OPENCHAIN

커리큘럼

이 슬라이드는 OpenChain Curriculum 공식 번역본입니다.
공동번역 : 박종백/법무법인태평양/jb.park@bkl.co.kr, 장학성/LG전자/hakssung@gmail.com

OpenChain 설명서 1.2을 위한 FOSS 교육 참조 슬라이드

CC0-1.0에 따라 제공됨
제한없이 이 슬라이드를 사용, 수정 및 공유할 수 있음.
보증 없이 제공됨.

이 슬라이드는 미국 법률을 따름. 법적 관할국에 따라 법적 요구 사항이 달라질 수 있음. 이 슬라이드를 컴플라이언스 교육 프로그램의 일부로 사용할 때는이 점을 고려하여야 함.

이 슬라이드에는 법률 자문이 포함되어 있지 않습니다.

OpenChain 커리큘럼이란?

OPENCHAIN

- OpenChain 프로젝트는 FOSS(Free and Open Source Software) 컴플라이언스 프로그램의 핵심 구성 요소를 식별하고 공유하는데 도움이 된다.
- OpenChain 프로젝트의 핵심은 **설명서(Specification)**이다. 이는 FOSS 컴플라이언스 프로그램이 충족해야하는 핵심 요구 사항을 식별하고 공표한다.
- OpenChain **커리큘럼**은 자유롭게 사용할 수있는 교육 자료를 제공함으로 설명서를 지원한다.
- 이 슬라이드는 회사들이 설명서 1.2의 요구 사항을 충족하는 데 도움이 된다. 또한, 일반 컴플라이언스 교육에도 사용할 수 있다.

자세한 내용 참조: https://www.openchainproject.org

목차

1. 지식 재산권이란 무엇인가?
2. FOSS 라이선스 소개
3. FOSS 컴플라이언스 소개
4. FOSS 검토를 위한 소프트웨어 핵심 개념
5. FOSS 검토 실행
6. 컴플라이언스 관리 전과정 (프로세스 예시)
7. 컴플라이언스 함정 피하기
8. 개발자가이드라인

FOSS 정책

- <<이것은 당신의 FOSS 정책이 어디서 발견될 수 있는지 알려주기 위한 견본용 슬라이드이다. (OpenChain 설명서 1.1, 1.1.1절)>

- https://www.linux.com/publications/generic-foss-policy 에 게시된 Linux Foundation Open Compliance Program에서 FOSS 정책의 예시를 얻을 수 있다.

1장

지식 재산권이란 무엇인가?

"지식 재산권"이란 무엇인가?

OPENCHAIN

- 저작권: 저자의 원본 저작물을 보호
  - 표현(바탕에 깔린 아이디어가 아님)을 보호
  - 소프트웨어, 서적 및 이와 유사한 저작물을 다룬다.
- 특허: 새롭고 뻔하지 않은 유용한 발명품
  - 혁신에 보상을 주기 위한 제한된 독점
- 영업 비밀: 가치있는 기밀 정보 보호
- 상표 : 제품의 출처를 식별하는 표시(단어, 로고, 슬로건, 색상 등)를 보호
  - 소비자 및 브랜드 보호; 소비자 혼란과 브랜드 가치 저하 방지

이 장에서는 FOSS 컴플라이언스와 가장 관련 있는 분야인,
저작권 및 특허권에 초점을 맞춘다.

소프트웨어의 저작권 개념

OPENCHAIN

- 기본 규칙 : 저작권은 창의적 작품을 보호
- 저작권은 일반적으로 책, 영화, 그림, 음악, 지도와 같은 어문 작품에 적용된다.
- 소프트웨어는 저작권의 보호를 받는다.
  - 기능(특허로 보호됨)이 아니라 표현(구현된 세부 사항의 창의성)이 보호를 받는다.
  - 바이너리 코드 및 소스 코드 포함한다.
- 저작권 소유자는 자신이 창작한 저작물에 대해서만 통제권을 가지고, 타인의 독립적인 창작물에 대해서는 통제권을 가지지 않는다.
- 저자의 허락없이 복사하는 경우 저작권 침해가 발생할 수 있다.

소프트웨어와 밀접한 관련있는 저작권상의 권리

- 소프트웨어를 *복제* 할 권리 – 복사하기
- “ *파생 저작물*”을 만들 권리 – 수정하기
  - 파생 저작물이라는 용어는 미국 저작권법에서 비롯된다.
  - 사전 정의가 아닌 법령에 근거한 특별한 의미를 갖는 "전문 용어"이다.
  - 일반적으로 충분한 저작 활동이 추가된 독창적인 저작물을 기반으로 한 새로운 저작물을 말하며, 이는 새로운 저작물의 사본이 아닌 원본 저작물을 나타내는 것이다.
- 배포할 권리
  - 배포는 일반적으로 바이너리 또는 소스 코드 형식의 소프트웨어 사본을 다른 주체 (회사 또는 조직 외부의 개인 또는 조직)에게 제공하는 것으로 간주된다.

참고 : "파생 저작물"또는 "배포"를 구성하는 내용에 대한 해석은 FOSS 커뮤니티 및 FOSS 법률 집단 내에서 토론의 대상이 된다.

소프트웨어의 특허 개념

OPENCHAIN

- 특허는 기능을 보호한다. – 컴퓨터 프로그램과 같은 작동 방법을 포함 할 수 있다.
  - 추상적인 아이디어, 자연의 법칙을 보호하지는 않는다.
- 해당 국가에서 특허를 얻기 위해서는 특정 관할 국가에서 특허 신청을 해야한다. 특허가 부여되면 독립적인 창작에 의한 것인지에 상관없이 특허소유자는 모든 사람에게 그 기능을 행사하지 못하게 할 권리가 있다.
- 그 기술을 사용하고자 하는 다른 당사자는 특허 라이선스(기술을 사용, 제작, 매도, 매도 청약 및 수입할 권한을 부여)를 요청할 수 있다.
- 다른 당사자가 독자적으로 동일한 발명을 창작하더라도 침해가 발생할 수 있다.

라이선스

- "라이선스"는 저작권 또는 특허권 소유자가 다른 사람에게 허가 또는 권리를 부여하는 방식이다.
- 라이선스는 다음 사항에 제한될 수 있다:
  - 허용된 사용 유형 (상업적 / 비상업적, 배포, 파생 저작물 / 제조)
  - 독점적 또는 비독점적인 조건
  - 지리적 범위
  - 영구적 또는 제한적 존속 기간
- 라이선스에는 부여조건이 있을 수 있는바,
  특정 의무를 준수하는 경우에만 라이선스를 취득한다는 의미이다.
  - 예: 저작자 표시 또는 상호적 라이선스 제공
- 보증, 배상, 지원, 업그레이드, 유지 보수와 관련된 계약 조건도 포함될 수 있다.

이해도 점검

OPENCHAIN

- 저작권법은 어떤 유형의 대상을 보호하는가?
- 소프트웨어에 대해 가장 중요한 저작권의 내용은 무엇인가?
- 소프트웨어가 특허 대상이 될 수 있는가?
- 특허가 특허 소유자에게 부여하는 권리는 무엇인가?
- 자신의 소프트웨어를 독자적으로 개발하는 경우,
  해당 소프트웨어에 대한 제3자의 저작권 라이선스가 필요할 수 있는가?
  특허 라이선스가 필요할 수 있는가?

2장

FOSS 라이선스 소개

FOSS 라이선스

OPENCHAIN

- FOSS 라이선스의 정의는 수정 및 재배포를 허용하는 조건하에서 소스 코드를 사용할 수 있게 하는 것이다.
- FOSS 라이선스는 저작자 고지, 저작권 표시 보존 또는 소스 코드 제공 서면 청약과 관련된 조건을 가질 수 있다.
- 널리 쓰이는 라이선스 집합은 Open Source Initiative (OSI)에서 Open Source Definition(OSD)을 기반으로 승인한 라이선스 집합이다. OSI 승인 전체 라이선스 목록은 다음에서 볼 수 있다: http://www.opensource.org/licenses/

Permissive FOSS 라이선스

- Permissive FOSS 라이선스: 제한이 최소한인 FOSS 라이선스를 설명하기 위해 자주 사용되는 용어
- 예: BSD-3-Clause
  - BSD 라이선스는 저작권 고지 및 라이선스 면책 조항이 유지되는한 어떠한 목적으로든 소스 코드 또는 오브젝트 코드 형식으로 무제한 재배포를 허용하는 Permissive 라이선스의 한 예이다.
  - 특정 허가 없이 파생 저작물을 인증하기 위해 기여자 이름을 사용하는 것을 제한하는 조항이 포함되어 있다.
- 다른 예: MIT, Apache-2.0

라이선스 상호주의(성) 및 Copyleft 라이선스

- 일부 라이선스는 파생 저작물(또는 동일 파일, 동일 프로그램 또는 다른 영역의 소프트웨어)이 배포되는 경우 원본 저작물과 동일한 조건으로 배포 할 것을 요구한다.
- 이를 "Copyleft" 또는 "상호주의" 효과라고 한다.
- GPL-2.0의 라이선스 상호주의 예:

배포하거나 공표하려는 저작물의 전부 또는 일부가 대상 프로그램의 일부를 포함하거나 대상 프로그램으로부터 파생된 것이라면, 본 라이선스의 규정에 따라 […] 의 사용허락을 해야 한다.

- 상호주의 또는 Copyleft 조항을 포함하는 라이선스에는 GPL, LGPL, AGPL, MPL 및 CDDL의 모든 버전이 포함된다.

독점 라이선스 또는 비공개 소스

OPENCHAIN

- 독점 소프트웨어 라이선스 (또는 상업용 라이선스 또는 EULA)는 소프트웨어의 사용, 수정 및 배포에 대한 제한을 부과한다.
- 독점 라이선스는 각 공급 업체마다 고유함 – 공급 업체 수 만큼 다양한 독점 라이선스가 존재하며 각각은 개별적으로 평가되어야한다.
- FOSS라이선스 및 독점 라이선스가 모두 지적 재산을 기반으로 하고 해당 자산에 대한 라이선스를 부여함에도, FOSS개발자들은 상업적 비 FOSS 라이선스를 설명하기 위해 종종 "독점"이라는 용어를 사용한다.

기타 비 FOSS 라이선스 상황

- 프리웨어 – 독점 라이선스 하에서 무료 또는 매우 저렴한 비용으로 배포되는 소프트웨어
  - 소스 코드가 사용 가능하거나 사용 가능하지 않을 수 있으며, 파생 저작물 생성은 일반적으로 제한된다.
  - 프리웨어 소프트웨어는 일반적으로 완벽하게 작동하며 (기능 제한 없음) 무제한으로 사용할 수 있다 (사용일수 제한 없음).
  - 프리웨어 소프트웨어 라이선스는 일반적으로 사용 유형 (개인, 상업, 학문 등)에 대한 제한 뿐만 아니라 소프트웨어의 복제, 배포 및 파생 저작물 제작과 관련하여 제한을 부과한다.
- 쉐어웨어 – 제한된 시간 동안 제한된 기능을 사용자에게 시범적으로 무료 제공하는 독점 소프트웨어
  - 쉐어웨어의 목표는 잠재적인 구매자에게 프로그램의 전체 버전에 대한 라이선스를 구입하기 전에 프로그램을 사용하고 그 유용성을 판단 할 기회를 제공하는 것이다.
  - 쉐어웨어 공급 업체는 소프트웨어가 조직 내에서 자유롭게 전파 된 후 대규모 라이선스료 지급을 받기 위해 회사에 접근하기 때문에 대부분의 회사는 쉐어웨어를 매우 경계한다.

기타 비 FOSS 라이선스 상황

- “비상업적” – 어떤 라이선스들은 FOSS 라이선스의 특성의 대부분을 갖고 있지만 비상업적 용도에 제한된다(예. CC-BY-NC).
  - FOSS의 정의상 소프트웨어의 사용분야를 제한할 수 없다.
  - 상업적 용도는 용도분야이어서 그에 대한 어떤 제한을 하는 라이선스는 FOSS가 되지 못한다.

퍼블릭 도메인

- **퍼블릭 도메인** 은 법의 보호를 받지 않아서 공중이 라이선스없이 사용할 수 있는 소프트웨어를 가르킨다.
- 개발자들은 자신이 개발한 소프트웨어에 대해 *퍼블릭 도메인으로 선언*할 수 있다.
  - 예) “이 소프트웨어안의 모든 코드와 문서는 저작자가 퍼블릭 도메인에 제공한다.”
  - 퍼블릭 도메인 선언은 FOSS 라이선스와 다르다.

