이트로는 openhub.net, wikipedia.org 등이 있음
성숙성	변수 = {기간, 버전 출시, 관리 체제, 평가 방법, 위원회 운영} 지표 = 충족하는 성숙지표의 종류 / 전체 성숙 지표의 종류 개수 1 점: 0.0 (<= 지표 < 0.2) 2 점: 2.0 (= 지표 < 0.4) 3 점: 4.0 (<= 지표 < 0.6) 4 점: 6.0 (<= 지표 < 0.8) 5 점: 0.8 (<= 지표 <= 1.0)	전체 성숙 지표의 종류 개수 = 5 1. 최초 버전 출시 이후 3년 이상 지속적으로 신규 버전 출시 2. 최근 배포한 안정된 버전의 넘버가 1.0 이상 3. 관리 운영자(maintenance operator), 커미터(심의자), 개발자 등의 운영 체제 확립 4. 기여도 및 참여도에 따른 개발자의 등급 체제 확립 5. 이사회 운영 - 개인의 독단적 판단이 아닌 위원회에 의한 의사 결정 방식
기능 적합성	변수 = {표준기능, 기능 별 가중치} 기능 적합성 = $SUM(\text{표준 기능} \times \text{표준 기능 가중치})$	표준 기능은 오픈 소스가 속하는 소분류 카테고리 안에서 표준적으로 제공하는 기능임 가중치는 평가자에 의해 결정됨
가용성	변수 = {연간 가동률} 지표 = 연간 실행 시간 / 1년 1점: server - 99% 이하 application - 90% 이하 2점: server - 99.9% 이하 application - 94% 이하 3점: server - 99.99% 이하	연간 가동률은 제품의 추천 사양 환경에서 운영이 중단되지 않고 실행되는 수준임 일반적으로 1년 동안 초단위의 가용한 시간 비율이 사용됨

속성	변환 공식	적용 방법
	application - 96% 이하 4점: server - 99.999% 이하 application - 98% 이하 5점: server - 100 % application - 100%	
회복성	변수 = {회복기간} 1점: 1일 < 회복기간 2점: 1시간 < 회복기간 (<= 1일) 3점: 10분 < 회복기간 (<= 1시간) 4점: 1분 < 회복기간 (<= 10 분) 5점: 0초 < 회복기간 (<= 1분)	1일: 참을 수 없음 1시간: 참을 수 있으나 불만이 큼 10분: 불만 없이 참을 수 있음 1분: 지연을 느끼지 못함
상용 대체 용이성	변수 = {대체기간} 1점: 12주 (<= 대체기간) 2점: 8주 (<= 대체기간 < 12주) 3점: 4주 (<= 대체기간 < 8 주) 4점: 2주 (<= 대체기간 < 4 주) 5점: 0주 (<= 대체기간 < 2 주)	상용 제품을 대체하는데 걸리는 시간 = 데이터 이전을 포함하여 테스트까지 완료 하는데 소요되는 주단위 기간임
대체 후 기능성	변수 = {확장기능, 기능 별 가중치} SUM(확장 기능 X 확장 기능 가중치)	확장기능 = 오픈 소스가 속하는 소분류 카테고리 안에서 표준 이외에 제공하는 기능 가중치는 평가자에 의해 결정됨
대체 방법론	변수 = {대체 방법론} 1점: 대체 방법론 없음 3점: 개념적 방법론만 있음 5점: 실용적 방법론도 있음	개념적 방법론: 유즈 케이스, 다이어그램, 흐름도, 문장 서술, 활동 단계(phase) 수준의 전환 방법론 실용적 방법론: Data type, parameter, procedure, API, 활동 단위(activity) 수준의 전환 방법론
라이선스 충돌	변수 = {라이선스 충돌 개수} 1점: 충돌 존재 5점: 충돌 부재	충돌: 배포하는 제품과 그 내부에 사용된 오픈 소스 간의 라이선스 의무사항이 불일치함
면책 서비스	변수 = {면책 서비스 조건} 1점: 면책 서비스 불가 5점: 면책 서비스 제공	면책 서비스: 사용자가 법적인 문제에 연루되지 않도록 공급자가 모든 책임을 지는 서비스
특허 침해성	변수 = {특허 침해 개수} 1점: 특허 침해 있음 5점: 특허 침해 없음	특허 소송 보복 조항이 있는 라이선스라 할지라도 제3자의 특허에 대해서는 침해가 발생한 경우 보호받을 수 없음
SLA 충실도	변수 = {기술 서비스 정책 충실도} 지표 = 제공하는 기술서비스의 종류 / 전체 기술 서비스의 종류 개수 1 점: 0.0 (<= 지표 < 0.2) 2 점: 2.0 (<= 지표 < 0.4) 3 점: 4.0 (<= 지표 < 0.6) 4 점: 6.0 (<= 지표 < 0.8) 5 점: 0.8 (= 지표 (= 1.0)	전체 기술 서비스의 종류 개수 = 5 1. 기술지원 서비스의 가격 정책이 수립되어 있음 2. 최초 지원 요청 후 응답 시간이 지정되어 있음 3. 상시 패치 및 업데이트 서비스가 지정되어 있음 4. 고객 창구가 개설되어 있음 5. 교육 또는 컨설팅 서비스가 제공되고 있음

속성	변환 공식	적용 방법
문제해결 능력 만족도	변수 = {해결기간} 1 점: 2주 <= 해결기간 2 점: 1주 <= 해결기간 < 2 주 3 점: 2일 <= 해결기간 < 1 주 4 점: 1일 <= 해결기간 < 2 일 5 점: 해결기간 < 1 일	2주 이상은 참을 수 없는 기간으로 설정함 당일 접수된 문제를 당일 해결할 수 있으면 최고 수준으로 설정함
기술보증	변수 = {보증 조건, 보상 조건} 1점: 제품 보증 및 보상 없음 3점: 제품 보증 제공 및 보상 없음 5점: 제품 보증 제공 및 보상 제시	오픈 소스 자체는 보증(warranty)가 없으나 사업적인 목적으로 문제 발생 시 즉각적인 해결 서비스를 제공함 제품 자체의 하자로 인하여 사용자의 손해 가 발생할 경우 보상을 약속할 수 있음
선호도	변수 = {점유율, 레퍼런스 개수} 1 점: 점유율 = 0%, 레퍼런스 0개 2 점: 점유율 < 2%, 레퍼런스 < 2개 3 점: 점유율 < 5%, 레퍼런스 < 5개 4 점: 점유율 < 8%, 레퍼런스 < 8개 5 점: 점유율 > 10%, 레퍼런스 > 10개	점유율은 상용을 포함한 시장에서 copy수 기준으로 산정함 레퍼런스는 상용 서비스를 제공하는 시스 템으로 사용되는 사이트 개수임

3. 공개소프트웨어 조직관리 수준조사 양식

(1) 응답자의 배경정보.

- 이 름 :
- 회사명 :
- 부서명 :
- 직 위 :
- 프로젝트 이름(선택) :
- 프로젝트 코드크기(선택) :
- 프로젝트 상세(선택) :
- 기 타 :
- 응답자의 주요 직무 :

- | | |
|------------------------------------|--------------------------------------|
| <input type="checkbox"/> 공개소프트웨어관리 | <input type="checkbox"/> 보안 |
| <input type="checkbox"/> 법무 | <input type="checkbox"/> 릴리즈 엔지니어링 |
| <input type="checkbox"/> 컴플라이언스 | <input type="checkbox"/> 도구와 프로세스 개발 |
| <input type="checkbox"/> 소프트웨어개발관리 | <input type="checkbox"/> 제품관리 |
| <input type="checkbox"/> 소프트웨어 설계 | <input type="checkbox"/> 공급망 관리 |
| <input type="checkbox"/> 소프트웨어 개발자 | <input type="checkbox"/> 사업관리 분야 |
| <input type="checkbox"/> QA | <input type="checkbox"/> 기타: _____ |

(2) 조직내 공개소프트웨어 활용실태 : 귀하가 인지하고 있는 범위 내에서 귀하의 조직이 공개소프트웨어를 어떻게 이용하고 있습니까? (해당 사항을 모두 선택)

내부/운영

- 운영 IT
- 데스크탑 제품
- 소프트웨어개발, 테스트, QA
- 하드웨어 개발
- 기타: _____

외부/고객에게 제공

- 서비스 제공
- 소프트웨어 제품 제공
- 하드웨어 제품 제공
- 기타: _____

- 사업 및 공개소프트웨어 사용 현황 파악을 위해 8가지 주요영역에 대한 질문 구성은 다음과 같다.

