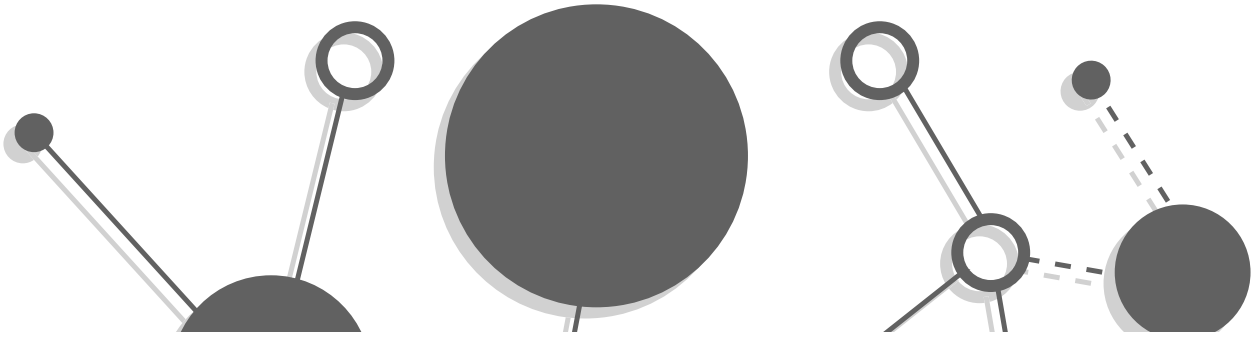
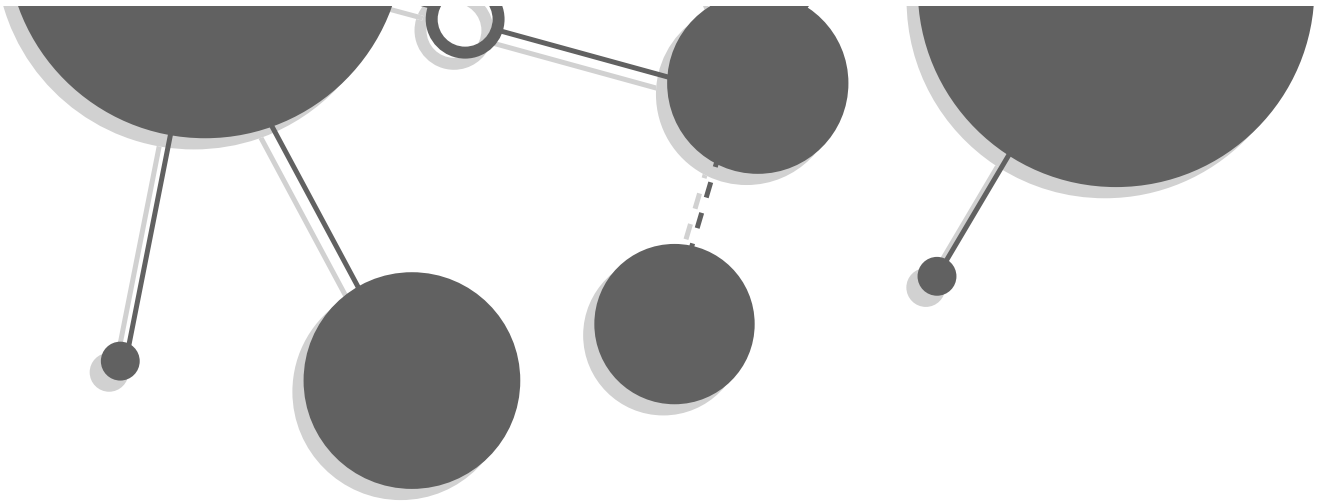


M&A取引における
オープンソース
監査
必須となるその基礎知識



THE LINUX FOUNDATION

イブラヒム ハダド(Ibrahim Haddad), Ph.D.

M&A取引におけるオープンソース監査

必須となるその基礎知識

イブラヒム ハダド(Ibrahim Haddad), Ph.D. 著

本書は、企業の合併・買収(M&A)取引におけるオープンソース 監査の全体像および実践的ガイドを提供するものです。また、買収企業、買収対象企業の双方においてオープンソース コンプライアンスへの備えを強化していく上で必要な基礎的ガイドラインについても説明します。

免責事項:本書は著者の法務専門職とは異なる経歴・知見に基づくものであり、法的アドバイスを提供するものではありません。また、本書は、著者自身の見解を示したものであり、現在もしくは過去の著者所属企業の見解を反映したものではありません。

THE LINUX FOUNDATION

目次

1 はじめに

私たちはソフトウェアによって定義された時代に生きています。私たちが行うことのほぼすべては、何らかの方法でソフトウェアによって計画、具体化、分析、および管理されています。その大きなソフトウェアという傘の下でも、オープンソース ソフトウェアは最も重要なものでしょう。あらゆる産業の企業が競ってオープンソース プロジェクトを使い、参加し、コントリビュートして、プロジェクトの恩恵を受けようとしています。その恩恵には、社外エンジニアリング リソースの有効活用による市場投入時間短縮や、イノベーションの加速など、さまざまなものがあります。

このような考え方は、企業の合併・買収取引にも当てはまります。テクノロジー企業の買収はどんなものであれ、何らかの形でソフトウェアに関係してくるからです。買収企業が買収対象企業のソフトウェアやコンプライアンス プロセスに対して包括的レビューを行うソフトウェア デュー デリジェンス(適正評価、精査)のプロセスは、合併・買収において標準的なものになりつつあります。このプロセスでオープンソース ソフトウェアが対象になることも多いのですが、これらにはプロプライエタリ ソフトウェアとは異なる検証課題があります。

本書では、企業の合併・買収(M&A)取引におけるオープンソース ソフトウェアの監査プロセスの全体像について触れていきます。

2 一般的なオープンソース使用シナリオ

オープンソースのデュー デリジェンスの話に入る前に、買収対象企業の開発プロセスでオープンソース ソフトウェアが取り込まれ得るさまざまな局面を理解することが有益です。これは、企業が意識的または無意識に、自社のソースコード ベースにオープンソース ソフトウェアを組み入れる場合にも当てはまります。交通違反キップを切られた際に、自らの義務を知らなかったというのは言い訳になりません。これと同じように、複数の提供元のソフトウェアが使用される可能性のあるさまざまな局面を理解しておくことが賢明です。最も一般的なオープンソース ソフトウェア使用シナリオは、取り込み(Incorporation)、リンク(Linking)、および改変(Modification)です。

オープンソースのコンポーネントに変更を加えることや、オープンソースのコードをプロプライエタリ コンポーネントやサードパーティ コンポーネントに注入(Inject)することは、監査サービス プロバイダーのコード発見・報告手法に影響を与える可能性があります。オープンソース監査のサービス プロバイダーと関わる際には、彼らの発見アプローチがどのようにオープンソースのコードを捕捉するのかを理解することが、しばしば助けとなります。

2.1 取り込み(Incorporation)

開発者は、オープンソース コンポーネント全体を使うこともあれば、コンポーネントの一部(スニペット)をソフトウェア製品の中にコピーすることもあります。そのようなシナリオは、取り込まれたオープンソースのコードのライセンスや、それを取り込むソフトウェア コンポーネントのライセンスによっては、許容することができ、ライセンス リスクもないでしょう。しかし、コピーされたオープンソースのコードがプロプライエタリのコードベースのライセンスと相入れないような場合など、取り込みが問題を生じさせることもあります(図1)。