- 퍼블릭도메인 선언은, 소프트웨어가 제한없이 사용될 수 있음을 명확하게 하기 위해 개발자가 소프트웨어에 대해 가진 모든 지적재산권을 포기 또는 제거하려는 것이다. 그러나 이 선언의 집행가능성은 FOSS 커뮤니티 내의 분쟁에 휘말릴 수 있고, 그 법적 유효성은 관할국가마다 다르다.
- 대개 퍼블릭 도메인 선언은 보증 면제 같은 다른 조건을 수반하는데, 그 경우 소프트웨어는 퍼블릭 도메인에 있지 않고 라이선스의 적용을 받는 것으로 볼 수 있다.

라이선스 양립가능성

- 라이선스 양립가능성은 라이선스 조건들이 서로 충돌하지 않도록 하는 절차이다.
- 어느 라이선스가 귀하에게 무엇을 행할 것을 요구하고 다른 라이선스는 그를 금지할 경우 그 라이선스들은 충돌하고, 두 소프트웨어 모듈의 결합이 어느 하나의 라이선스의 의무를 발생시키는 경우 양립가능하지 않다.
  - GPL2.0과 EPL1.0은 배포되는 "2차적 저작물"에 자신의 의무를 확대한다.
  - GPL-2.0 모듈이 EPL-1.0 모듈과 결합되어 통합된 모듈이 배포될 경우, 그 모듈은 반드시
    - (GPL-2.0에 의하면), GPL-2.0에 의해서만 배포되어야 하고,
    - (EPL-1.0에 의하면), EPL-1.0에 의해서만 배포되어야 한다.
  - 배포자는 동시에 두가지 조건을 충족시킬 수 없으므로, 그 모듈은 배포될 수가 없다.
  - 이것이 라이선스 양립불가능의 예이다.

"2차적 저작물"의 정의에 대해서는 FOSS 커뮤니티 내에 다른 견해들이 있고, 법적 해석은 관할 국가별로 차이가 있다.

고지

OPENCHAIN

파일 헤더 안의 커멘트 문구같은 고지는 종종 저작자와 라이선싱 정보를 제공한다. FOSS 라이선스는 또한 저작자에게 크레딧을 주거나 소프트웨어가 수정 내용을 포함함을 명백히 하기 위하여 고지문구를 소스코드 안에 또는 소스코드와 병행하여 위치시킬 것을 요구할 수 있다.

- **저작권 고지** –세상에 대해 저작권 소유권을 알리기 위해 저작물의 사본에 위치시키는 식별자. 예: Copyright © A. Person (2016)
- **라이선스 고지** – 제품에 포함되는 FOSS의 라이선스 조건을 특정하고 인정하는 고지.
- **저작자 고지** – 제품에 포함되는 FOSS의 원 저작자와/또는 스폰서의 동일성을 인정하는, 제품 출시에 포함시키는 고지.
- **수정 고지** – 파일의 제일 윗부분에 귀하의 저작권 고지를 추가하는 것과 같은, 파일의 소스 코드를 수정하였다는 고지.

다중-라이선싱

- 다중-라이선싱은 동시에 두 개 이상의 서로 다른 조건으로 소프트웨어를 배포하는 관행을 지칭한다.
  - 예, 소프트웨어가 “듀얼 라이선스”로 된 경우, 저작권 소유자는 각 수령인에게 두개 라이선스중 하나를 선택권을 준다.
- 주의: 라이선서가 하나 이상의 라이선스를 부과하고, 귀하는 복수의 모든 라이선스를 준수해야 하는 상황과 혼동되어서는 안된다.

이해도 점검

OPENCHAIN

- 무엇이 FOSS 라이선스인가?
- Permissive FOSS 라이선스의 전형적인 의무는 무엇인가?
- Permissive FOSS 라이선스를 거명하시오.
- 라이선스 상호성은 무슨 의미인가?
- Copyleft 스타일의 라이선스를 거명하시오.
- Copyleft 라이선스하에서 사용되는 코드에 대해서는 무엇이 배포되어야 하는가?
- 프리웨어와 쉐어웨어 소프트웨어는 FOSS로 여겨지는가?
- 다중-라이선스는 무엇인가?
- FOSS 고지에서 어떤 정보를 찾을 수 있는가? 그 고지는 어떻게 이용되는가?

CHAPTER 3

FOSS 컴플라이언스 소개

FOSS 컴플라이언스 목적

OPENCHAIN

- **귀하의 의무 파악.** 귀하는 귀하의 소프트웨어에 존재하는 FOSS 컴포넌트를 식별하고 추적하는 절차를 가져야 한다.

- **라이선스 의무 충족.** 귀하의 절차는 귀하의 조직의 비즈니스 실무에서 생성되는 FOSS 라이선스 의무를 관리할 수 있어야 한다.

어떤 컴플라이언스 의무가 충족되어야 하나?

OPENCHAIN

해당 FOSS라이선스에 따라, 컴플라이언스 의무는 다음으로 구성된다.

- **출처와 고지.**귀하는 하위 사용자들이 소프트웨어의 출처와 라이선스하의 자신들의 권리를 알 수 있도록 소스코드 및/또는 제품 문서 또는 사용자 인터페이스에 저작권과 라이선스 본문을 제공하거나 유지할 필요가 있다. 귀하는 또 수정에 관한 고지와 또는 라이선스의 전체사본을 제공할 필요가 있다.
- **소스 코드 입수 가능성.** 귀하는 FOSS 소프트웨어, 귀하가 작성한 수정 저작물, 결합되거나 링크된 소프트웨어에 대한 소스코드와 빌드 프로세스를 제어하는 스크립트를 제공할 필요가 있을 수 있다.
- **상호성.** 귀하는 특정 FOSS 컴포넌트의 수정물 또는 2차적 저작물에 그 컴포넌트에 적용되는 라이선스와 동일한 라이선스가 유지되도록 할 필요가 있을 수 있다.
- **기타 조건.** FOSS 라이선스는 저작권자의 이름이나 상표의 사용을 제한할 수 있고 혼란을 피하기 위하여 수정버전에는 다른 이름을 사용할 것을 요구하거나, 모든 위반시에 해지될 수 있다.

FOSS 컴플라이언스 쟁점: 배포

- 외부 주체에 대한 전파
  - 사용자의 기계 또는 모바일 기기에 다운로드된 애플리케이션
  - 사용자의 기계에 다운로드된 자바스크립트, 웹 클라이언트 또는 다른 코드
- 어떤 FOSS 라이선스에 대해서는 컴퓨터 네트워크를 통한 접근이 “의무발생”사유가 될 수 있다.
  - 어떤 라이선스들은 “의무발생”사유를 서버에서 작동하는 소프트웨어에 접근 허용하는 것이나(예, 소프트웨어가 수정되는 경우 모든 Affero GPL버전) “컴퓨터 네트워크를 통해 원격으로 소프트웨어와 상호작용하는 사용자들”의 경우를 포함하는 것으로 정의한다.

FOSS 컴플라이언스 쟁점: 수정

OPENCHAIN

- 현재 프로그램에 대한 변경(예, 추가, 파일안의 코드 제거, 컴포넌트들의 결합)
- 어떤 FOSS 라이선스하에서는 수정으로 배포시에 다음과 같은 추가 의무가 발생할 수 있다:
  - 수정 고지 제공
  - 해당 소스 코드 제공
  - 수정물에 FOSS 컴포넌트에 적용하는 라이선스와 동일한 라이선스를 적용한다.

FOSS 컴플라이언스 프로그램

OPENCHAIN

FOSS 컴플라이언스를 성공적으로 갖춘 조직들은 다음 목적을 위하여 자신들만의 *FOSS 컴플라이언스 프로그램* (정책, 절차, 교육과 툴로 구성됨)을 구축해 왔다:

1. 제품(상업적 또는 그외)내에서 효과적인 FOSS 사용을 가능하게 하고
2. FOSS 개발자/소유자의 권리를 존중하고 라이선스상 의무를 준수하고
3. FOSS 커뮤니티에 기여하고 참가한다.

컴플라이언스 실무 실행

OPENCHAIN

다음 사항을 처리할 비즈니스 절차와 충분한 관리직원을 준비:

- 모든 내,외부 소프트웨어의 출처와 라이선스 식별
- 개발절차 내에서 FOSS 소프트웨어를 추적
- FOSS 검토 이행과 라이선스 의무 식별
- 제품 배송시 라이선스 의무 이행
- FOSS 컴플라이언스 프로그램을 위한 감독, 정책수립과 컴플라이언스 결정
- 교육

컴플라이언스 혜택

OPENCHAIN

견고한 FOSS 컴플라이언스 프로그램의 혜택은 다음과 같다:

- FOSS의 혜택과 그것이 조직에 미치는 영향에 대한 이해 증가
- FOSS 사용과 관련된 비용과 위험에 대한 이해 증가
- 이용 가능한 FOSS 해결에 대한 지식 증가
- 위반 위험의 감소와 관리, FOSS개발자/소유자의 라이선싱 선택에 대한 존중 증가
- FOSS 커뮤니티 및 FOSS 조직과의 관계 강화

이해도 점검

OPENCHAIN

- FOSS 컴플라이언스는 무엇을 의미하는가?
- FOSS 컴플라이언스 프로그램의 두가지 주요 목적은 무엇인가?
- FOSS 컴플라이언스 프로그램의 중요한 비즈니스 실무를 열거하고 설명하시오.
- FOSS 컴플라이언스 프로그램의 혜택은 무엇인가?

CHAPTER 4

FOSS 검토를 위한 소프트웨어 핵심 개념

FOSS 컴포넌트를 어떻게 사용하길 원하는가?

일반적인 시나리오는 다음을 포함한다:

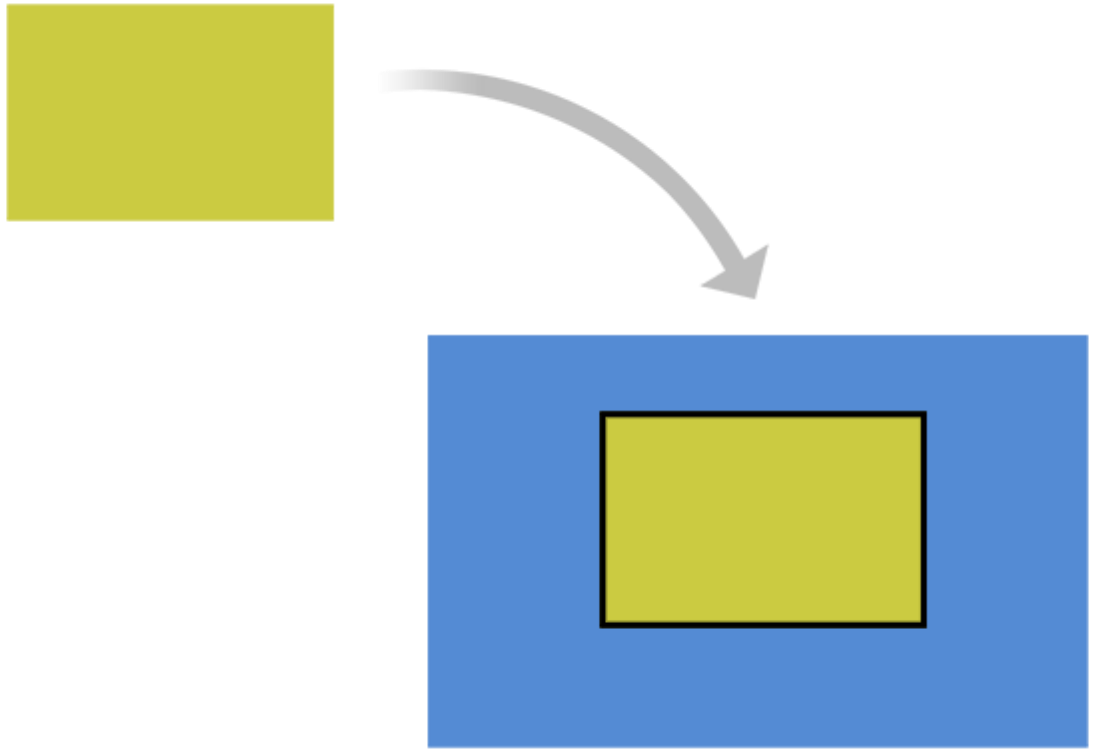
- 편입
- 링킹
- 수정
- 번역

편입

개발자는 FOSS 컴포넌트의 일부를 소프트웨어 제품에 복사해 넣을 수 있다.