(1) 서드파티와 공개소프트웨어 발견

- 조직 내에 개발자들이 프로젝트에 포함하기 위한 서드파티 소프트웨어(인터넷에서 다운로드 받은 코드, 상용 공개소프트웨어, 독점 상용 소프트웨어 포함)를 어떻게 찾기 시작합니까? (해당 사항을 모두 선택)
- 조직내에 공식적인 가이드라인 혹은 프로세스가 준비되어 있지 않다.
- 약간의 가이드라인이 개발자들에게 제공된다.
- 각 개발조직들은 그들 자신의 가이드라인을 가지고 있다.
- 조직에서 상용 서드파티 공개소프트웨어 컴포넌트들을 위한 검증된 소스에 대한 문서화된 가이드라인을 제공한다.
- 조직에서 어떤 소스에서 직접 획득되어 질 수 있는 공개소프트웨어 유형에 대한 문서화된 가이드라인을 제공한다.
- 특정 서드파티/공개소프트웨어 컴포넌트들을 구체적으로 명시하는 업무를 담당하는 조직의 개발자들은 서드파티 소프트웨어 획득 정책에 대해 교육을 받는다.
- 우리의 개발자들은 서드파티 컴포넌트들의 속성을 검증하고 검색하는데 도움이 되는 도구들(내부적으로 혹은 서드파티/ 공개소프트웨어)을 가지고 있다.
- 우리 조직은 소프트웨어 컴포넌트들을 위한 표준을 제정하고 그들 속성을 공표하는 산업협회에 활발히 참여한다.
- 잘 모르겠다.
- 기타의견

(2) 서드파티/공개소프트웨어 컴포넌트 검토와 선택

- 귀하의 조직은 서드파티 컴포넌트들이 조직의 소프트웨어 개발 코드베이스에 포함되어 지기 전에 어떻게 평가하고 승인합니까? (해당 사항을 모두 선택)

	상용독점 소프트웨어	상용 공개 소프트웨어	인터넷에서 다운로드된 코드
• 우리의 개발자 혹은 개발팀들은 어떤 컴포넌트가 사용되어야 할지를 그들 자신이 결정한다.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• 약간의 검토와 선택 가이드라인이 개발자들에게 제공된다.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

- 각 개발 조직은 그들 자신의 가이드라인을 가지고 있다. □ □ □
- 우리는 수용 가능한 소스들과 속성들(공개소프트웨어 요소와 라이선스 포함)에 대한 문서화된 정책과 공식적인 승인 프로세스를 가지고 있다. □ □ □
- 우리는 내부 정책 요구사항에 부합하지 않는 어떤 서드파티 컴포넌트 요청에 대해 결정하고 감독하는 검토위원회 가지고 있다. □ □ □
- 우리는 서드파티 소프트웨어 요청, 승인과 기록 유지를 관리하기 위한 자동화된 프로세스를 가지고 있다. □ □ □
- 우리는 조직내 사용된 고빈도/고품질의 서드파티 소프트웨어를 관리하고 있고 해당 커뮤니티에 참여한다. □ □ □
- 잘 모르겠다 □ □ □
- 기타의견

(3) 공급망관리/공개소프트웨어 결합

– 귀하의 구매 조직은 공개소프트웨어가 소프트웨어 공급 산출물에 유입되는 것과 관련된 위험을 관리하게 위해 어떤 조치를 취하고 있습니까? (해당 사항을 모두 선택)

- | | 상용독점
소프트웨어 | 계약 개발자
코드 | 인터넷에서
다운로드된 코드 |
|---|---------------|--------------|-------------------|
| • 우리는 공개소프트웨어 내용과 관련하여 공급자의 특정 요구사항을 가지고 있지 않다. | □ | □ | □ |
| • 우리는 소프트웨어 공급자들이 그들의 산출물에 | □ | □ | □ |

결합된 공개소프트웨어 코드를 보고하는 것을 요구한다.

- 우리는 우리의 소프트웨어 공급자에게 위반 혹은 기타 라이선싱 클레임에서 우리를 보호해 줄 보증과 법적책임을 제공하는 것을 요구한다.
- 우리는 모든 유입된 소프트웨어가 우리 자신의 라이선스와 우리가 사용하는 기타 소프트웨어 들과 호환되는지를 검증한다.
- 우리는 소프트웨어 공급자들에게 코드 분석 보고서를 요청하거나 소프트웨어 보고서에 명시된 서드파티의 정확성을 검증하기 위해 자체적으로 코드를 분석한다.
- 우리는 소프트웨어 공급자들에게 공급하는 코드에 유입된 모든 서드파티 소프트웨어에 대한 라이선스 원문 제공을 요구한다.
- 우리는 소프트웨어 공급자들에게 공급하는 코드에 유입된 모든 서드파티 소프트웨어에 관련된 라이선스 준법 요구사항들에 대한 리스트 작성을 요구한다.
- 우리는 공급자의 공개소프트웨어 거버넌스 프로그램을 검토하고 위험에 따라 공급자를 평가한다.
- 우리는 소프트웨어 공급자들에게 공급하는 코드에 유입된 모든 서드파티 소프트웨어에 관련된 라이선스 준법 요구사항을 확인하는 자동화된 프로세스를 가지고 있다.
- 우리는 조직내부에 사용된 고빈도/고품질의 공개 소프트웨어를 관리하고 있고 해당 커뮤니티의 참

여속에 역량있는 공급자와의 관계를 생성한다.

• 잘 모르겠다

• 기타의견

(4) 공개소프트웨어 코드관리

- 귀하의 조직은 귀하의 내부 혹은 고객에게 전달되는 소프트웨어에 포함되어져 온 서드파티 컴포넌트들에 있는 공개소프트웨어를 어떻게 관리하고 파악하고 있습니까?

(해당 사항을 모두 선택)

- 우리는 서드파티/공개소프트웨어를 다르게 취급하지 않는다; 공개소프트웨어는 독점 소프트웨어와 동일한 방식으로 관리, 유입 및 포함된다.
- 서드파티/공개소프트웨어 코드 컴포넌트들은 독점코드와 동일한 방식으로 관리되지만, 개별적으로 해당 코드 컴포넌트들을 추적한다.
- 우리는 완전한 컴포넌트들 뿐만 아니라 관련된 공개소프트웨어 코드 요소들(부분 일치와 기타 지점 유입된 코드)을 추적한다.
- 우리의 정책은 각 공개소프트웨어 컴포넌트에 대한 소유자와 해당 소유자의 코드 관리 책임을 명확히 한다.
- 우리는 특정 저장소에 있는 코드 요소들과 모든 서드파티 공개소프트웨어 컴포넌트들의 원본과 수정된 복사본을 유지 관리한다.
- 이미 승인된 서드파티/공개소프트웨어 컴포넌트들의 재사용은 권장되고 각각의 신규 사용을 추적하고 검토한다.
- 우리는 우리의 소프트웨어에 포함된 각각의 서드파티/공개소프트웨어 컴포넌트를 위한 라이선스 준법성 요구사항들과 사용, 속성, 소스를 추적하는 자동화된 프로세스를 가지고 있다.
- 우리의 저장소와 관리 프로세스는 서드파티 벤더들 혹은 공개소프트웨어 커뮤니티에 버그수정(조직에 의해 개발된 기타 코드와)의 외부 릴리즈를 위한 지원을 제공한다.
- 잘 모르겠다.
- 기타의견

(5) 공개소프트웨어 유지보수와 지원

- 귀하의 조직은 조직의 내부 혹은 고객에게 전달하는 소프트웨어 기반에 포함되어 온 서드파티/공개소프트웨어 컴포넌트들을 어떻게 유지보수 하는가? 그리고 귀하의 개발자들과 지원 엔지니어들은 이들 컴포넌트들을 위해 어떻게 기술지원을 습득하는가? (해당 사항을 모두 선택)
- 우리의 개발자들은 그들이 결정한대로 한다.
- 소프트웨어가 설치 및 사용되면, 우리는 어떠한 유지보수도 하지 않는다.
- 우리는 우리의 소프트웨어 기반에 사용하는 서드파티 컴포넌트들의 신규 릴리즈와 버그수정, 보안취약점 등에 대처하기 위한 정형화된 처리방법을 가지고 있다.
- 우리의 정책은 신규 릴리즈와 버그 수정, 보안취약점 등에 대처하기 위해 각각의 책임을 정의하고 컴포넌트를 위한 담당자를 명확히 하고 있다.
- 각 서드파티/공개소프트웨어 컴포넌트를 위한 지원 전략은 사용 승인시 식별된다.
- 지원은 조직내에 모든 컴포넌트의 사용을 위해 통합된다.
- 우리는 각 서드파티/공개소프트웨어 컴포넌트들을 위한 호환 요구사항과 버전, 버그수정,이슈 등을 추적하는 자동화된 프로세스를 가지고 있다.
- 우리는 고객 및 외부 커뮤니티 지원을 위해 각 서드파티/공개소프트웨어 컴포넌트들을 위한 호환 요구사항과 버전, 버그수정,이슈 등을 추적하는 자동화된 프로세스를 가지고 있다.
- 잘 모르겠다.
- 기타의견

(6) 라이선스 준법성 프로그램

- 귀하의 조직은 귀하의 내부 혹은 고객에게 전달되는 소프트웨어 기반에 포함된 서드파티/공개소프트웨어 준법성을 어떻게 확인합니까? 공개소프트웨어가 상용 라이선스 패키지에 결합되어 있을 때, 상용 라이선스 준수가 결합된 공개소프트웨어 라이선스 준수를 보장하기에 충분하지 않을 수 있습니다. (해당 사항을 모두 선택)
- 정형화된 프로세스가 준비되어 있지 않다..
- 우리는 조직의 소프트웨어 기반에 포함된 서드파티/공개소프트웨어 컴포넌트들을 파악하고 있다.
- 우리는 각각의 릴리즈 혹은 출시 검사를 통해 라이선스 준법성 요구사항들의 목록을 취합하고 있다.
- 우리의 공개소프트웨어 선택과 코드관리 프로세스들은 릴리즈 혹은 출시 시점에

민감한 준법성 결함을 예방함에 있어서 지금까지 효과적이다.