オープンソース ライセンスは、企業の法的責任や、自社コードのプロプライエタリ性に影響し得るさまざまな義務を伴うため、こういった取り込みは、サードパーティ ライセンスのソフトウェアと同様のプロセスに従い、追跡、申告、および社内承認されるべきです。

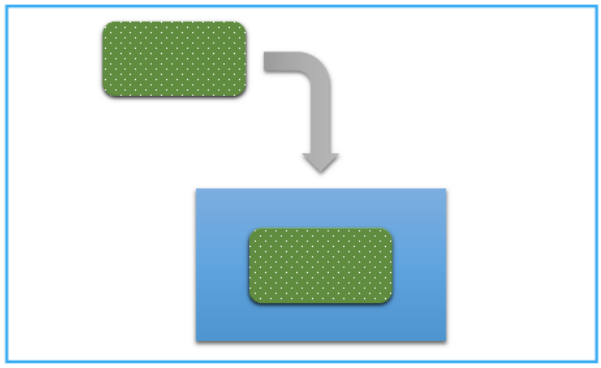


図1:オープンソース コード(緑色ドット)の別コード体系(青色)への取り込み(Incorporation)

ソースコード監査は、申告されていないオープンソースのコードベースへの取り込みを発見し、買収後の好ましくないサプライズを回避するためのものです。申告されない取り込みは、買収対象企業がオープンソース コンプライアンスについて開発者のトレーニングを十分に実施していなかった場合や、コントラクターやインターンといった長期にわたる記録管理を行わない期間作業者に依存し続けていた場合に起こる可能性が高くなります。

取り込みが行われた場合、人間の目でソースコードを見てもはっきりとわからないことが多いのですが、スニペットを発見・照合する機能のあるソースコード スキャンツールであれば、容易に発見できます。

2.2 リンク(Linking)

リンク(Linking)は、たとえばオープンソースのライブラリを使用する時などによくあるシナリオです。このシナリオでは、開発者がオープンソース ソフトウェアのコンポーネントと自社ソフトウェア コンポーネントをリンクさせます(図2)。このシナリオに対応する用語は、静的リンク(Static link)、動的リンク(Dynamic link)、結合(Combining)、パッケージング(Packaging)、相互依存性の生成(Creating interdependency)などを

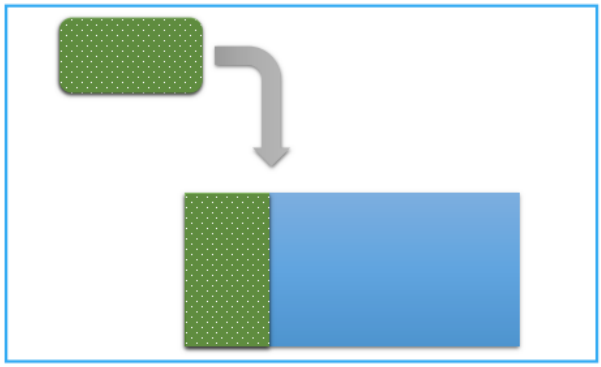


図2:オープンソース コード(緑色ドット)の別のコード体系(青色)へのリンク(Linking)

含めていくつかあります。リンクは通常、ソースコードの目視確認で簡単に検出されます。なぜなら、ライブラリがファイルの先頭部分でインクルードされていたり、リンクされるコードが別名のディレクトリやファイルにあったりするためです。

リンクが取り込み(Incorporation)と異なるのは、ソースコードがコピーされても一体化せず、分離されていることです。リンクの相互作用は、コードが1つの実行バイナリにコンパイルされる時(静的リンク)、あるいは主プログラムが実行され、リンクされたプログラムを呼び出す時(動的リンク)のいずれかの時点で生じます。

2.3 改変(Modification)

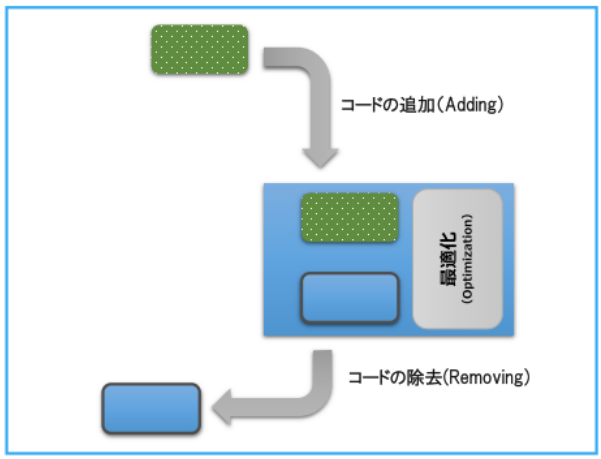


図3:オープンソースのコードへの開発者によって適用された改変(緑色・ドット)

改変は、開発者がオープンソース ソフトウェアのコンポーネントに変更を加える一般的なシナリオで(図3)、以下のようなものがあります。

- オープンソース ソフトウェア コンポーネントに新しいコードを追加(Adding)/注入(Injecting)する
- オープンソース ソフトウェア コンポーネントに対しバグ修正(Fixing)、最適化(Optimizing)、または変更(Making change)する
- コードを削除(Deleting)または除去(Removing)する

2.4 開発ツールに関する留意事項

このような作業を開発者に気づかれずに実施してしまう開発ツールがあることを認識しておくことが重要です。たとえば開発者は、開発プロセスの一部を自動的に行ってくれるツールを使うことがあります。例を挙げるなら、ユーザー インターフェイス

のテンプレートを提供するグラフィックスのフレームワークや、物理エンジンを提供するゲーム開発用のプラットフォーム、クラウド サービスへのコネクターを提供するソフトウェア開発キット(SDK: Software Development Kit)などです。これらのツールは、前述の処理を提供する目的のために、開発者の作成物のビルド時に開発ツールのコードの一部をその作成物に注入しなければなりません。特に、生み出された作成物がしばしば静的リンクされていることを考慮すると、このように開発ツールによって注入されたコードのライセンスを検証することが必要です。

3 オープンソース監査

M&A取引にはさまざまな形がありますが、買収に伴ってオープンソースの義務を引き継ぐことによるインパクトを検証する必要があることは共通しています。オープンソース監査は、オープンソースの使用の深さと依存度について理解するために実行されます。また、オープンソース監査は、コンプライアンスのあらゆる課題や、買収対象企業のエンジニアリング実務についても、重要な洞察を与えてくれます。

3.1 なぜオープンソース監査を行うのか？

オープンソース ライセンスはソフトウェアの再頒布方法に制約を課すことがあります。このような制約は買収企業のビジネスと相反するかもしれないため、早期に発見されるべきです。オープンソース ソフトウェアがあることで買収対象企業の資産に影響し得る例として、次のようなものがあります。

- オープンソース ライセンスは、一般的にコード頒布の際に何らかの義務を課します。その一例がGNU General Public License (GNU GPL)で、派生物もしくは結合物を同じライセンスの下で利用できるようにすることを要求します。また、ドキュメント内での通知や告知を求めたり、製品の販売促進のやり方に制約を課したりするライセンスもあります
- オープンソース ライセンスの義務の不履行が、訴訟、費用のかさむ再設計、製品リコール、悪評などにつながる可能性もあります

