공개 소프트웨어 연구개발 수행 가이드라인

2018. 4.

과학기술정보통신부

nipa 정보통신산업진흥원
National IT Industry Promotion Agency

목 차

요약 공개SW 연구개발 시 핵심 고려사항

1 가이드라인의 대상 및 목적

○ **(대상)** 공개SW 연구개발 사업(과제)을 수행하고자 하는 기관(기업) 담당자

○ **(목적)** 공개SW 연구개발 수행 시 고려하거나 검토해야 할 핵심 사항을 안내

2 핵심 고려사항

단계	주요 활동		공개SW 개발 관련 핵심 검토사항
제안 단계	사업계획서 작성 및 제출		- 사업 제안서(RFP)를 참조하여 공개SW 개발 방식을 적용한 연구개발 계획 작성 - 공개SW 라이선스 선정 - 평가기준 확인 및 관련내용 반영
수행 단계	연구개발 계획 수립		- 소스코드(산출물) 공개 범위 설정 - 소스코드(산출물) 공개 시점 설정 - 소스코드(산출물) 공개 방법 설정
	연구개발 수행 (반복)	요구사항 분석	- 기존 공개SW 활용 시 조사, 분석, 평가, 계약
		설계	- 모듈라 설계 방식 적용
		구현	- 소스코드 통합 및 형상관리 - 이슈 추적 및 관리 - 소스코드 저장소 및 호스팅 환경 구축 - 패키징
		테스트	- 테스트 자동화 환경 구축 - 공개SW 라이선스 검증 및 보안 취약점 점검 - 소스코드 릴리즈 및 프로젝트 공개
	결과(연차) 보고서 작성		- 최종(연차) 결과보고서 작성 - 사업화 및 커뮤니티 운영 방안 포함
활용 단계	커뮤니티 운영		- 연구개발 산출물을 일반에 공개하고, 외부 개발자 참여 유도를 위한 커뮤니티 개설 - 커뮤니티 행동규칙 및 의사결정 방안 수립 - 커뮤니티 멥버쉽 관리 및 작업 할당 - 커뮤니티 활성화를 위한 홍보활동 수행
	사업화		- 사업화를 위한 공개SW 비즈니스 모델의 구체화 및 실행

제1장 개 요

1. 가이드라인 개요

공개 소프트웨어 연구개발 수행 가이드라인(이하 동 가이드라인)은 공개 소프트웨어(이하 공개SW) 연구개발 사업 또는 과제를 수주하거나, 수주된 사업을 수행하고자 하는 기관(기업)에서 알아야 할 절차 및 내용을 안내해 주고자 한다.

다만, 동 가이드라인에서는 일반적인 연구개발을 수행할 때의 프로세스 및 개발 방법론 등에 대해서는 설명하지 않고, 연구개발 수행 중 공개SW 특성을 고려하여야 하는 부분을 중심으로 설명한다.

동 가이드라인은 연구개발과 관련하여 “정보통신·방송 연구개발 관리 규정(이하 관리규정)” 및 부속 규정을 따르며, 관리규정 제40조에 따라 사업 공고 시 공개SW 과제로 지정된 과제는 동 가이드라인을 참조할 수 있다(관리규정 제40조에 따라 공개SW 과제는 기술료 및 민간부담금 면제 가능).

2. 공개SW의 개념 및 중요성

동 가이드라인에서 설명하고 있는 공개SW의 개념 및 관련 주요 용어는 아래와 같다.

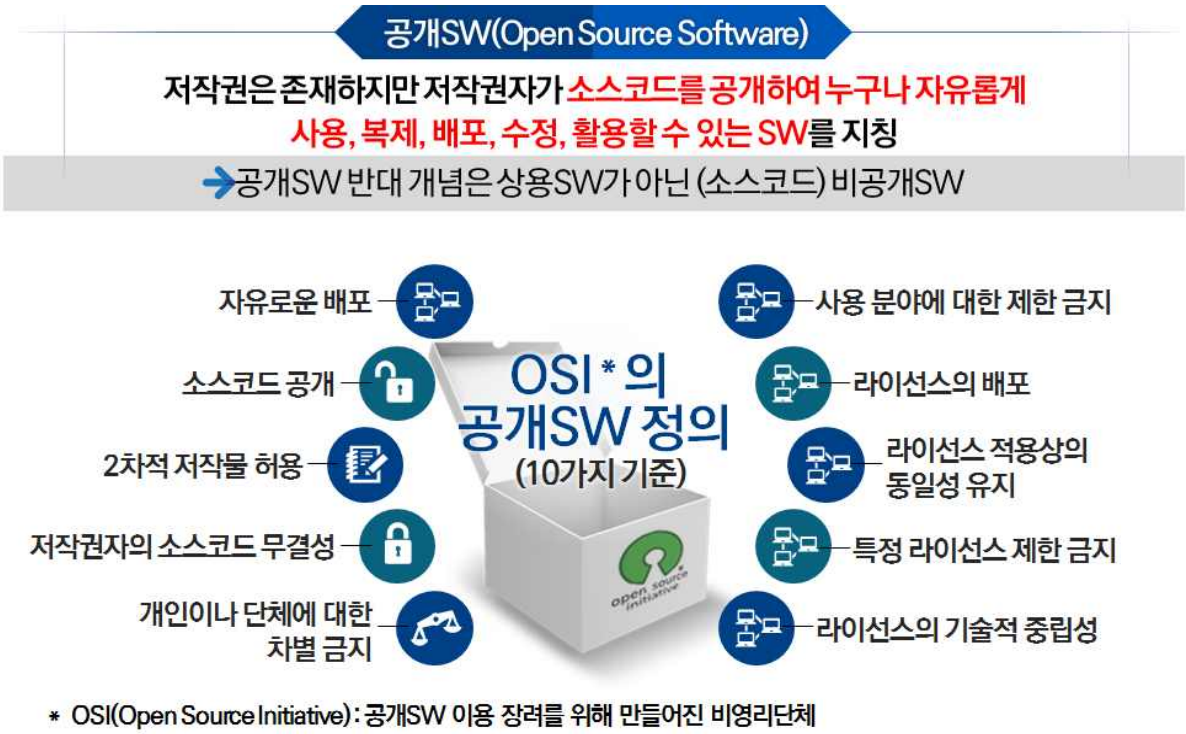
○ "**공개SW**"라 함은 "오픈소스 소프트웨어" 또는 "오픈 소스" 등 그 명칭과 상관없이 소프트웨어(SW)의 저작권자가 해당 소스코드를 공개하여 이를 사용, 복제, 수정, 배포할 수 있는 권한을 부여한 SW를 말한다.

○ "**공개SW 라이선스**(License)"라 함은 공개SW의 사용, 복제, 수정, 배포와 관련하여 허용되는 권한 범위를 명시한 이용허락 조건을 말한다.

○ "**공개SW 개발방식**"이라 함은 SW의 소스코드를 공개하고 SW를 개발 및 유지 관리하는 전 과정에 최초 개발한 자 외에도 누구나 자유롭게 참여할 수 있도록 하는 개발방식을 말한다.

○ "**공개SW 연구개발 사업 또는 과제**"란 SW 연구개발 중 그 결과물을 공개SW로 배포할 목적을 가지고 공개SW 개발방식으로 추진되는 연구개발 사업 또는 과제를 말한다.

○ "**공개SW 커뮤니티**"란 공개SW 연구개발 사업 또는 과제를 수행하기 위한 온·오프라인 협업 공간으로서, 과제 개발 과정에서의 소스코드 공유, 의사소통 등을 위한 공간을 말한다.



[그림 1] 공개SW의 정의 및 특징

공개SW를 활용한 연구개발 수행은 다음과 같은 장점을 가진다.

[그림 2] 공개SW의 중요성

3. 공개SW 연구개발 과제의 개발 유형

공개SW 연구개발 과제는 개발 유형에 따라 아래 두 가지 유형으로 구분할 수 있다.

○ **연구개발 완료 후 최종 산출물만을 공개하는 과제**

○ **연구개발 초기부터 산출물을 공개하고, 참여연구원 이외의 외부 개발자 참여를 유도하여 협업하는 과제**

전자의 경우 최종 산출물의 공개 여부를 제외하고는 기존 일반적인 비공개 연구개발 프로세스 및 방법론(예 : 폭포수 모델)과 유사하며, 이는 엄밀히 얘기하면 공개SW 개발방식을 적용하였다고 보기는 어렵다.

후자의 경우 실제 대부분의 공개SW 커뮤니티에서 적용하는 개발방식이며, 반복·순환 방법론(예 : 애자일)을 적용한다.

공개SW 개발방식의 핵심 개념인 공유·기여·협업의 가치실현을 위해서는 일부 불가피한 경우(단년도 과제 등)를 제외하고 후자의 개발방식을 적용하는 것이 타당하며, 다년도 과제의 경우 1년차 종료시점 전까지는 산출물을 공개하도록 한다.

다만, 공개SW 개발방식은 관련 경험이 부족할 경우 시행착오 등을 격을 수 있으므로, 한시적으로 프로젝트 상태를 비공개로 하되 산출물 저장소에서는 이력이 지속적으로 쌓이도록 유지한 후, 일부라도 공개가능한 시점이 되었을 때 프로젝트를 공개 상태로 전환하는 방식도 고려해 볼 수 있다.

마지막으로 상기의 산출물 공개 및 관리를 위해 커뮤니티를 개설하여야 하며, 기술료 등을 면제받는 특혜를 고려하여 사업 종료 후에도 해당 공개SW 프로젝트가 자생할 수 있도록 지속적인 커뮤니티 유지 및 운영 방안이 필요하다.

4. 공개SW 특성을 반영한 연구개발 수행절차

공개SW 연구개발 사업 및 과제는 아래와 같은 절차로 수행할 수 있으며, 제2장부터는 공개SW 개발 특성을 반영한 핵심 프로세스 중심으로 세부적인 내용을 설명한다.

단계	전담기관	수행기관	공개SW 관련 주요 수행내용
기획 단계	사업공고		- 공개SW 연구개발 과제 여부 및 관련 내용을 공고문 및 RFP에 명시
		사업계획서 작성 및 제출	- 사업계획서 작성 시 공개SW 특성을 고려한 연구개발 계획 작성 - 공개SW 라이선스 선정 - 평가기준 참고 및 관련내용 반영
	사업계획서 접수 및 평가		- 공개SW 연구개발 특성을 고려한 평가 기준 적용
수행 단계	협약체결	연구개발 수행	- 협약서 상 공개SW 연구개발 관련 내용 확인 및 협약 체결 - 공개SW 개발방식 및 특성을 적용한 연구개발 수행 - 커뮤니티 개설 및 소스코드 공개 - 공개SW 라이선스 검증 및 보안취약점 점검 등 수행
	진도점검		
종료 단계		연차/최종 보고서 작성 및 제출	- 보고서 작성 시 사업종료 후 커뮤니티 운영 및 발전 방안 등 향후 계획 등 추가 작성 - 공개SW 라이선스 검증 결과보고서 등 추가 제출
	연차/최종 평가		- 공개SW 연구개발 특성을 고려한 평가 기준 적용

[표 1] 공개SW 연구개발 프로세스

제2장 사업계획서 작성 및 제출

1. 사업계획서 작성

사업계획서 양식은 수행하고자 하는 연구개발 사업이 공고될 때 배포된다. 일반적으로 공개SW 과제일 경우 배포된 사업계획서 양식 내에 공개SW 개발 특성을 반영한 항목이 기술될 수 있도록 안내된다.

사업계획서 양식은 사업의 특성, 목적, 전담기관 등에 따라 상이할 수 있으므로, 해당 사업 공고 시 첨부된 사업계획서 양식을 반드시 확인하여야 한다.

공개SW 연구개발 사업계획서의 경우 일반적인 양식에 각 목차별로 공개SW 특성을 고려한 항목이 일부 삽입되거나, 별도의 독립된 장으로 구성될 수 있다.

아래 예시는 설명의 편의를 위해 공개SW 관련 항목이 별도의 독립된 장으로 구성된 사업계획서 양식을 기준으로 설명하고 있으며, 배포된 양식과 다를 경우 해당 양식에 맞도록 아래 내용을 참고하여 해당 부분을 작성토록 한다. 또한, 배포된 양식에서 별도의 공개SW 관련 항목 작성의 요구가 없을 경우 아래 항목을 참고하여 가능한 자세히 기술토록 한다.

항목	핵심 기술내용
필요성	- 공개SW 개발과제로서의 본 과제의 타당성, 필요성, 파급효과 등 기술 - 해당 분야 유사 상용SW와의 비교 및 차별성

계획수립	- 공개SW 개발 방법 및 전략, 공개 시점, 공개 범위, 공개 방법 등 - 사업종료 후 고도화 및 사업화 방안
공개SW 라이선스 정의	- 본 과제에 적용할 공개SW 라이선스 정의 - 해당 라이선스 선정 사유 및 타당성 - 사업 종료 후 라이선스 정책 및 활용 방안
개발환경 구축	- 소스코드 형상 관리, 통합 빌드방안, 협업체계 구축 등 - 사용할 소스코드 저장소(깃허브(Github) 등) 등 - 외부 개발자와의 협업 방안
라이선스/보안 취약점 검증	- 공개SW 라이선스 및 보안취약점 점검 계획 등
커뮤니티 운영	- 커뮤니티 운영 조직 및 방법, 이슈리스트, 버그트래킹, Q&A 관리방법, 코드리뷰 등 - 커뮤니티 운영 및 발전 계획
홍보 및 세미나 계획	- 커뮤니티(기술) 활성화를 위한 홍보 및 세미나 개최 계획 등
참여인력	- 총괄책임자 또는 참여연구원의 공개SW 개발 프로젝트 참여 이력(프로젝트명, 기간, 역할, 기여도 등)

[표 2] 공개SW 연구개발 사업계획서 작성 항목 예시

상기 내용은 일반적인 공개SW 연구개발 특성을 고려한 사업계획서 작성 목록의 예시이며, 수행하고자 하는 사업 유형 및 특성에 따라 내용이 추가・삭제될 수 있다.

2. 평가지표 확인

일반적으로 공개SW 연구개발 사업 공고 시 지원과제 선정 평가 기준도 공개되므로 이를 반드시 참조하여야 하며, 평가기준에 부합하는 내용을 사업계획서에 가능한 한 구체적으로 명시하기를 권장한다.

아래는 공개SW 연구개발 사업 선정평가 시 적용 가능한 평가항목의 예시로, 전담기관에 따라 평가기준이 상이하므로 반드시 해당 사업의 평가기준을 재차 확인하도록 한다.

구분	평가내용	배점
기술성	o 제안 과제의 기술적 우수성 - 공개SW 개발과제로서의 타당성, 필요성, 파급효과 등 - 기술개발 목표(지표 등)의 적절성 o 공개SW 프로젝트 개발 경험 - 기존 공개SW 개발 프로젝트 참여 및 기여도 등	
창의성	o 공개SW 기술로서의 창의적 활용성 - 공개SW 기반 기술로의 창의적 활용 가능성 등 o 개발과제의 독창성 - 기존 기술·제품과의 차별성 및 공개SW 개발의 장점	
실용성	o 공개SW 라이선스의 적절성 - 적정 라이선스 적용 여부 및 적용 사유의 적정성 o 개발 계획 및 방법의 적절성 - 공개SW 개발 계획 및 방법의 적절성 - 개발 산출물 공개 범위의 적정성	
발전성	o 향후 사업화 실현 가능성 - 사업 종료 후 사업화 성공 가능성 o 사업종료 후 고도화 및 발전계획의 적절성 o 공개SW 커뮤니티로의 성장 및 지속 가능성	
합계		100

[표 3] 공개SW 연구개발 평가기준(선정) 예시

3. 공개SW 라이선스 선정

공개SW 연구개발 과제는 관리규정 제40조 4항에 따라 본 사업에 적용할 공개SW 라이선스를 선택해야 한다.