관련 용어는 다음을 포함한다:

- 통합 (Integrating)
- 병합 (Merging)
- 붙여넣기 (Pasting)
- 각색 (Adapting)
- 삽입 (Inserting)



링킹

개발자는 FOSS 컴포넌트를 귀하의 소프트웨어 제품과 링크하거나 연결할 수 있다.

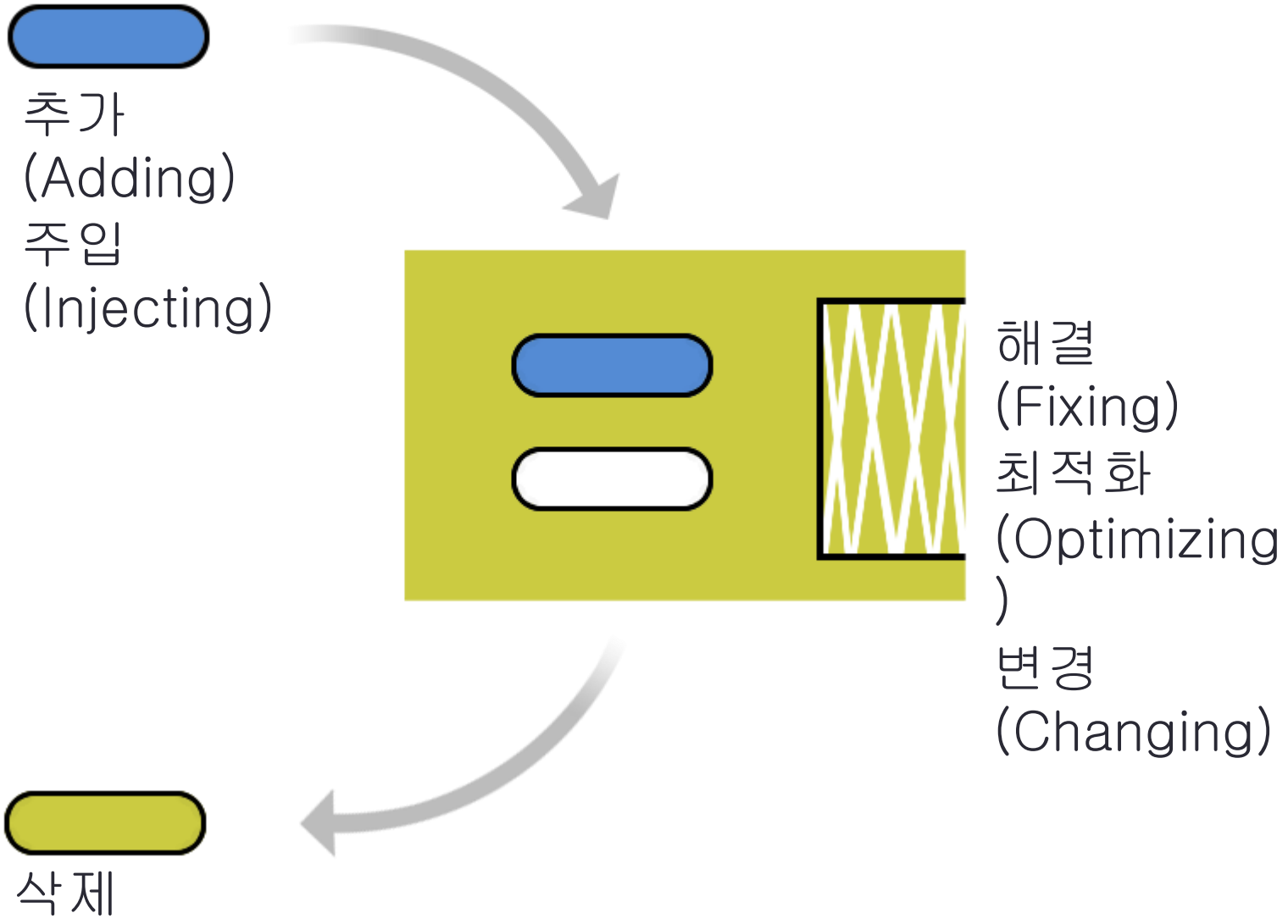
관련 용어는 다음을 포함한다:

- 정적/동적 링킹
- 페어링
- 결합 (Combining)
- 활용 (Utilizing)
- 패키징
- 상호의존성 생성

수정

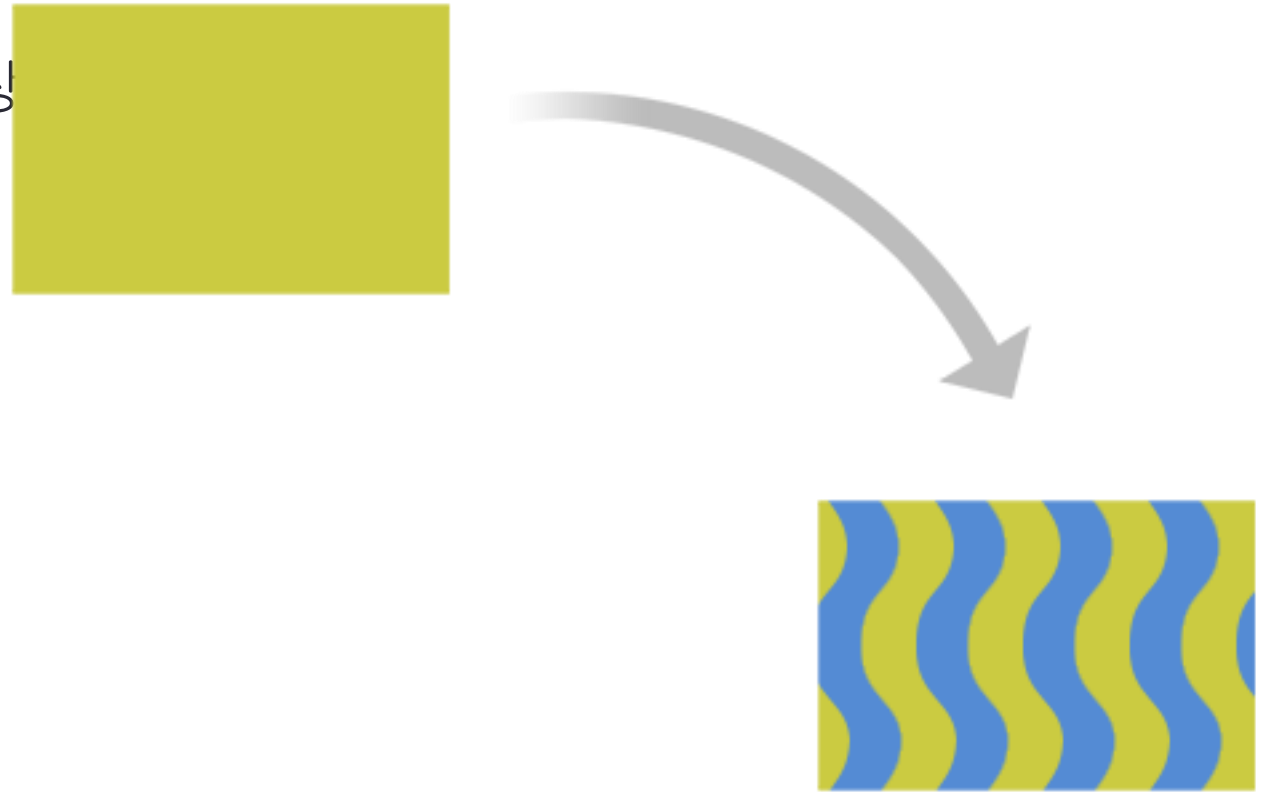
개발자는 다음을 포함하여 FOSS 컴포넌트를 변경할 수 있다:

- FOSS 컴포넌트에 새로운 코드 추가 / 주입
- FOSS 컴포넌트의 수정, 최적화 또는 변경
- 코드 삭제 또는 제거



번역

개발자는 코드를 어느 상태에서 다른 상태로 변환 할 수 있다.



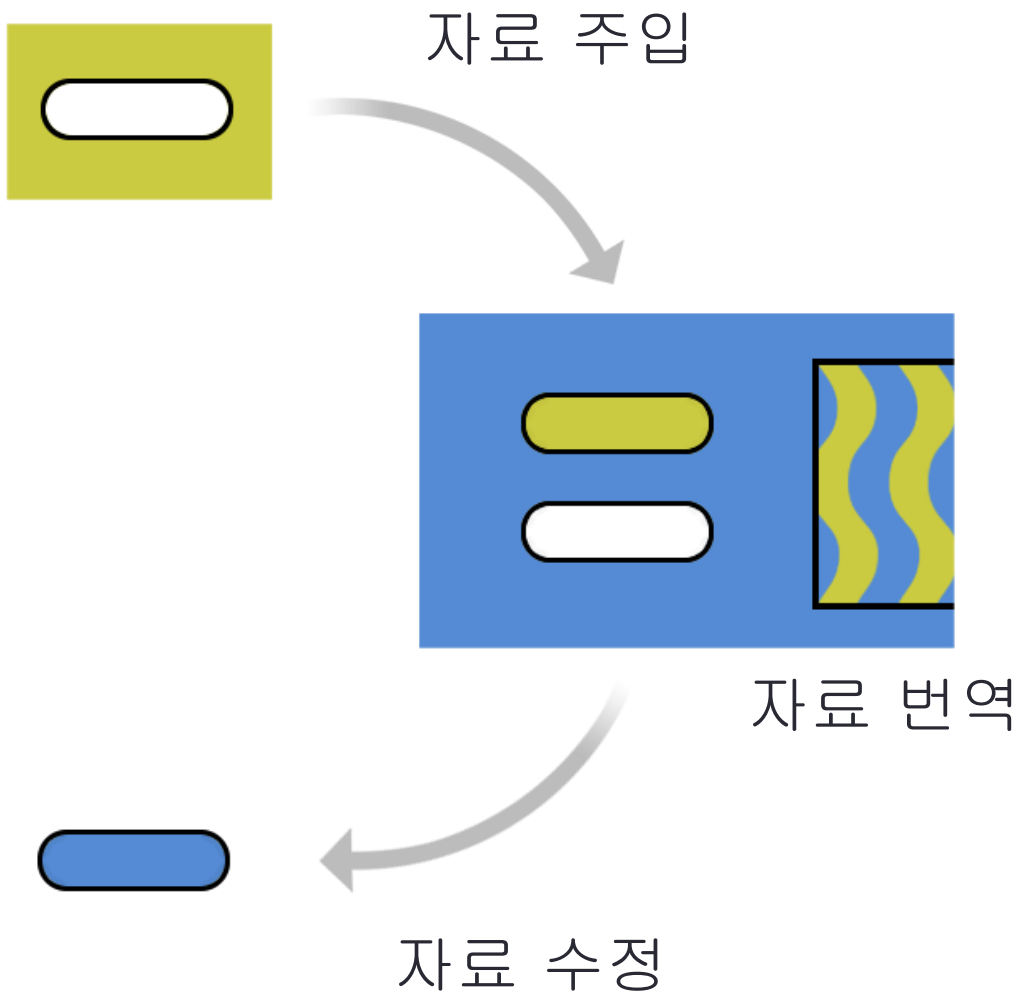
예:

- 중국어를 영어로 번역
- C++에서 Java로 변환
- 바이너리로 컴파일

개발 도구

개발 도구가 이러한 작업 중 일부를 내부적으로 수행할 수 있다.

예를 들어, 어떤 도구는 그 출력물에 자체 코드의 일부를 주입할 수 있다.



FOSS 컴포넌트가 어떻게 배포되는가?

OPENCHAIN

- 누가 소프트웨어를 받는가?
  - 고객/파트너
  - 커뮤니티 프로젝트
  - 비지니스 그룹 내 다른 법인 (이는 배포로 여겨질 수 있다)

- 어떤 형태로 전달되는가?
  - 소스 코드 제공
  - 바이너리 제공
  - 하드웨어에 사전 탑재됨

이해도 점검

OPENCHAIN

- 편입이란 무엇인가?
- 링킹이란 무엇인가?
- 수정이란 무엇인가?
- 번역이란 무엇인가?
- 배포인지를 평가하는데 중요한 요소는 무엇인가?