3.2 オープンソース監査を委託すべきか？

よくある質問が、そもそもオープンソース監査が必要なのか、というものです。その質問に対する答えは、企業によって、買収の目的によって、またソースコードのサイズによって異なります。たとえば、小規模な買収の場合、買収対象企業からオープンソースの部品表(Bill of Material, BoM)の提供を受けることができれば、それをレビューし、エンジニアリング リーダーと共にオープンソース実務について議論

を実施するだけでよしとする企業もあります。買収の目的がたとえ人材の獲得にあったとしても、監査をすることは、出荷済み製品のライセンス義務に起因する非開示の責任の有無を明らかにするのに有益です。

3.3 インプットとアウトプット

監査プロセスでは、主となるインプットが1つ、主となるアウトプットが1つあります（図4）。プロセスのインプットは、買収取引に関係するソフトウェア スタック全体です。ここにはプロプライエタリ、オープンソース、そしてサード パーティ ソフトウェアがあります。プロセスの出力側、つまり主となるアウトプットは、以下のものを列挙した詳細なオープンソースの部品表です。

- コンポーネントとして使用されているすべてのオープンソース ソフトウェア、それらの起源およびライセンス
- プロプライエタリ ソフトウェアやサードパーティ ソフトウェアで使用されたすべてのオープンソースのスニペット、その起源となるコンポーネントおよび確認されたライセンス

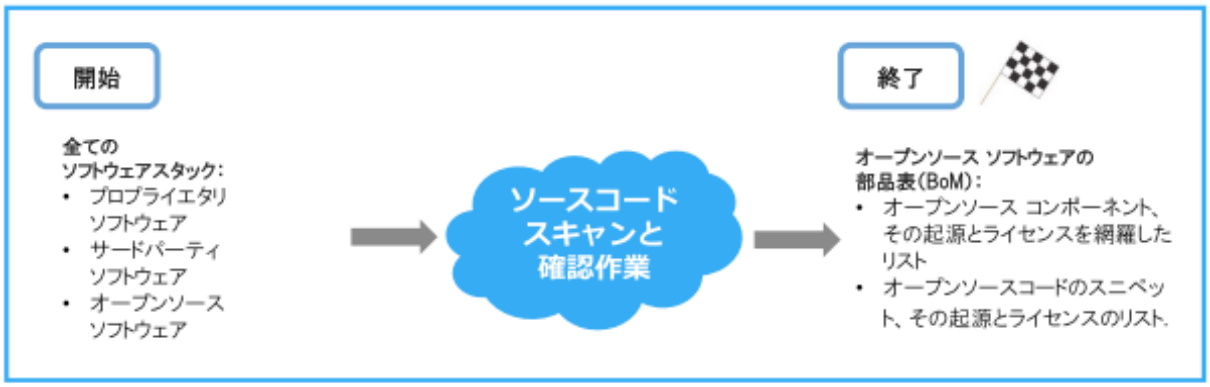


図4：監査プロセスのインプットとアウトプット

4 監査業務のスコープを評価する

監査の規模、スコープ、およびコストは、合併・買収取引ごとに異なり、一般的にはソースコードのサイズや複雑さに比例して増加します。オープンソース監査に対する(コストと時間の)見積もりを出すために、監査人はコードベースのサイズとその特徴、およびプロジェクトの緊急性について、基本的に理解する必要があります。

監査人が挙げる最初の質問は、ソースコード ベースのメトリクスに関するものでしょう。たとえば、監査対象のコードベースのサイズ、ソースコードのライン数、ファイルの数などです。また彼らは、コードベースがソースコードだけなのか、あるいはバイナリ ファイルやコンフィグレーション ファイル、ドキュメント、およびその他のファイルフォーマットのものを含んでいるのかも質問するでしょう。監査対象のファイルの拡張子を知ることが、監査人にとって有益になることもあります。

成熟した企業では、自社製品やプロジェクトで使われているオープンソース のコンポーネントやバージョンについて、記録を残しています。こういった情報は非常に有益で、想定されるワークロードに関する監査人の理解を高めます。

監査費用の議論は、規模やスコープに基づいて監査プロセスの中で早期に起こるため、買収企業は前述のような情報のすべてにアクセスできないかもしれません。しかし少なくとも、監査人はスキャンするファイルの数を作業開始前に理解しておく必要があり、追加の情報が見積もりの精度を上げる助けになります。監査人は、作業スコープを理解するための十分な情報を得たら、緊急性も理解する必要があるでしょう。それが監査のコストに大きく影響するからです。

5 監査手法

オープンソース監査を実施する際に、買収企業にとって価値のあるツール機能があります。最も重要な機能の1つが、買収対象企業のプロプライエタリ コードに混入したオープンソース コードのスニペット(もしくはその逆)を検索する機能です。また、検知した結果から誤検知(False positive)を自動的に削除する機能もあり、手作業を最少化できます。

監査の手法には以下の3つがあります。

1. 伝統的な監査。監査人がすべてのコードに完全アクセスし、実地もしくはリモートで監査を実施します
2. ブラインド監査。監査人はソースコードを見ることなく、リモートで作業を行います
3. DIY(Do It Yourself)監査。買収対象企業もしくは買収企業が、自分自身で大半の監査作業を実際に行います。監査企業からは、監査ツール、サポート、さらには、監査結果に対する無作為的な検証などが提供されることがあります

5.1 伝統的な監査手法

私がこの手法を「伝統的(Traditional)」と呼ぶのは、これがオープンソース コンプライアンスを目的としたソースコード スキャンのもともとの手法だったためです。伝統的な監査では、サードパーティの監査企業の監査人が、ソースコードにクラウド システム経由でリモートからアクセスしたり、物理的に現地へ足を運んでソースコード スキャンを実施したりします。