공개SW 연구개발 과제에서 라이선스 선정은 매우 중요한 문제이며, 특히 사업 종료 후 사업화 방향과도 직결되므로 어떤 라이선스를 적용할지에 대해 매우 신중히 검토하여야 한다.

공개SW 라이선스 관련 내용은 매우 방대하므로, 본 가이드라인에서는 공개SW 연구개발 수행과 관련하여 핵심적인 내용만 기술할 것이며, 추가적인 내용은 본 문서의 '**[붙임1] 공개SW 라이선스 개요**' 및 공개SW 라이선스 가이드라인(www.oss.kr에서 확인 가능)을 참조하기 바란다.

공개SW 라이선스 선정에 있어 우선 검토되어야 할 사항은 연구개발 수행 시 기존 공개SW의 활용 유무이며, 이는 아래의 두 가지 유형으로 구분할 수 있다.

- **전체 시스템을 자체 개발하여 외부에 공개할 경우**
- **기존 공개SW를 일부 또는 전체적으로 활용하여 추가적인 개작 후 외부에 공개할 경우**

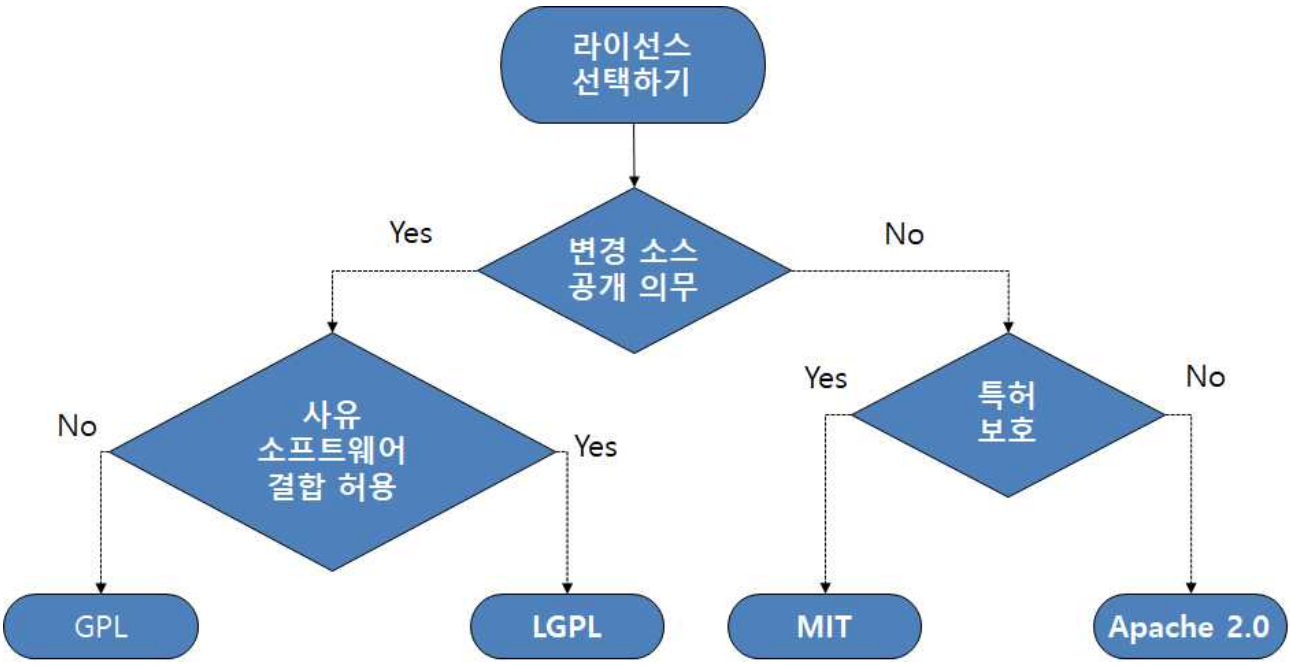
전자의 경우 연구개발 종료 후의 사업화 및 비즈니스 모델에 따라 전략적으로 라이선스를 선정하여야 한다. 연구개발 과제의 수행기관이 기업이고, 향후 과제 산출물 기반의 비즈니스를 계획하고 있을 경우에는 '**[붙임2] 공개SW 비즈니스 기획**' 을 참조한다.

후자의 경우 전자에 더하여 본 사업에서 선정하고자 하는 라이선스와 기존 공개SW 라이선스의 양립가능성이 문제될 수 있으므로, 반드시 활용하고자 하는 기존 공개SW 라이선스를 확인하여야 한다. 이와 관련된 세부적인 내용은 '**[붙임3] 공개SW 라이선스의 양립성**'을 참조한다.

연구개발 과제에서 활용할 기존 공개SW는 목록으로 만들어 관리되어야 하며, 필요 시 조사, 분석, 평가, 계약(듀얼 라이선스일 경우) 등의 과정을 거쳐 자체적인 검증과정을 거칠 수 있다.

상기 내용을 충분히 참고하고 신중히 검토하여 과제에 적용할 라이선스를 선정해야 하며, 선정될 라이선스의 종류에 따라 소스코드의 공개범위 및 특허권리 등이 결정된다.

아래 그림은 공개SW 라이선스를 선정하기 위한 기초적인 절차의 예시이다.



[그림 3] 공개SW 선정절차 예시

제3장 공개SW 연구개발 수행

1. 공개SW 개발방식의 특성

본 장에서는 연구개발 수행을 위한 협약체결 완료 후 본격적인 기술개발 단계에서의 공개SW 개발방식 특성을 설명하며, 일반적인 SW 개발 프로세스 및 방법론에 대해서는 설명하지 않는다.

공개SW 개발방식에서 우선적으로 고려되어야 할 주요사항은 아래와 같다.

- ○ **소스코드(산출물) 공개 범위 결정**
- ○ **소스코드(산출물) 공개 시점 결정**
- ○ **소스코드(산출물) 공개 방법 결정**

소스코드 공개 범위는 해당 과제의 사업화 방향과 밀접한 관련이 있으며, 개발자는 사업계획서 작성 시에 고려되었던 사항을 재검토하여 최종 공개 범위를 결정한다. 설계 단계에서도 공개 범위에 대한 고려가 필요하다.

소스코드 공개 시점은 본 가이드라인의 서두에서 설명하였다시피 다년도 과제의 경우 늦어도 1차년도 연차평가 시점까지는 공개되어야 하며, 이후에는 완전한 공개SW 개발 방식을 적용한다.

소스코드 공개 방법은 일반적인 공개SW라면 누구나 쉽게 접근가능하고, 기본적인 소스코드 관리가 용이한 공개된 저장소(Repository) 중 한 곳을 선택하여 공개한다. 이와 관련된 세부적인 내용은 전 세계적으

로 가장 많은 공개SW 프로젝트가 관리되고 있는 깃허브(Github)를 예시로 설명한 '**[붙임5] Github 개요**'를 참조한다. 또한 공개 시 그 자체로 빌드 및 수행이 가능한 독자적인 형태(self-contained)가 되어야 한다. 만일 일부 모듈이 미개발되었거나 아직 공개시점이 아니라고 판단한 경우에는 빈 함수(empty function)로 대체 하거나 라이브러리 형태로 포함하여, 전체 소스코드가 저장소에서 다운로드, 컴파일, 빌드가 가능하여야 한다.

아래는 연구개발 각 단계에서 공개SW 개발방식의 특성을 고려한 주요 수행 항목들을 전반적으로 정리한 내용이다.

<table>
<tr><th>단계</th><th colspan="2">핵심 검토사항</th><th>비고</th></tr>
<tr><td rowspan="3">사업계획서
작성</td><td colspan="2">공개SW 관련 항목 작성</td><td rowspan="3"></td></tr>
<tr><td colspan="2">과제선정 평가지표 확인 및 반영</td></tr>
<tr><td colspan="2">공개SW 라이선스 선정</td></tr>
<tr><td rowspan="3">연구개발
계획 수립</td><td colspan="2">소스코드 공개 범위 설정</td><td rowspan="3"></td></tr>
<tr><td colspan="2">소스코드 공개 시점 설정</td></tr>
<tr><td colspan="2">소스코드 공개 방법 설정</td></tr>
<tr><td rowspan="7">연구개발</td><td colspan="2"></td><td></td></tr>
<tr><td rowspan="4">요구사항 분석</td><td>조사</td><td rowspan="4">기존 공개SW
활용 시</td></tr>
<tr><td>분석</td></tr>
<tr><td>평가</td></tr>
<tr><td>계약</td></tr>
<tr><td>설계</td><td>모듈라 설계</td><td></td></tr>
<tr><td rowspan="3">구현</td><td>소스코드 통합 및 관리</td><td rowspan="3"></td></tr>
<tr><td>소스코드 형상 관리</td></tr>
<tr><td>이슈 추적 및 관리</td></tr>
</table>

단계	핵심 검토사항		비고
		저장소 및 호스팅 설비	
		패키징	
	테스트	테스트 자동화	
		공개SW 라이선스 검증	
		보안 취약점 점검	
		릴리즈	
		론칭	
커뮤니티 운영	커뮤니티 운영 목적		
	커뮤니티 구조		
	커뮤니티 운영		
	커뮤니티 활성화(홍보 등)		

[표 4] 연구개발 프로세스 상 공개SW 특성 관련 주요 고려사항

2. 요구사항 분석 단계에서의 공개SW 특성

일반적인 SW개발 프로세스 상 첫 번째 단계인 요구사항 분석 단계에서 공개SW 관련 고려해야 할 주요 사항으로 본 가이드라인에서는 아래 5개 단계에 대해 설명한다. 본 단계는 해당 과제에서 기존 공개SW를 활용하거나, 이를 기반으로 개작 또는 고도화 과제를 수행할 경우에만 해당된다.

단계	태스크	연구개발 수행 유형	
		전체 시스템을 자체 개발하여 외부에 공개할 경우	기존 공개SW를 일부 또는 전체적으로 활용하여 추가적인 개작 후 외부에 공개할 경우
요구사항 분석	요구분석	O	O
	조사	N/A	O
	분석	N/A	O
	평가	N/A	O
	계약	N/A	O

[표 5] 기존 공개SW 활용 유무에 따른 수행 활동

○ **요구 분석** : 해당 연구개발 과제 제안서(RFP), 사용자, 개발자 등의 고민과 요구사항을 기반으로 기능, 성능, 연동, 호환, 보안 등의 관점에서 필요한 사항을 분류하고 해석하는 활동이다.

○ **조사** : 요구사항 분석 결과를 기반으로 본 과제의 속성과 이를 위한 기존 공개SW 정보를 취합하는 활동이다. 이 활동을 통해 반드시 취합해야 할 속성으로는 기능, 용도, 라이선스, 개발언어, 적용분야, 연동 및 연계 SW, 유스 케이스(use case) 등이다. 또한, 기존 공개SW 관련 정보로는 최초 등록일, 소스코드 개발 속도, 다운로드 횟수, 레퍼런스 개수, 핵심 개발자 및 커미터(committer) 등이 있다.

○ **분석** : 기능성, 이식성, 신뢰성, 사용성, 유지보수성, 커뮤니티 영속성 등 개발자가 활용할 공개SW의 수준과 가치를 분석하기 위한 데이터 취합 및 정리 활동이다.

○ **평가** : 본 단계는 도입 및 활용할 공개SW의 성숙도와 적용성을 정량적으로 산출하는 활동이다. 이러한 평가는 수치로 산출될 수 있어야 하며 이를 위해 정량적인 지표를 정의하여야 한다.

○ **계약** : 평가를 통해 선정된 공개SW의 사용조건과 의무사항 이행을 약속하는 활동이다. 공개SW는 라이선스에 대한 비용 청구 조건이 없으며, 배포받고 사용하는 순간부터 해당 라이선스의 의무사항 준수에 동의하는 것으로 간주된다. 단, 기존 공개SW가 GPL(General Public License) 등을 포함한 듀얼 라이선스 정책을 가지고 있고, 추후 사업화 시 일부 산출물의 비공개 필요성이 있을 경우 비용을 지급하고 정식 계약을 체결하여야 한다.

3. 설계 단계에서의 공개SW 특성

일반적인 SW개발 프로세스 상 두 번째 단계인 설계 단계에서는 공개SW 개발 방식의 특성 상 아키텍처가 매우 중요하다. 즉, 소스코드 공개 이후 외부 개발자의 참여가 이루어지므로 반드시 소스코드 응집도가 낮은, 즉 가능한 많이 분리된 모듈들로 시스템을 설계하여야 한다. 실제 리눅스 커널을 최초로 개발한 리누스 토발즈(Linus. B. Torvalds)는 MS 윈도우가 소스코드를 오픈한다고 하여도 리눅스가 이룩한 만큼의 발전을 이룰 수 없을 것이라고 한 바 있다. 즉, 윈도우는 소스코드 복잡도 및 응집도가 매우 높아 소스코드를 공개해도 발전이 더딜 것이라는 것이다.

실제 많은 대형 공개SW 프로젝트의 핵심 아키텍처는 시스템에 필수적인 모듈들을 지원하는 플랫폼과 그 상위에 있는 서로 다른 모듈들의 집합으로 구성된다. 예를 들어 리눅스 커널은 플랫폼의 일부이고, 디바이스 드라이버가 독립적인 모듈이다. 또한, 이클립스 통합개발도구(IDE)는 전체가 플러그인 모듈들로 구성되어 있다.

이러한 모듈라 설계 방식은 아래와 같은 특징이 있다.

- 설계 디자인이 뚜렷하고 명확하며 이해하기 쉽다.
- 각기 다른 모듈들 및 작업들 사이의 느슨한 결합은 설계 시에 다른 모듈들에 영향을 주지 않고 작업을 수행할 수 있게 한다. 이는 더 많은 자율성을 제공하며 개발자(기여자 포함)들 사이의 보다 적은 상호작용을 요구한다.
- 좀 더 자발적인 기여를 끌어들인다.
- 시너지 효과와 더불어 협업의 기회를 촉진한다.
- 실험적이고 탐험적인 구현 시도들이 보다 안전하게 수용될 수 있으며, 코드의 변경 및 개선이 전체 시스템을 위태롭게 하지 않고도 수행될 수 있다.

4. 구현 단계에서의 공개SW 특성

일반적인 SW개발 프로세스 상 세 번째 단계인 구현 단계에서는 설계 단계에서 정의된 기능 구현 단위의 소스코드를 작성하고, 각 구성 요소를 묶어서 하나의 통합 SW를 만드는 활동이다. 이러한 활동을 위해 공개SW 개발 방식에서는 아래 5가지 활동을 특히 중요하게 고려하여야 한다.

ㅇ 소스코드 통합 및 관리

공개SW는 대부분 메인 라인을 바탕으로 개발 및 배포를 하게 되므로, 메인 라인을 기준으로 분기(branch), 복제(fork) 작업 등을 통한 추가적인 개발과 지속적인 코드 통합 및 관리가 필요하다.

실제 저장소에 등록된 새로운 코드들을 통합하기 위한 방법으로 다양한 방법 및 도구가 있으나, 이를 위한 예시로 '**[붙임6] Github Pull Request 활용 예시**'를 참조한다.

ㅇ 소스코드 형상 관리

다음으로 중요한 요소 중 하나가 형상 관리이다. 이를 위한 분산 환경 방식의 도구 중 하나가 깃(Git)이며, 이를 온라인으로 구현한 Github를 참조하여 필요 시 활용토록 한다.

ㅇ 이슈 추적 및 관리

다음으로는 이슈 추적 및 관리이다. 모든 SW개발 프로젝트는 이슈를 관리해야 할 필요성이 있다. 이에 프로젝트마다 이슈 추적 시스템을 활용하게 되는데, Github에서 이슈 추적 시스템 기능이 'Issues'라는 이름으로 제공하고 있으므로 필요 시 이를 활용할 수 있다.