- 우리는 각각의 승인된 컴포넌트들을 위한 라이선스 준법성 요구사항들의 목록을 유지관리하고 조직원들은 각 릴리즈/출시와 관련하여 특정 준법성 요구사항 목록을 생성하기 위해 이 목록을 사용한다.
- 우리는 릴리즈 빌드 시스템 혹은 QA시스템이 모든 서드파티 컴포넌트들을 검증하고 식별하는 부분으로서 자동화된 감사 기능을 개발했다.
- 우리는 보고와 추적기능을 가진 공식적인 준법성 프로세스를 수행해 왔다.
- 우리는 검토, 코드관리, 검사 그리고 준법성 기능을 통합하는 자동화된 프로세스를 개발했다.
- 우리의 준법성 시스템은 고객들을 위해 그리고 내부관리를 위해서 결합된 소프트웨어와 라이선스 및 준법성 요구사항들에 대한 자동화된 보고서를 생성한다.
- 우리의 정책은 커뮤니티 기여를 위한 준법성 요구사항들에 대한 자동화된 프로세스를 포함하고 있다.
- 잘 모르겠다.
- 기타의견

(7) 공개소프트웨어 커뮤니티 상호작용

- 귀하의 조직은 귀하의 핵심사업과 밀접하게 관련되거나 조직의 소프트웨어 기반에 포함되어서 온 컴포넌트들을 생성하는 공개소프트웨어 커뮤니티와 어떻게 상호작용합니까? (해당 사항을 모두 선택)
 - 우리는 커뮤니티 사이트에서 코드를 다운로드 한다.
 - 우리의 개발자들은 버그수정과 업데이트에 보조를 맞추어 사용하고 있고 서드파티 컴포넌트들에 포함된 공개소프트웨어를 추적한다.
 - 우리의 개발자들은 지원을 받고 문제를 요청하기 위해 커뮤니티포럼에 참여하지만, 내부 사항에 대한 공개는 허용되지 않는다.
 - 우리의 개발자들은 조직의 연관성을 확인하고 지원을 획득하고 질문을 요청하기 위해 커뮤니티포럼에 참여한다.
 - 우리는 제공된 컴포넌트들에 대한 버그 수정과 코드관리 그리고 공개소프트웨어 커뮤니티들과의 상호작용과 추적을 위해 서비스 공급자와(혹은) 상용 비공개공개 소프트웨어에 의존한다.
 - 우리의 개발자들은 내부적으로 혹은 고객에게 전달되는 소프트웨어에 사용된 컴포넌트들에 대한 버그수정을 우리에게 공급했던 벤더들에게 다시 공헌한다.

- 우리의 개발자들은 우리가 사용하는 컴포넌트들에 대한 버그 수정을 원래 커뮤니티에 다시 공헌한다.
- 우리는 신규 컴포넌트 혹은 프로젝트들을 기존 혹은 새롭게 설립된 공개소프트웨어 커뮤니티들에 공헌한다.
- 우리는 개발자 자원들, 시설 혹은 재정지원을 통해 조직 전략에 중요한 공개소프트웨어 커뮤니티들에 후원자로서 활동한다.
- 잘 모르겠다.
- 기타의견

(8) 의사결정자 감독

- 적절한 감독기능을 제공하기 위해 사업관리 부문, 엔지니어링 관리와 법무에서는 서드파티 소프트웨어(독점상용 소프트웨어, 상용 공개소프트웨어, 인터넷에서 다운로드된 코드를 포함)관리에 어떻게 참여합니까? (해당 사항을 모두 선택)
- 우리의 의사결정자들은 단지 지휘계통을 통해 참여하고 공개소프트웨어와 관련된 특정 역할을 가지고 있지 않다.
- 우리의 법무부서는 내부적 그리고 고객에게 제공되는 서드파티 컴포넌트들에 포함된 각각의 공개소프트웨어 라이선스를 승인해야만 한다.
- 책임있는 의사결정권자들(전형적으로 엔지니어링, QA, 법무와 사업관리)은 정기적으로 소프트웨어에 포함된 공개소프트웨어 보고서를 받는다.
- 우리는 어떤 공개소프트웨어 예외 승인과 프로세스, 정책을 위한 책임이 있는 검토 위원회(전형적으로 엔지니어링, QA, 법무와 유관부서들)를 수립하고 있다.
- 우리의 의사결정자들은 외부 공개소프트웨어 커뮤니티 활동들에 참여하기 위해 문서화된 정책을 수립하고 있다.
- 우리의 의사결정자들은 버그 수정의 공헌을 위한 승인 프로세스에 참여한다.
- 우리의 의사결정자들은 승인 프로세스에 참여하고 신규 프로젝트 혹은 컴포넌트들을 공헌하기 위해 문서화된 정책과 프로세스를 수립하고 있다.
- 잘 모르겠다.
- 기타의견

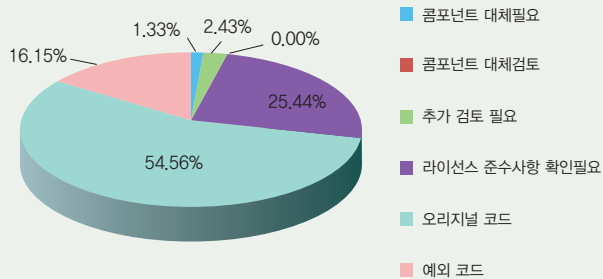
4. 공개소프트웨어 검증 권장 개선조치 보고서 샘플

(1) 권장 개선조치 요약

본 권장 개선조치는 다음과 같은 가정을 전제로 하고 있습니다 : 본 솔루션은 일반적인 상용 독점 라이선스에 의해 고객 및 파트너들에게 배포됩니다. 즉, 본 프로젝트는 상용, 독점 권한을 가지고 공급(판매 및 출시)하는 응용프로그램입니다. 본 프로젝트 분석 및 상기 배포 시나리오에 근거해서 발견된 공개소프트웨어 컴포넌트들에 대해 다음과 같은 개선사항을 권장합니다.

- 3개의 컴포넌트들은 대체가 필요합니다. (우선순위 1)
- 0개의 컴포넌트들은 라이선스 변경 혹은 대체가 필요할 수 있습니다. (우선순위 2)
- 3개의 컴포넌트는 거의 대체가 필요 없는 것으로 판단되지만, 상황에 따른 고려사항과 검토, 추가적인 조사가 필요합니다.(우선순위 3)
- 4개의 컴포넌트는 대체할 필요는 없지만, 해당 라이선스들은 약간의 개발 요구사항에 반영해야 할 의무사항을 가지고 있습니다. 해당 라이선스 의무사항 준수에 대해 적절히 검토하기를 권장합니다. (우선순위 4)

우선순위에 따른 컴포넌트들



총 합

해당 프로젝트에는 컴포넌트 대체가 필요한 1.33%의 공개소프트웨어 코드가 있으며, 추가검토가 필요한 공개소프트웨어가 2.43%, 라이선스 준수 확인이 필요한 공개소프트웨어가 25.44%, 공개소프트웨어지만 공개소프트웨어 라이선스 적용이 필요하지 않은 예외 공개소프트웨어 코드가 16.15% 포함되어 있습니다.

권장 개선조치에 대한 상세 정보는 권장 개선조치 세부사항을 참조하시기 바랍니다.
 코드 분석을 통해 확인된 공개소프트웨어 컴포넌트들과 해당 라이선스들의 의무사항과 결합 형태를 파악하여 준법성 준수를 위한 문제해결을 위해 다음과 같은 각 컴포넌트별 라이선스 현황에 따른 권장개선 조치 세부사항을 제시합니다. 권장개선 조치 세부사항의 각 범주는 컴포넌트 대체(or 코드 삭제 및 재개발)가 필요한 우선순위 1, 라이선스 변경 혹은 컴포넌트 대체(or 코드삭제 및 재개발)가 필요할 수 있는 우선순위2, 컴포넌트 대체는 필요 없지만 추가적인 검토가 필요한 우선순위 3, 해당 라이선스 의무사항 준수여부에 대한 확인이 필요한 우선순위 4로 구성되어 집니다.

우선순위 1: 컴포넌트 대체 필요

해당 공개소프트웨어 컴포넌트들은 제품 및 서비스에 포함되고 배포되는 방식에 따라 상용 라이선스와 호환되지 않는 라이선스들로 구성되어 있습니다. 일반적으로 해당 공개소프트웨어 컴포넌트들과 라이선스들의 코드를 상용, 독점 라이선스 코드와 결합사용 하게 되면 상용, 독점 코드의 일부 혹은 전체를 공개하고 해당 라이선스 의무사항에 따라 공개소프트웨어 코드를 포함한 전체 코드를 공개해야 합니다.

컴포넌트	사용형태	라이선스	의견
embedded-network-w5200	Snippet (+ File)	GPL 2.0	해당 컴포넌트는 ARM CortexM3를 위한 W5200 driver입니다. http://code.google.com/p/embedded-network-w5200/ 조치사항 : 칩배포사에 문의 및 사용검토가 필요합니다.
FreeRTOSRealTimeKernel	Snippet	GPL 2.0	해당 컴포넌트는 32개의 각각 마이크로컨트롤러 코어와 14개의 다른 개발 도구 체인을 지원하는 크로스플랫폼 표준 RTOS입니다. http://sourceforge.net/projects/freertos/ 조치사항 : 개작 사용되고 있는 만큼 수정 및 삭제가 요구됩니다.
LinuxKernel	Snippet	GPL 2.0 Only	해당 컴포넌트는 리눅스 커널입니다. http://www.kernel.org/ 조치사항 : 개작 사용되고 있는 만큼 수정 및 삭제가 요구됩니다.

우선순위 2: 컴포넌트 대체 검토

해당 공개소프트웨어 컴포넌트들은 결합방법 및 배포방법에 따라 상용라이선스와 호환되거나 호환되지 않을 수 있습니다. 따라서 해당 컴포넌트들의 사용(usage), 연결(linkage), 배포(distribution)에 대한 검토가 필요하며 해당 라이선스와 양립할 수 없다면 반드시 대체되어야 합니다.

컴포넌트	사용형태	라이선스	의견
			해당사항 없음

우선순위 3: 추가 검토 필요

해당 공개소프트웨어 컴포넌트들은 상용라이선스와 호환되는 컴포넌트들로 예상됩니다. 하지만 해당 라이선스 정보가 불명확하거나 추가 검토가 필요한 컴포넌트들이 있습니다. 해당 컴포넌트들에 대한 출처 및 사용결합 형태 등에 대한 추가 제약조건에 대한 조사가 필요합니다.

컴포넌트	사용형태	라이선스	의견
ATMELlicense	Component	Commercial License	상용라이선스입니다. http://www.atmel.com/About/legal.aspx 조치사항 : 해당 기업에 로열티 발생여부 등 사용권한에 대한 문의가 필요합니다.
FatFsModule	Component	FatFs License	해당 라이선스는 BSD-style 라이선스의 하나이지만, 임베디드 프로젝트를 위한 것이기 때문에 바이너리형태의 재배포의 조건은 사용성을 높이기 위해 명시되지 않습니다. http://elm-chan.org/fsw/ff/en/appnote.html#license 조치사항 : 해당 라이선스는 GNU GPL과 호환되기 때문에 사용자에 따라 재배포시에 GPL로 배포가 가능합니다. 향후 사용에 있어서 참조하시기 바랍니다.
MicroC/OS-2	Snippet (+ Component)	Commercial License	상용라이선스입니다. http://micrium.com/buy/licensing/ 조치사항 : 해당 기업에 로열티 발생여부 등 사용권한에 대한 문의가 필요합니다.