図5は、次に挙げる伝統的な監査手法のプロセスを表しています。このプロセス

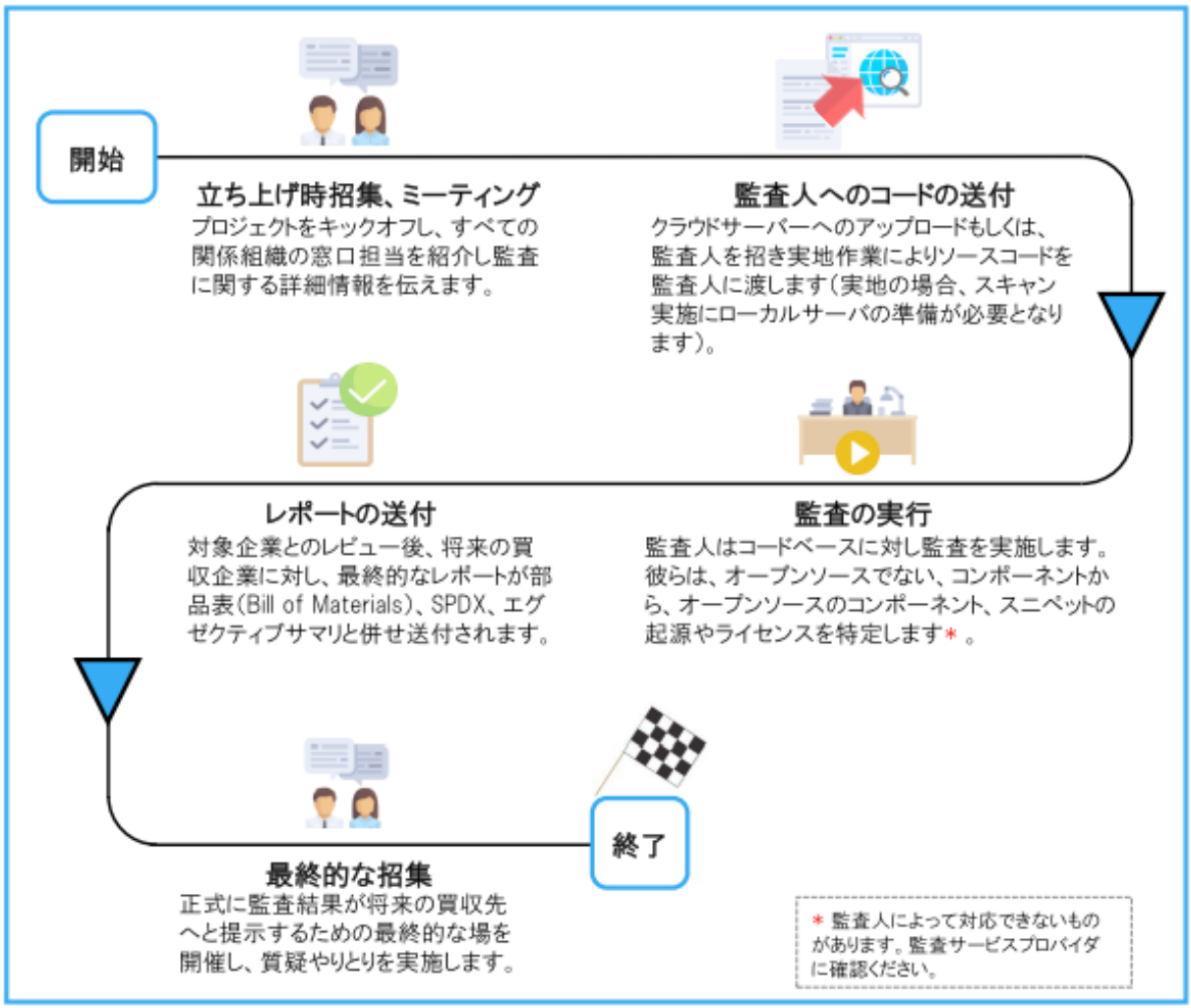


図5:M&A取引における伝統的な監査手法

は、サービス プロバイダーごとに微妙に異なってくるので留意してください。伝統的な監査プロセスの典型的なものとして、以下のようなステップがあります。

- 監査人が、作業内容をよりよく理解するために、買収企業に質問状を送付する
- 買収企業は、監査人が監査スコープとパラメーターをよりよく理解できるように、これに答える
- 監査人が、この応答をもとに見積もりを提供する
- 見積もりについて合意され、サービス契約書、作業明細書、守秘義務契

約書(NDA)などにサインされる(注:図5、6、7にある「開始」は合意文書すべてにサインされた後の実際の監査プロセスの開始を想定しています。)

- セキュアなクラウド経由のアップロード、もしくは実地訪問に基づく監査によって対象企業のコードにアクセスする権利が、与えられる
- 監査人が、対象企業のソースコードをスキャンし、誤検知分を処理し、結果を評価する
- 監査人が、レポートを生成し依頼主に送付する
- 電話会議、もしくはフェイス ツー フェイスのミーティングによって、監査人とともに結果をレビューし、質疑のやり取りを実施する

この手法は、ほとんどの監査サービス プロバイダーで一般的なものです。そのため、同じ監査業務に対して複数企業からの入札を集め、要求に合った最良の入札者を選択することもできます。このモデルでは、買収対象企業は進んでコードを監査人に提供するか、監査人が業務を遂行するために実地に訪れるのを許可しなければなりません。

5.2 ブラインド監査

ブラインド監査は、ストックホルムを拠点としたFOSSID AB社によって開発された、M&A取引における守秘義務要求に対応するための手法です。(FOSSID ABは会社名、FOSSIDはツール名です。)

FOSSID AB社のプロプライエタリな技術を用いることで、ソースコードを見ることなく監査を実施し、レポートを生成できます。図6は、FOSSID AB社が用いる、M&A取引においてソースコードの機密を保つようデザインされたブラインド監査のプロセスを表しています。ブラインド監査の1つの大きなメリットは、監査人がソースコードにアクセスせずにレビューを完了できることです。さらに、買収企業が十分な予防措置をとることで、監査人でも買収対象企業を特定できないほど高度な機密性が提供されます。著者の知る限り、オープンソース コンプライアンス サービスの提供企業でこのような手法をとれるところは他にありません。