ㅇ 저장소 및 호스팅 설비

일반적인 공개SW 개발 프로젝트는 성격(소스포지(sourceforge), Github) 혹은 주제(자바넷, CPAN, CTAN)에 따라 해당 웹 기반 저장소에 호스팅 된다. 이들은 각종 파일 저장, 문서 제작 및 프레젠테이션 등의 시설, 메일링 리스트 호스팅, 온라인 포럼, 소스 코드 탐색 클라이언트, 버전 관리 시스템, 이슈 트래킹 데이터베이스, 다운로드 지원과 같은 다양한

기능을 제공한다. 유명한 저장소에 호스팅된 프로젝트는 상당한 가시성 및 홍보 효과가 있다. 반면 독립적인 웹사이트에 호스팅된 프로젝트는 보다 뚜렷한 존재성과 자율성을 가진다. 이를 고려하여 개발자는 해당 과제의 산출물을 공개할 저장소를 신중히 검토하여 선정한다.

ㅇ 패키징

패키징은 개발된 공개SW의 설치, 복제, 업데이트 등이 편리하도록 관련된 요소 프로그램을 모으고, 구성에 필요한 보조 프로그램 또는 유틸리티를 추가하는 작업이다. 이 단계에서는 직접 개발한 소스코드와 외부에서 획득한 공개SW를 결합하게 된다. 패키징은 플랫폼 및 애플리케이션 레벨에 따라 다양한 방법이 존재한다.

플랫폼 레벨은 리눅스 배포판의 경우 dpkg, RPM, tgz, Pacman 등이 있으며, 오픈솔라리스(OpenSolaris)의 경우는 pkgadd, IPS를 사용한다. 크로스 플랫폼(Cross-platform)의 경우는 dpkg, IPS, OpenPKG 등이 있다.

애플리케이션 레벨의 패키징 방식은 프로그래밍 환경에 따라 다양한 방식이 있다. 예를 들어, Composer(PHP를 위한 의존성 관리 도구), PEAR(PHP 프로그래밍 라이브러리), CPAN(Perl 프로그래밍 라이브러리 및 프로그램 관리자), CRAN(R 프로그래밍 라이브러리 및 프로그램 관리자), Easyinstall(Python 프로그래밍 라이브러리 및 프로그램 관리자), PyPI(Python 패키지 관리자), RubyGems(Ruby 저장소 및 패키지 관리자), Maven(Java 빌드 및 패키지 관리자) 등이 있다.

5. 테스트 단계에서의 공개SW 특성

일반적인 SW개발 프로세스 상 마지막 단계인 테스트 단계는 요구사항 반영 여부와 설계 단계에서 정의한 사양이 오류나 장애 없이 정상적으로 동작하는지를 기능과 성능 그리고 품질 등의 확인을 통해 규명

하는 활동이다. 공개SW 개발 방식에서는 아래 5가지 활동을 특히 중요하게 고려하여야 한다.

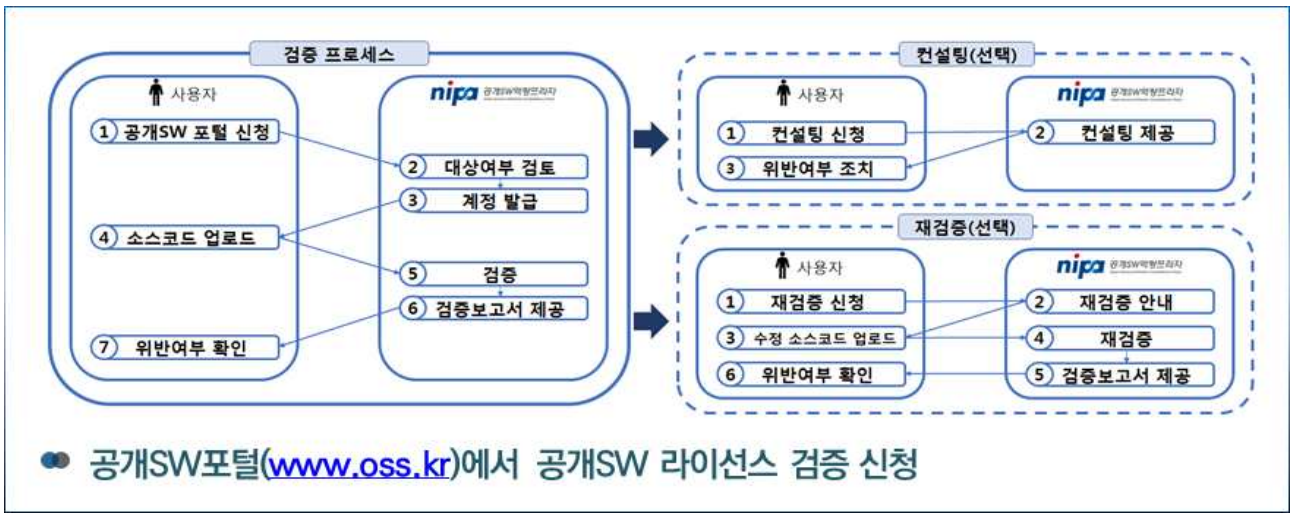
○ 테스트 자동화

공개SW 개발 방식의 특성 상 신속하고 반복적인 테스트를 위해 자동화가 더욱 필요하다. 테스트 자동화를 위해서는 unittest, pytest, mocha, JUnit 등과 같은 테스트 도구를 활용하여 테스트하고, CI 도구를 활용하여 테스트 실행 모듈을 관리하면 된다. 이 또한 Github 와의 연동으로 많이 사용되는 Travis CI가 있으니 필요 시 활용토록 한다.

○ 공개SW 라이선스 검증

개발자는 SW 개발 시 활용한 기존 공개SW의 라이선스 종류 및 공개 범위 등을 명확히 하여 향후 발생될 수 있는 법적분쟁을 사전에 방지하여야 한다. 연구개발 전담기관에 따라서는 라이선스 검증 결과보고서를 요구하기도 하므로 개발 과정에서 반드시 이를 고려하여야 한다.

이와 관련하여 정보통신산업진흥원(NIPA)에서 운영하는 공개SW 포털에서는 관련 서비스를 제공(공공기관 및 중소기업 대상)하고 있으니 필요 시 활용토록 한다.



[그림 4] 공개SW 라이선스 검증 서비스 절차(NIPA)

ㅇ 보안 취약점 점검

개발자는 SW 개발 시 활용한 공개SW에 대한 보안취약점을 점검해야 하며, 이는 활용한 공개SW에 대한 라이브러리 및 모듈 등에 대한 보안 취약점 패치여부 모니터링 및 점검 활동이다.

이 또한 공개SW 포털에서 해당 서비스를 제공하고 있으니, 필요 시 활용토록 한다.

ㅇ 릴리즈

릴리즈 활동은 개발된 공개SW를 외부 사용자에게 전달해 주는 활동으로 전달 매체(인터넷, 저장소, 문서), 배포물의 형태(소스코드, 실행코드), 사용자 유형(고객, 불특정 개인, 단체) 등에는 제한이 없다.

SW 관리에서 “의존성 지옥”이라 불리는 문제가 있다. 이는 시스템 규모가 커질수록, 그리고 더 많은 패키지를 가져다 쓸수록 더욱 쉽게 발생하는데, 의존성이 높은 시스템에서는 새 패키지 버전을 배포하는 일이 매우 어렵다. 의존성 명세를 너무 엄격하게 관리하면, 버전에 갇히게 될 위험이 있고(의존하는 모든 패키지의 새 버전을 배포하지 않고는 업그레이드 할 수 없게 된다), 의존성을 너무 느슨하게 관리하면, 버전이 엉켜서 괴롭게 될 것이다(지나치게 나중 버전까지 호환될 거라 가정한 경우). 버전에 갇히거나 엉켜서 쉽고 안전하게 프로젝트를 계속 진행할 수 없다면 의존성 지옥에 빠진 것이다.

이러한 문제의 해결책으로 버전 번호를 어떻게 정하고 올려야 하는지를 명시하는 규칙과 요구사항을 제안한 것이 있는데 이를 유의적 버전 명세라고 하며 아래와 같이 요약 할 수 있다.

버전을 주. 부. 수 숫자로 하고, - 기존 버전과 호환되지 않게 API가 바뀌면 “주(主) 버전”을 올리고,

- 기존 버전과 호환되면서 새로운 기능을 추가할 때는 “부(副) 버전”을 올리고,
- 기존 버전과 호환되면서 버그를 수정한 것이라면 “수(修) 버전”을 올린다.
- 주. 부. 수 형식에 정식배포 전 버전이나 빌드 메타데이터를 위한 라벨을 덧붙이는 방법도 있다.

여기서 주, 부, 수는 major, minor, patch를 번역한 것으로 주된 버전, 부차적 버전, 수정 버전을 의미한다.

ㅇ 론칭

공개SW 개발 후 공개 가능한 시점에서 해당 프로젝트를 론칭하기 전에 최종 준비 사항을 점검하는 차원에서 아래 체크리스트를 활용하면 도움이 될 것이다.

<문서>
- 오픈소스 라이선스 내용이 포함된 LICENSE 파일 준비
- 기본 문서(README, CONTRIBUTING, CODE OF CONDUCT) 준비
- 프로젝트 네이밍
- 분류 작업 및 라벨링을 포함한 이슈 현행화

<코드>
- 일관된 코드 컨벤션과 명확한 함수, 메서드, 변수 이름 사용
- 명료한 주석 처리
- 개정(Revision) 이력, 이슈, Pull Request에 민감한 내용이 없는지 확인

<사람>
- 개인일 경우 : 법적 자문을 구하던지 회사원이라면 회사의 지적재산권과 오픈소스 정책을 이해
- 회사 또는 기관일 경우 : 법무팀과 상의, 프로젝트 공지 및 프로모션을 위한 마케팅 계획 준비, 커뮤니티 활동(이슈 대응, Pull Request 리뷰 및 취합)을 관리할 전담 인력 확보, 프로젝트 Admin 접속 가능한 인력 최소 두 명 확보

제4장 공개SW 커뮤니티 운영

1. 커뮤니티의 목적

본장에서는 공개SW 개발방식의 핵심적인 특성으로 프로젝트의 성공 여부에 중요한 영향을 미치는 커뮤니티 운영과 관련된 사항을 설명한다.

커뮤니티 활동에는 개발에 참여하여 사양을 정하고 소스코드를 기여하며 문제를 해결하는 것뿐만 아니라, 재정적 지원, 교류 참여 등도 포함된다. 이처럼 커뮤니티를 운영하는 이유는 외부의 리소스를 적극적으로 활용하기 위해서이다. 이미 알려진 대로 대표적인 외부 리소스인 공개SW 개발자 중에는 역량과 열정을 갖추고 있는 고급 인력들이 많다. 특히 리눅스 커널, Gnome, Apache Http Server 등 전 세계적으로 매우 인정받고 많이 사용되고 있는 공개SW일수록 핵심 개발자들의 역량은 상상을 초월하는 경우가 많다.

일반적인 규모의 기업에서는 대규모의 SW 인력을 운영하기가 어렵기 때문에 고도의 기술을 요구하는 프로젝트에 대해서 마음껏 인력을 투입하기 어렵다. 인력이 있다고 하더라도 원하는 기술을 달성하기 위한 고급 개발자는 더더욱 찾기 어렵다. 따라서 공개SW 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.

공개SW 커뮤니티 참여자에는 개발자, 관리자, 사용자 등이 있으며, 개발자 간 소통을 위해서는 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 강화해야 커뮤니티 조직이 구성되며, 버그 리포트에 대한 신속한 대응과 사용자 및 참여 개발자의 문제점을 안정적으로 해결함으로써 커뮤니티는 활성화된다. 커뮤니티는 해당 프로젝트의 문제 및 요구사항을 스스로 해결할 수 있는 핵심 주체이다.

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

2. 커뮤니티의 구조

커뮤니티 이해관계자는 크게 코어 멤버, 액티브 멤버, 주변 멤버로 구성(양파 모델)되며, 엄격한 의미에서 커뮤니티 멤버는 아니지만 SW 시스템의 사용을 통해 프로젝트에 기여하는 수동적인 사용자가 있다.

ㅇ 코어 멤버

소유자 : 소유자는 비전을 정하고, 분명한 기술적 개발 작업 외에 프로젝트가 나아갈 방향에 대한 의사 결정, 배포와 적용될 라이선스 체계 관리, 프로젝트의 생존능력을 보장하기 위해 요구되는 비즈니스 모델을 수립한다.

코어개발자 : 코어개발자는 투표 권리와 코드 승인 권한을 가진다. 코드에 대한 가치 있는 기술 기여 외 배포관리 위원회를 포함하여 각종 위원회를 만들고 참여하는 것과 같은 좀 더 구체적인 책임을 가진다. 이 위원회는 프로젝트의 향후 배포 버전에 어떤 기능들이 포함될 것인지를 결정하는 책임을 가지며, 결정은 투표과정을 통해 이루어진다. 몇몇 코어 개발자들과 위원회는 관리자로도 알려져 있는데, 이들은 특정한 모듈 또는 코드 영역을 다루는 데에 대한 책임이 있다.

ㅇ 엑티브 멤버

액티브 개발자 : 액티브 개발자는 정기적으로 버그를 수정하고, 새로운 기능을 위한 코드 작업을 할 뿐만 아니라, 관련된 문서 작업까지 수행하는 프로그래머들이다. 이들에게는 소스 코드에 대한 폭넓은 이해와 프로젝트 아키텍처에 대한 깊은 이해도가 요구된다.

버그 수정자 : 버그 수정자는 버그 보고자에 의해 보고된 버그들을 수정한다. 그러므로 소스 코드에 대한 충분한 이해가 필요하다. 버그 수정자들은 특권을 행사하지는 않지만 액티브 개발자들에게 패치를 제출한다.

ㅇ 주변 멤버

주변 멤버는 커뮤니티의 대다수를 구성하며 프로젝트에 산발적으로 참여하고 지식을 교류하고 형성하는데 기여한다. 주변 멤버 중에 적극적인 참여 및 기여를 하는 경우 코어 멤버에 의해 승급될 수 있다.

주변 개발자 : 일부 국한된 버그 수정이나 마이너한 새로운 기능 개발에 가끔씩 기여한다.

버그 보고자 : 버그와 문제점들을 확인하고 보고한다.

액티브 유저 : 소스 코드에 관심을 갖고 있으며 때때로 검토와 논평을 통해 학습하기도 하며, 포럼에 참여한다.

제휴 단체나 기업 : 재정적 후원, 비즈니스 개발, 프로모션 등의 다양한 지원을 제공한다.

3. 커뮤니티의 운영

공개SW 커뮤니티를 운영하기 시작하면 새로운 개발자와 사용자가 외부에서 유입되기 때문에 그들이 본 프로젝트에 어떻게 기여해야 하는지에 대한 가이드라인이 필요하다. 예를 들어 페이스북의 react 프로젝트의 경우 CONTRIBUTING.md 파일에 기여자가 알아야 할 사항을 기술하고 있는데 대표적인 내용을 보면 아래와 같다.

- 행동 규칙(규범)
- 오픈 개발 : 리뷰 프로세스에 대한 적용이 외부 기여자뿐만 아니라 내부 핵심 개발자에도 적용됨을 명시함
- 브랜치 전략
- 유의적 버전 명명법을 사용함을 공지
- 버그 처리 방안
- 온라인 채팅 및 토론 방법
- Pull Request 사용 방법
- Contributor License Agreement(CLA) 작성 가이드 : Pull Request가 수용되려면 그 전에 기여자로서의 라이선스에 대한 합의가 전제되어야 함
- 기여 전에 전제되어야 할 사전 준비 사항
- 개발 워크플로우
- 스타일 가이드
- 소개 비디오

상기 내용 중 행동규칙은 특히 중요하다. 이는 커뮤니티에 참여하는 기여자뿐만 아니라 커뮤니티를 리딩하는 관리자의 입장에서도 매우 중요한데, 그 이유는 행동규칙을 사전에 정의하고 강제하면 커뮤니티에서 발생할 수 있는 좋지 못한 일들을 예방할 수 있기 때문이다.