CHAPTER 5

FOSS 검토 실행

FOSS 리뷰

OPENCHAIN

- 제안된 FOSS 컴포넌트의 유용성및 품질에 대해 프로그램 및 제품 관리자와 엔지니어가 검토 한 후, 선택한 컴포넌트의 사용과 관련된 권리 및 의무에 대한 검토가 시작되어야 한다.
- FOSS 컴플라이언스 프로그램의 핵심 요소는 *FOSS 리뷰* 프로세스이다. 이 프로세스를 통해 회사가 사용하는 FOSS 소프트웨어를 분석하고 권리와 의무를 이해할 수 있다.
- FOSS 리뷰 프로세스에는 다음 단계가 포함된다:
  - 관련 정보 수집
  - 라이선스 의무 분석 및 이해
  - 회사 정책 및 비즈니스 목표와 양립 가능한 가이드 제공

FOSS 리뷰 시작하기

프로그램 또는 제품 관리자, 엔지니어, 법무팀을 포함하여 회사내에서 FOSS 관련 일을 하는 사람은 누구든지 FOSS 리뷰를 시작할 수 있어야 한다.

참고: 프로세스는 주로 엔지니어 또는 외부 공급 업체가 새로운 FOSS 기반 소프트웨어를 선택한 때 시작된다..

어떤 정보를 수집해야하는가?

FOSS 사용을 분석할 때, FOSS 컴포넌트의 정체, 출처 및 FOSS 컴포넌트가 어떻게 사용될지에 대한 정보를 수집하시오. 여기에는 다음이 포함될 수 있다:

- 패키지 이름
- 패키지를 제공하는 커뮤니티의 상태 (활동, 다양한 회원, 반응도)
- 버전
- 다운로드 또는 소스 코드 URL
- 저작권 소유자
- 라이선스 및 라이선스 URL
- 저작자 및 다른 고지와 URL
- 만들고자하는 수정사항에 대한 설명

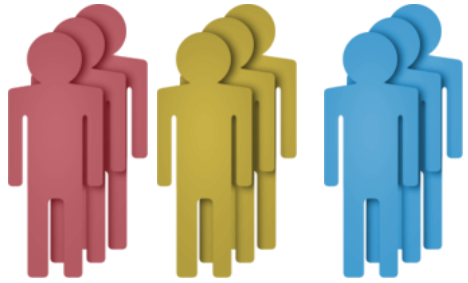
- Dependency 목록
- 제품 내 의도된 용도
- 패키지를 포함하는 첫번째 제품 출시
- 소스 코드가 유지될 위치
- 다른 상황에서 있었던 이전의 승인
- 외부 공급 업체로부터 입수한 경우:
- 개발팀의 연락 지점
- 라이선스 의무를 충족시키기 위해 필요한 경우, 공급업체 수정에 대한 저작권 고지, 출처 , 소스 코드

FOSS 리뷰 팀

FOSS 리뷰 팀에는 FOSS 사용을 지원, 안내, 조정 및 검토하는 회사 대표자가 포함된다. 대표자에는 다음이 포함 될 수 있다:

- 라이선스 의무를 확인하고 평가하는 법무팀
- FOSS 사용을 식별하고 추적할 수 있는 소스 코드 스캐닝 및 도구 지원
- 사업적 이해관계, 상용 라이선싱, 수출 컴플라이언스 등 관련 일을 하면서 FOSS 사용으로 영향 받을 수 있는 엔지니어링 전문가

제안된 FOSS 사용 분석



FOSS 리뷰 팀은 문제에 대한 가이드를 제공하기 전에 수집한 정보를 평가해야한다. 여기에는 정보의 정확성을 확인하기 위해 코드를 스캔하는 작업이 포함될 수 있다.

FOSS 리뷰 팀은 다음을 고려해야한다.

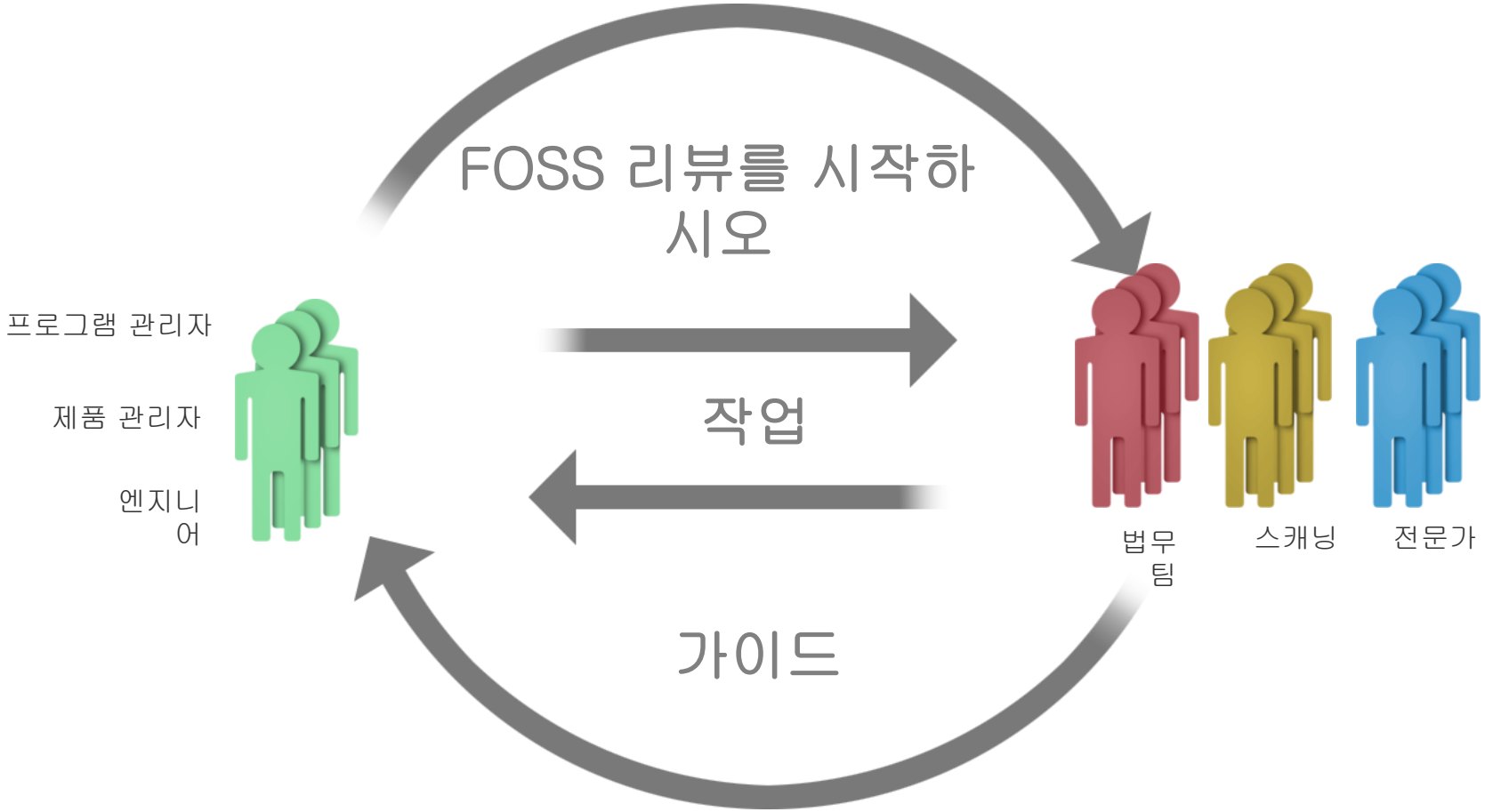
- 코드 및 관련 정보가 완전하고 일관되며 정확한 것인가?
- 선언된 라이선스가 코드 파일에 있는 것과 일치하는가?
- 라이선스는 소프트웨어의 다른 컴포넌트와 사용을 허용하는가?

소스 코드 스캐닝 도구

OPENCHAIN

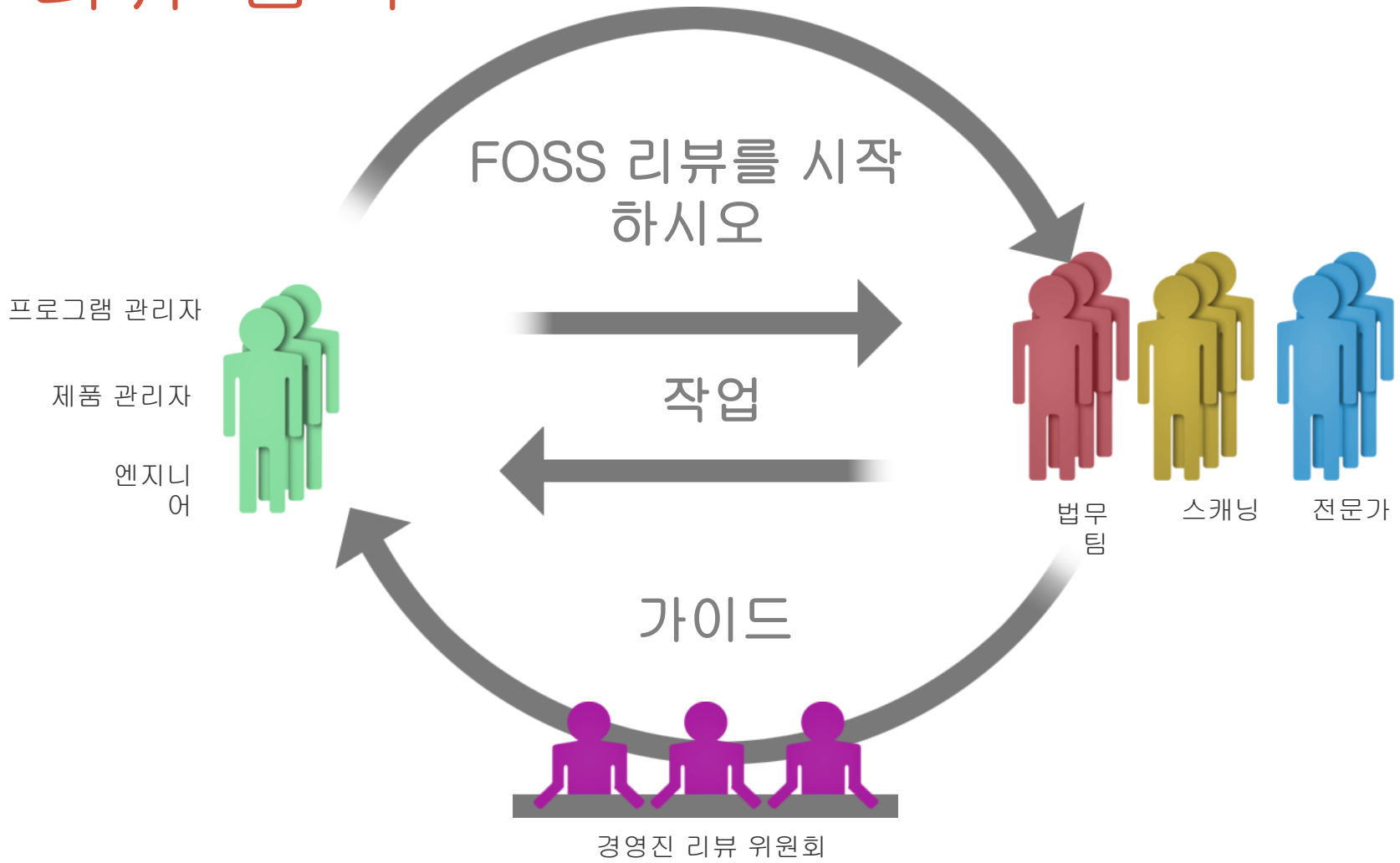
- 다양한 자동화된 소스 코드 스캐닝 도구가 있다.
- 모든 솔루션은 특정 요구 사항들을 처리하기 때문에 어떤 것도 모든 문제를 해결할 수 없다.
- 회사들은 특정 시장 영역 및 제품에 가장 적합한 솔루션을 선택해야 한다.
- 많은 회사들은 자동화된 도구와 수동 검토를 모두 이용한다.
- 무료로 사용할 수 있는 소스 코드 스캐닝 도구의 좋은 예는 Linux Foundatoin에서 호스팅하는 프로젝트인 FOSSology이다. : http://fossology.org

FOSS 리뷰를 통한 작업



FOSS 리뷰 프로세스는 엔지니어링, 비지니스, 법무팀을 비롯한 여러 분야를 걸치게 된다. 모든 그룹이 문제를 정확히 이해하고 명확하고 공유된 가이드를 만들 수 있도록 상호 작용해야 한다.

FOSS 리뷰 감독



FOSS 리뷰 프로세스는 의견 불일치를 해결하고 가장 중요한 결정을 승인할 수 있는 경영진의 감독을 받아야 한다.