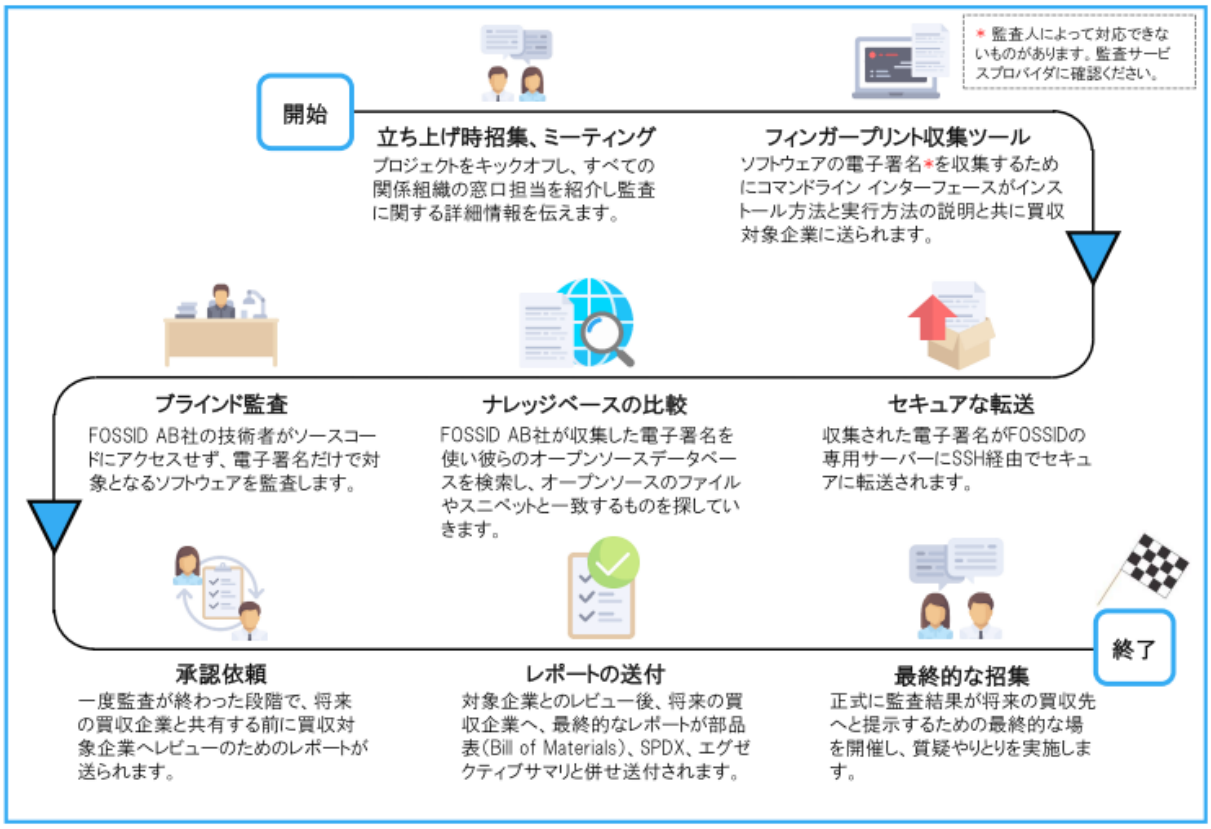


図6:M&A取引を想定したFOSSIDを用いたブラインド監査

5.3 DIY監査

Do-It-Yourself(DIY)監査は、買収企業もしくは買収対象企業が自らスキャンを実施できるように、時間限定でクラウドのコンプライアンス ツールへのアクセスを提供します。すべてのナレッジベースやレポート機能にアクセスすることで、内部監査を実施できるようになります。このアプローチは、スキャン結果を解釈してその是正手続きを提案する十分な知見がある従業員を持つ企業にとって、特に興味深いものとなります。M&Aプロセスを年間数回実施するような企業は、この手法により迅速に費用効率を上げることができます。さらなる監査の完全性確保を目的として、監査ツールのサービス プロバイダーが発見事項の検証を実施し、独立した形で認定を行う手法もあります。

図7はFOSSID AB社のツールを用いた監査手法を表しています。このアプローチにはいくつかのメリットがあります。たとえば、社内リソースを使用するため、サードパーティの監査人の対応可能状況に依存せず、必要な時にすぐに監査を開始できます。時間の短縮や、社外コストの削減も期待できます。直接コードにアクセスして解決できる人によって実施されるため、あらゆるコンプライアンス問題に即座に対応できます。そして最後に、この監査における正確性や網羅性を確保するために、監査ツール提供者によって検証することができます。FOSSID AB社は、DIYサービスの一環として、買収対象企業で監査されるべきと示されたファイルのX%(このX%は見積もりの合意の一部として決定)の無作為抽出検証を提供しています。

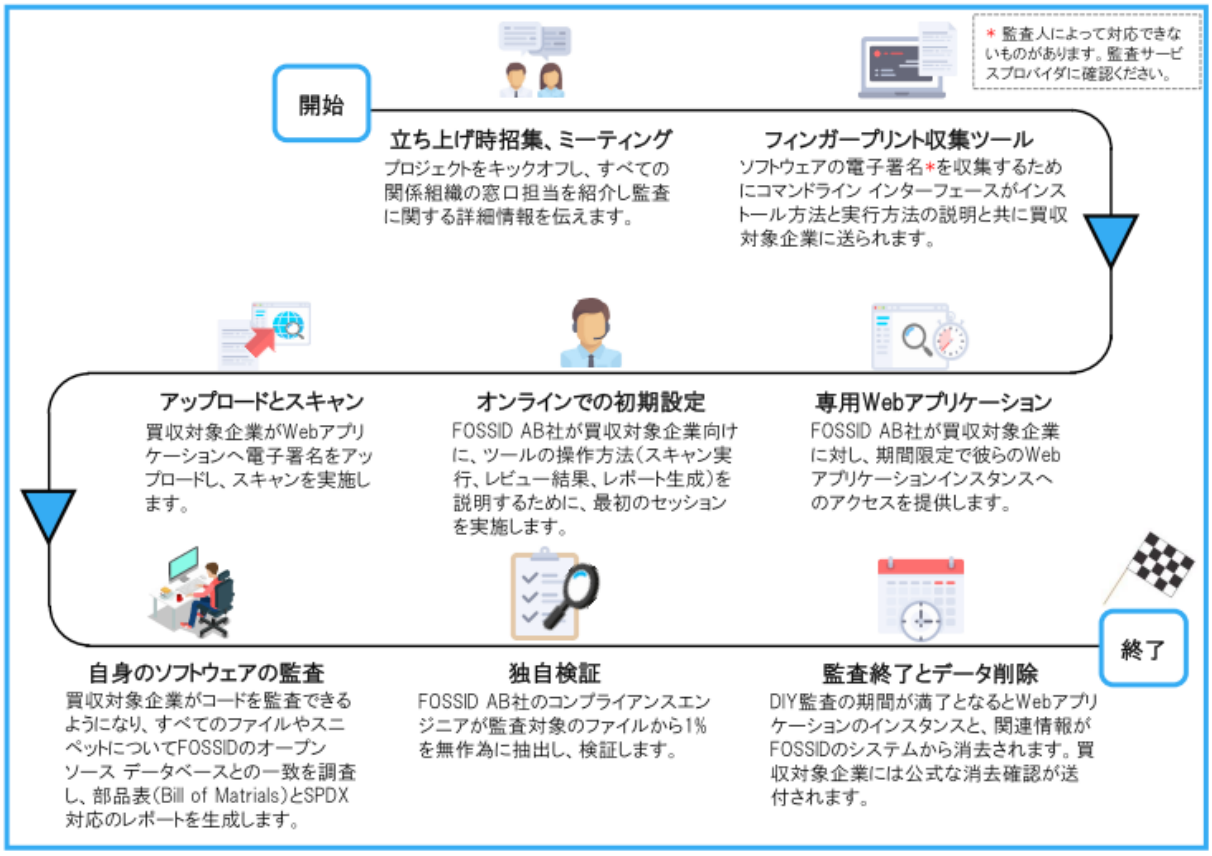


図7：M&A取引を想定したFOSSIDを用いるDIY監査プロセス

6 最終レポートに関する留意事項

多くの監査ツールは、潜在的な問題に焦点を当てるようにチューニングできます。結果を入念にレビューしてみると、その多くは問題ではないかもしれませんが、多くのノイズが出ることを覚悟しておくべきです。ノイズは、コードツリー内にあっても使われていない残存コードのようなものから出ることもあります。したがって初期のレポートは冗長になる場合があるので、真の問題を発見するためにレポートをフィルターしてノイズを除去する時間を準備しておくべきです。

通常、Software Package Data Exchange (SPDX) に準拠したレポートは、要求に応じて提供されます。したがって、監査サービス提供者にそのようなレポートを提供して欲しい場合は、要求する必要があるでしょう。

7 セキュリティとバージョン管理

ソフトウェアはワインのようなものでなく牛乳のように経年劣化するもの、というのは一般的に受け入れられている事実です。そしてオープンソースであれ、それ以外のものであれ、すべてのコードにはセキュリティの脆弱性がついて回ります。しかし、オープンソースのプロジェクトでは、そのような脆弱性をその修正プロセスとともに公開します。この公開は、修正実行の前に行われることもあれば、後に行われることもあります。また、更新されていないオープンソース コードは、すでに広範囲でアクティブに悪用されている脆弱性を含んでいる可能性があります。セキュリティとバージョン管理は、オープンソース コンプライアンスのデュー デリジェンスの範疇ではありませんが、ソースコード スキャン サービスを提供する企業が、特定したオープンソースのコンポーネントを既知のオープンソースのセキュリティ脆弱性とマッピングするサービスを提供することもあります。