우선순위 4: 라이선스 준수사항 확인 필요

해당 공개소프트웨어 컴포넌트들은 일반적인 상용라이선스와 호환되는 컴포넌트들입니다. 하지만, 해당 컴포넌트들은 다양한 의무사항이 있습니다. 일반적인 의무사항은 저작권 유지, 라이선스 원문 배포, 고지의무 등입니다(참조3. 상용라이선스 의무사항).

컴포넌트	사용버전	라이선스
embox	Unspecified	BSD 2.0
SanosOperatingSystemKernel	20090611	BSD 2.0
avropendous	Unspecified	MIT License V2
micropendoux	Unspecified	MIT License V2

참 조

사용형태

- Original : 해당 파일들은 공개소프트웨어가 아니라 사용자가 직접 개발한 코드로 구성되어 있습니다.
- Snippet : 해당 파일들은 해당 공개소프트웨어 컴포넌트로 배포된 코드를 일부 수정하여 재사용된 코드로 구성되어 있습니다.
- File : 해당 파일들은 해당 공개소프트웨어 컴포넌트로 배포된 코드를 100% 복제하여 File 결합형태로 재사용된 코드로 구성되어 있습니다.
- Library : 해당 파일들은 해당 공개소프트웨어 컴포넌트로 배포된 코드를 100% 복제하여 Library 결합형태로 재사용된 코드로 구성되어 있습니다.
- Separate Work : 해당 파일들은 해당 공개소프트웨어 컴포넌트로 배포된 코드를 100% 복제하여 Separate Work 결합 형태로 재사용된 코드로 구성되어 있습니다.
- Module : 해당 파일들은 해당 공개소프트웨어 컴포넌트로 배포된 코드를 100% 복제하여 Module 결합형태로 재사용된 코드로 구성되어 있습니다.

(2) 사용 라이선스 비율

컴포넌트	라이선스	사용형태	파일수	%
최승연_VMS_20130124	[template] Basic Proprietary Commercial License	Original Code	247	54.65%
embox	BSD 2.0	Snippet	13	2.88%
Sanos Operating System Kernel	BSD 2.0	Snippet	1	0.22%
embedded-network- w5200	GPL 2.0	Snippet (+ File)	1	0.22%
FreeRTOSRealTimeKernel	GPL 2.0	Snippet	1	0.22%
LinuxKernel	GPL 2.0 Only	Snippet	3	0.66%
avropendous	MIT License V2	Snippet (+ File)	50	11.06%
micropendousx	MIT License V2	Snippet (+ File)	51	11.28%
akiduki-drunk	Unspecified	Snippet	3	0.66%
beerbug	Unspecified	Snippet	13	2.88%
CommonDriverInterface	Unspecified	Snippet (+ File)	15	3.32%
Original_General_Use	Unspecified	Component	1	0.22%
Original_Miss_Match- Define	Unspecified	Component	37	8.19%
Original_Not_Source_ Code-Text	Unspecified	Component	3	0.66%
tinynos-2.x-contrib	Unspecified	Snippet	1	0.22%
ATMELlicense	Commercial License	Component	3	0.66%
FatFsModule	FatFs	Component	5	1.11%
MicroC/OS-2	Commercial License	Snippet (+ Component)	3	0.66%
합계			452	100%

(3) 사용 라이선스 의무사항

No.	라이선스 의무사항	Proprietary Commercial License	GPL 2.0	LGPL 2.1	MPL 1.1	Public Domain	BSD 2.0	MIT 2.0	Borland JBuilder License as applied to Redistribut ables
1	배포권리(오브젝트/바이너리코드 배포)	X	X	X	X	X	X	X	X
2	코드배포에 의해서만 부여되는 의무사항)	X	◎	◎	◎	◎	◎	X	◎
3	소스코드배포/강제적 공유 의무사항	X	◎	◎	◎	X	X	X	X
4	복제 권한 허용	X	◎	◎	X	X	X	X	X
5	수정(개작)권한 허용	X	◎	◎	X	X	X	X	X
6	역설계 권한 허용	X	◎	◎	X	X	X	X	X
7	차별적 제한 금지 ◎ or 차별적 제한 있음 ●	X	◎	◎	◎	X	X	X	X
8	추가 복제에 대한 로열티 및 수수료 금지	X	◎	◎	◎	X	X	X	X
9	특허보복(특허소송 제기시 라이선스 종료)	X	X	X	◎	X	X	X	X
10	명시적 특허라이선스(특허소송을 제기하지 않음)	X	X	X	◎	X	X	X	X
11	DRM금지	X	X	X	X	X	X	X	X
12	고지(특정법률 혹은 속성)	X	◎	◎	X	X	X	X	X
13	변경사항 고지	X	◎	◎	◎	X	X	X	X
14	변경사항에 대해 원저작자에게 사용허가	X	X	X	◎	X	X	X	X
15	다른사람을 대신한 보증의 부인	◎	◎	◎	◎	X	X	◎	X
16	다른사람의 책임의 제한	◎	◎	◎	◎	X	X	◎	X
17	배포/사용으로 인해 발생한 원저작자의 클레임에 대한 배상	X	X	X	◎	X	X	X	◎
18	배포시 라이선스 사본 포함	X	◎	◎	◎	X	◎	◎	◎
19	광고/홍보시 배포자,저작자, 특정상표사용 금지	X	X	X	◎	X	◎	X	◎
20	원코드와 동일조건 허가	X	◎	◎	◎	X	X	X	◎
21	라이선스 확장 범위(공개범위)	X	코드 기반의 산출물 (perGPL)	동적라 이브리리 (per LGPL)	파일 (per MPL)	X	X	X	X
결합사용된 라이선스별 권장개선 조치 우선순위			1	1,2	2	3	4	4	4

(4) 검증 확인된 파일현황

1. Project License : [template] Basic Proprietary Commercial License												
2. Review												
Identified Files												
License Conflict	Discovery Type	File/Folder	Total Lines	Component	Version	License	Usage	%	Matched File	Matched File Line	Comment	Developer Comment
Component License Conflict	Code Match	/Gapyeong BIT Source/firm Main.ddp		Borland Auto-generated Code	Unspecified	Borland JBuilder License as applied to Redistributables	Component	100%	borland-arch-5.tar /borland/ Reports Preview.ddp			
Component License Conflict	Code Match	/Gapyeong BIT Source/firm Setting.ddp		Borland Auto-generated Code	Unspecified	Borland JBuilder License as applied to Redistributables	Component	100%	borland-arch-5.tar /borland/ Reports Preview.ddp			
Declared License Conflict and Component License Conflict	Code Match	/Gapyeong BIT Source/firm Window Ctrl.cpp	26	Shareaza	Unspecified	GPL 2.0	Snippet	8%	borland-arch-5.tar /borland/ Reports Preview.ddp		프로젝트 컴파일시 자음중 생성코드중 최대화 최소 등업선들을 통해 개작사 용(RES 폴더)	
Declared License Conflict and Component License Conflict	Code Match	/Gapyeong BIT Source/firm Window Ctrl.cpp	43	MLDonkey	Unspecified	GPL 2.0	Snippet	7%	borland-arch-5.tar /borland/ Reports Preview.ddp		프로젝트 컴파일시 자음중 생성코드중 최대화 최소 등업선들을 통해 개작사 용(RES 폴더)	

5. Linux Foundation에서 국제표준으로 권장하는 SPDX 보고서

Spreadsheet Version	Version SPDX	Creator	Created	Data License	Creator Comment										
1.1.0	SPDX-1.1	Person: Gary O'Neill Organization: Source Auditor Inc. Tool: SourceAuditor-V1.2	2010-02-03 0:00	CC0-1.0	This is an example of an SPDX spreadsheet format										
Package Name	Package Version	Package File Name	Package Supplier	Package Originator	Package Download Location	Package Checksum	Package Verification Code	Verification Code Excluded Files	Source Info	License Declared	License Concluded	License Info From Files	License Comments	Package Copyright Text	Package Summary
SPDX Translator 0.9.2		spdxtranslator-1.0.zip	Organization: Linux Foundation	Organization: SPDX	http://www.spdx.org/tools	2fd4e1c67a2d28fce0849ee1bb76e7391b931e6a	4e3211c67a2d28fce0849ee1bb76e7391b931e6a	SpdxTranslator, SpdxTdf, SpdxTranslator, SpdxTdf	Version 1.0 of the SPDX Translator application	(LicenseRef-3 AND LicenseRef-4 AND Apache LicenseRef-1 AND MPL-1.1 AND LicenseRef-1 AND LicenseRef-2)	(LicenseRef-3 AND LicenseRef-4 AND Apache LicenseRef-1 AND MPL-1.1 AND LicenseRef-1 AND LicenseRef-2)	Apache-1.0, Apache-2.0, LicenseRef-1, LicenseRef-3, LicenseRef-4, MPL-1.1	The declared license information can be found in the NOTICE file at the root of the archive	Copyright 2010, 2011 Source Auditor Inc.	SPDX Translator utility
File Name	File Type	File Checksum	License Concluded	License Info in File	License Comments	File Copyright Text	Artifact of Project	Artifact of Homepage	Artifact of URL	File Comment	User Defined Columns...				
src/org/spdx/parser/DOAPP/Project.java	SOURCE	2fd4e1c67a2d28fce0849ee1bb76e7391b931e6a12	Apache-2.0	Apache-2.0	This license is used by Jena	Copyright 2010, 2011 Source Auditor Inc.	Jena	http://www.openjena.org/	UNKNOWN	This file belongs to Jena					
Jena-2.6.3/jena-2.6.3-sources.jar	ARCHIVE	3apb4e1c67a2d28fce0849ee1bb76e7391b931e6a125	LicenseRef-1	LicenseRef-1		(c) Copyright 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009 Hewlett-Packard Development Company, LP									
Reviewer												Reviewer Comment			
Person: Joe Reviewer		2010-02-10 0:00		This is just an example. Some of the non-standard licenses look like they are actually BSD 3 clause licenses											
Person: Suzanne Reviewer		2011-03-13 0:00		Another example reviewer.											