이해도 점검

OPENCHAIN

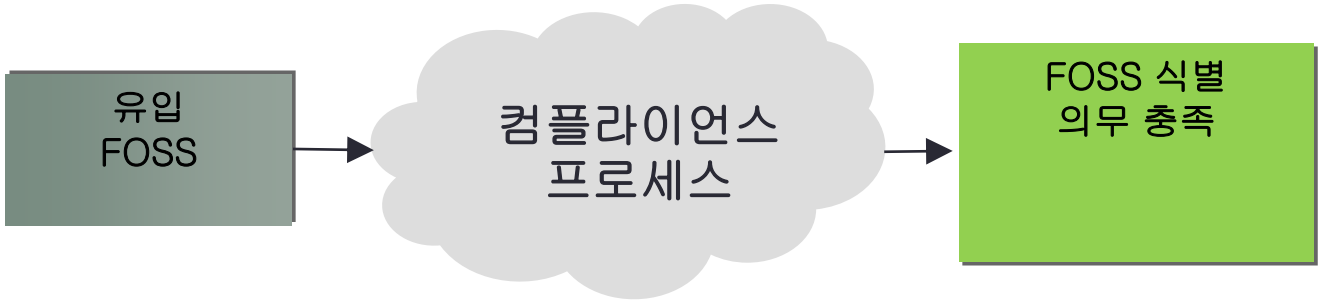
- FOSS 리뷰의 목적은 무엇인가?
- FOSS 컴포넌트를 사용하고자 할 때 취해야 할 첫 번째 액션은 무엇인가?
- FOSS 사용에 관한 질문이 있으면 무엇을 해야 하는가?
- FOSS 리뷰를 위해 어떤 종류의 정보를 수집하는가?
- 누가 소프트웨어 라이선스를 부여했는지 식별하는데 도움이 되는 정보는 무엇인가?
- 외부 공급 업체로부터 유입된 FOSS 구성 요소를 검토 할 때 중요한 추가 정보는 무엇인가?
- FOSS 리뷰에서 수집한 정보의 품질을 평가하기 위해 어떤 단계를 수행할 수 있는가?

CHAPTER 6

컴플라이언스 관리 전과정 (프로세스 예시)

소개

- 컴플라이언스 관리는 제품에 사용되는 FOSS 컴포넌트를 관리하는 일련의 작업이다. 회사는 독점 컴포넌트를 위한 유사한 프로세스를 가질 수 있다. FOSS 컴포넌트는 OpenChain 설명서에서 "공급 대상 소프트웨어 (Supplied Software)"라고 불린다.
- 이러한 활동에는 주로 다음이 포함된다.:
  - 공급 대상 소프트웨어에 사용된 모든 FOSS 컴포넌트 식별
  - 해당 컴포넌트에 의해 발생된 모든 의무를 식별하고 추적
  - 모든 의무가 충족되었거나 충족될 것임을 확인
- 소규모 회사는 간단한 체크리스트를, 큰 기업들은 상세한 절차를 사용할 수 있다.



예: 중소 기업 체크리스트

OPENCHAIN

지속적인 컴플라이언스 과제:

1. 구매/개발 주기 초반에 모든 FOSS를 발견.
2. 사용된 모든 FOSS 컴포넌트 리뷰 및 승인.
3. FOSS 의무를 이행하는데 필요한 정보를 확인.
4. FOSS 프로젝트에 대한 외부로의 기여에 대해 리뷰 및 승인.

지원 요구사항:

1. 적절한 컴플라이언스 인력을 확보하고 명확한 책임 사항을 지정.
2. FOSS 컴플라이언스 프로그램을 지원하는데 맞춰 기존 비지니스 프로세스를 수정.
3. 조직의 FOSS 정책에 대한 교육을 모든 사람이 받을 수 있게 함.
4. 모든 FOSS 컴플라이언스 활동의 진행 사항을 추적.

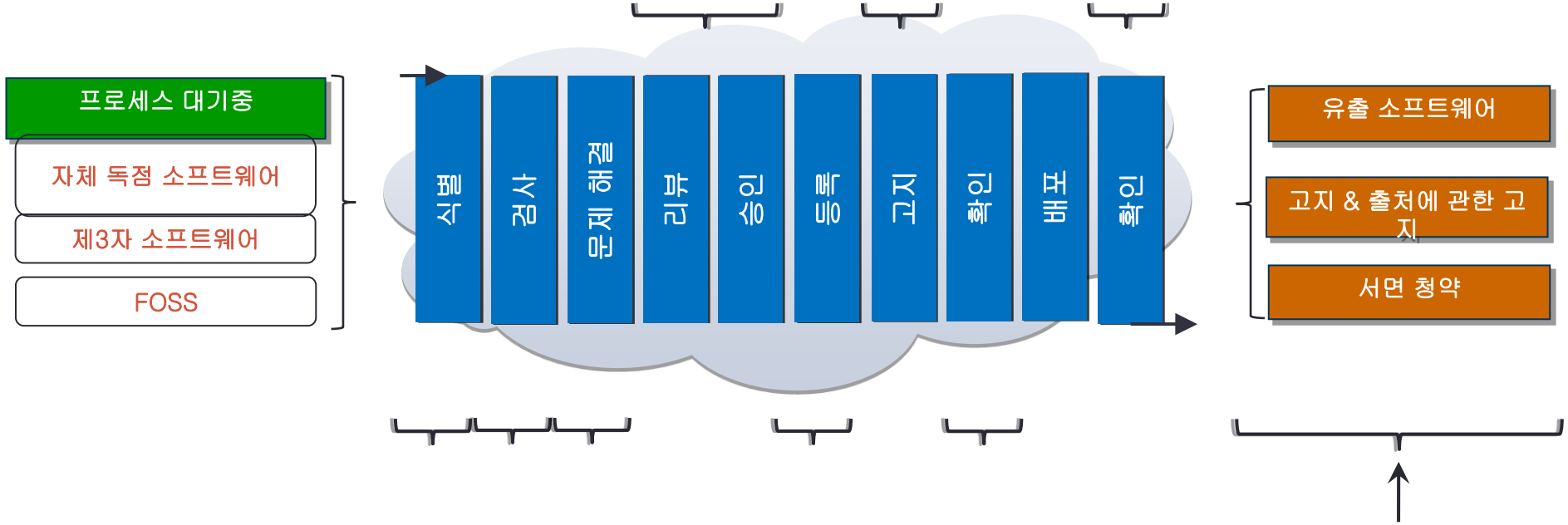
이 항목에 대한 자세한 체크리스트는 여기서 받을 수 있음: https://www.linuxfoundation.org/projects/opencompliance/self-assessment-compliance-checklist

예: 기업 프로세스

FOSS 컴포넌트의 컴플라이언스 기록 리뷰 및 승인

공개를 위한 고지문 수집

공개 후 확인



리뷰를 위해 FOSS 컴포넌트 식별

스캔 또는 소스 코드 검회사 – 그리고 – 소스 코드의 출처와 라이선스 확인

FOSS 정책에 따라 모든 검사 문제 해결

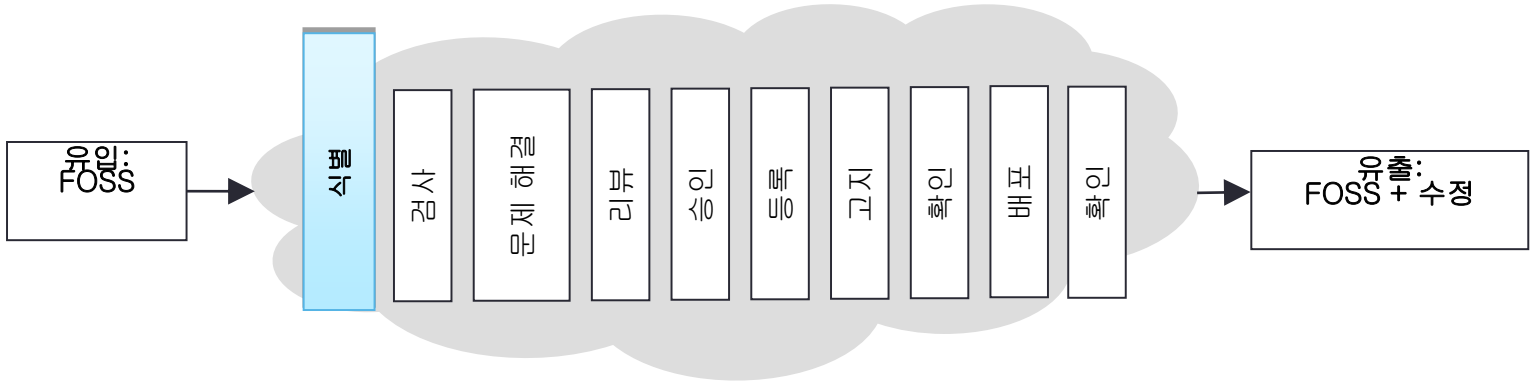
각 제품 및 출시별 승인된 소프트웨어/버전을 목록에 기록

배포용 소스 코드 패키지 검증 – 그리고 – 적절한 고지 제공 검증

소스 코드 공개,고지 및 서면 청약 제공

예 : 컴플라이언스 관리 전과정 프로세스

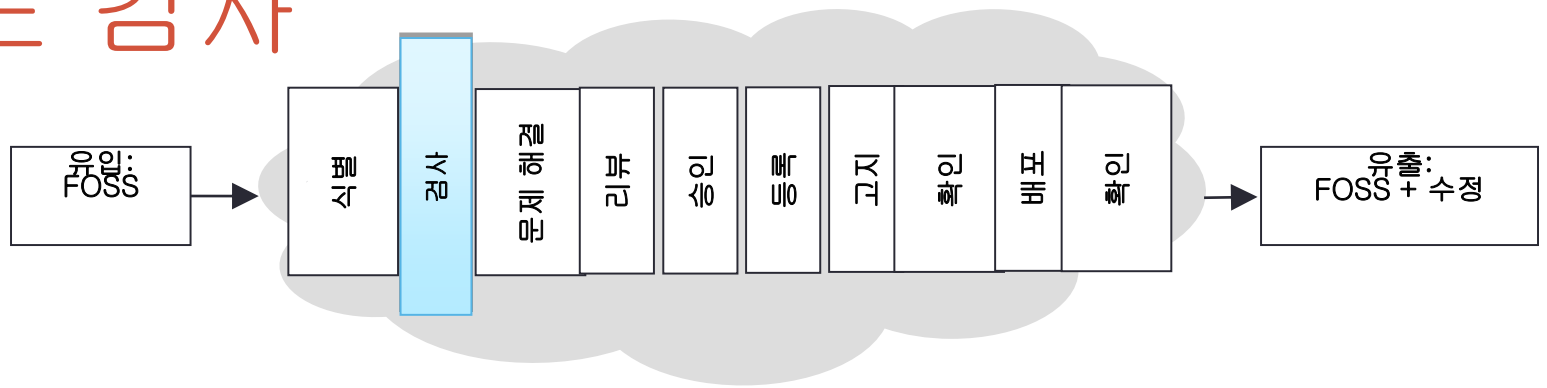
FOSS 사용의 확인 및 추적



FOSS 컴포넌트 식별

- 단계:
  - 엔지니어링으로부터 요청 접수
  - 소프트웨어 스캔
  - 제3자 소프트웨어의 실사
  - 저장소에 추가된 새로운 컴포넌트에 대한 수동 인식
- 결과:
  - FOSS에 대한 컴플라이언스 기록이 생성(또는 업데이트)됨.
  - FOSS 정책 요구사항에 따라 포괄적이거나 제한적으로 정의된 범위로 소스 코드를 리뷰하도록 검사가 요청됨.