행동규칙은 아래와 같이 만들 수 있다.

- 행동규칙이 적용되는 영역(오직 이슈 및 Pull Request에만 적용하는지, 이벤트와 같은 커뮤니티 활동에 적용하는지?)
- 누구에게 행동규칙이 적용되는지에 대한 사항
- 행동규칙이 지켜지지 않을 시에 어떤 제재 조치가 시행되는지에 대한 사항
- 행동규칙 위반 보고 방법
- 4만 여개의 오픈소스 프로젝트가 사용하고 있는 베스트 프랙티스 행동규칙인 Contributor Covenant를 가져도 쓰는 것도 대안이 될 수 있다.
- 행동규칙 파일(CODE_OF_CONDUCT)은 프로젝트 루트 디렉토리에 위치시키고 CONTRIBUTING 혹은 README 파일에 링크를 달아 가시성을 확보해 주는 것이 바람직하다.

행동규칙은 아래와 같이 시행할 수 있다.

- 행동규칙 위반 시에는 문제가 되는 사람에 대해 공식적인 경고를 주고 개별적으로 왜 문제가 되었는지를 설명해준다.
- 경고에도 불구하고 행동규칙 위반이 계속될 시에는 커뮤니티 참여를 일시적으로 중단시키거나 영구 제명을 할 수 있다.

커뮤니티 운영을 통한 개발 시 수행해야할 활동에는 멤버십 관리, 작업 할당, 의사 결정이 있다.

ㅇ 멤버십 관리

멤버십은 모두에게 열려 있으나 가입의 승인과 참여 수준의 관리는 프로젝트의 코어 멤버나 지정된 그룹에 의해 이루어진다. 이 그룹들은 멤버의 기여 수준과 스킬에 대한 평가에 기반을 두고 멤버를 승인하는 전체적인 권한을 가지고 있으며, 대개 투표 또는 합의절차를 거친다.

ㅇ 작업 할당

프로젝트에 참여하는 개발자의 개인적 선호는 가능하면 최대한 고려되고 존중되어야 한다. 개발자들이 할 수 있는 일을 최대한 많이 제시하는 대신 이를 엄격하게 통제하지 않고 개발자들의 선택으로 남겨둔다.

일반적으로 코드 소유권의 개념은 강력하며, 한 개발자가 프로젝트의 모듈 또는 섹션에 대해 대부분의 책임을 지며 다른 개발자는 2차적 기여를 한다.

원하는 협업과 분업이 이루어진 프로세스에서는 각 팀 간에 메일링 리스트 또는 버그와 이슈 저장소를 통해 의사소통하기 시작한다. 개인의 관심, 스킬과 역량, 그리고 자발적인 작업 할당에 기초하여 작업의 선택이 이루어지며, 개발자들의 기여를 촉진하는 동기 부여 또한 필요하다.

ㅇ 의사 결정

향후 릴리스 시 무엇을 포함할 것인지, 아키텍처 이슈들과 중요한 버그들을 어떻게 다룰 것인지 등과 같은 개발 방향에 대한 결정은 코어 개발팀이 수행하며, 이러한 결정들은 투표와 합의의 과정을 통해 이루어진다.

4. 커뮤니티의 활성화

공개SW의 지속적인 발전 및 개선을 위해서는 커뮤니티 운영뿐만 아니라 실제 활성화가 더욱 중요하다. 이를 위해 다양한 홍보방법이 존재하며, 아래에서는 이를 위한 구체적인 팁을 설명한다.

방법	홍보 팁
페이스북 활용	1) 현재 진행하고 있는 아이템과 유관한 내용의 페이스북 그룹은 무조건 가입한다.

방법	홍보 팁
	2) 그리고 그곳에 많은 글들을 올리고 일정 부분 활동 및 기여를 한다. 3) 또한, 해당 그룹이 진행하는 컨퍼런스나 세미나 등에 참여하여 얼굴을 익히는 것도 좋은 방법이다. 4) 마지막으로 당사의 아이템을 주제로 새로운 페이스북 그룹을 개설하여 회원들을 유도하여야 한다. 5) 커뮤니티 인원을 모으는 방법은 해당 주제에 관심이 있어 하는 사람들을 위한 글들을 많이 올리는 방법이 있다.
슬라이드 쉐어에 일정 부분 기여	1) 인지도를 높이기 위하여 슬라이드쉐어에 당사가 가진 노하우 중 일부를 공개하면서 기여하는 것도 하나의 방법이다. 2) 또한, 당사의 특정 개발자의 인지도를 높이기 위해 노력해야 한다. 3) 슬라이드 쉐어에 PPT를 올리다 보면 자연적으로 사람들이 알아보기 시작한다.
유튜브를 통해 공개	1) 당사의 활동이나 지식 관련 내부 회의의 모습을 영상화하고, 이를 유튜브에 공개하면 어느 정도 효과를 볼 수 있다. 2) 당연히 회사명과 오픈하는 아이템의 주제로 검색이 되도록 등록해야 한다.
세미나 및 밋업(meet-up) 행사 개최	1) 적어도 1년에 1 ~ 2번 정도는 세미나 또는 밋업을 통해 주변의 여러 기업들이나 커미터들이 관심을 가질 수 있도록 해야 한다. 2) 회사가 살아 움직인다는 느낌을 줄 수 있는 가장 저렴한 비용의 홍보 방법 중 하나이다.
국외 컨퍼런스에 관심과 참여	1) 가까운 나라 일본에서부터 멀리 미국까지 모든 나라의 컨퍼런스를 인터넷의 간단한 조회로 확인할 수 있다. 2) 이런 컨퍼런스는 참여가 쉬울 뿐만 아니라 한국의 KOTRA(대한무역투자진흥공사)를 활용하면 연계가 더 쉬울 것이다. 3) 컨퍼런스 발표를 할 수 있다면 반드시 Live demo 혹은 동영상 demo를 발표자료에 넣어 관심을 높이도록 한다. 4) 외국 개발자나 혹은 유명 개발자들을 직접 얼굴 보고 만나서 이야기할 수 있는 좋은 기회로써 아무래도 콜드 이메일(cold email)을 그냥 보내는 것보다는 한번이라도 얼굴 봤던 사람이면 같은 내용의 이메일을 보내도 읽어주고 관심 가져줄 확률이 높다. 5) 개발 프로젝트가 어느 정도 사용 가능한 수준이 되었다면, 직접 프로젝트를 활용한 튜토리얼(Tutorial) 세션을 진행하는 것도 좋다. 6) 국외의 특정 협회나 커뮤니티와 인연이 닿게 되면 역으로 한국에서 좋은 기회를 가질 수 있게 될 것이다.
Documentation API를 오픈	1) 오픈소스로 된 것들 중 Documentation API가 없이 성공하는 회사는 없다.

방법	홍보 팁
	2) 항상 마지막에는 Documentation API가 있어야 한다. 3) 또한 실행을 따라해 볼 수 있는 Step By Step 코드도 존재해야 한다.
기존 공개SW 활용에 대한 기여/협업 활동	1) 프로젝트에 기존 공개SW를 활용 시, 해당 SW의 Contributor 사이트에서 기능보완 등의 이슈를 요청하여 커미터들이 관심을 가질 수 있도록 해야 한다. 2) 가능하면 협업 체계를 갖도록 참여활동을 높여 프로젝트에 대한 해당 분야의 커미터들에게 간접적인 홍보효과를 기대할 수 있다.
메일링 리스트 활용	1) 국내에서는 별로 익숙하지 않지만, 외국 공개SW의 경우 프로젝트 단위 혹은 여러 프로젝트를 다루는 조직 단위의 메일링 리스트가 있어, 자신의 이메일 주소를 메일링리스트에 등록하는 식으로 구독한 후, 메일링 리스트로 메일을 보낼 수 있다. 2) 아무 때나 자신의 프로젝트 홍보를 하면 스팸으로 의심받기 쉬우므로 보통 Q&A에서 사람들이 필요로 하는 무언가를 해결해줄 수 있는 대안의 하나로 포지셔닝 해서 답변을 다는 식으로 진행한다. (지속적인 커뮤니케이션이 중요함) 3) 메일링 리스트의 분위기에 따라 관련 프로젝트로 직접 홍보하는 것도 가능할 수 있다.(예 : Python asyncio 기반 라이브러리들을 만드는 github.com/aio-libs는 asyncio 라이브러리 제작자들의 커뮤니티인데, 여기는 메일링 리스트와 gitter라는 Github 연동 채팅 룸을 운영한다. 이곳 메일링에는 자신이 만든 asyncio 기반의 라이브러리나 혹은 업데이트가 있을 때 릴리즈 소식을 공유하는 경우가 많은데, 이 메일링을 통해 홍보를 진행하여 다수의 외국 개발자들에게 star를 받는 효과 사례가 있음)
기타	1) 공개SW 진영의 유명 인사를 초청해서 강연을 듣고 친분을 쌓는 것도 좋은 방법. 이런 인사들과 워크숍을 함께 가면서 주변의 다른 사람들을 초대하게 만들면서 지인의 영역이 넓어지도록 하는 방법이다. 2) IT 신문을 통해 홍보하는 방법도 있으나 이는 비용부담이 크다. 3) 프로젝트의 첫 기여자나 사용자가 나왔을 때 - 이 프로젝트가 망하거나 버려지지 않을 거라는 믿음을 주어야 한다. - Name value가 없는 초반에는 프로젝트 초기 개발자가 집중적으로 이슈 대응을 해주는 것이 중요하다. - 사람들이 써보고 싶은 어떤 Painkiller가 있어야, 프로젝트에 부족한 점(버그 등)이 있을 때 사람들이 "아 이거 내가 고쳐서라도 써야겠구나"하는 생각을 하면서 기여자가 된다고 한다.

[표 6] 공개SW 커뮤니티 홍보 팁

또한, 상기에서 언급한 홍보 방법 등을 통해 새롭게 방문한 개발자 및 사용자들에게 우호적인 행동을 하고, 반갑게 맞이하는 것이 해당 프로젝트의 평판과 미래를 위한 중요한 투자이다. 이를 위한 간단한 팁을 아래와 같이 설명한다.

<사람들로 하여금 환영 받는다고 느끼게 하라.>

- 친절한 README 파일과 명확한 코드 예제는 프로젝트에 첫 발을 내딛는 사람들에게 큰 도움이 된다.
- CONTRIBUTING 파일을 사용하여 기여하는 방법을 명확히 설명하라.

<모든 것을 문서화하라.>

- 프로젝트 로드맵, 찾고 있는 기여 타입, 기여가 리뷰 되어 지는 과정
- 여러 사용자가 동일 문제를 겪고 있다면 README에 답변을 문서화할 것

<적절한 반응을 보이라.>

- 모질라 조사에 의하면 기여자가 48시간 이내 코드 리뷰를 받으면 해당 커뮤니티에 재방문하고 기여를 계속하는 비율이 훨씬 높아진다.

<모일 수 있는 공간을 커뮤니티에 제공하라.>

- 커뮤니티에 모이는 사람들은 동일한 관심사를 가지고 있는데, 이들이 서로 얘기할 수 있는 공간이 불가피하게 요구된다. 의사소통이 공개되고 접근 가능하면 이들이 지금까지 쌓인 자료를 읽고 빠르게 할 수 있게 된다.
- 프로젝트에 대해 얘기할 공간이 없으면 커뮤니티에 유입되는 사람은 직접 개인적인 메일을 발송하게 되고 프로젝트가 유명해지면 일일이 대응할 수 없는 어려운 지경에 이른다.

다음은 지속적으로 커뮤니티를 가꾸어 나가기 위해 유념할 사항이다.

<나쁜 행동을 용납하지 마라.>

- 어떤 유명 프로젝트도 불가피하게 도움이 되기보다는 해가 되는 사람을

끌어들이게 된다. 이들은 불필요한 논쟁과 사소한 기능에 대해 부각시킴으로써 중요한 일에 집중하지 못하게 하고 다른 사람들을 괴롭힌다.

- 받아들여질 수 없는 행동임을 분명히 말하고 커뮤니티를 떠나도록 요구한다. 이때 행동규칙 문서가 유용하다.

<기여자가 수행할 수 있는 일을 마련하라.>

- 이슈 중에 새로 참여한 사람이 해결할 수 있는 것에 예를 들어 “frist timers only”와 같은 라벨을 붙인다.

<프로젝트 소유권 공유>

- Pull Request를 요청한 기여자들에게 커밋 권한 주기 등 소유 의식을 갖게 하면 보다 적극적인 참여를 유도할 수 있다.

붙임1 공개SW 라이선스 개요

1. 라이선스 개요

공개SW 라이선스를 이해하기 위해서는 SW에 관한 지적재산권(Intellectual Property Rights, IPRs)과 라이선스(License)의 의미부터 이해해야 한다.

우선 SW는 SW에 관한 지적재산권에 의해 보호받고 있는데, 원칙적으로 저작권자만이 해당 SW에 대한 독점 사용 권리를 갖는다. 라이선스(License)는 이러한 독점 사용 권리에 대해 SW 개발자와 사용자 간의 이용방법 및 조건의 범위를 명시한 대여 규칙을 정의해 놓은 사용허가권(License)을 말한다. 대여의 방법은 유료 및 무료로 구분된다.

상용SW 판매 시 SW에 포함되는 최종 사용자 라이선스 협정(End User License Agreement)이 대표적인 유료 사용 허가권(Proprietary License)이며, 공개SW에 포함되어 배포되는 GPL, Apache 등의 라이선스가 대표적인 무료 사용 허가권(FOSS License)이다.

2. 지적재산권의 종류

지적재산권(IPRs)은 발명자에게 주어지는 창작물에 대한 배타적 독점권으로 SW에 관한 지적재산권은 저작권, 특허권, 상표권, 영업비밀 등으로 구성된다.

※ 단, 지적재산권의 보호는 공공의 발전을 목적으로 하므로 일정 보호기간이 지나면 퍼블릭 도메인(Public Domain : 저작권이 없거나 소멸)에 속하게 됨

○ **저작권(Copyright)** : 어문저작물(시, 소설, 논문 등), 음악저작물(노래 등), 연극저작물(연극, 무용, 무언극 등), 영상저작물, 컴퓨터프로그램 저작물 등 저작물에 대해 부여되는 권리이다. 저작권은 저작물을 창작한 때부터 발생하며 어떠한 절차나 형식의 이행을 필요로 하지 아니한다. 즉, 어떤 프로그래머가 SW를 개발하면 컴퓨터 프로그램 저작권이 자동으로 발생하며, 그 권리가 부여된다. 저작권은 저작자의 권리를 보호하고 저작물의 공정한 이용을 도모하기 위해 저작권법에 의해 보호된다.

○ **특허권(Patent)** : 발명을 했을 때 발생하는 권리를 말하며, 발명이란 특허법 상 "자연법칙을 이용한 기술적 사상의 창작으로서 고도한 것"이라고 정의된다. 특허권은 저작권과는 달리 발명을 하면 자연적으로 부여되지 않으며, 특허청에 특허출원하여 심사를 받고 설정등록을 해야만 권리가 발생한다.

○ **상표권(Trademark)** : 상표권은 상표권자가 지정 상품에 관하여 그 등록상표를 사용할 독점적인 권리로서, 상표란 상표법 상 "상품을 생산·가공 또는 판매하는 것을 업으로 영위하는 자가 자기의 업무에 관련된 상품을 타인의 상품과 식별되도록 하기 위하여 사용하는 표장"으로 정의된다. 상표권은 상표등록 출원하여 심사를 받고 설정등록을 해야만 권리가 발생한다.