6. 용어집

용어	설명
저작권	<p>저작권(copyright)은 문학·학술 또는 예술의 범위에 속하는 창작물(저작물)의 창작에 의하여 그 창작물에 대하여 창작자(저작자)가 취득하는 권리로서 창작물의 아이디어를 보호하는 것이 아니라 그 표현(expression)의 결과물을 보호하는 것이다. 저작권은 권리의 발생에 있어 등록과 같은 요건이 필요하지 않고 창작과 동시에 권리가 발생한다(무방식주의). 따라서 어떤 프로그래머가 특정 SW를 개발하게 되면 컴퓨터 프로그램 저작권이 자동적으로 발생하며, 그 권리는 프로그래머 또는 그가 속한 회사에 부여된다. 저작권이 있는 저작물의 경우 누구도 원저작자나 저작권자의 허가가 없는 해당 저작물을 사용·복제·배포·수정할 수 없다.</p>
특허권	<p>특허권(patent)은 발명에 관하여 발생하는 독점적·배타적 지배권으로 법에 정해진 절차에 의해 출원을 하여야 하며, 심사를 통해 부여되는 권리이다. 특허기술을 사용하기 위해서는 반드시 특허권자의 허락을 얻어야만 한다. 특허권자는 자신이 허락하지 않은 사람이 해당 특허를 활용한 제품을 만들거나, 사용하거나, 판매하는 것을 막을 수 있다. 특허는 무엇인가 유용하게 하는 방식(method)이므로 특허받은 방식을 구현하는 SW라면 프로그래밍 언어가 다르거나 소스코드가 다르더라도 해당 특허권자의 명시적인 허락을 받아야 하며 이는 공개소프트웨어 뿐만 아니라 상용 SW에도 마찬가지이다.</p>
상표권	<p>상표권(trademark right)이란 상표권자가 지정상품에 관하여 그 등록상표를 사용할 독점적인 권리로서 일정한 절차에 따라 등록하여야 효력이 발생한다. 이러한 상표를 사용하기 위해서는 반드시 상표권자의 허락을 얻어야 하며 허락받지 않고 상표를 이용할 경우 처벌을 받게 된다. 상표권을 취득한 SW의 경우 상표를 사용하려면 상표권자의 명시적인 허락을 받아야 한다.</p>
영업비밀	<p>영업비밀(tradeseecret)이란 공언히 알려져 있지 아니하고 독립된 경제적 가치를 지니는 것으로서 상당한 노력에 의하여 비밀로 유지되는 생산방법, 판매방법, 기타 영업활동에 유용한 비밀 유지 의무가 있음에도 다른 사람에게 이를 누출하는 경우 처벌받게 된다. 공개되지 않은 SW의 경우 영업비밀로서 보호를 받을 수 있으며, 공개된 SW라 하더라도 아이디어에 대한 부분은 영업비밀로 보호를 받을 수 있는 가능성이 있다. 단, 영업비밀로서의 SW보호는 널리 공개되어 유통되는 경우에는 보호받기가 어렵고, 이를 알지 못하고 사용한 제3자에게 법적으로 문제를 삼을 수 없는 한계가 있다.</p>
공개소프트웨어(OSS)	<p>저작권이 있으면서 소스코드가 공개되어 있는 소프트웨어를 말하며, 일반적으로 자유롭게 복제, 사용, 수정, 배포 할 수 있다. 공개소프트웨어의 대표적인 예로는 Linux 커널 및 관련 GNU SW, 아파치 웹서버, FireFox 웹브라우저, MySQL 데이터베이스시스템, Python/PHP/Perl 언어, Eclipse 툴 등이 있다.</p>

용 어	설 명
공개소프트웨어 라이선스	<p>오픈 소스 SW 개발자와 이용자간에 사용방법 및 조건의 범위를 명시한 계약을 말한다. 공개소프트웨어를 이용하기 위해서는 공개소프트웨어 개발자가 만들어놓은 사용방법 및 조건의 범위에 따라 해당 SW를 사용해야 하며, 이를 위반할 경우에는 라이선스를 위반함과 동시에 저작권 침해로 인해서 이에 대한 처벌을 받게 된다. 대표적인 라이선스로는 GPL, LGPL, BSD, MPL 등의 라이선스가 있으며, 이런 오픈 소스 SW 라이선스는 기본적으로 사용자의 자유로운 사용,수정,배포를 보장하고 있다.</p>
Adware	<p>프로그램의 기능이나 날짜상의 제한없이 무료로 사용하는 대신 해당 SW로 작업하는 동안 광고 창을 통해 지속적으로 새로운 광고가 노출되도록 한 마케팅 기법이 적용되어 제작된 소프트웨어.</p>
Freeware	<p>프리웨어는 실행파일의 형태로 무료로 배포되는 SW를 말한다. 일반적으로 프리웨어의 형태로 제공되는 경우에는 SW의 계속적인 단순사용과 재배포는 허용하지만 변경이나 2차 저작물 작성은 허용되지 않는다. 국내에서는 흔히 공개판이라고 부르기도 한다.</p>
Shareware	<p>실행파일의 형태로 무료 배포되지만 일정기간의 시험기간 이후 유상으로 전환되는 SW를 의미한다. 누구든지 임의대로 복제, 배포할 수 있다. 셰어웨어와 프리웨어는 무료로 자유롭게 이용할 수 있다는 측면에서 공개SW와 일면 유사한 점이 있지만, 셰어웨어나 프리웨어는 저작권자가 상업적 또는 개인적 이유에서 해당 SW를 이용할 수 있도록 배려하는데 불과하다는 점에서 공개소프트웨어의 공유정신과는 매우 상이하다.</p>
복제	<p>원래의 저작물을 재생하여 표현하는 모든 행위. 인쇄, 사진, 복사, 녹화 등이 해당 *프로그램을 유형물에 고정시켜 새로운 창작성을 더하지 아니하고 다시 제작하는 행위</p>
배포	<p>저작물의 원작품 또는 그 복제물을 일반공중에게 대가를 받거나 받지 아니하고 양도 또는 대여하는 행위이다. 또한, 저작물을 이용하는 방법이자 저작물을 시장에 유통시키는 방법으로 B2B, B2C, 제품형태의 유통 등을 포함한다.</p> <ul style="list-style-type: none"> - 직접적인 실행파일 또는 소스코드의 제공 - Active-X, CD, 인터넷 다운로드 등의 형태로 실행파일 또는 소스코드 제공 - 전송을 통한 저작물 이용제공 등
복사	<p>파일을 디스켓 따위의 다른 곳으로 옮김</p>
전송	<p>일반 공중이 개별적으로 선택한 시간과 장소에서 수신하거나 이용할 수 있도록 저작물을 무선 또는 유선통신의 방법에 의하여 송신(Send)하거나 이용에 제공(Upload)하는 것</p>

용 어	설 명
고지	소송법에서, 법원이 결정사항이나 명령을 당사자에게 알리는 일 *게시나 글을 통하여 알림
반환의무	공개소프트웨어를 사용하여 수정한 소스코드를 공개해야 하는 의무를 말함. 공개 의무와 동일한 의미.
공개	공개소프트웨어는 라이선스에 따라서 공개소프트웨어를 사용하여 수정한 부분이 있을 때 해당 부분의 소스코드를 공개하여야 한다고 명시하는 경우가 있으며, 공개의무는 공개소프트웨어를 배포함으로써 공개의 의무가 발생하고 공개범위는 각각의 라이선스에서 정하고 있는 범위에 따라 달라질 수 있다.
공개 대상	공개소프트웨어의 공개 대상은 라이선스 별로 차이가 있으며 일반적인 공개 대상자는 공개소프트웨어를 공급(배포)받은 자를 의미함 - 일반적으로 공개소프트웨어를 공급받지 않은 자에게 반드시 소스코드를 제공해야 하는 의무는 없음 - 어떤 라이선스(MPL 등)는 수정시 원저작자에게 반드시 공개하여야 함 - SW를 배포받지 않고 사용권만 허용된 일반인(기업)에게는 공개 의무 없음
공개 범위	공개소프트웨어 라이선스별 소스코드 공개범위는 상이하며 라이선스에 따라 원본 사용 소스와 수정(변경, 개작 포함) 소스, 관련 Link 소스 등에 대한 공개의무가 발생할 수 있음 - 소스코드 공개를 요구하는 라이선스는 GPL, LGPL, MPL, CPL, EPL, IBM, OSL, Qt등임 - 기타 공개의 의무가 있으나 범위가 명확하게 기술되어 있지 않는 라이선스는 정확한 공개 범위가 정해져 있지 않으므로 파일 단위 공개를 가정함
공개 방법	소스코드의 공개방법은 일반적으로 배포시에 소스코드를 함께 동봉하는 것이 원칙이나, 실행파일만 배포해도 가능함(단, 최저 3년 동안 별도비용 없이 소스코드를 신청할 수 있는 배포신청서 동봉하거나 소스코드를 Download받을 수 있는 웹사이트를 공지해야 함) - 매뉴얼에 소스코드를 요청할 수 있는 연락처를 기입하여 두거나, 혹은 FTP 서버, 웹서버 등에 소스코드를 업로드해 두고 매뉴얼에 해당 주소를 기입하기도 함
파일	컴퓨터에서 프로그램을 저장하는 최소 단위
모듈	잘 정의된 한 가지 일을 수행하는 프로그램의 논리적인 일부분. 주프로그램은 논리적으로 몇 개의 모듈로 나눌 수 있다. 모듈은 여러 프로그램 작성자에 의해 나뉘어 작성되는 성질을 지닌다.