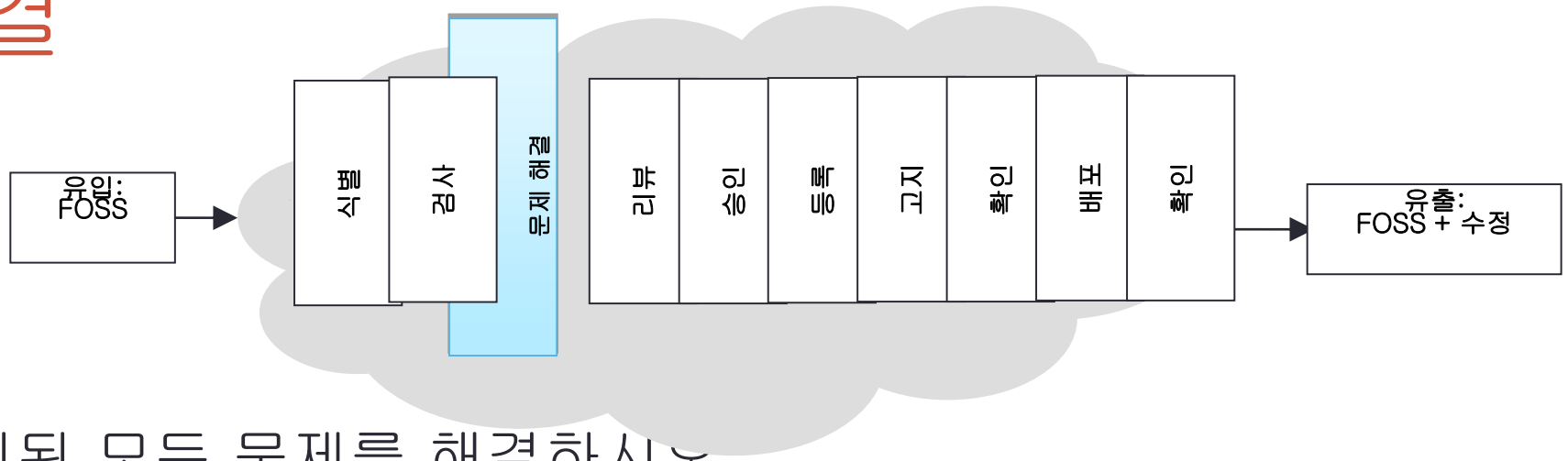
소스 코드 검사



FOSS 라이선스 식별 및 검사

- 단계:
  - 검사할 소스 코드가 식별됨.
  - 소스는 소프트웨어 도구로 스캔할 수도 있음.
  - 검사 또는 스캔을 통한 "히트 (Hits)"가 리뷰되고 코드의 적절한 출처에 대해 확인됨.
  - 검사 또는 스캔은 소프트웨어 개발 및 출시 주기에 따라 반복적으로 수행됨.
- 결과:
  - 다음을 식별하는 검사 보고서:
    1. 소스 코드의 출처와 라이선스
    2. 해결이 필요한 문제

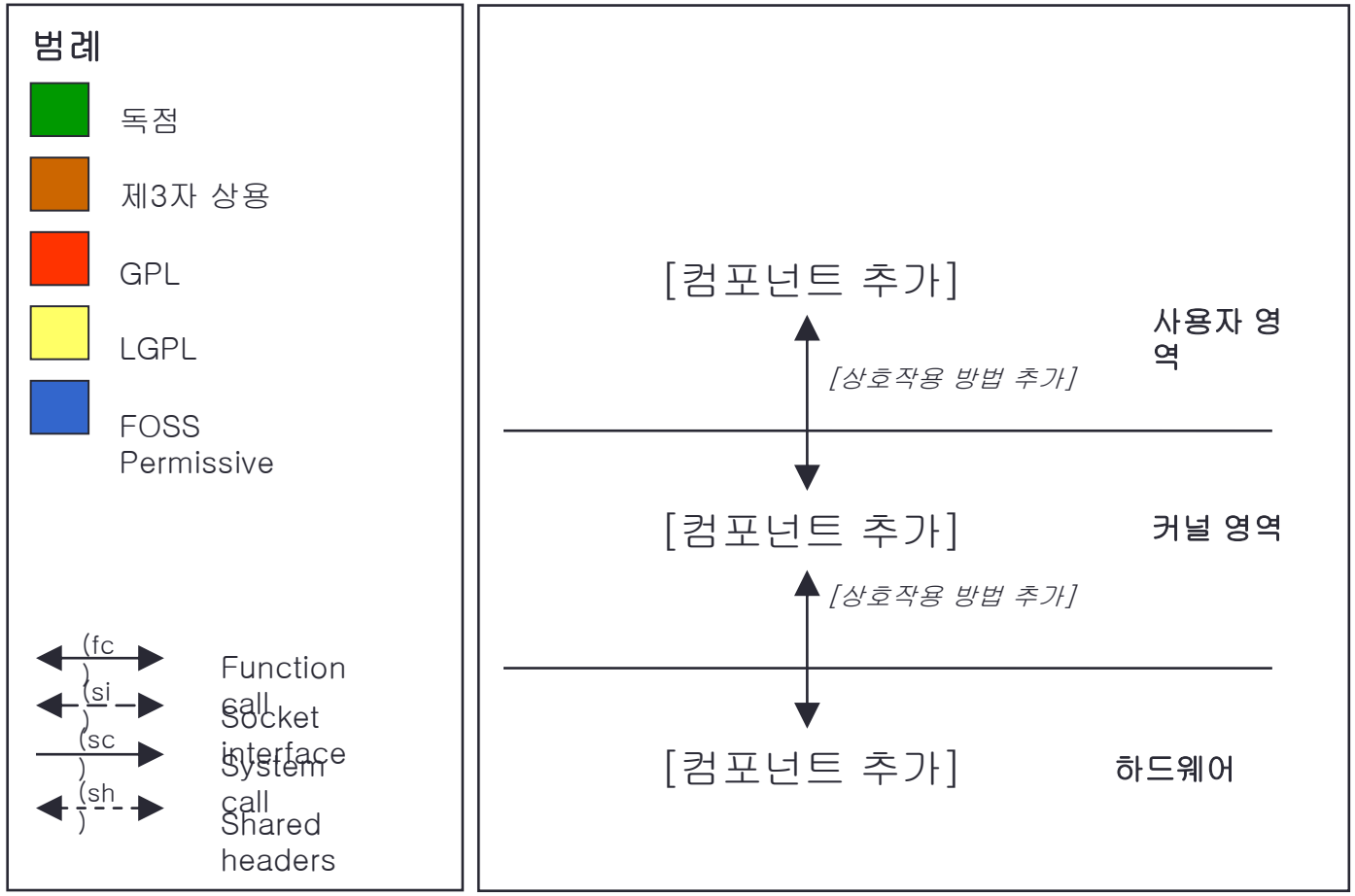
문제 해결



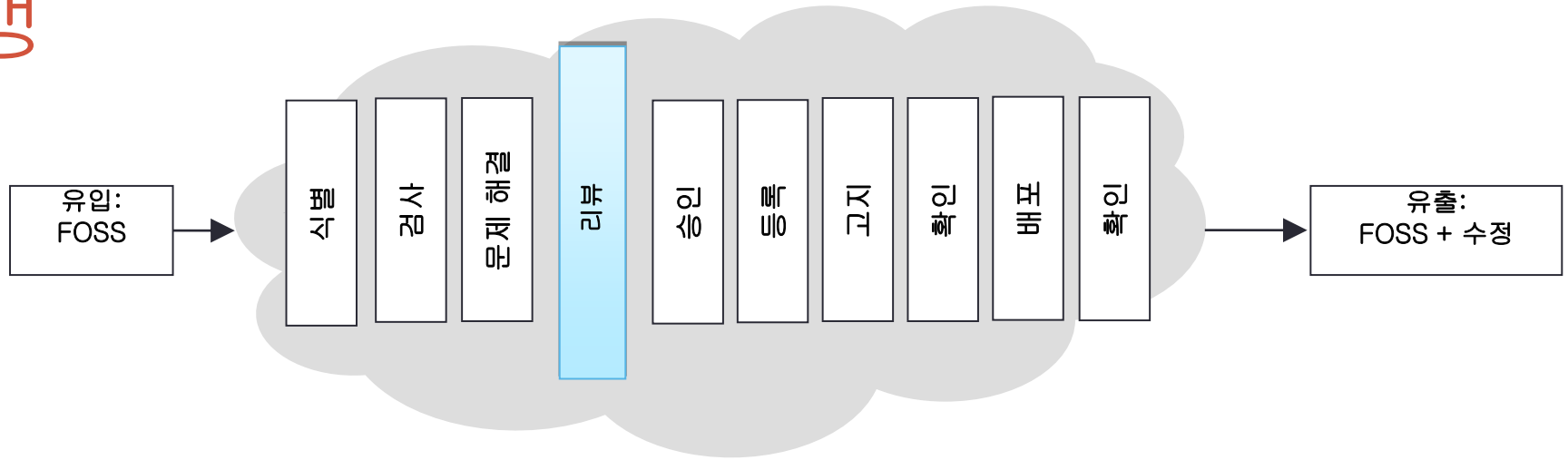
검사에서 확인된 모든 문제를 해결하시오

- 단계:
  - FOSS 정책과 충돌하는 검사 보고서 내의 문제를 해결하기 위해 적절한 엔지니어에게 피드백을 제공하시오.
  - 그러면 엔지니어는 관련 소스 코드에 대한 FOSS 리뷰를 수행한다. (다음 슬라이드의 템플릿 참조)
- 결과:
  보고서 내 플래그가 표시된 각 파일에 대한 해결 방법 및 플래그가 표시된 라이선스 충돌에 대한 해결 방법

아키텍처 리뷰 (예시 템플릿)



리뷰 수행



해결된 문제를 리뷰하여 FOSS 정책과 일치하는지 확인한다.

단계:

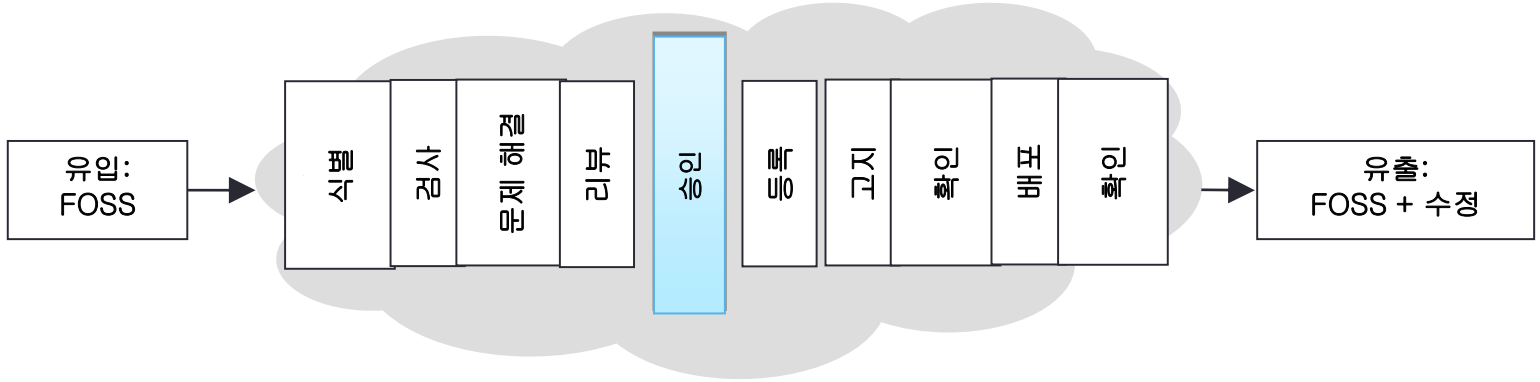
- 리뷰 직원에게 적절한 수준의 권한을 부여한다.
- FOSS 정책을 참고하여 리뷰를 수행한다.

결과:

- 검사 보고서 내의 소프트웨어가 FOSS 정책을 준수하도록 한다.
- 검사 보고서 발견사항을 보존하고, 해결된 이슈에 대해서는 다음 단계(승인)를 위해 준비된 것으로 표시한다.