8 買収前、買収後の改善

この段階で買収企業は、買収対象企業がどのようにオープンソース ソフトウェアを使用および管理し、どのように確実にオープンソース ライセンスの義務を履行してきたかについて、明確な情報を得るべきです。両社はこの情報を使用して、あらゆるコンプライアンス問題の是正策を協議することになります。監査において問題が明らかになった場合、未解決の処理の一部としてそれらを解決するためには、いくつかの選択肢が考えられます。最初の選択肢は問題を引き起こすコードすべてを単純に削除することです。そのオープンソース ソフトウェアが単にプロプライエタリのコードを補っているだけというのであれば、完全に削除できるかもしれません。もう1つの選択肢は、問題となっているコンポーネント周辺をくまなく設計するか、またはクリーンルーム方式（他社の著作権やトレード シークレットを侵すことなく独自開発する手法）ですべてのコードを書き直すことです。

そのコードが必須のものである場合、または以前に頒布されていたものである場合、残される選択肢は、そのコードをコンプライアンスな状態にすることだけです。それぞれの選択肢に要するコストは、対象企業の買収価格を決定する際に使用できます。いずれを選択しても、オープンソース コードを取り込むのに関わった人物を特定し、是正作業に参加してもらうことが重要です。彼らは、問題を解決するための有益な資料や知識を持っている可能性があります。

9 買収対象企業として監査に備える

オープンソース コンプライアンス監査にパスすることは、備えていれば難しくはありません。しかし買収企業が買収への関心を示してから初めてその準備を始める場合は、簡単ではないでしょう。ここで示す作業は、日常のビジネスや開発と密接に関係しています。その目的は、企業が必ずすべてのオープンソース コンポーネントを追跡し、オープンソース コンポーネントの使用によって生じたオープンソース ライセンスの義務を確実に尊重するようにすることです。これらの取り組みは、その企業が企業取引の対象になった時に、好ましくないサプライズのリスクを最小にしてくれる点で大きな助けとなります。

9.1 コードの中身を知る

コードの中に何があるのかを知ることは、コンプライアンスにおける黄金律（Golden rule）です。すべてのソフトウェア コンポーネントについて、起源やライセンス情報などを含む目録を保持する必要があります。目録には、自身の組織で作成されたコンポーネント、オープンソース コンポーネント、およびサードパーティ起源のコンポーネントを記載します。ここで最も重要なのは、オープンソースのコンポーネントを特定および追跡するプロセスを持つことにあります。必ずしも複雑なコンプライアンス プログラムは必要ありませんが、「ポリシー」、「プロセス」、「スタッフ」、「トレーニング」、「ツール」の5つの基本要素は具備しておくべきでしょう。

9.1.1 ポリシーとプロセス

オープンソース コンプライアンス ポリシーは、オープンソース ソフトウェアの管理（使用とコントリビューションの両方）を統制する一連のルールです。プロセスは、企業がこれらのルールを日常ベースで実践していく方法に関する具体的な仕様です。コンプライアンスのポリシーとプロセスが、オープンソース ソフトウェアの使用、コントリビューション、監査、頒布といったさまざまな側面を統制します。

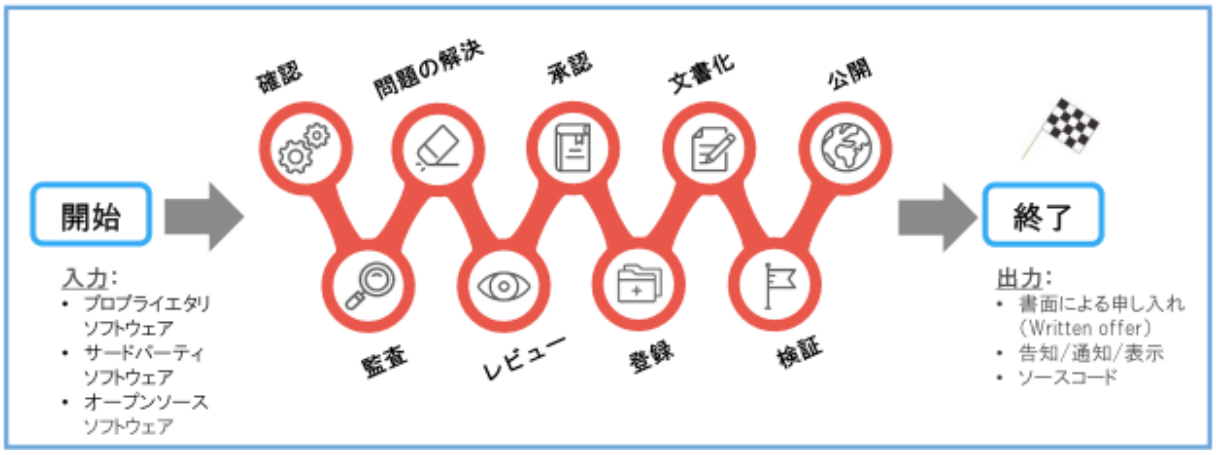


図8:オープンソース コンプライアンス プロセスの開始からから終了まで(サンプル)

図8は、サンプルのコンプライアンス プロセスです。企業が製品やソフトウェア スタックを開発する際に、各ソフトウェア コンポーネントはデュー デリジェンスの一環としてこれらのさまざまなステップを経ることになります。

1. 外部から入ってくるすべてソースコードを特定する
2. ソースコードを監査する
3. 監査で明らかにされたあらゆる問題を解決する
4. 適切なレビューを実施し、これを完遂する
5. オープンソースの使用についての内部承認を得る
6. ソフトウェア目録にオープンソース ソフトウェアを登録する
7. 製品の関連文書にオープンソース ソフトウェアの使用状況を反映する
8. 頒布に先立ちすべてのステップに対する検証を行う
9. ソースコードを頒布し、頒布に関する最終検証を行う

このプロセスからのアウトプットは、公開可能なオープンソースの部品表(BoM: Bill of Materials)ですが、それに書面による申し出(Written offer)、著作権、ライセンス、帰属表示の告知文など、部品表にあるコンポーネントの法的義務を履行していることを示すものを伴います。オープンソース コンプライアンス プロセスの詳細については、The Linux Foundationから公開されているフリーの電子書籍「Open Source Compliance in the Enterprise」を参考にしてください。

9.1.2 スタッフ

大企業におけるオープンソース コンプライアンス チームは、オープンソース コンプライアンスを確実にするという目標を持つさまざまな個人で構成される分野横断的なグループになります。中核となるチーム(Core team)は、通常「オープンソース レビュー ボード(Open Source Review Board: OSRB)」と呼ばれ、エンジニアリングや製品チームからの代表者、1人以上の法務専門家、およびコンプライアンス オフィサーで構成されます。また、ドキュメント、サプライチェーン、経営企画、情報システム、ローカライゼーションなど、コンプライアンスの取り組みに継続的な貢献をする複数部門のさまざまな人々によって、拡張チーム(Extended team)が形成されます。しかし、小規模企業やスタートアップにおいては、1人のエンジニアリング マネージャーと1人の法務専門家というシンプルな構成もあるでしょう。どのような構成になるかは、会社によって異なります。

9.1.3 トレーニング

教育は、コンプライアンス プログラムの重要な構成要素です。教育によって、従業員は確実にオープンソース ソフトウェアの使用を統制するポリシーを正しく理解できるようになります。オープンソースとコンプライアンスのトレーニングを提供する目的は、オープンソースのポリシーや戦略に対する意識を高め、オープンソース ライセンスの事実と課題についての共通理解を構築することです。製品やソフトウェア ポートフォリオへのオープンソース ソフトウェア取り込みに関するビジネス上および法務上のリスクについても、カバーすべきでしょう。