용 어	설 명
특허무효조항	특허와 관련하여 라이선스에 관한 특허 소송이 제기되는 경우 소송을 제기한 날에 특허소송을 제기한 SW의 공개소프트웨어 라이선스는 종료됨
소스코드	컴퓨터 프로그램의 텍스트 부분임 - 프로그램은 소스코드로 구성됨
2차적 저작물	원래 소스코드를 수정하여 만든 프로그램. 공개소프트웨어 라이선스에는 프로그램 원저작물의 개작이나 이를 이용한 2차적 프로그램의 창작이 허용되어야 하며, 이러한 파생적 프로그램들은 최초의 프로그램이 갖고 있던 라이선스의 규정과 동일한 조건하에서 재배포될 수 있어야 한다.
FSF	Free Software Foundation, 자유로운 SW의 개발과 보급을 위해 리처드 스톨만이 1984년 설립한 비영리 민간단체다. 컴퓨터 프로그램의 복제, 배포, 개작의 자유와 이를 위한 소스코드의 사용에 대한 제한 철폐 등을 목적으로 하며 본부는 미국 보스턴에 있다.
GNU	Gnu is Not Unix. 'GNU는 유닉스가 아니다' 라는 뜻의 재귀적 약어이다.
라이선스	권리자가 다른 사람에게 일정한 내용을 조건으로 하여 특정 행위를 할 수 있는 권한을 부여하는 행위
저작권 관련 문구 유지	SW의 경우 소스코드 상단을 보면 프로그램 이름, 개발자 이름, 버전, 연락처, 라이선스명 등이 기록되어 있으며, 이러한 정보는 저작권에 의해 보호받기 때문에 수정 또는 배포하는 사람이 임의로 삭제하거나 수정하는 것을 금지하는 조항 잘 관리되는 공개소프트웨어들의 경우 거의 대부분 소스코드 상단에 개발자 정보와 연락처 등이 기록되어 있다
제품명 중복 방지	공개소프트웨어의 제품명은 상표의 의미이므로 동일 이름으로 다른 제품명 및 서비스 명에 사용하는 것을 금지하는 조항
사용여부 명시 (고지)	많은 공개소프트웨어 라이선스들은 소스코드를 자유롭게 열람하고 수정 및 재배포 할 수 있는 권리를 부여하는 한편, SW를 사용할 때 해당 공개소프트웨어가 사용되었음을 명시적으로 표기하는 것을 의무사항으로 채택하고 있다. 이것은 마치 논문을 쓸 때 인용을 하는 것과 비슷하여, 이 SW는 공개소프트웨어인 무엇무엇을 사용하였습니다. 라는 식으로 사용여부를 명확히 기술하라는 것이다.
서로 다른 라이선스의 조합	서로 다른 공개소프트웨어간의 라이선스 의무조항이 상충하는 문제에 따라 이를 해결할 목적으로 타 라이선스의 수용 여부를 명시하는 조항

용 어	설 명
양립성 (Compatibility)	<p>두 라이선스의 요구조건이 서로 달라서 배포하는 것이 불가능하게되는 문제로 함께 사용이 가능하면 compatible, 사용이 불가능하면 incompatible이라고 한다.</p> <p>예를 들어, SW를 작성하고자 할 경우 기존에 만들어진 코드를 재사용하거나 결합하는 경우가 많은데, 결합되는 각 코드의 라이선스가 상호 상충되는 경우가 있다.</p> <p>실질적으로 MPL 조건의 A코드와 GPL조건의 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.</p>
상호주의 (Reciprocal)	<p>Reciprocal 공개소프트웨어 라이선스는 공개소프트웨어를 자유롭게 사용했으면 동일하게 본인이 만든 소스코드도 공개소프트웨어로 내어 놓으라는 의미로 GPL 라이선스가 대표적인</p>
Permissive	<p>Permissive 공개소프트웨어 라이선스의 경우는 제한없이 마음대로 사용이 가능하고, 해당 소스코드를 이용하여 상업적인 용도의 사용도 가능하다는 것을 의미한다.</p>
Copyleft	<p>Copyleft는 SW, 문서, 음악, 예술 등 작업에 적용되는 라이선스로 특정한 작업의 복제나 재배포를 제한하는 Copyright와 상반되는 개념임.</p> <p>Copyright에 대한 언어적인 유희로서 SW의 소스코드를 공개하기만 하면 법적인 구속 없이 원본을 자유롭게 설치, 운영, 변경, 재배포 등이 가능한 라이선스임</p>
라이브러리	<p>라이브러리란 메인 프로그램과 링크(link)되어 실행되는 오브젝트(object) 코드를 의미함</p>

7. 공개소프트웨어 기반의 개발 사업을 위한 위탁계약서 양식

계 약 서

〈발주사〉(이하"갑"이라함)와 〈공급사〉(이하"을"이라함)는 "공개소프트웨어 기반의 ○○○○ 시스템"의 개발과 관련하여 다음과 같이 위탁개발계약(이하 "본계약"이라함)을 체결한다.

-다 음-

제1조(목적)

본계약은 "갑"이 을"에게" 공개소프트웨어 기반의 ○○○○ 시스템"의 개발을 의뢰하고 을"은 이를 수행함에 있어 "갑"과 "을"의 권리·의무 등 필요한 제반사항을 규정함을 그 목적으로 한다.

제2조(용어정의)

- ① "개발" 이라 함은 "갑"이 제공하는 제반 정보를 이용하여 을"이 행하는 "공개소프트웨어 기반의 ○○○○ 시스템"의 개발에 대한 연구, 분석, 실험과 데이터 처리, 소프트웨어 제작, 샘플 제작 및 성능 개선을 포함하는 일체의 연구 활동을 지칭한다.
- ② "지적재산권"이라 함은 본 계약을 수행함에 따라 발생하는 특허권, 실용신안권, 디자인권, 저작권, 프로그램 저작권 등을 말하며 블루 프린트, 레이아웃, 영업비밀, 노우하우, 정보에 관한 일체의 권리를 포함한다.
- ③ "개발결과물"이라 함은 을"이 본 계약에 따라 개발을 수행하여 얻은 소프트웨어, 하드웨어, 소스코드, 개발 관련 문서 등을 포함하는 일체의 결과물을 말한다.
- ④ "정보"라 함은 본 계약 이행을 위해 제공된 기술적 사실 데이터, 실험 자료 및 결과, 아이디어, 기술적 사양, 노우하우, 영업비밀, 보고서를 포함하는 일체의 정보 자료와 본 계약을 수행함에 따라 취득하거나 발생하는 개발 결과물의 기능과 기술적 사실, 디자인을 포함하는 일체의 기술 자료와 지적 재산권, 노우하우, 영업비밀, 아이디어, 실험자료 및 결과 보고서를 포함하는 일체의 정보자료를 말한다.

제3조(개발 범위 등)

본 계약상의 개발 범위, 목표 등의 상세 내역은 별도 문서로 정한다.

제4조(개발기간)

본 계약의 개발기간은 201X년 XX월 XX일부터 201X년 XX월 XX일까지로 한다.

단, 다음 각 호의 1에 해당하는 경우 위 개발기간을 조정할 수 있다.

1. "갑"의 요구에 의해 개발 사양을 변경하는 경우
2. "갑", "을"이 상호 인정하는 사유가 발생하는 경우

제5조(개발결과물 제출)

① "을"은, "갑"과 "을"이 협의한 개발일정에 따라 각 단계별 결과물을 "갑"이 지정하는 장소에 제출하여야 한다.

② 제4조의 개발기간 종료 후 "갑"과 "을"이 협의한 개발 일정에 따라 "을"이 "갑"에게 제출해야 할 개발결과물은 다음 각 호와 같다.

1. 개발 시 "갑"이 제공한 물품
2. 소프트웨어에 포함된 공개소프트웨어의 소스코드
3. 개발과 관련하여 "을"이 실시한 검사의 결과물 및 관련 서류
4. 공개소프트웨어의 사용명세서 (SPDX 규약 준수)

② "을"은 "갑"이 위 제1항의 개발결과물을 사용, 적용하는데 필요한 기술적 정보 및 "갑"이 그러한 필요가 있다고 판단하여 제공을 요청하는 기술적 정보도 함께 제출해야 한다.

제6조(권리와 의무)

① "갑"의 권리 및 의무

1. "갑"은 "을"에게, 개발목표를 달성하는 데 필요한 범위 내에서, 각종 개발 관련 자료의 제출을 요구할 수 있다.
2. "갑"은 "을"에게, "갑"이 제공하는 검사 절차서에 의거하여, "개발결과물"에 대한 각종 성능 시험을 요구할 수 있다.
3. "갑"은 "을"에게 본 계약 제8조의 개발비를 지급하여야 한다.
4. "갑"은, "을"이 개발을 진행하는 데 차질이 없도록, 개발 관련 자료 또는 정보를 제공하도록 한다.

② "을"의 권리 및 의무

1. "을"은 "갑"에게 본 계약 제8조의 개발비를 청구할 수 있다.

2. "을"은, "갑"이 본 계약 체결 과정에서 제시한 제반 기술수준, 개발방법, 관리지침 및 기타 요구사항에 따라 본 계약 개발을 수행하여야 한다.
3. "을"은 "갑"에게 개발업무 수행현황에 관하여 다음과 같은 보고를 해야 한다.
 - 가. 정기보고 : 매월 X일
 - 나. 수시보고 : "갑"의 요청이 있는 경우, 즉시
4. "을"은 본 계약 개발과 관련하여 관계당국, 언론 등과 접촉하는 등의 주요 대외 업무에 관하여 항상 "갑"과 사전 협의하여 제반 조치를 취하도록 한다.
5. "을"은, "갑"이 "개발결과물"을 제품에 적용하여 출시할 때까지 문제가 발생할 경우 이의 해결을 위하여 적극적으로 지원해야 한다.
6. "을"은, 본 계약 개발을 정상적으로 수행할 수 있는 능력을 가진 개발인력을 본 계약 개발업무에 투입하여야 하고, 해외출장 결정·동종업계 퇴직 후 X 년 미만 등의 결격사유 있는 자를 위 개발 인력에서 제외하여야 한다.