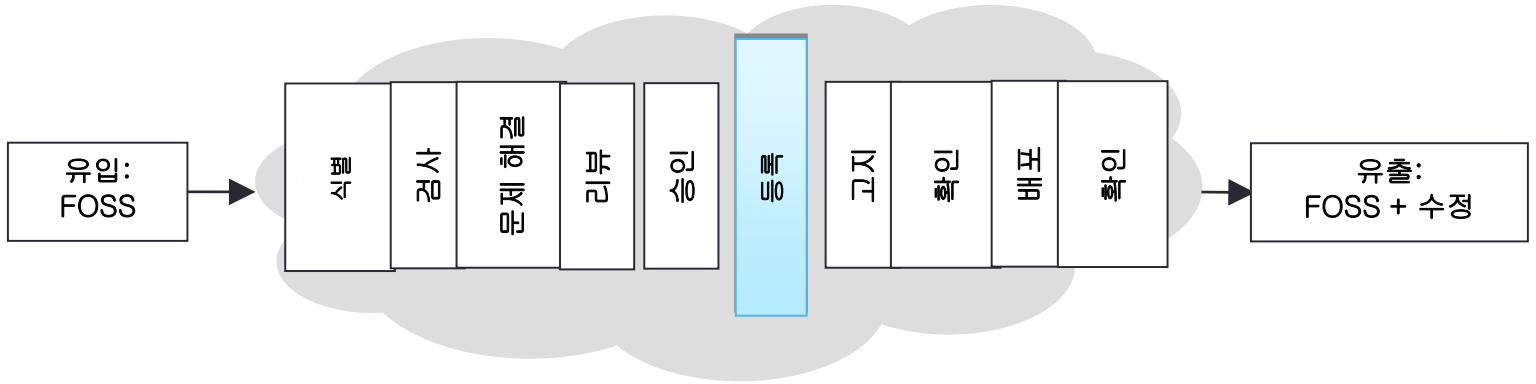
승인

- 이전 단계의 소프트웨어 검사 및 리뷰 결과에 따라 소프트웨어 사용이 승인되거나 승인되지 않을 수 있다.
- 승인에서는 승인된 FOSS 컴포넌트의 버전, 컴포넌트의 승인된 사용 모델 및 FOSS 라이선스 하의 기타 해당되는 모든 의무 사항을 명시해야 한다.
- 승인은 적절할 수준의 권한에서 이루어져야 한다.

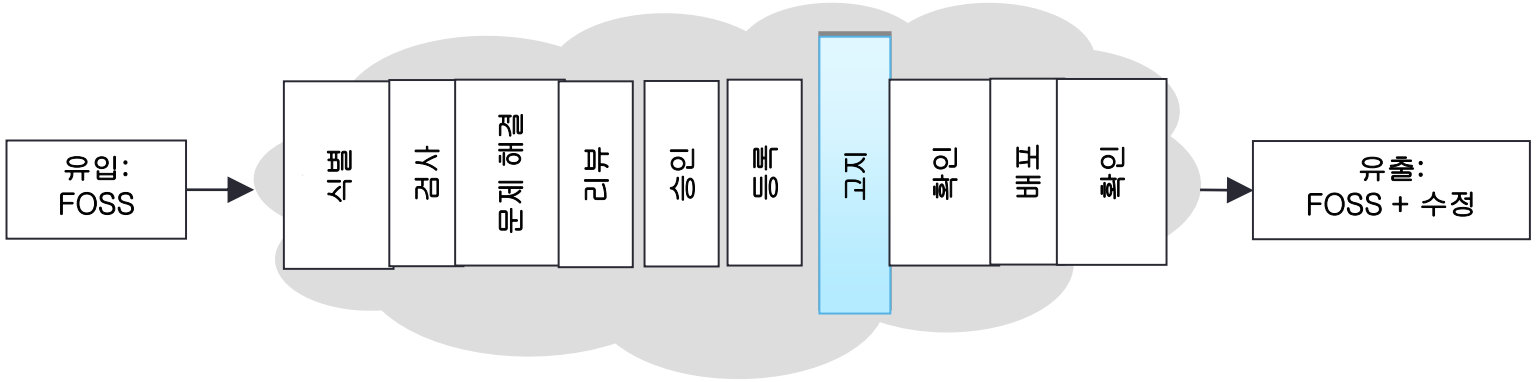


등록 / 승인 추적

- 제품 사용에 대한 FOSS 컴포넌트가 승인되면 해당 제품의 소프트웨어 목록에 추가해야한다.
- 승인 및 이에 대한 조건은 추적 시스템에 등록해야 한다.
- 추적 시스템은 새로운 버전의 FOSS 컴포넌트에 대해서나 새로운 사용 모델이 제안되는 경우는 새로운 승인이 필요하다는 점을 명확히 해야 한다.

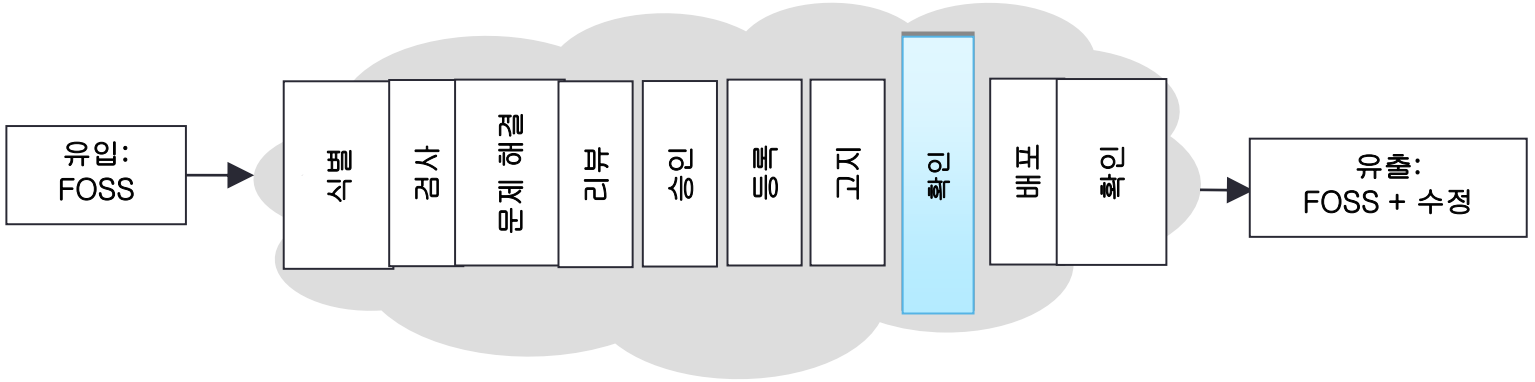


고지



- 제품 출시에 사용되는 FOSS에 대한 적절한 고지를 준비하시오.
  - 모든 저작권 및 출처에 관한 고지를 제공함으로써 FOSS의 사용을 인정한다.
  - FOSS 소스 코드의 사본을 얻는 방법에 대해 제품의 최종 사용자에게 알린다(해당하는 경우. 예: GPL 및 LGPL).
  - 필요에 따라 제품에 포함된 FOSS 코드에 대한 라이선스 계약의 전체 텍스트를 복제한다.

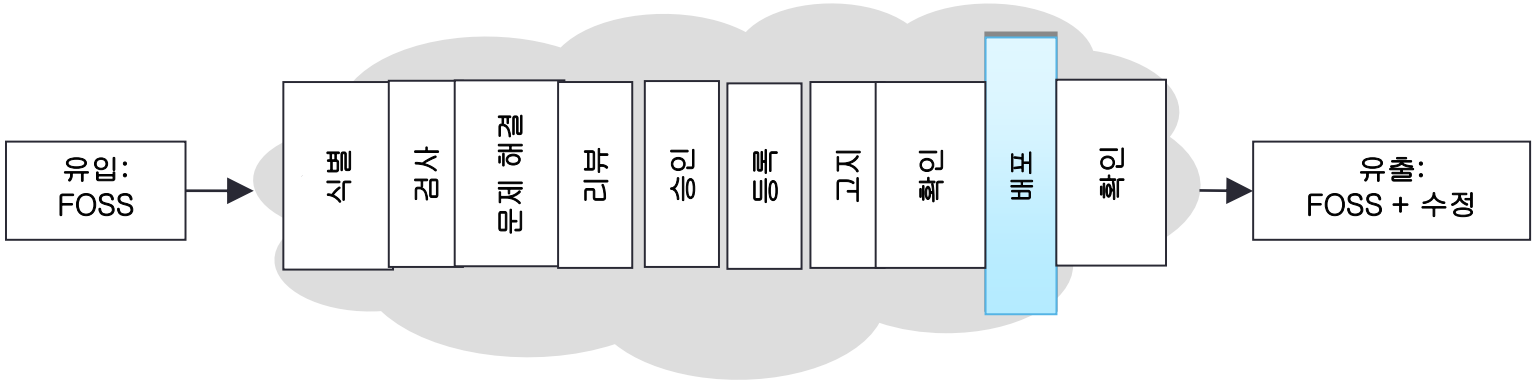
배포 전 검증



배포되는 소프트웨어가 리뷰되고 승인되었는지 확인한다.

- 단계:
  - 배포할 FOSS 패키지가 식별되고 승인되었는지 확인한다.
  - 리뷰된 소스 코드가 제품에 탑재된 바이너리와 일치하는지 확인한다.
  - 최종 사용자에게 식별된 FOSS에 대한 소스 코드를 요청할 수 있는 권리를 알리기 위한 모든 적절한 고지 사항이 포함되어 있는지 확인한다.
  - 식별된 다른 의무를 준수하는지 확인한다.
- 결과:
  - 배포 패키지에는 리뷰 및 승인된 소프트웨어만 포함된다.
  - 적절한 고지 파일을 포함하는(OpenChain 설명서에 정의된) "배포할 컴플라이언스 결과물"은 배포 패키지 혹은 다른 전달 방법에 포함된다.

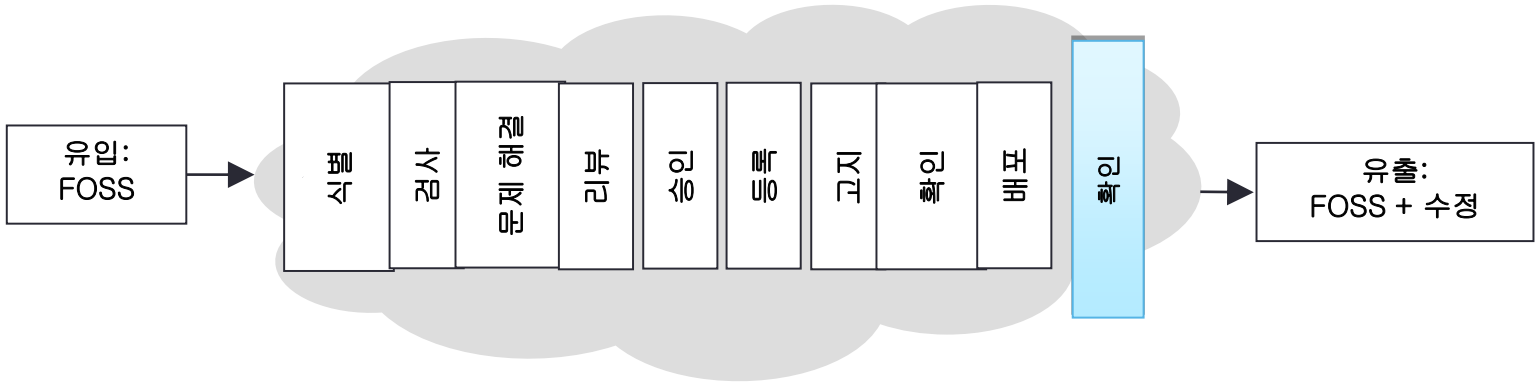
동반 소스 코드 배포



요구되는 동반소스 코드를 제공한다.

- 단계:
  - 동반 소스 코드를 관련 빌드 도구 및 문서와 함께 제공한다. (예: 배포 사이트에 업로드하거나 배포 패키지에 포함).
  - 동반 소스 코드는 해당하는 어느 제품 및 버전에 대한 레이블로 식별된다.
- 결과:
  - 동반 소스 코드를 제공해야 할 의무가 충족된다.

최종 검증



라이선스 의무 준수 인증

- 단계:
  - 동반 소스 코드 (있을 경우)가 올바르게 업로드 또는 배포되었는지 확인한다.
  - 업로드되거나 배포된 소스 코드가 승인된 버전과 동일한 것인지 확인한다.
  - 고지가 올바르게 공표되고 사용 가능하게 되었는지 확인한다.
  - 기타 식별된 의무가 충족되었는지 확인한다.
- 결과:
  - 확인된 배포할 컴플라이언스 결과물이 적절하게 제공됩니다.

이해도 점검

OPENCHAIN

- 컴플라이언스 실사 (Due Diligence)에 관여된 것은 무엇인가 (예시 프로세스에서 높은 수준에서의 단계를 설명하시오)?
  - 식별
  - 소스 코드 검사
  - 문제 해결
  - 리뷰 수행
  - 승인
  - 추적 등록 / 승인
  - 고지
  - 배포 전 검증
  - 동반 소스 코드 배포
  - 검증
- 아키텍처 리뷰는 무엇을 찾기 위한 것인가?