公式および非公式なトレーニングの両方を活用できます。公式な手法には、コース修了のために従業員が確認試験に合格する必要のあるインストラクター指導型トレーニング コースがあります。非公式なものには、ウェビナーやブラウンバッグ セミナー(昼食持参セミナー)、そして新規採用従業員に対するオリエンテーション セッションの一環で行う新入社員向けプレゼンテーションなどがあります。

9.1.4 ツールの活用

オープンソース コンプライアンス チームは、ソースコードの監査の自動化、オープンソース コードの発見、およびそのライセンスの特定のためにツールを頻繁に用います。これらのツールとしては、コンプライアンス プロジェクト管理のためのもの、ソフトウェア目録のためのもの、ソースコードやライセンスを特定するためものなどが挙げられます。

9.2 「コンプライアンス」の状態にある

オープンソース ソフトウェアを含む製品を出荷した場合、意図的であるかどうかにかかわらず、それらソフトウェア コンポーネントを統制する各種ライセンスを順守している必要があります。コードの中に何があるかを正確に知ることが重要であり、整備された部品表(BoM)の存在がコンプライアンスの履行を容易にします。

コンプライアンスの状態にある、というのはそう単純な話ではなく、ライセンスやコードの構造に応じて製品ごとに変わってくるものです。ハイレベルでは、コンプライアンスにある状態とは、以下を意味しています。

1. オープンソース ソフトウェアのすべての使用を追跡している
2. 製品として出荷したイメージファイルにあるすべてのソフトウェアのオープンソース部品表(BoM)を作成している
3. オープンソース ライセンスの義務を履行している
4. ソフトウェアのアップデートを発行するたびに同じプロセスを繰り返している
5. コンプライアンスに関する問い合わせに対し真摯にかつ迅速に対応している

9.3 セキュリティのために最新版を使用する

包括的なコンプライアンスプログラムのメリットの1つは、安全でないバージョンのオープンソース コンポーネントを含む製品を簡単に見つけて置換できることです。最近では大抵のソースコード スキャンツールが、古いソフトウェア コンポーネントで明らかになったセキュリティ脆弱性にフラグを付ける機能を提供しています。オープンソース コンポーネントをアップグレードする際に考えるべき大事なことの1つは、前のバージョンと同じライセンスが維持されていることを常に確認することです。オープンソース プロジェクトでは、メジャー リリースのタイミングでライセンスが変更されることがあります。セキュリティ上の問題があるバージョンを使う状況を回避するためにも、企業はオープンソース プロジェクトのコミュニティに積極的に参加することが望まれます。使用しているオープンソース プロジェクトのすべてにおいてアクティブに活動することは合理的・現実的ではないため、最重要コンポーネントを特定するためにある程度の優先順位付けをする必要があります。参加のレベルは多岐にわたり、メーリング リストへの登録や技術的議論への参加から、バグ修正や小さめの機能のコントリビューション、さらには大規模なコントリビューションまでさまざまです。少なくとも、特定のオープンソース プロジェクトに取り組んでいる企業内開発者がメーリング リストを閲覧し、セキュリティの脆弱性や修正対応に関するレポートに目を配っておくことは有益です。

9.4 コンプライアンスの取り組みを測る

あらゆる規模の組織にとって、最も簡単で効果的な最初のステップは、OpenChainプロジェクトに参加し、「OpenChain適合（OpenChain Conformant）」のステータスを得ることです。オンラインもしくは机上で一連の質問を埋めることでこの作業は実施できます。OpenChain適合に使われる質問は、組織のオープンソース ソフトウェア コンプライアンスに対するプロセスやポリシーをチェックする助けにもなります。OpenChainはISO9001に似た業界標準です。大局を重視し、詳細なプロセスやポリシーの実装については、個々の組織の裁量に任せています。OpenChain適合とは、オープンソース コンプライアンスのプロセスやポリシーが存在していること、およびサプライヤーや顧客の求めに応じて詳細情報を提供できることを意味します。OpenChainは、グローバルなサプライチェーンにまたがる組織間の信頼を築き上げるために設計されているのです。

The Linux Foundationの「**自己査定用チェックリスト(Self-Assessment Checklist)**」はオープンソース コンプライアンス プログラムを成功させるために必要な要素に加え、コンプライアンスのベストプラクティスを提供した、幅広いチェックリストです。企業はぜひこれを社内の自主運営型チェックリストとして使用し、自社のコンプライアンスをベストプラクティスと比較して評価してください。

10 買収企業として監査に備える

買収企業は、監査を依頼する前は行動を起こして意思決定を行う必要があり、監査結果を受領した後は新たな義務が発生します。

10.1 ニーズに合わせ適切な監査モデル・監査人を選択する

前述のとおり監査では主に3つの手法を用いることができますが、そのどれが自社の具体的状況と合っているのかを決める必要があります。

10.2 何に留意すべきかを知る

スキャンしたコードの複雑さによっては、ソースコード監査レポートが膨大な情報を提供することがあります。どのライセンスやユースケースが大事だとみなすかを明確にする際に重要になります。

10.3 適切な質問をする

オープンソース監査レポートは、買収対象企業のソースコードや関連するライセンスについての大量の情報を提供します。しかし、コンプライアンス関連の懸念事項を明確化および確認するために、その他の多くのデータ要素にさらなる調査が求められるでしょう。本節では、買収企業にとって重要なことが何か、買収対象企業とともに取り組むべき問題は何か、といった枠組みを作るためのスタート地点としての設問集を提示します。

- 買収企業や買収対象企業の知財を危険にさらすようなライセンスのコードを買収対象企業が使っていないか？
- よくわからない起源やライセンスのコードのスニペットはないか？
- 買収対象企業のオープンソース コンプライアンスの実務は、十分成熟しており、包括的なものか？
- 買収対象企業は、自社のオープンソース コンポーネントの既知の脆弱性を追跡しているか？
- 製品を提供する際、買収対象企業はオープンソース ライセンスの義務を履行するために必要なすべての資料（書面による申し出、必要な告知/通知/表示、ソースコードなど当てはまるもの）を提供しているか？
- 買収対象企業のコンプライアンス プロセスが、製品リリース計画に基づく開発スピードと合っているか？
- 買収対象企業は、社内外からのソースコード要求に対しタイムリーな形で対応することができるプロセスを用意しているか？

10.4 買収取引実行前に解決すべき項目を特定する

オープンソース監査は、ライセンスやコンプライアンスの履行について、買収企業が受け入れられないような実態を明らかにすることもあります。その場合、買収企業は、そのような事実の影響を軽減することを契約締結の条件として要求することが可能です。たとえば、買収対象企業は「Aライセンス」というライセンス下にあるコンポーネントのコードを使っているが、買収企業は「Aライセンス」のソースコードの使用を禁止する厳格なポリシーを持っているようなケースです。こういった状況では、両社が議論して、可能な解決策を導き出すことが必要になります。

10.5 買収後のコンプライアンス是正計画を策定する

コンプライアンス是正計画の策定は、大規模企業が小さいスタートアップを買収し、子会社として運営し続けるような場合は特に重要です。このシナリオでは、買収企業はしばしば買収対象企業に対し、秩序だったコンプライアンス ポリシーやプロセスの確立を支援し、買収企業で使われているトレーニングを提供し、継続的な指導や支援を行います。