제7조(검수 등)

- ① "갑"은, "을"로부터 제5조의 개발결과물을 제출받으면 위 제출일로부터 XX일 이내에 개발 내용을 검수하여 그 결과를 통보 하여야 한다. 위 기간까지 아무런 통보가 없는 경우, 위 개발결과물이 검수에 합격된 것으로 본다.
- ② "갑"이 개발결과물을 위 제1항에 따라 검수한 결과 특별한 문제점이 없는 경우 이를 승인하고 서면 또는 전자문서를 통하여 "을"에게 통지함으로써 개발결과물을 인수한다.
- ③ "을"이 제출한 개발결과물이 "갑"이 요청한 사양과 차이가 나거나 승인 기준에 미달하는 경우, "갑"은 "을"에게 필요한 조치를 요구할 수 있다. 이 경우, "을"은 지체없이 "갑"이 요구한 조치를 취한 후 재검수를 받아야 한다.
- ④ "을"이 제출한 개발결과물이 "갑"이 요청한 사양과 현저한 차이가 있어 계약의 목적을 달성 할 수 없는 경우에는 "갑"은 본 계약을 해지 할 수 있다.

제8조(개발비 및 지급조건)

- ① "갑"은 "을"에게 본 계약 개발업무 수행에 대한 대가로 총 금 XX 의 개발비를 다음 각 호와 같이 대금지급기준 약정에 따라 현금 지급한다. 단, 위 개발비에는 개발결과물 인수 후 X 년간 품질보증, 유지·보수 비용이 포함되고, 규격인공과 관련된 직접 비용은 "갑"이 별도로 부담한다.
 1. 선 금 (XX %) : 금 XX 원
계약 체결 후 30일 이내에 지급

2. 중도금 (XX %) : 金 XX원

"을"의 중간 산출물 제출 및 "갑"의 승인 통지가 도달한 날로부터 XX 일 이내에 지급

3. 잔 금 (XX %) : 金 XX 원

"을"의 개발결과물 제출 및 "갑"의 승인 통지가 도달한 날로부터 XX 일 이내에 지급

- ② "갑"의 요청에 의하여 "을"이 해외에 출장하여 본 계약 개발업무를 수행한 경우, "갑"은 "을"에게 위 해외출장에 소요된 비용을 "해외출장 정산기준"에 따라 정산하여 지급한다.

제9조(개발결과물 귀속 등)

- ① 본 계약에 따른 개발 수행으로 발생한 개발결과물에 대한 소유권은 "갑"에게 귀속된다. 따라서, "갑"은 위 개발결과물을 사용, 수익, 처분할 수 있다.
- ② "갑"이 본 계약의 수행을 위해 "을"에게 제공한 모든 자료에 관한 권리는 "갑"에게 있다.
- ③ "을"은 위 제2항의 자료를 "갑"의 사전 서면 동의 없이 본 계약 개발수행 이외의 용도로 사용할 수 없다.

제10조(품질보증)

- ① "을"은 개발결과물에 관하여 제7조 제2항의 개발결과물 인수 후 X 년간 그 품질을 보증한다.
- ② "을"은 위 제1항의 기간 내에 개발결과물에 하자가 발생하거나 유지보수가 필요한 경우, "갑"의 요청에 따라 무상으로 이를 수행하여야 한다.

제11조(개발비의 조정)

- ① "갑"과 "을"은 개발업무 내용, 작업기간, 필요인력 등과 같이 개발비 산정의 기초가 되는 조건이 변경된 때에는 상대방에게 개발비의 조정을 요청할 수 있다.
- ② 위 제1항의 요청은 서면 또는 전자문서로 하고, 대금 조정이 필요한 사유를 기재하여야 한다.
- ③ 위 제1항의 요청이 있는 경우, "갑"과 "을"은 위 요청이 있는 날로부터 XX 일 이내에 대금 조정이 필요한 사유의 발생 여부 및 그 범위에 관하여 상호 협의하여 가격 조정에 관한 사항을 결정하기로 한다.

제12조(개발기간의 변경)

제4조의 개발기간 내에 본 계약 개발을 완료할 수 없을 경우, "을"은 즉시 그 사유 및 완료 일정을 구두 및 문서로 "갑"에게 통보하여야 한다. 이 경우, "갑"에 대한 통보로 인하여 개발일정 지체에 대한 "을"의 책임이 면제되는 것은 아니다.

제13조(지적재산권)

- ① 본 계약 개발업무 수행으로 발생되는 모든 지적재산권(개발 진행 중에 또는 개발 완료 후 X 년 이내에 본 계약 개발과 관련하여 출원되어질 지적재산권 포함)은 "갑"에게 귀속된다. 단, 본 계약 개발업무 수행 전부터 "을"이 이미 보유하고 있던 지적재산권 또는 공개소프트웨어의 저작권은 그러하지 아니하다.
- ② 본 계약 개발업무 수행으로 지적재산권이 발생된 경우, "을"은 지체 없이 "갑"에게 발생사실 및 그 내용을 통지하여야 한다.
- ③ 위 제1항 본문의 지적재산권은 "갑"의 명의로 출원 및 등록한다. 단, 출원일정 등의 사유로 부득이 우선 "을"의 명의로 출원할 경우에는 "을"은 사전에 "갑"의 합의를 얻어야 하며 출원 후 즉시 "갑"에게 무상 양도해야 한다.
- ④ 위 제3항 본문에 따른 지적재산권의 출원 및 등록에 소요되는 비용은 "갑"이 부담한다.
- ⑤ "을"은, 본 계약 개발업무 수행 전부터 "을"이 이미 보유하고 있던 지적재산권 중 본 계약 개발과 관련된 지적재산권으로 "갑"이 본 계약 개발결과를 실시하는 것을 제약하지 않는다.
- ⑥ "을"이 위 제5항의 지적재산권을 양도하고자 하는 경우, "갑"은 이에 대한 우선협상권을 가진다.

제14조(보증 및 면책)

- ① "을"은 본 계약에 의한 어떠한 결과도 제3자의 특허권, 실용신안권, 디자인권, 저작권, 프로그램저작권, 영업비밀을 포함하는 일체의 권리를 침해하지 않는다는 것을 보증한다. 만일 "을"이 제3자의 권리(공개소프트웨어 포함)를 사용하여야 할 경우에는 사전에 "갑"에게 서면으로 통보해야 하고, "을"은 위 제3자의 권리를 정당하게 취득하여 사용해야 하며 이 경우 그 제3자가 향후 "갑"에 대하여 아무런 문제 제기를 하지 않는다는 점에 관한 서면 확인을 받아 "갑"에게 제출해야 한다.
- ② "을"이 제출한 개발결과물에 관련되어 "갑"과 제3자 사이에 제3자의 특허권을 포함한 일체의 권리에 대해 분쟁이나 소송이 발생할 경우 "을"은 "을"의 비용과

책임으로 "갑"을 방어하여야 하며 이로 인해 "갑"에게 손해가 있을 경우 "갑"이 입은 손해를 배상하여야 한다. 단, "갑"은 위 분쟁이나 소송의 발생, 진행사항을 즉시 "을"에게 알려야 한다.

- ③ 다음 각 호의 경우에는 본 조 제2항이 "을"에게 적용되지 않는다.
 - 1. "을"이 개발하지 않은 것으로서 사전에 "갑"에게 이러한 사실을 알리고 "갑"의 승락을 받아 개발결과물에 포함시킨 것.
 - 2. "갑"이 개발결과물을 "을"과의 협의없이 임의로 수정한 것.

제15조(신의성실 및 상호협조)

- ① "갑"과 "을"은 상호 신의를 갖고 본 계약의 각 조항을 성실히 이행해야 한다.
- ② "을"은 전 개발 과정에 걸쳐 "갑"의 요청이 있을 경우 수시로 개발 내용에 관하여 "갑"과 협의해야 하며, "갑"은 "을"의 요청이 있을 경우 필요한 사항에 관하여 "을"에게 협조한다.
- ③ "을"은 본 개발기간의 완료 후에도 "갑"의 새로운 제품에 본 계약으로 개발된 기술의 적용과 관련하여 "갑"의 요청이 있을 경우 적절한 비용으로 기술자문이나 기술지원을 제공한다.

제16조(비밀유지)

- ① "을"은 본 개발 수행기간 중에 개발과제 관련 사항 및 본 계약내용 일체를 자사 홈페이지나 신문, 잡지, 방송 등 언론매체를 포함하여 그 어디에도 공표하거나 누설할 수 없다. 개발이 완료된 후에도 개발내용 및 이와 관련된 일체의 사실을 공개하고자 할 경우 사전에 "갑"의 서면 동의를 얻어야 한다.
- ② "을"은 본 계약의 체결 및 이행과 관련하여 알게 된 "갑"의 영업비밀을 "갑"의 사전 서면 동의 없이 제3자에게 공표하거나 누설하여서는 안 되며, 위 비밀정보의 보안을 유지하기 위하여 필요한 조치를 해야 한다.
- ③ "을" 및 "을"의 임직원("을"의 수급업체 직원 포함)이 본조를 위반하는 경우 "을"은 "갑"에게 제8조에 정한 총 개발비를 위약금으로 지급하여야 한다.
- ④ 다음 각호의 어느 하나에 해당하는 정보는 비밀정보로 취급되지 아니한다.
 - 가. 비밀정보의 제공 이전부터 을이 보유하고 있던 정보
 - 나. 을의 과실없이 공지의 사실로 된 정보
 - 다. 을이 적법하게 제3자로부터 제공받은 정보
 - 라. 을이 독자적으로 용역한 정보
 - 마. 갑이 사전에 유출을 서면으로 허락한 정보

- ⑤ 본조의 의무는 본 계약기간 중은 물론 본 계약이 해제, 해지되거나 기간만료 등으로 종료된 경우에도 계속하여 유효한 것으로 한다.