CHAPTER 7

컴플라이언스 함정 피하기

컴플라이언스 함정

OPENCHAIN

이 장에서는 컴플라이언스 프로세스에서 피해야 할 잠재적인 함정들에 대해 설명한다.:

1. 지식 재산권 (IP) 함정
2. 라이선스 컴플라이언스 함정
3. 컴플라이언스 프로세스 함정

지식 재산권 함정

유형 및 설명	발견	방지
Unplanned inclusion of copyleft FOSS into proprietary or 3rd party code: 이러한 유형의 실패는 개발 프로세스 중에 엔지니어가 FOSS정책에 위반하여 독점화할 소스 코드에 FOSS 코드를 추가할 때 발생한다.	이러한 유형의 오류는 다음과 일치하는 소스 코드를 스캔하거나 검사하여 발견할 수 있다: • FOSS 소스 코드 • 저작권 고지 이를 위해 자동화된 소스 코드 스캐닝 도구가 사용될 수 있다.	이러한 유형의 실패는 다음과 같은 방법으로 방지할 수 있다: • 엔지니어링 직원에게 컴플라이언스 이슈, 다양한 유형의 FOSS 라이선스 및 FOSS를 독점 소스 코드에 포함시키는 의미에 대해 교육한다. • 빌드 환경 내 모든 소스 코드에 대해 정기적인 소스 코드 스캔 및 검사를 수행한다.

지식 재산권 함정

유형 및 설명	발견	방지
Copyleft FOSS와 독점 소스 코드와의 계획하지 않은 링킹: 이러한 유형의 실패는 충돌하거나 양립가능하지 않는 라이선스하의 소프트웨어와 링킹한 결과로 발생합니다. 링킹의 법적 효력은 FOSS 커뮤니티에서 논쟁의 대상이다.	이러한 유형의 실패는 서로 다른 컴포넌트간의 링킹을 보여주는 종속성 추적 도구(dependency tracking tool)를 사용하여 발견할 수 있다.	이러한 유형의 실패는 다음과 같은 방법으로 방지할 수 있다: 1. 엔지니어링 직원들이 소프트웨어 컴포넌트를 FOSS정책과 충돌하는 라이선스와 링킹하여 이러한 법적 위험에 직면하지 않도록 교육을 제공한다. 2. 빌드 환경에서 종속성 추적 도구를 지속적으로 실행한다.
소스 코드 수정을 통해 독점 코드를 Copyleft FOSS에 포함	이러한 유형의 실패는 검사 또는 스캔을 사용하여 FOSS 컴포넌트에 도입한 소스 코드를 식별하고 분석하여 발견할 수 있다.	이러한 유형의 실패는 다음과 같은 방법으로 방지할 수 있다: 1. 엔지니어링 직원에게 교육을 제공한다. 2. 정기적인 코드 검사를 수행한다.

라이선스 컴플라이언스 함정

OPENCHAIN

유형 및 설명	방지
동반하는 소스 코드 / 적절한 라이선스, 저작자 또는 고지 정보의 제공 실패	이러한 유형의 실패는 제품이 시장에 출시되기 전에 제품 출시 주기에 맞추어 소스 코드를 캡쳐하고 체크리스트 항목을 게시함으로 방지할 수 있다.
동반하는 소스 코드의 잘못된 버전 제공	이러한 유형의 실패는 바이너리 버전에 맞는 동반하는 소스 코드가 공개되고 있음을 보장하기 위한 확인 단계를 컴플라이언스 프로세스에 추가함으로 방지할 수 있다.
FOSS 컴포넌트 수정에 대한 동반하는 소스 코드 제공 실패	이러한 유형의 실패는 FOSS 컴포넌트의 원본 소스 코드가 아닌 수정 소스 코드가 공개되고 있음을 보장하기 위한 확인 단계를 컴플라이언스 프로세스에 추가함으로 방지할 수 있다.

라이선스 컴플라이언스 함정

유형 및 설명	방지
FOSS 소스 코드 수정 표시 실패: FOSS 라이선스가 요구하는 대로 FOSS 소스 코드의 변경 사항을 표시하는 것의 실패 (또는 자세함과 명확성에 있어서 라이선스를 만족시키기에 불충분한 수준의 수정에 대한 정보 제공)	이러한 유형의 실패는 다음과 같은 방법으로 방지할 수 있다: 1. 소스 코드를 배포하기 전 확인 단계에 소스 코드 수정 표시 과정을 추가한다. 2. 엔지니어링 직원에게 대중에게 공개될 모든 FOSS 또는 독점 소프트웨어의 저작권 표시 또는 라이선스 정보가 업데이트 되는 것을 보장하기 위해 교육을 제공한다.

컴플라이언스 프로세스 실패

설명	방지	예방
개발자의 FOSS 사용 승인 신청 실패	이러한 유형의 실패는 회사의 FOSS 정책 및 프로세스에 대하여 엔지니어링 직원에게 교육을 제공함으로 방지할 수 있다.	이러한 유형의 오류는 다음과 같은 방법으로 예방할 수 있다: 1. "신고되지 않은" FOSS 사용을 발견하기 위해 소프트웨어 플랫폼에 대해 정기적으로 전체 스캔을 수행한다. 2. 회사의 FOSS 정책 및 프로세스에 대해 엔지니어링 직원에게 교육을 제공한다. 3. 직원 성과 리뷰에 컴플라이언스를 포함한다.
FOSS 교육의 실패	This type of failure can be avoided by ensuring that the completion of the FOSS training is part of the employee's professional development plan and it is monitored for completion as part of the performance review	This type of failure can be prevented by mandating engineering staff to take the FOSS training by a specific date

컴플라이언스 프로세스 실패

설명	방지	예방
소스 코드 검사 실패	이러한 유형의 실패는 다음과 같은 방법으로 방지할 수 있다: 1. 정기적인 소스 코드 스캔 / 검사 수행 2. 반복적인 개발 프로세스 검사가 하나의 중요한 단계가 되도록 보장	이러한 유형의 오류는 다음과 같은 방법으로 예방할 수 있다: 1. 일정이 지연되지 않도록 적절한 인력 제공 2. 정기적인 검사 시행
검사상 발견사항의 해결 실패 (스캔 도구 또는 검사에 의해 보고된 "발견(hits)"에 대한 분석)	이러한 유형의 실패는 검사 보고서가 완료되지 않은 경우 컴플라이언스 티켓 해결(i.e. Close)을 허용하지 않음으로 방지할 수 있다.	이런 유형의 실패는 FOSS 컴플라이언스 프로세스 내 승인 과정에서 블럭을 구현함으로써 예방할 수 있다.
FOSS 리뷰의 적시 요구 실패	이러한 유형의 실패는 엔지니어링 부서가 FOSS 소스 코드의 채택을 아직 결정하지 않았더라도 FOSS 리뷰 요청을 조기에 시작함으로써 방지할 수 있다.	이러한 유형의 실패는 교육을 통해 예방할 수 있다.

제품 출하 전 컴플라이언스 보장

OPENCHAIN

- 기업은 제품 (어떤 형태이든지) 출하 전에 컴플라이언스를 우선시 해야 한다.
- 컴플라이언스에 우선순위를 부여함으로 다음이 촉진된다 :
  - 조직 내 FOSS의 보다 효과적인 사용
  - FOSS 커뮤니티 및 FOSS 조직과의 관계 개선

커뮤니티와의 관계 수립

OPENCHAIN

상용 제품에 FOSS를 사용하는 회사라면 FOSS 커뮤니티(특히 상용 제품에 사용하고 배포하는 FOSS 프로젝트와 관련된 특정 커뮤니티)와 좋은 관계를 만들고 유지하는 것이 가장 좋다.

또한 FOSS 기관과의 좋은 관계는 컴플라이언스에 대한 최선의 방법에 대해 조언을 받을때 매우 유용할 수 있으며, 컴플라이언스 이슈가 발생할 경우에도 도움이 된다.

소프트웨어 커뮤니티와의 좋은 관계는 양방향 커뮤니케이션에 도움이 될 수 있다: 업스트림 개선 및 소프트웨어 개발자로부터의 지원

이해도 점검

OPENCHAIN

- FOSS 컴플라이언스에서 어떤 유형의 함정이 발생할 수 있는가?
- 지식 재산권의 실패 사례를 제시하시오.
- 라이선스 컴플라이언스의 실패 사례를 제시하시오.
- 컴플라이언스 프로세스의 실패 사례를 제시하시오.
- 컴플라이언스에 우선순위를 두는 것의 이점은 무엇인가?
- 커뮤니티와 좋은 관계를 유지하는 것의 이점은 무엇인가?

CHAPTER 8

개발자 가이드라인

개발자 가이드라인

- 고품질이면서 FOSS 커뮤니티 지원이 잘 되는 코드를 선택한다.
- 가이드를 요구한다.
  - 사용 중인 각 FOSS 컴포넌트에 대한 공식적인 승인을 신청한다.
  - 리뷰되지 않은 코드를 내부 소스 트리로 포함시키지 않아야 한다.
  - FOSS 프로젝트로의 외부 기여에 대해 공식 승인을 신청한다.
- 기존 라이선스 정보를 보존한다.
  - 사용하는 FOSS 컴포넌트의 기존 FOSS 라이선스 저작권 또는 라이선스 정보를 제거하거나 어떤 식으로든 훼손하지 않아야 한다. 모든 저작권 및 라이선스 정보는 모든 FOSS 컴포넌트에서 그대로 유지해야 한다.
  - FOSS 라이선스에 의해 요구 (예: 수정 버전의 이름 변경 요구)되지 않는한 FOSS 컴포넌트의 이름을 변경하지 않아야 한다.
- FOSS 리뷰 프로세스에서 요구되는 FOSS 프로젝트 정보를 수집하고 유지한다.

컴플라이언스 프로세스 요구사항을 예측

OPENCHAIN

- 작업 계획에 수립된 FOSS 정책을 준수하는데 필요한 시간을 포함시킨다.
  - FOSS 소프트웨어 사용에 대한, 특히, FOSS 코드를 독점 소스 코드 또는 제3자 소스 코드에 통합하거나 링킹하는 경우 등에 대한 개발자 가이드라인을 따른다.
  - 아키텍처 계획을 리뷰하여 양립가능하지 않는 FOSS 라이선스가 적용되는 컴포넌트들을 결합시키는 것을 방지한다.
- 모든 제품에 대해 컴플라이언스 검증을 항상 업데이트한다.
  - 제품별로 컴플라이언스 여부를 확인합니다: FOSS 패키지가 한 제품에서 사용하도록 승인되었다고 해서 반드시 두번째 제품에서도 사용하도록 승인된 것은 아니다.
- 그리고 새로운 버전의 FOSS로 업그레이드 할때마다
  - 동일한 FOSS 컴포넌트의 새로운 버전이 모두 검토되고 승인되었는지 확인한다.
  - FOSS 패키지의 버전을 업그레이드할 때 새 버전의 라이선스가 이전에 사용된 버전의 라이선스와 동일한지 확인한다.(버전 업그레이드간에 라이선스 변경이 발생할 수 있다.)
  - FOSS 프로젝트의 라이선스가 변경되었다면, 컴플라이언스 기록을 업데이트하고 새로운 라이선스가 충돌을 일으키는지 확인한다.

모든 FOSS 컴포넌트에 대해 컴플라이언스 프로세스 적용

- 유입된 소프트웨어
  - 공급업체가 제공하는 소프트웨어에 FOSS가 포함되었는지 파악하기 위한 조치를 수행한다.
  - 귀사의 제품에 포함될 모든 소프트웨어에 관한 귀사의 모든 의무를 평가한다.
  - 항상 소프트웨어 공급 업체로부터 받은 소스 코드를 검사하거나 또는 소프트웨어 공급업체는 제공하는 모든 소스 코드에 대하여 소스 코드 검사 보고서를 제공해야 한다는 회사 정책을 만든다.

이해도 점검

OPENCHAIN

- 개발자가 FOSS로 작업할 때 실행해야할 몇가지 일반적인 가이드라인을 나열하시오.
- FOSS 라이선스 헤더 정보를 삭제하거나 변경해야 하는가?
- 컴플라이언스 프로세스의 중요한 단계를 나열하시오.
- 이전에 리뷰된 FOSS 컴포넌트의 새로운 버전이 어떻게 새로운 컴플라이언스 이슈를 야기할 수 있는가?
- 유입되는 소프트웨어에 대하여 어떠한 위험에 대해 다뤄야 하는가?

Linux Foundation에서 주최하는 무료 교육(Compliance Basics for Developers)에 대해 자세히 알아보십시오:
https://training.linuxfoundation.org/linux-courses/open-source-compliance-courses/ compliance-basics-for-developers