○ **영업비밀** : 영업비밀은 부정경쟁방지 및 영업비밀보호에 관한 법률 상 "공공연히 알려져 있지 아니하고 독립된 경제적 가치를 가지는 것으로서, 상당한 노력에 의하여 비밀로 유지된 생산방법, 판매방법, 그 밖의 영업활동에 유용한 기술상 또는 경영상의 정보"로 정의된다. SW 분야 대표적인 영업비밀로는 소스코드를 들 수 있다.

위에서 설명한 지적재산권을 요약 정리하면 아래와 같다.

지적재산권 유형	정의	유의사항
저작권 (copyright)	창작물에 대하여 창작자(저작자)가 취득하는 권리로서 창작의 결과물을 보호하며, 창작과 동시에 권리가 발생한다. 따라서 어떤 프로그래머가 소프트웨어를 개발하면 저작권이 자동으로 발생하며, 그 권리는 프로그래머 또는 그가 속한 회사에 부여된다.	저작권이 있는 저작물은 저작권자의 허락이 없이는 누구도 해당 저작물을 복제, 배포, 수정할 수 없다.
특허권 (patent)	발명에 관하여 발명자(특허권자)가 갖는 독점배타권이다. 저작권과는 달리 일정한 방식으로 출원해야 하며, 심사를 통과한 후 등록되어야만 권리가 발생한다.	특허기술을 사용하기 위해서는 반드시 특허권자의 허락을 얻어야만 한다. 특허받은 방식을 구현하는 소프트웨어라면 프로그래밍 언어에 상관없이 특허권의 범위에 속한다.
상표권 (trademark)	상표권자가 지정 상품에 관하여 그 등록상표를 사용할 독점적인 권리로서 일정한 절차에 따라 등록하여야 효력이 발생한다.	상표를 사용하기 위해서는 반드시 상표권자의 허락을 얻어야 하며 허락받지 않고 상표를 이용할 경우 상표권의 침해에 해당한다.
영업비밀	공개되지 않은 정보	영업비밀은 일단 공개되면 더 이상 보호받기 어렵고, 또한 영업비밀을 알지 못하고 사용한 제3자에게 법적으로 문제를 삼을 수 없다는 한계가 있다

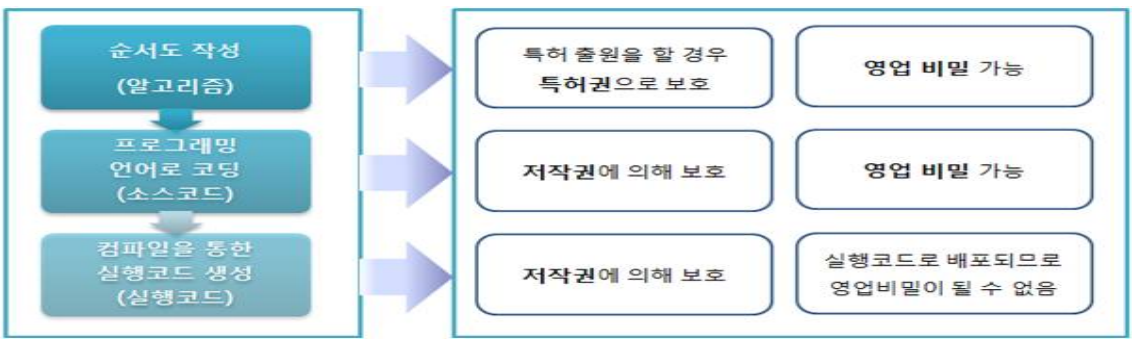
[표 7] 지적재산권의 유형

3. 공개SW 라이선스란?

위에서도 언급하였지만 **라이선스(License)**란 저작권을 가지고 있는 저작권자가 자신의 권리 중 일부분(사용, 복제, 재배포 등)을 일정 조건으로 이용자가 사용할 수 있도록 권한을 부여한 이용 허가권이다. SW의 경우 SW를 사용하고자 하는 자에 대해 일정한 범위와 방법으로 프로그램을 사용할 수 있도록 인정해주는 것이다. 그리고 이러한 계약은 저작권으로 보호되고 있어 이를 위반할 때에는 법적 책임을 지게 된다.

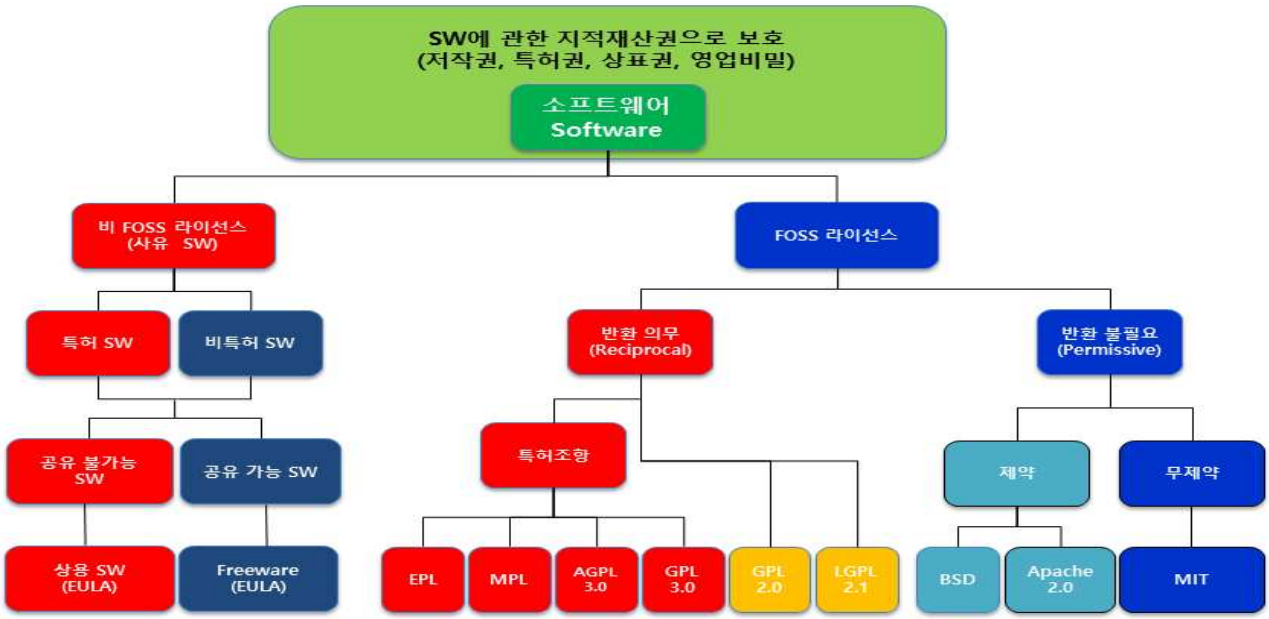
라이선싱(Licensing, 사용허락계약)은 저작권자(Licensor, 라이선서)가 저작권 또는 특허권 자체는 이전하지 않은 상태에서 일정한 계약기간 동안 일정한 지불 조건(유·무상)으로 사용이 허가된 상품 또는 서비스업에 대한 제한된 권리를 제3자(Licensee, 라이선시)에게 허락하는 계약을 의미한다. 즉, 저작권자가 가치 있는 상업적 자산권의 일정 영역을 계약기간 동안 양도하는 행위로 규정할 수 있다.

SW는 저작권과 특허권으로 중복하여 보호되는 경우가 있으므로 동일 라이선스 내에 저작권과 특허권에 관한 사항을 함께 기재할 수 있다. [그림5]는 SW 개발단계에 따른 법적 보호범위를, [그림6]은 지적재산권의 보호범위와 SW 라이선스의 구분을 보여주고 있다.



[그림 5] SW 개발 단계에 따른 법적 보호 범위

공개SW 라이선스란 공개SW 개발자와 이용자 간의 사용 방법 및 조건의 범위를 명시한 계약을 말한다. 따라서 공개SW를 이용하려면 공개SW 개발자가 만들어놓은 조건의 범위에 따라 해당 소프트웨어를 사용해야 하며, 이를 위반할 경우에는 라이선스 위반 및 저작권 침해에 대한 법적 책임을 져야 한다.



[그림 6] 지적재산권의 보호범위와 SW 라이선스의 구분

4. 주요 공개SW 라이선스별 의무사항

공개SW 라이선스는 소스코드 공개 의무 및 독점 SW와의 결합방식을 기준으로 세 가지 유형으로 분류가 가능하다. 공개SW의 라이선스 유형은 아래와 같다.

free-for-all	원 저작물(original)에 대한 저작권 표시만 한다면, **자유롭게 독점SW와 결합이 가능한** 유형의 라이선스
keep-on	독점 라이선스 등 다른 라이선스 조건하의 소스코드와 결합 가능하지만, **공개SW를 이용하여 만든 개작물은 반드시 공개**해야 하는 유형의 라이선스
share-alike	**원 저작물 및 개작 저작물 모두를 무조건 공개**해야하는 유형의 라이선스

[표 8] 공개SW 라이선스 유형

또한, 주요 공개SW 라이선스의 의무사항을 비교하면 아래와 같다.

의무사항 \ 라이선스	Commercial License	GNU General Public License v2.0 or later	GNU General Public License v3.0 or later	GNU Lesser General Public License v2.1 or later	GNU Lesser General Public License v3.0 or later	GNU Affero General Public License v3.0	Eclipse Public License 1.0	Mozilla Public License 1.1	Apache License 2.0	BSD 3-clause "New" or "Revised" License
배포권리(오브젝트/바이너리코드 배포)	X	X	X	X	X	X	X	X	X	X
배포(코드배포에 의해서만 부여되는 의무사항)	X	O	O	O	O	O	X	O	X	O
소스코드 배포 /강제적 공유 의무사항	X	O	O	O	O	O	O	O	△	△
복제 권한 허용	X	O	O	O	O	O	△	△	△	△
수정(개작)권한 허용	X	O	O	△	O	O	△	△	△	△
역설계 권한 허용	X	O	O	O	O	O	△	△	△	△
차별적 제한 금지	X	O	O	O	O	O	△	O	△	X
추가 복제에 대한 로열티 및 수수료 금지	X	O	O	O	O	O	O	O	△	△
특허보복(특허소송 제기 시 라이선스 종료)	X	X	O	X	O	O	O	O	O	X
명시적 특허라이선스 (특허소송을 제기하지 않음)	X	X	O	X	O	O	O	O	O	X
DRM 금지	X	X	O	X	O	O	O	O	O	X
고지(특정법률 혹은 속성)	X	O	X	O	X	O	X	X	X	X
변경사항 고지	X	O	O	O	O	O	O	O	O	X
변경사항에 대해 원저작자에게 사용허가	X	X	O	X	O	O	O	O	X	X
다른 사람을 대신한 보증의 부인	O	O	X	O	X	X	O	O	O	X
다른 사람의 책임의 제한	O	O	X	O	X	X	O	O	O	X
배포 /사용으로 인해 발생한 원저작자의 클레임에 대한 배상	X	X	X	X	X	X	O	O	O	X
배포 시 라이선스 사본 포함	X	O	O	O	O	O	O	O	O	O
광고/홍보 시 배포자, 저작자, 특정상표사용 금지	X	X	X	X	X	X	X	O	O	O
원 코드와 동일조건 허가	X	O	O	O	O	O	O	O	X	X
라이선스 확장 범위 (공개범위)	X	코드 기반의 산출물 (per GPL)	코드 기반의 산출물 (per GPL)	동적 라이브러리 (per LGPL)	동적 라이브러리 (per LGPL)	코드 기반의 산출물 (per GPL)	모듈 (per EPL/CPL)	파일 (per MPL)	X	X

O : 필수 의무사항, X : 필수 의무사항 없음, △ : 명문화된 조항 없음

[표 9] 공개SW 라이선스별 의무사항 비교

개발된 산출물 공개 시 법률적 문제를 겪지 않기 위해서는 공개SW 라이선스의 의무사항을 반드시 이해해야 한다. 아래 표는 주요 라이선스별 의무사항과 소스코드 공개 범위를 설명한다.

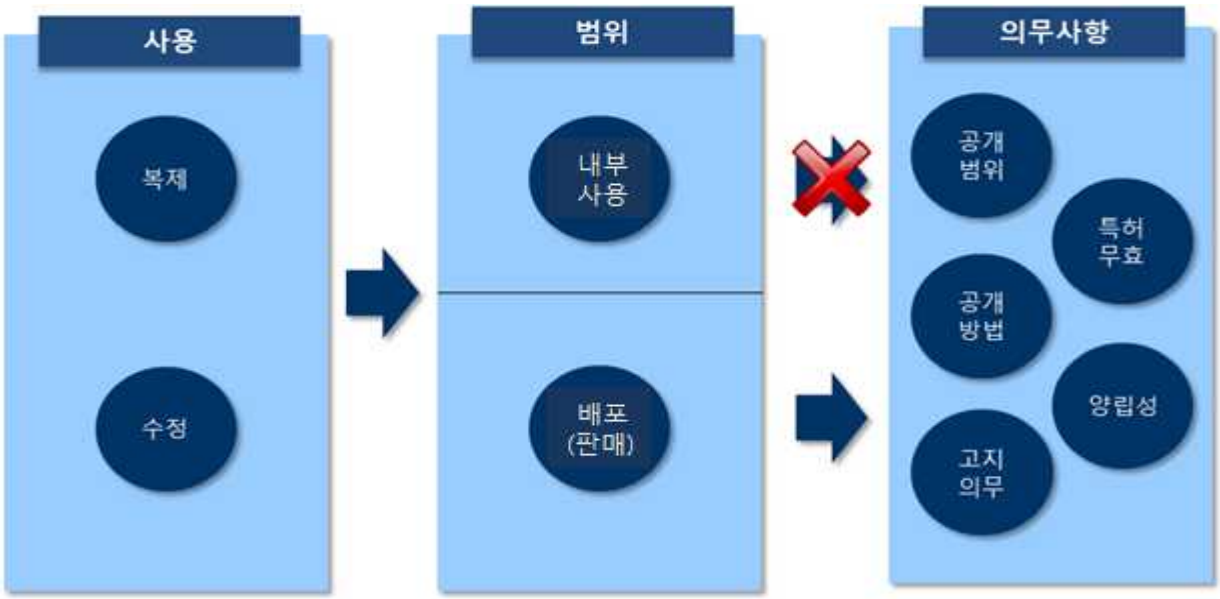
주요 라이선스	주요 의무사항	공개 범위
GPL 2.0	- 자유로운 사용, 복제, 배포, 수정 허용 - 저작권 표시, 보증 책임이 없다는 표시 - 영문 라이선스 사본 포함 - 배포 시 GPL로 배포 - 바이너리로 배포 시 약정서 제공	- GPL 코드(수정코드 포함) 및 링크 시 모든 코드를 GPL에 의해 공개 (단, 결합형태에 따라 예외조항 있음)
GPL 3.0	- 자유로운 사용, 복제, 배포, 수정 허용 - 저작권 표시, 보증 책임이 없다는 표시 - 영문 라이선스 사본 포함 - 배포 시 GPL로 배포 - DRM 금지 - GPL3.0 코드에 포함된 특허를 사용자에게 무상 허용 - 사용자에게 특허 소송시 보복(라이선스 취소, 특허 허여 취소) - 바이너리로 배포 시 약정서 제공	- GPL 코드(수정코드 포함) 및 링크 시 모든 코드를 GPL에 의해 공개 (단, 결합형태에 따라 예외조항 있음)
LGPL 2.1	- 자유로운 사용, 복제, 배포, 수정 허용 - 라이브러리 링크만을 허용 - 역설계 권한 허용 - 저작권 표시, 보증 책임이 없다는 표시 - 영문 라이선스 사본 포함 - 배포 시 LGPL로 배포 - 바이너리로 배포 시 약정서 제공	- LGPL 라이브러리 코드(수정코드 포함) 및 정적링크된 응용프로그램의 목적코드 공개(단, 동적링크시 소스코드 공개 없으며, 정적링크 시 목적코드만 제공)
AGPL 3.0	- 자유로운 사용, 복제, 배포, 수정 허용 - 저작권 표시, 보증 책임이 없다는 표시 - 영문 라이선스 사본 포함 - 배포 시 APL로 배포 - 바이너리로 배포 시 약정서 제공	- AGPL 코드(수정코드 포함) 및 링크 시 모든 코드를 AGPL에 의해 공개 - 네트워크로 통신하는 SW에 포함시 GPL 3.0이 적용되어 GPL 3.0에 의한 소스코드 공개의무 발생
MPL	- 자유로운 사용, 복제, 배포, 수정 허용 - 저작권 표시, 보증 책임이 없다는 표시, MPL 명시 - 영문 라이선스 사본 포함 - 배포 시 MPL로 배포 - MPL 코드에 포함된 특허를 사용자에게 무상 허용 - 사용자에게 특허 소송 시 보복(라이선스 취소, 특허 허여 취소	- MPL 소스코드 수정 사용 시 (Modifications) MPL 코드가 포함된 파일을 공개
EPL	- 자유로운 사용, 복제, 배포, 수정 허용 - 저작권 표시, 보증 책임이 없다는 표시, EPL 명시 - 영문 라이선스 사본 포함 - 배포 시 EPL로 배포 - EPL 코드에 포함된 특허를 사용자에게 무상 허용 - 사용자에게 특허 소송 시 보복(라이선스 취소, 특허 허여 취소	- EPL 소스코드 수정 사용 시 (changes to the Program) EPL코드가 포함된 모듈을 공개