제17조(자료의 사용 및 반환)

- ① "을"은 본 계약의 수행을 위하여 "갑"으로부터 제공 받은 자료를 선량한 관리자의 주의 의무를 가지고 관리하여야 하며, 위 자료를 본 계약 수행 이외의 용도에 사용할 수 없다.
- ② "을"은, 본 계약 개발 수행이 완료되거나 본 계약이 해지되는 즉시 또는 "갑"의 요청을 받는 즉시, "갑"으로부터 제공 받은 모든 자료 및 그 복사본을 "갑"에게 반환해야 한다.
- ③ "갑"은 광고, 판매촉진 기타 선전의 목적으로 "을"이 "갑"에게 제출한 보고서나 문서의 일부 또는 전부에 대한 원본, 복제물, 복사물 등을 사용할 수 있다.

제18조(권리 의무의 양도금지)

"갑"의 서면에 의한 사전 동의 없이 "을"은 본 계약상의 어떠한 권리와 의무도 제3자에게 양도하거나 처분할 수 없다.

제19조(개발수행 제한)

- ① "을"은 본 계약 개발업무 수행으로 습득한 제반 기술 또는 정보를 "갑" 이외의 이 동통신단말기 개발·제조업체를 위하여 이용하거나 제공할 수 없다.
- ② "을"이 본 계약 개발결과물 인수일로부터 또는 본 계약 해지일로부터 1년 이내에 "갑" 이외의 이 동통신단말기 개발·제조업체를 위하여 본 계약 개발과제와 동일 또는 유사한 개발과제를 수행하거나 위 개발과제와 관련된 협력 활동을 하기 위하여는 "갑"의 사전 서면 동의를 있어야 한다.

제20조 (계약의 해제 및 해지)

- ① "갑" 또는 "을" 양 당사자 중 일방에게 다음 각호에 해당하는 사유가 발생한 경우에는 상대방은 최고 없이 본 계약을 해제 또는 해지할 수 있다.
 - 1. 발행한 어음이나 수표가 부도 또는 거래 정지된 경우
 - 2. 감독관청으로부터 영업정지 또는 영업면허, 영업등록 등의 취소처분을 받은 때
 - 3. 파산절차 또는 회생절차가 시작되거나 이러한 신청이 있는 경우
 - 4. 가압류·가처분 등으로 본 계약의 목적달성이 곤란하다고 판단될 경우
- ② "갑" 또는 "을" 중 일방이 본 계약을 위반한 경우 상대방은 XX 일의 기간을 두고

이를 최고한 후, 시정되지 않을 경우 본 계약을 서면통지에 의해 해제 또는 해지할 수 있다.

- ③ "갑"은 "을"에게 다음 각호에 해당하는 사유가 발생하였을 경우 XX 일의 기간을 두고 이를 최고한 후, 시정되지 않을 경우 본 계약을 서면통지에 의해 해제 또는 해지할 수 있다.
1. 개발용역 수행이 정지 상태가 되어 소기의 개발 성과를 기대하기 곤란하거나 완수할 능력이 없어졌다고 객관적으로 판단될 때
 2. "갑"이 추구하는 개발 목표가 다른 개발 수행에 의하여 성취되어 개발용역을 계속할 필요성이 인정되지 아니할 때
 3. 기타 중대한 사유로 인하여 개발용역을 계속할 수 없다고 판단될 때
- ④ 본 계약이 해지된 경우 양 당사자는 개발결과물의 완성 정도에 따라 해지일까지의 개발비를 정산하고, 해지일까지 발생한 개발결과물 및 본 개발용역과 관련하여 발생하는 모든 지적재산권은 "갑"에게 귀속한다. 단, 해지일까지의 개발결과물이 "갑"과 "을"이 상호 합의한 내용을 충족하지 못할 경우 "갑"은 위 개발결과물을 인수하지 않고 "을"에게 기 지급한 개발비의 반환을 요구할 수 있다.
- ⑤ 본 계약의 해제 또는 해지로 인하여 손해가 발생한 경우에 손해를 입은 당사자는 상대방에게 손해배상을 청구할 수 있다.

제21조(지체상금)

- ① "을"의 책임 있는 사유로 인하여 개발 일정이 지연되었을 경우, "을"은 "갑"에게 지체일수 1일당 개발비 총액의 1000분의 XX 에 해당하는 금액을 지급하여야 한다.
- ② "갑"이 제8조의 개발비 지급을 지체한 경우, "갑"은 "을"에게 "을"의 주거래은행의 평균 연체금리에 따른 지연이자를 지급하여야 한다.

제22조(손해배상책임)

- ① "갑"과 "을" 중 어느 일방이 본 계약서에 규정된 의무 이행을 게을리하거나 또는 지연시킬 때에는 그 행위를 야기한 당사자는 그러한 불이행에 대해 전적으로 책임을 지며, 그에 따른 일체의 손해를 배상함과 동시에 상대방이 요구하는 모든 적절한 조치를 취해야 한다.
- ② 본 계약상의 어떠한 조항에도 불구하고, 채무불이행 이외의 다른 어떠한 청구권원에 의하더라도, 을의 귀책사유로 발생하는 모든 손해배상액은 계약금액을 초과하지 못하며, 을은 간접적, 파생적 손해에 대해 일체의 책임을 지지 않는다.

제23조(계약의 변경)

"갑"과 "을"은 서면합의에 의하여 본 계약의 내용을 변경할 수 있다.

제24조(불가항력에 의한 면책)

"갑" 또는 "을"이 화재, 홍수, 지진, 폭풍, 전쟁, 혁명 등 불가항력에 의해 본 계약에 기초한 의무를 이행할 수 없는 경우, 해당 당사자는 위 사유를 원인으로 하여 이행 지체 또는 불이행의 책임을 부담하지 않는다.

제25조(합의관할)

본 계약과 관련된 일체의 분쟁은 XX 법원 또는 "갑"과 "을"이 별도 합의한 법원을 관할법원으로 한다.

제26조(존속조항)

본 계약의 제9조(개발결과물 귀속 등), 제10조(품질보증), 제13조(지적재산권), 제14조(보증 및 면책), 제16조(비밀유지)는 본 계약이 해지되거나 기간 만료 기타 사유로 종료되어도 유효하다.

제27조 (통지)

양 당사자간의 모든 통지는 우편 또는 이메일로 다음의 연락처에 전달되어야 한다.

"갑" : <사업장 주소> <부서명>
전화번호 : <유선 번호>
수 신 자 : <대표자 성명> <대표자 이메일 주소>

"을" : <사업장 주소> <부서명>
전화번호 : <유선 번호>
수 신 자 : <대표자 성명> <대표자 이메일 주소>

제28조 (기타)

- ① 본 계약 이전에 서면, 구두로 본 계약과 관련한 합의사항들은 모두 무효이고, 본 계약서에 규정된 내용만이 유효하며, 최종적으로 합의된 내용이다.
- ② 첨부된 문서들은 본 계약의 일부를 이룬다.

제29조 (계약의 효력)

본 계약은 201X년 XX월 XX일로 소급되어 효력이 발생한다.

첨부 <문서명>

위와 같이 계약을 체결하고 그 성립을 증명하기 위하여 본 계약서 2부를 작성하여 "갑"과 "을"이 각 기명 날인한 후 각 1부씩 보관한다.

- 이상 -

8. 참고문헌 및 사이트

- “공개소프트웨어 거버넌스 가이드라인”, 김병선, 2013
- “공개소프트웨어 활용 법적 문제 대비해야”, 디지털타임스, 2013
- “오픈소스SW 라이선스 분쟁 대응방안 가이드”, 문화체육관광부, 2009
- “공개소프트웨어 활성화 정책의 현황과 방향”, 이철남, 2003
- “공개소프트웨어 기술 및 표준화 동향”, 전자통신연구원, 2006
- “공공기관을 위한 공개소프트웨어안내서Ⅳ”, 정보통신부, 2005
- “오픈소스SW 라이선스 가이드”, 정보통신부, 2007
- “공개소프트웨어 유지보수 가이드라인”, 정보통신부, 2007
- “공개소프트웨어도입가이드(o-ISP)”, 정보통신산업진흥원, 2009
- “공개소프트웨어 Governance v2.0”, 정보통신산업진흥원, 2009
- “공개소프트웨어 백서”, 정보통신산업진흥원, 2012
- “공개SW 라이선스 가이드”, 정보통신산업진흥원, 2014
- “공개소프트웨어 도입저해요인 실태조사 연구”, 한국소프트웨어진흥원, 2003
- “공개소프트웨어 가이드 호주”, 한국소프트웨어진흥원, 2005
- “공개소프트웨어 가이드”, 한국소프트웨어진흥원, 2006
- “공공기관 공개소프트웨어 도입가이드”, 한국소프트웨어진흥원, 2007
- “Assessing Open Source Software Projects”, Gartnet, 2005
- “The governance of free/open source projects”, M. Lyne Markus, 2007
- “A framework for information systems architecture”, J. A. Zachman, 1987
- “Consulting Group Open Source Software Assessment Survey”, Black Duck Software, Inc



Open Source Software Governance Framework and Its Applying Guide

공개소프트웨어 거버넌스 프레임워크 및 적용가이드

2015년 8월 25일 인쇄

2015년 8월 31일 발행

발행인 윤종록

발행처 정보통신산업진흥원

(27872) 충청북도 진천군 덕산면 정통로 10

TEL. 043-931-5000 FAX. 043-931-5129

디자인/인쇄 사회복지법인 흥애원

비매품



9 788961 083072

ISBN 978-89-6108-307-2



공개소프트웨어 거버넌스 프레임워크 및 적용가이드는
크리에이티브 커먼즈 저작자표시-비영리-변경금지 2.0
대한민국 라이선스에 따라 이용할 수 있습니다.

ISBN 978-89-6108-307-2 93000