주요 라이선스	주요 의무사항	공개 범위
Apache 2.0	- 자유로운 사용, 복제, 배포, 수정 허용 - 저작권 표시, 보증 책임이 없다는 표시 - 영문 라이선스 사본 포함 - 사용 및 변경사항에 대한 고지 - 배포 시 Apache 라이선스 2.0으로 배포 - Apache 코드에 포함된 특허를 사용자에게 무상 허용 - 사용자에게 특허 소송 시 보복(라이선스 취소, 특허 허여 취소	- 코드 공개의무는 없으나 사용고지 및 변경사항에 대한 고지의무 준수 필요
BSD	- 자유로운 사용, 복제, 배포, 수정 허용 - 저작권 표시, 보증 책임이 없다는 표시 - 영문 라이선스 사본 포함	- 소스코드 공개 의무 없음
MIT	- 자유로운 사용, 복제, 배포, 수정 허용 - 저작권 표시, 보증 책임이 없다는 표시 - 영문 라이선스 사본 포함	- 소스코드 공개의무 없음

[표 10] 주요 공개SW 라이선스별 의무사항과 공개 범위

5. 공개SW 라이선스 의무사항 적용 범위

공개SW는 사용 범위에 따라 의무사항 적용이 달라지는데, **내부에서 사용할 경우에는 의무사항의 적용을 받지 않고,** 외부에 배포 및 판매할 경우에 의무사항의 적용을 받게 된다.



[그림 7] 공개SW 라이선스의 의무사항 적용 범위

6. 듀얼 라이선스

공개SW는 원래의 라이선스 외에 **예외적 사용을 허용하는 듀얼 라이선스**(Dual License) 정책을 운영하는 경우가 있다. 듀얼 라이선스 정책 적용 시 개발자는 자신에게 유리한 라이선스를 선택할 수 있다.

※ 예를 들어 'GPL or MIT' 형식의 듀얼 라이선스는 해당 소스코드를 GPL로 가져오거나 MIT로 가져올 수 있다는 의미로, 자신이 개발하고 있는 SW가 상용SW일 경우 소스코드 공개의무가 없는 MIT로 가져와서 사용할 수 있다는 의미이다.

※ 'GPL or Commercial' 형식의 듀얼 라이선스는 Commercial 라이선스의 구입을 통해 GPL의 소스코드 공개 등 의무사항 적용 없이 상용SW에 포함할 수 있다.

- (사례) MySQL은 GPL 2.0으로 배포되고 있으나, 오라클사의 상용 라이선스를 구입하면 응용SW의 소스코드를 공개하지 않아도 됨

붙임2 공개SW 비즈니스 기획

공개SW 라이선스 선정 시 향후 사업화 방향을 고려하여야 하는데, 이는 라이선스의 종류에 따라 비즈니스 모델이 달라지기 때문이다. 이에 본 장에서는 공개SW 기반의 비즈니스 모델을 살펴본다.

비즈니스 모델은 일반적으로 수익의 원천, 타겟 시장, 제품의 개발 및 영업 방식 그리고 경쟁(또는 협력)자들과의 관계로 특징지어진다. 특히 수익의 원천은 비즈니스 모델을 분류하는 기준이며, 그 성과 또한 수익을 토대로 판단된다.

그럼에도 불구하고, 기업들이 공개SW 개발에 대한 투자를 어떤 방법으로 회수하는지(즉, 수익을 내는지)에 대한 관심 이외에도, 그들이 공개SW 개발 모델이 제공해 줄 수 있는 많은 장점들을 어떤 방식으로 최대한 활용하는가를 분석하는 것도 중요하다. 그 장점들 중 하나가 협력적 경쟁이라는 개념인데, 이는 독점적SW 산업 생태계와 구분되는 공개SW 산업 생태계의 독특한 특징이다. 이는 공개SW 개발과 그에 따른 비즈니스 전략을 수립하는데 있어서 매우 중요하다.

공개SW 모델의 적용은 기업 비즈니스에 많은 전략적 이점들을 제공할 수 있으며, 다양한 방법으로 회사 또는 조직 운영에 영향을 줄 수 있다.

사용자 기반과 커뮤니티	시장 위상과 경쟁	수익 모델 및 재정
- 사용자 기반 개발 - 시장에 대한 정보 - 혁신의 전파 - 생산성 증가 - 고객 요구사항 만족 - 외부 개발자 활용 - 새로운 스킬 및 프랙티스 습득	- 제한된 시장으로의 접근 - 명성 - 경쟁자 공격 - 비공개 표준에 대항한 개발 선점 - 약자 포용 정신 - 벤더 종속 탈피	- 새로운 서비스 창출 - 보완 서비스 요구 증가 - 개발 비용 절감 - 낮아진 손익 분기점 - 새로운 수익모델 도입

[표 11] 공개SW 비즈니스의 전략적 이점

본 장에서는 공개SW 비즈니스 모델을 살펴보고, 기업 측면에서 공개SW 비즈니스를 위해 무엇을 고려해야 할 것인지 살펴본다. 이후 공개SW 활용 목적에 따라 적합한 모델을 선정하고 이를 위한 전략수립을 모색하는 것을 지원한다.

1. 공개SW 비즈니스 시 고려사항

기업에서는 경쟁사와의 차별화, 시장 확대, 수익 구조 확보 등의 목적으로 공개SW를 활용할 수 있는데, 제품을 공개SW로 제공할 것인지에 대한 의사결정 과정에서 아래와 같은 사항들을 반드시 고려하여야 한다.

항목	고려사항
타겟 제품의 시장 평가	- 상업적 제품과 공개SW로의 제공, 또는 두 가지 결합 고려 - 공개SW가 될 제품에 대한 시장의 관심도 판단
개발 커뮤니티의 관심도 판단	- 제품이 공개SW가 되었을 때 커뮤니티의 개발자들이 얼마나 형성되고 전문 지식과 개발 노력을 제공해 줄 것인지 고려 - 포럼, 메일링 리스트, 커뮤니케이션 채널 등에 대한 통찰 필요
제품의 어떤 부분들을 공개SW로 제공할 것인지 결정	- 일부만 공개SW로 제공, 나머지는 사유로 유지 - 영업비밀, 미공개 알고리즘 증
단기 전환 비용의 균형	- 새로운 공개SW 버전과 이전 버전과의 호환성 유지 - 기존 고객 이탈 등 손실 비용
새로운 프로세스, 인프라와 환경을 고려	- 오픈 소스 프로젝트와 커뮤니티를 형성
조직 내의 올바른 정신이 존재하는지 확인	- 조직원의 지적재산권 존중
코드 정제	- 코드 배포를 위한 재작성, 리팩토리 등

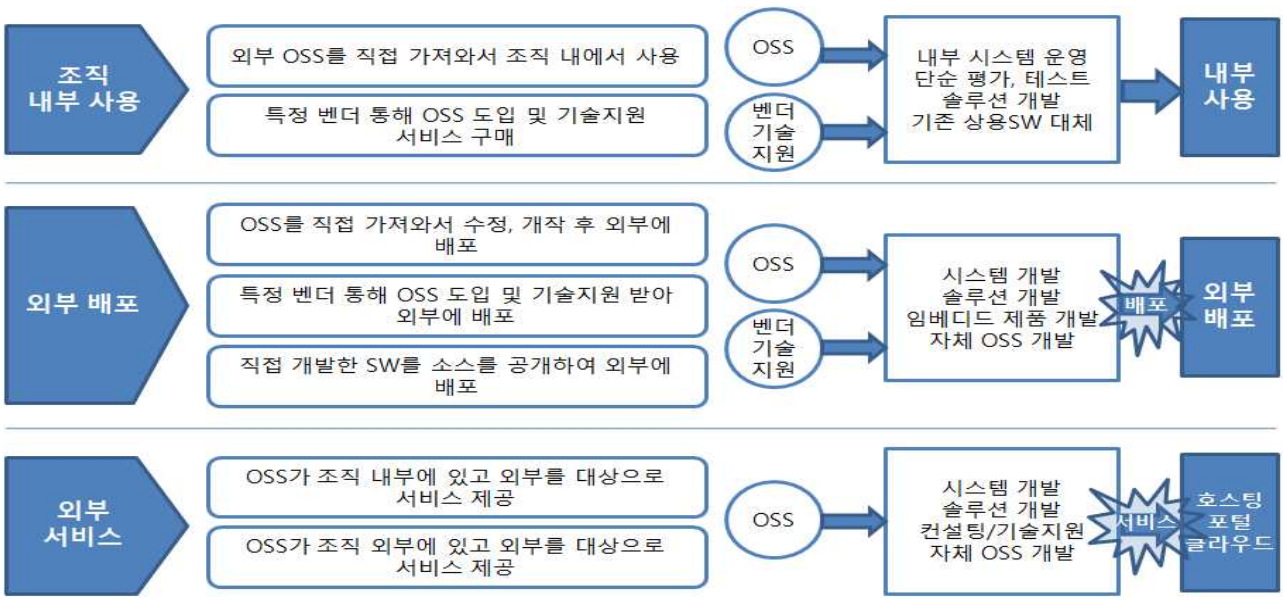
항목	고려사항
적절한 비즈니스 모델의 선택	- 목적에 따른 비즈니스 모델 선정
적절한 라이선스의 선택	- 목적을 충족하는 라이선스 채택
마케팅 방식의 결정	- 공개SW 제품에 대한 인지도를 구축

[표 12] 공개SW 여부 판단을 위한 고려사항

이러한 고려사항은 공개SW 비즈니스를 위한 모델 선정, 전략수립, 개발 계획 수립, 개발 활동 전반에 걸쳐 관심에 가지고 검토하여야 한다.

2. 공개SW 사용자 유형

공개SW 비즈니즈 모델을 살펴보기 전에 먼저 공개SW 사용자 유형을 살펴보자. 사용자 유형은 내부사용, 외부 배포, 외부서비스로 분류할 수 있다.



[그림 8] 공개SW 사용자 유형

각 사용자 유형별 특징을 정리하면 아래와 같다.

유형		특징
내부 사용	1) 외부의 공개SW를 직접 가져와서 조직 내에서 사용	- 공개SW 커뮤니티로부터 소스코드를 직접 가져와 외부의 지원이나 간섭이 없이 자체적으로 설치하고 운영
	2) 특정 벤더사를 통해 공개SW도입 및 기술지원을 받으며 사용하는 경우	- 자체적인 기술이나 역량이 부족하여 외부 전문업체의 도움을 받아 쌍방간의 계약 조건에 따라 차별화된 서비스 제공
외부 배포	3) 공개SW를 직접 가져와서 개작 후 외부에 배포하는 경우	- 공개SW 활용단위 : 일부 또는 전체 - 라이선스 정책 등 거버넌스 활동이 중요
	4) 특정 벤더사를 통해 공개SW 도입 및 기술지원을 받아 외부에 배포하는 경우	- SI업체가 전문적인 공개SW 업체를 통해서 SW를 납품하거나, 사용 주체와 배포 주체 간의 공급 계약에 의해서 지리적으로 떨어진 사용 주체 등을 대상으로 지원
	5) 직접 개발한 소프트웨어를 공개SW로 외부에 배포하는 경우	- 외부의 공개SW를 재사용하여 내부적으로 독점SW를 개발한 이후 특정 공개SW 라이선스를 붙여서 해당 SW의 소스코드를 공개 - 상용SW를 공개SW로 전환하여 소스코드를 공개
외부 서비스	6) 공개SW가 내부에 있고 외부 서비스를 제공하는 경우	- 사용자가 공개SW를 직접 개발하는 생산자가 아니지만 사용자의 조직 내부에 공개SW를 획득하고 이를 활용하여 외부 고객에게 다양한 서비스를 제공
	7) 공개SW가 외부에 있고 외부 서비스를 제공하는 경우	- 서비스 공급자가 기업 외부에 있는 공개SW를 활용하여 외부 고객에게 다양한 서비스를 제공

[표 13] 공개SW 사용자 유형별 특징

본 가이드라인은 공개SW 개발에 중점을 두고 있어 사용자 유형 중 외부 배포만을 대상으로 하며, 그 중 비중이 높은 3), 5) 유형만 다룬다.

3. 공개SW 비즈니스 모델

개방형 협력을 통해 품질향상, 비용절감, 고객참여 등과 같은 많은 이점들을 얻을 수 있음에도 불구하고 경쟁자들이 일부 혜택을 보는 것은 불가피하다. 그렇기 때문에 공개SW 비즈니스 모델의 성공사례들을 통해 이와 같은 우려를 사업상의 장점으로 승화시키기 위한 방법을 찾는 것이 필요하다. 그러나 이에 대한 해답을 찾아 나가는 과정에서 명심해야 할 것은

- **공개SW 비즈니스에 있어서의 핵심 가치는 경쟁기업들과의 관계에서가 아닌, 고객(커뮤니티)들과의 관계에 있다는 점**

- **공개SW 비즈니스에 접근하려는 기업들의 핵심적인 이슈는 '고객 가치와 비즈니스 가치의 일치'** 이다.

아래 그림은 일반적인 SW 비즈니스의 세 가지 모델(개발 모델, 서비스 모델, 개발과 서비스를 병행하는 하이브리드 모델)을 기준으로 분류한 공개SW 기반 비즈니스 모델들이다.

[그림 9] 공개SW 기반 비즈니스 모델 분류

○ 개발모델 : 특정 프로젝트의 기획자 또는 리더로서 공개 소프트웨어를 개발하는 모델

○ 서비스 제공 모델 : 규모 및 솔루션의 세분화, 산업 분야의 세분화, 특정 서비스에 대한 전문화를 통한 여러 가지 세부적인 모델이 존재

- 맞춤형 개발
- 선정
- 설치 및 통합
- 기술적인 인증, 지원, 유지보수
- 교육/훈련

○ 개발-서비스 혼합 모델 : 개발한 제품을 공개SW로 공개하고 수익은 서비스를 통해 창출하는 유형

○ 하드웨어 보조 모델 : 하드웨어 제품을 판매하기 위해 SW 공개

○ 기타 보조 모델 : 커뮤니티 호스팅, 서적 판매, 상품판매 등

각 비즈니스 모델별 세부 분류와 내용은 아래와 같다.

비즈니스 모델			설명	고려사항
개발 모델	순수 OSS 개발 모델	사용 가치 획득 모델	- 판매가치보다 사용가치가 소프트웨어 개발의 주된 동인 - 폐쇄 소스보다 오픈소스 개발 모델이 더 효율적이고 효과적인 방법 - 예상되는 사용가치만으로도 오픈소스 커뮤니티를 통한 개발이 이득이 있다고 판단	핵심 자산을 보호하면서, 커뮤니티를 통한 개발로 비용, 보안 등 제품 개선 목적 달성 예) 회계프로그램 : DB엔진을 공개하면서, 핵심자산인 스키마(사업정보)는 보호
		분산 (소액) 판매 모델	- 완전히 자원봉사자로 구성되는 비영리 오픈소스 소프트웨어 프로젝트일 경우	기업 비즈니스로 부적합

비즈니스 모델			설명	고려사항
	OSS - 독점적 제품 혼합 모델	듀얼 라이선싱 모델	- 해당 공개SW를 제품에 내장하여야 하는 상업적인 목적의 사용자층이라는 시장이 있을 때 가능 - 아무런 반대급부도 받지 못하면서 기여하는 자발적 참여자들의 동기에 부정적인 영향을 미칠 수 있으므로 공개SW 제품을 위하여 협업하는 사람들의 커뮤니티를 유지하기 위한 각별한 노력이 필요함	1. 개작 후 배포 경우 : 개작 또는 추가하는 기능을 독점SW로 하기 위한 방안 검토 2. 독점SW를 공개SW로 전환 : 소스 공개 여부, 비즈니스 형태 등을 고려해 어떤 라이선스를 선정할 것인지가 중요
		독점적 부가 제품 모델	- 하나의 프로그램이 기본 버전과 기본 버전에 기반을 둔 플러그인 또는 액세서리의 형태로 추가적인 기능을 갖춘 독점적 상업적 버전이라는 두개의 버전을 가짐	- 무료와 독점적인 버전을 모두 유지하기 위한 라이선스 선정 - 무료버전의 커뮤니티 유지 중요
서비스 제공 모델	플랫폼 유통 모델		- 다양한 컴포넌트들을 인증되고 미리 테스트된 스택 형태로 제공 - 기업들을 위한 통합된 SW 패키지 (주로 플랫폼)	- 기업을 대표하는 브랜드의 신뢰성이 중요 - 가입제도(Subscription) - 설치 및 지원 서비스에서 차별화 중요
	컨설팅 및 시스템 통합 모델		- 당면한 문제 또는 필요에 대한 완전한 해결책을 제공	- 기존 SI보다 뛰어난 가격과 품질을 보장 중요
	기술지원 및 교육/훈련 서비스 모델		- 소규모의 기술 인력만으로 서비스	- 공개SW의 공개성으로 해당 소프트웨어 전문가면 제공 - 교육/훈련 비즈니스 업체
개발-서비스 혼합 모델	OSS-전문가 서비스 모델		- 사용자 직접 대응이 아닌 IT서비스 기업(협력업체)에 대한 전문서비스	- 제품에 대한 신뢰성을 제공하고 파트너사에게 다양한 지원이 중요
	OSS-SaaS 모델		- 인터넷을 통해 사용자 기능(SW 및 HW 인프라까지 포함)을 제공하면서 서비스 가입비의 형태로 수익을 창출	- 소스코드 공개에 대한 결정 필요

비즈니스 모델	설명	고려사항
하드웨어 보조모델	- 하드웨어 공급업체가 원가를 절감하는 하나의 방법으로써 오픈소스를 개발 또는 채택	- 기업 비즈니스로 부적합
기타 보조모델	- 커뮤니티 호스팅 법률적 인증서적 판매, 상품판매, 광고 등	- 기업 비즈니스로 부적합

[표 14] 공개SW 기반 비즈니스 모델 분류

본 가이드라인은 공개SW 개발 및 서비스에 대한 내용을 다루고 있으므로 상기 모델 중 기업 비즈니스로 부적합한 모델은 제외한다.

개발모델에서는 해당 기업들이 공개SW에 관여하는 것이 매우 중요하며, 커뮤니티 관리와 그 과정에서 혁신, 확산 및 자발적 기여라는 기회 요인을 획득하는 것이 사업전략의 핵심이다.

이러한 모델들은 커뮤니티를 위한 오픈소스 제품과 상업적 제품 또는 관련 서비스를 함께 가지고 있는 경우가 많으며, 따라서 이들 둘 사이의 균형을 취하는 것이 핵심(즉, 공개SW 라이선스와 결합된 모델을 유지하면서 어떻게 개발에 대한 최초의 투자를 회수하느냐가 중요함)이다.

이러한 공개SW 전략을 채택하는데 따르는 장점은

- 관련 제품의 매출을 촉진하는 것 외에도
- 해당 기술을 사실상의 표준으로 자리 잡게 하거나,
- 제품을 보다 매력적인 것으로 만들기 위한 개선 및 보완을 달성하거나,
- 제품의 잠재적인 고객을 포함한 청중들의 공감대를 형성하거나,
- 유지보수 원가를 절감하는 등이 있다.

4. 공개SW 비즈니스 모델별 전략 수립 시 고려사항

본 장에서는 공개SW 사용자 유형 중 외부배포 유형을 비즈니스 모델과 연계하고, 이에 따른 고려사항을 정리하였다. 외부 배포 유형 중 공개SW를 직접 가져와서 개작 후 외부에 배포하는 경우, 저작권 침해 및 컴플라이언스 정책, 재배포 이슈, 서비스 범위 합의, 해당 커뮤니티 활성화 및 고객의 VoC 반영, 커뮤니티 참여 및 기여 수준에 따른 정책, 커밋 유도 등의 활동이 중요하다.

반면, 직접 개발한 SW를 공개SW로 외부에 배포하는 경우, 소스 공개범위 결정, 자체 개발 부분의 라이선스 지정, 배포를 위한 새로운 커뮤니티 또는 기존 커뮤니티 활용 등이 중요하다.

아래 표는 공개SW 사용자 유형을 비즈니스 모델로 매핑하고 각 모델별 비즈니스 케이스와 전략수립 시 고려사항을 정리하였다. 향후 비즈니스 모델 설계 시 유사 케이스 검토를 통해 비즈니스 전략 수립에 활용하면 된다.

공개SW 사용자 유형		비즈니스 모델	비즈니스케이스	전략수립 시 고려사항
외부배포	기존 공개SW를 활용하여 개작 후 외부에 배포하는 경우	듀얼 라이선싱 모델	공개SW를 개작 후 공개 및 판매로 수익 창출	저작권 침해 및 플라이언스 정책 중요 - 소스코드 배포없이 상용으로 하려면 라이선스 비용을 지불 - 커뮤니티 참여 및 기여 수준에 따른 정책, 커밋 유도 등 활동 중요 - 커뮤니티 형성 및 고객의 VoC 반영 중요
		독립적 부가제품 모델	공개SW를 기반으로 플러그인 제품 또는 액세서리 형태의 추가	저작권 침해 및 컴플라이언스 정책 중요 - 공개SW가 GPL계열이면 상용화를

공개SW 사용자 유형		비즈니스 모델	비즈니스케이스	전략수립 시 고려사항
			기능 제품 판매로 수익 창출	위한 듀얼 라이선스 여부 확인
		플랫폼 유통 모델	다양한 컴포넌트들을 인증되고 미리 테스트된 스택 형태로 제공	브랜드의 신뢰성과 지원서비스의 차별화가 중요
	직접 개발한 소프트웨어를 공개SW로 외부에 배포하는 경우	사용가치 획득 모델	핵심 기능은 제외하고 공개하여 지속적인 기능 및 성능 개선을 함	- 핵심기능을 분리하기 위한 설계기술 필요 - 커뮤니티 참여 및 기여수준에 따른 정책, 커밋 유도 등 활동 중요 - 고객의 VoC 반영 중요
		듀얼 라이선싱 모델	엔터프라이즈 버전은 보유하고, 저변 확대를 위해 공개SW로 공개	소스 공개 범위 및 사업 범위에 따라 라이선스 선정 중요 - 예) 무료버전은 GPL, 확장 버전은 독점
		독립적 부가제품 모델	공개SW를 기반으로 플러그인 제품 또는 액세서리 형태의 추가 기능 제품 판매로 수익 창출	- 독점 제품을 생성하기 위해서는 무료버전은 반드시 MPL 또는 BSD 형태의 라이선스를 사용 - 무료 버전은 기여, 원가 절감, 제품 혁신 및 진화와 같은 공개SW 장점을 최대한 활용하고, 독점적 제품은 사용 편의성, 인터페이스, 안정성 등에 집중
		컨설팅 및 시스템통합	개발 SW를 공개SW로 공개하고 SI(커스터마이징) 등 컨설팅	- 자체 개발 부분의 라이선스를 먼저 지정 - 기존 SI보다 뛰어난 가격과 품질을 보장 중요
		공개SW-전문가 서비스	개발 SW를 공개SW로 공개하고 전문가 지원으로 수익 창출	제품에 대한 신뢰성을 제공하고 파트너사에게 다양한 지원이 중요
		공개SW-	개발 SW를 공개SW로	SaaS 모델에 따라 제공되는

공개SW 사용자 유형		비즈니스 모델	비즈니스케이스	전략수립 시 고려사항
		SaaS 모델	공개하고 SaaS 서비스로 수익 창출	기능의 소스코드 공개를 요구할 경우 Aferro GPL 라이선스 채택

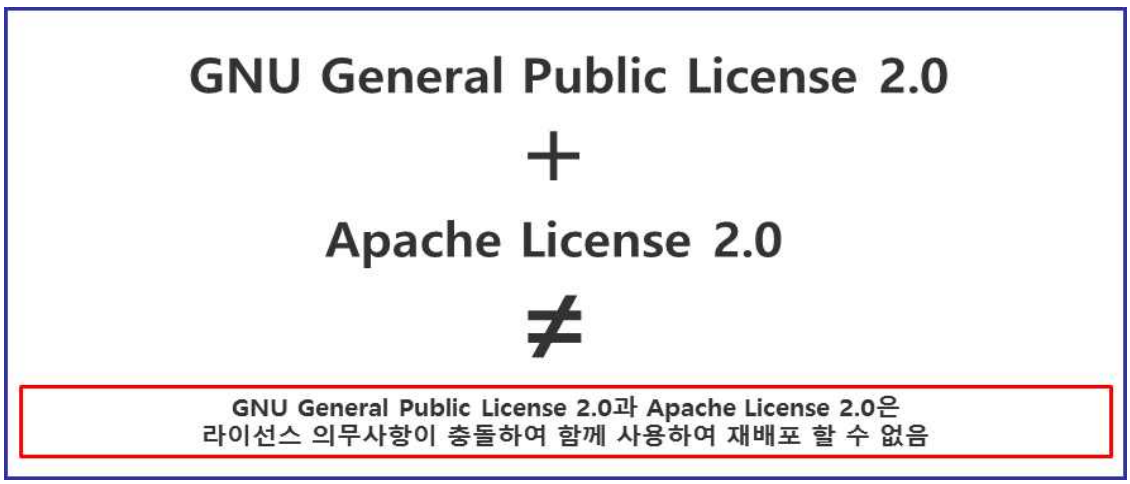
[표 15] 공개SW 비즈니스 모델별 비즈니스 케이스와 고려사항

공개SW를 개작하여 외부에 배포하는 경우가 저작권 침해나 컴플라이언스 정책이 매우 중요하며, 직접 개발한 SW를 공개SW로 외부에 배포하는 경우에도 영업기밀 등을 고려해 라이선스 정책을 지정하여야 한다.

붙임3 공개SW 라이선스의 양립성

1. 개요

공개SW 라이선스의 양립성이란 서로 라이선스 선정 시 다른 의무사항을 가진 라이선스 간 양립(Compatibility)이 가능하도록 검토해야 함을 의미한다.



[그림 10] 공개SW 라이선스의 양립성 검토

일반적으로 프로젝트 개발 시 GPL 라이선스와의 호환성만 검토하여도 상당부분의 라이선스 위반을 사전에 방지할 수 있다. 아래 표는 주요 공개SW 라이선스와 GPL 라이선스의 호환성을 보여주고 있다.

공개SW 라이선스	GPL 2.0 호환	GPL 3.0 호환
Academic Free License	No	No
Affero GNU General Public License version 3.0	No	Yes
Apache License version 1.0	No	No
Apache License version 1.1	No	No
Apache License version 2.0	No	Yes

Apple Public Source License version 1.x	No	No
Apple Public Source License version 2.0	No	No
Artistic License 1.0	No	No
Clarified Artistic License (draft 2.0)	Yes	Yes
Artistic License 2.0	Yes	Yes
Berkeley Database License	Yes	Yes
original BSD license	No	No
modified BSD license	Yes	Yes
Boost Software License	Yes	Yes
CeCILL	Yes	Yes
Common Development and Distribution License	No	No
Common Public License	No	No
Creative Commons licenses (Tags: by &sa)	No	No
Creative Commons licenses (Tags: nc &nd)	No	No
Cryptix General License	Yes	Yes
Do What The Fuck You Want To Public License (WTFPL)	Yes	Yes
Eclipse Public License	No	No
Educational Community License	No	Yes
Eiffel Forum License version 2	Yes	Yes
Fair Licence	Yes	Yes
GNU General Public License 2.0	Yes	No
GNU General Public License 3.0	No	Yes
GNU Lesser General Public License	Yes	Yes
Hacktivismo Enhanced-Source Software License Agreement	No	No
IBM Public License	No	No
Intel Open Source License	Yes	Yes
ISC license	Yes	Yes

LaTeX Project Public License	No	No
Microsoft Public License	No	No
Microsoft Reciprocal License	No	No
MIT license	Yes	Yes
Mozilla Public License version 1.1	No	No
Mozilla Public License version 2.0	Yes	Yes
Netscape Public License	No	No
Open Software License	No	No
OpenSSL license	No	No
PHP License	No	No
POV-Ray-License	No	No
Python Software Foundation License 2.0.1, 2.1.1 and newer	Yes	Yes
Q Public License	No	No
Sun Industry Standards Source License	No	No
Sun Public License	No	No
W3C Software Notice and License	Yes	Yes
XFree86 1.1 License	No	Yes
zlib/libpng license	Yes	Yes
Zope Public License version 1.0	No	No
Zope Public License version 2.0	Yes	Yes

※ 출처 : http://en.wikipedia.org/wiki/Comparison_of_free_software_licenses

[표 16] 주요 공개SW 라이선스의 GPL 호환성

붙임4 공개SW 라이선스 정책 수립 시 고려사항

1. 저작권 관련 문구 유지

저작권이란 표현된 결과물에 대해 자동적으로 발생하는 권리이다. SW의 소스코드에 대해서도 마찬가지이며 잘 관리되는 공개SW들의 경우 거의 대부분 소스코드 상단에 개발자 정보와 연락처 등이 기록되어 있는데 만약 이러한 개발자 정보를 임의로 수정하거나 삭제하여서는 안 된다.

특히 GPL 등 수정된 결과물을 다시 공개하도록 규정하고 있는 Strong Copyleft 라이선스의 경우 만약 소스코드 상에 개발자 정보가 수정/삭제한 채로 외부에 소스코드를 공개하였다가 그 사실이 밝혀질 경우 더 큰 문제가 발생할 수 있다.

2. 제품명 중복 방지

사용하는 공개SW와 동일한 이름을 제품명이나 서비스명으로 사용하여서는 안 된다. 특히 유명한 공개SW 일수록 상표로 등록되어 있는 경우가 많기 때문에 더욱 조심하여야 한다.

3. 서로 다른 라이선스의 조합

SW를 작성하고자 할 경우 기존에 만들어진 코드를 재사용하거나 결합하는 경우가 많은데, 결합되는 각 코드의 라이선스가 상호 상충되는 경우가 있다. 예를 들어 MPL 조건이 있는 A 코드와 GPL 조건의 B 코드를 결합하여 A+B 라는 프로그램을 만들어 배포하고자

하는 경우, MPL은 A+B의 A 부분을 MPL로 배포할 것을 요구하는 반면, GPL은 A+B 전체를 GPL로 배포 할 것을 요구하기 때문에 A+B 프로그램을 배포하는 것은 불가능하다.

이러한 문제를 라이선스의 양립성 문제라고 한다. 이러한 양립성의 검토에 있어서는 해당 라이선스의 버전에 따라서도 차이가 난다. 예를 들어 MPL 1.1과 GPL 2.0은 상호 호환되지 않지만 MPL 2.0에서는 1.12 "Secondary License"조항을 신설하여 GPL 2.0과 LGPL 2.1, AGPL 3.0으로 배포를 허용하고 있고 사용자가 선택한 라이선스에 따라 더 큰 저작물로 배포할 수 있도록 허용하고 있다.

4. 사용 여부 명시

많은 공개SW 라이선스들은 소스코드를 자유롭게 열람하고 수정 및 재배포할 수 있는 권리를 부여하는 한편, SW를 사용할 때 해당 공개SW가 사용되었음을 명시적으로 표기하는 것을 의무사항으로 채택하고 있다.

5. 소스코드 공개

공개SW 라이선스에 따라 수정하거나 추가한 부분이 있을 때 해당 부분의 소스코드를 공개하여야 한다고 명시하는 경우가 있다. 이에 해당하는 라이선스는 GPL이 가장 유명하다. 그러나 정확한 공개 범위는 각각의 라이선스에서 정하고 있는 범위가 다르고 SW를 개발하는 방법에 따라서도 달라질 수 있다.

6. 특허

특허에 대한 기본적인 내용은 만약 어떤 기술이 특허로 보호될 경우 해당 기술을 구현할 때 반드시 특허권자의 허락을 받아야 한다는 것이다. 이는 공개SW인지 여부에 관계없이 공통적으로 해당한다.

그러나 어떤 특허를 공개SW로 구현할 경우 해당 특허의 구현 결과는 공개SW 라이선스를 따르게 되는 등 공개SW와 관련된 특허권의 문제는 보다 복잡하게 전개된다.

특히 최근 SW특허가 급격히 증가하면서 문제가 심각해지고 있기 때문에 새롭게 만들어지는 공개SW 라이선스에서는 특허관련 조항을 포함하고 있는 경우가 많아지고 있다.

예를 들어, EPL 1.0 및 MPL 1.1 라이선스는 원저작권자가 사용자에게 저작권, 특허권을 부여하는데, EPL 1.0의 경우 사용자 중 누군가가 소송을 제기 시 그 사용자에게 부여한 무료 특허 권리를 종료시키며, MPL 1.1은 특허권리 뿐만 아니라 저작권리도 종료시키는 특징이 있다.

붙임5 Github 개요

깃허브는 깃(Git*) 원격 저장소이자 웹 그래픽 기반 환경에서 깃 도구를 편하게 이용할 수 있도록 만든 호스팅 서비스로, 전 세계 개발자들이 함께 공개SW 프로젝트의 소스코드를 수정하고 참여하며 공개SW 프로젝트 확산에 기여하고 있다.

* 깃(Git)은 2005년에 개발된 공개SW로 분산형 버전관리 시스템(DVCS)으로, 리누스 토발즈와 주니오 하마노가 개발

Github에서 공개된 오픈소스 프로젝트는 무료로 이용할 수 있고, 비공개 프로젝트의 경우 유료서비스 요금제를 선택하여 이용 가능(유료서비스 $7~$200)하다.

Github에서는 가장 강력한 기능인 '포크(Fork)'를 통해 소스코드를 자신의 저장소에 복사하고 배포하는 과정을 몇 번의 클릭으로 가능하게 한다.

Github는 프로그램의 소스코드 뿐만 아니라 설치 관련 문서, 활용 방법, 번역 페이지도 등록 가능하고, gitbook을 통해서 전자 출판으로 활용할 수도 있으며 소스코드 검색 및 관련 통계도 제공된다.

* 미 백악관은 Github에 레포지토리를 운영하며 사람들과 소통하는 창구로 이용하고, 한 해의 예산안을 Github를 통해 공개하고 있음

깃허브 관련 주요 용어는 아래와 같다.

- Watch : 해당 프로젝트의 관찰 기능, 각 이력들에 대한 알림제공
- Star : 해당 프로젝트에 대한 관심, 북마크와 같은 기능
- Fork : 내 Branch로 메인 프로젝트 코드 저장소 복사본 생성
- Issue : 기능을 논하거나 버그 추적하는 대화 공간

- Pull request : Fork한 Branch의 작업완료 후 Master Branch로 병합 요청
- Commit : 변경 내용 저장
- Branch : 테스트를 하거나 새로운 기능을 개발하기 위한 독립된 저장소
- Contributor : 오픈소스 프로젝트 기여자

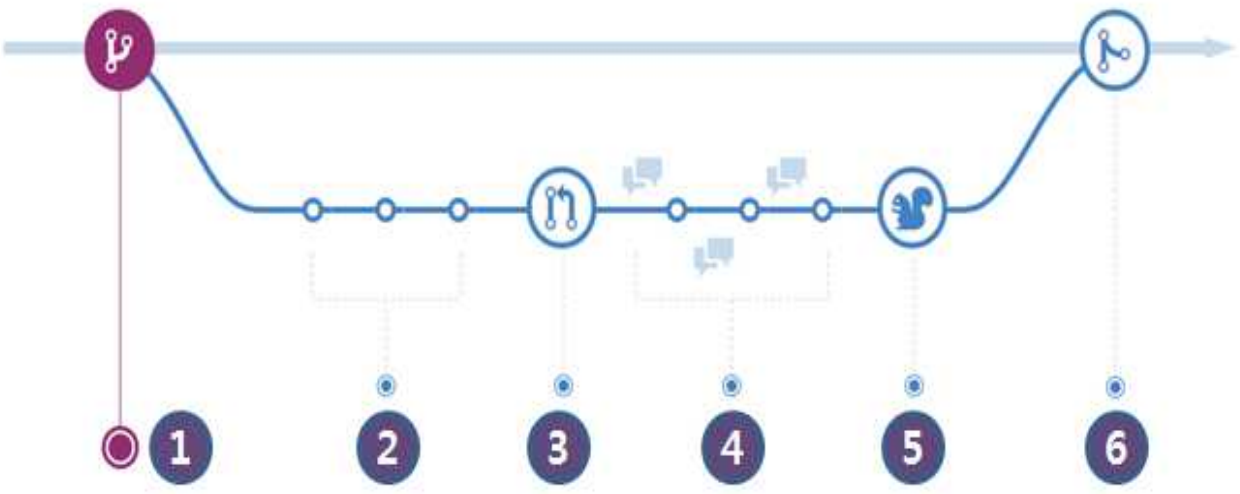
* Github 사용방법에 대한 자세한 내용은 https://guides.github.com/ 참고

붙임6 Github Pull Request 활용 예시

SW개발 시 공통적인 코드 통합 절차는 아래와 같다.

- 코드 리딩(reading) 및 숙지 : 개발자는 우선 프로젝트 저장소에 있는 기존의 관련 코드를 연구한다.
- 개발 및 테스트 : 개발자는 개인 작업 공간에서 필요한 코드의 수정 작업을 실시하고 그 구현 결과를 확인하기 위해 테스트를 수행한다.
- 패치 제출 : 개발자가 커밋 권한이 없는 경우, 코드 변경은 리뷰 및 프로젝트의 코드 베이스에 대한 최종 통합을 위해 코어 개발자나 유지 관리자에게 제출된다. 이것은 종종 이메일 또는 분산된 버전 관리 시스템을 통해 이루어진다.
- 검토 및 사전 커밋 테스트
- 코드 커밋

이제 저장소에 새로운 코드 조각들을 통합하기 위한 방법으로 Github의 Pull Request를 활용하는 방법을 살펴보자.



[그림 11] Github Pull Request 수행 순서

① 기존에 없던 다른 기능 및 아이디어를 시험해보고자 하면 처음 해야 할 것은 브랜치를 생성하는 일이다. 마스터라는 이름의 브랜치는 배포 가능한 것들로만 구성되어야하기 때문에 직접 마스터 브랜치에 작업을 해서는 안 되기 때문이다.

② 이제 새로운 실험을 마음껏 해볼 수 있는 브랜치를 만들었다면 추가하고 편집하고 삭제하는 등 개발자가 원하는 변경을 진행하는데. 이러한 변경 하나 하나를 커밋이라고 한다. 이때 커밋 이벤트를 발생하면서 생성하는 커밋 메시지는 추후 다른 개발자의 피드백을 위해 명확하게 기록해야 한다.

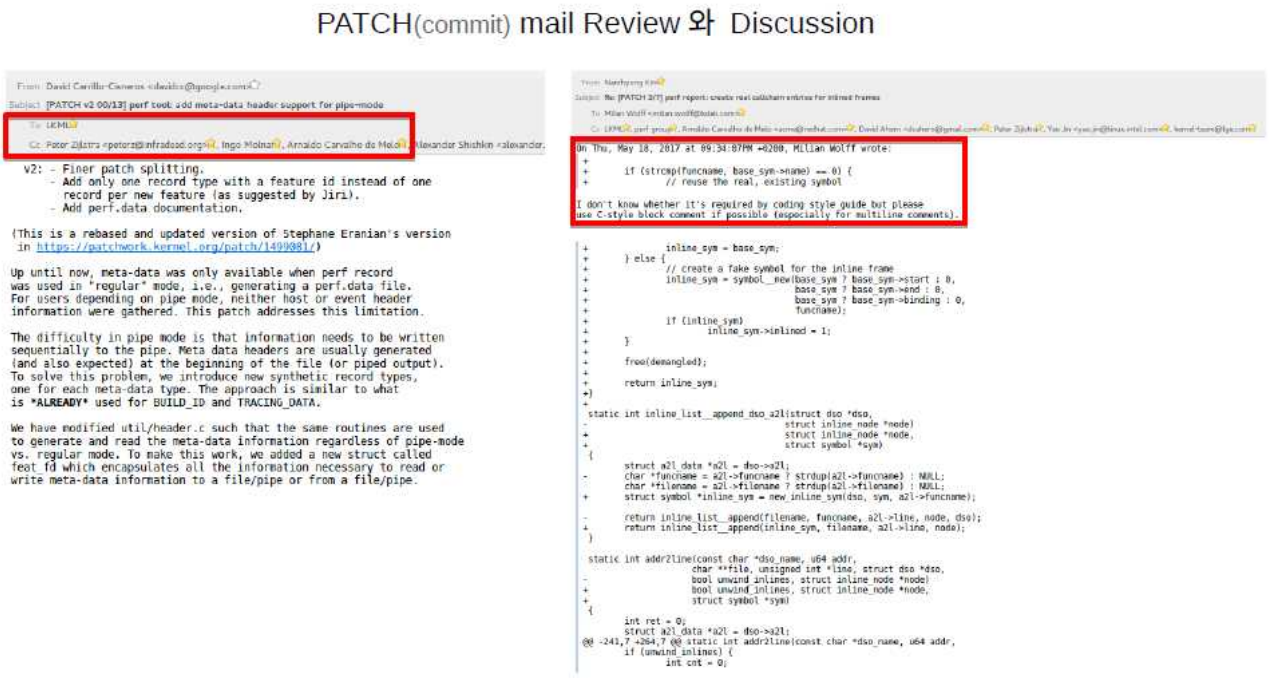
③ 어느 정도 커밋이 쌓여 변경이 일단락되었다면 그것에 대해 코드 리뷰를 요청하는 순서가 기다린다. 이것을 Github에서는 Pull Request라고 한다. 물론 코드 변경 건이 아니고 어떤 스크린샷에 대한 논의나 아이디어에 대한 논의가 필요할 시에도 이 기능을 활용할 수 있다. @mention 기능을 통해 특정 사람이나 팀을 지정하여 Pull Request를 요청할 수 있다.

④ 이제 코드에 대해 논의하고 리뷰를 진행한다. 즉 코딩 스타일이 맞지 않다던가, 단위 테스트가 누락되었다던가, 버그가 있으니 수정을 해달라는 논의 등이 그 예라 할 수 있다.

⑤ Pull Request를 통해 코드 리뷰가 완료되고 커밋을 통해 변경을 가한 개발자의 브랜치가 테스트까지 끝나면 제품 레벨에서 확인하기 위해 배포가 된다(deploy). 이때 당신의 브랜치가 이슈를 발생하면 기존 마스터 브랜치를 제품에 배포하여 원래대로 복구한다.

⑥ 제품 레벨에서 확인까지 마쳤으면 당신의 코드가 마스터 브랜치에 병합된다(merge).

간단한 예로 리눅스 등의 진영에서는 패치 메일을 중심으로 리뷰 및 토론을 전개한다.



[그림 12] 패치 메일을 통한 리뷰 및 토론

먼저 수정한 내용을 커밋으로 만들어야 한다. 이때 중요한 것은 이 커밋이 공개SW 라이선스를 따른다는 표시로 커밋에 sign을 남겨야 한다는 것이다.

다음은 패치 파일을 만들고 보내기 전에 먼저 patch가 코딩 스타일을 따라서 잘 만들어졌는지 확인한다.

다음으로는 패치를 보낼 담당자를 찾아야 한다. 물론 LKML(Linux Kernel Mailing List)로 보내긴 하지만 LKML은 하루에만 400~500개의 patch들이 쏟아져 들어오기 때문에 어느 누구도 이를 모두 다 살펴보고 검토하지는 않는다. 따라서 현재 작업한 내용을 담당하고 있는 메인테이너(관리자)를 찾아서 직접 메일을 보내고 LKML을 참조로 넣는다.

참고문헌

[1] 김종배, 오픈소스 소프트웨어 비즈니스 모델, 2017

[2] NIPA, 공개SW 라이선스 가이드, 2017

[3] NIPA, 공개소프트웨어 프레임워크 및 적용가이드, 2015

[4] NIPA, 오픈소스 라이선스 해설, 2013

[5] 윤석찬, 오픈소스 개발방법론, 2009

[6] 한국저작권위원회, 오픈소스SW 라이선스 분쟁 대응방안 가이드, 2015

[7] 한국저작권위원회, 오픈소스 라이선스, 2016

[8] 한국저작권위원회, 오픈소스 소프트웨어 라이선스 가이드 3.0, 2016

[9] 한국저작권위원회, 소프트웨어 관련 지적재산권, 2016

[10] Karl Fogel, Producing Open Source Software, 2016

[11] Stephanos Androutsellis-Theotokis , Diomidis Spinellis, Maria Kechagia, Georgios Gousios 지음, 김종배 옮김, (10,000 피트에서 바라본) 오픈소스 소프트웨어, 2014

[12] Open Source Guide, https://opensource.guide/

[13] 인터넷 urls (검색일 : 2018. 4.24)

- 2011 오픈소스SW 라이선스 인사이트 컨퍼런스 1부: https://www.youtube.com/watch?v=2BcMP5VU4wU

- Github Issues를 활용한 요구분석 Tip : https://robinpowered.com/blog/best-practice-system-for-organizing-and-tagging-github-issues/
- 공개SW 탐색 : https://stackify.com/source-code-repository-hosts/
- 이클립스 모듈라 설계: http://aosabook.org
- 오라일리 참여를 위한 아키텍처 : http://archive.oreilly.com/pub/a/oreilly/tim/articles/architecture_of_participation.html
- 리눅스 패치 메일 : https://www.kernel.org/doc/html/v4.11/process/submitting-patches.html
- 릴리즈 버전 관리 : https://github.com/hatemogi/semver,
- 저장소 및 호스팅 설비 : https://www.phacility.com/
- 소스코드 형상관리 : https://git-scm.com/book/en/v2/Getting-Started-Git-Basics