11 コンプライアンスにおいて推奨される開発実務

オープンソース ライセンスのコンプライアンス活動を支援する開発実務の確立については、詳細な推奨事項を添えた文書がいくつか執筆されています。この節では、その中でも最も重要なものに簡単に焦点を当てます。それらに従うことによって一般的なコンプライアンス問題の多くを排除することができるでしょう。

11.1 推奨プラクティス

- 製品のレポジトリにコードをコミットする前に、オープンソース ソフトウェアを使用するための承認を求める
- 当該オープンソース ライブラリのコードのライセンスが企業のポリシーによって事前承認されていない場合、プロプライエタリ コードをオープンソースのライブラリにリンクする、もしくはその逆のことをする前に、承認を求める
- 実施された変更について、変更日時、変更者、1行程度の内容説明といった変更ログ(Changelog)をすべてのファイルについてアップデートする
- 開発するすべてのコードとオープンソース ソフトウェアの間のインターフェイスを文書化する。これにより他者が(ソフトウェア間の)相互作用を理解し、コンプライアンス上の懸念事項を明確にすることができる
- ソースコード パッケージのライセンスが記載されているWebページをPDFで保管する。これによりパッケージをダウンロードした際のプロジェクトの状態を文書として保存できる
- パッケージを変更していない状態のコピーを、ライセンス情報と一緒にバックアップしておく

- オープンソース ソフトウェアのコンポーネントをアップグレードする際に、ライセンスが同じものかどうかを確認する。ライセンスはバージョン間で変わることがある
- ソースコード パッケージに記述されたライセンスがプロジェクトWebサイトで記述されているものと合っているかどうかを確認する。差異がある場合、プロジェクトにコンタクトし明確にする

11.2 間違いを回避する

- もともとあったライセンス情報・著作権情報の削除や修正をしないこと。そのような情報はすべて原形が保たれている必要がある
- オープンソース コンポーネントの名称を変更しないこと
- 事前承認なく、オープンソース コードをプロプライエタリ ソースコードやサードパーティ ソースコードにコピー/ペーストしないこと(その逆もしかり)
- 事前承認なく、オープンソース コードやサードパーティ ソースコードを自社製品のソース ツリーにコミットしないこと
- 事前承認なく、異なったライセンスのもとで外部から入手したコードをマージしたり、混合させたりしないこと
- 社外の人々と、コンプライアンスのプラクティスについて議論しないこと

12 結論

オープンソースのデュー デリジェンスは一般的に、M&A取引において成功裏に完了させる必要のある長いタスク リストの中の1つにすぎません。しかし、そうはいってもソフトウェアの中核的な役割と潜在的な知財リスクを考えると、デュー デリジェンス全般で重要な側面を有しています。オープンソースのデュー デリジェンスは、長期的なプロセスに感じられるかもしれませんが、特に両社が準備をしていて、かつ対応の早いコンプライアンス サービス プロバイダーと一緒に取り組めば、通常は迅速に完了させることができます。

どうすれば準備しておくことができるのでしょうか？

買収対象企業は、開発プロセスとビジネス プロセスを以下のように確実に実行することで、適切なオープンソース コンプライアンスのプラクティスを維持することができます。

- 社内・社外のソフトウェアすべてについて、起源とライセンスを明確にする
- 開発プロセスの中で、オープンソース ソフトウェアのコンポーネント・スニペットを追跡する
- ビルドの中で新規採用・アップデートされるコードに対し、ソースコード レビューを実施する
- 製品が出荷される時、あるいはソフトウェアがアップデートされる時に、ライセンスの義務を履行する
- 従業員にオープンソース コンプライアンスのトレーニングを提供する

買収企業は、探すべきものを知り、迅速に問題に取り組むためのスキルを手近に持っておく必要があります。

- 買収対象企業とともに、用いるべき妥当な監査手法とその監査に関与するサードパーティの企業を決定する。ただし、ブラインド監査を実施できないところもあれば、DIY監査に対応していないところもあり、またスニペットまでは発見できないところもあるので留意が必要となる
- 可能であれば、監査を複数社からの入札とし、監査サービス プロバイダーについて調査する。費用面だけでなく、買収企業が持ち得る懸念の対処に役立つ確かなアウトプットがあるかどうかを見ること。各社の入札を公平に比較するための社内専門家を確保した上で、各社の入札内容が以下の監査パラメーターすべてを含んでいることを確認する
    - 監査手法、インプット、およびアウトプット
    - 問題が生じた際の議論迅速化のための買収企業と買収対象に対する一次窓口担当者
    - スケジュールと物流(実地訪問が絡む場合は特に)
    - 機密事項に関するパラメーター
    - コードの脆弱性とバージョン コントロールに関する分析
    - 費用、通常プロセスと簡易版プロセス

オープンソース コンプライアンスは継続して行われるプロセスです。良いオープンソース コンプライアンス プラクティスを維持していくことで、企業は、起こり得る買収、売却、あるいは製品・サービスのリリースなど、ソフトウェア取引が行われるあらゆるシナリオに備えられるようになります。したがって、企業はぜひオープンソース コンプライアンス プログラムの構築・改善に力を注いでください。

参考文献

Open Source Compliance in the Enterprise(企業におけるオープンソース コンプライアンス)

The Linux Foundationで公開されている「Open Source Compliance in the Enterprise(企業におけるオープンソース コンプライアンス)」は、実践ガイドとして企業が製品やサービスにオープンソースをうまく活用し、法的責任を果たす形でオープンソース コミュニティに参加するする方法について触れています。

Practical GPL Compliance (実践GPLコンプライアンス)

The Linux Foundationで公開されている「Practical GPL Compliance(実践GPLコンプライアンス)」は、スタートアップや小規模事業者、そして技術者向けのコンプライアンス ガイドで、特にGNU General Public License (GPL)の各バージョンを順守することに焦点を当てています。本書の目的は、共通に抱える課題に実践的な情報提供をすることにあります。

OpenChain Curriculum (OpenChain カリキュラム)

「OpenChain Curriculum(OpenChainカリキュラム)」は、組織がOpenChain Specification(OpenChain仕様書)で定義されたトレーニングやプロセスの要求事項に準拠することを支援するために設計されています。一般的なオープンソースのトレーニングでも使用可能ですし、パブリックドメインのライセンスであることから部分的にでもすべてでも制約なく組織内外で再利用することができます。

Compliance Basics for Developers (開発者向けコンプライアンスの基礎)

The Linux Foundationが提供する、開発者を対象とした無償のオープンソースコンプライアンスコースです。

Software Package Data Exchange (SPDX)

SPDXは、ソフトウェア パッケージのコンポーネント、ライセンスおよび著作権を伝達していくための一連の標準フォーマットです。

オープンソース コンプライアンス ソリューションを提供している商用プロバイダー一覧*

- Black Duck Software
- Flexera Software
- FOSSA
- FOSSID AB
- nexB
- Protecode (Synopsys)
- Rogue Wave Software
- WhiteSource Software

オープンソース コンプライアンスツール*

- FOSSologyはオープンソース ライセンス コンプライアンスのためのオープンソースのソフトウェアシステムとツールキットです
- Binary Analysis Toolは、コンプライアンスのための活動を支援するオープンソースのツールです。バイナリ コードを試験し、コンプライアンスの問題を検出します。

* リストに挙げたプロバイダーやツールについて記載漏れがあるかもしれませんが、ご容赦下さい。

謝辞

OpenChain プロジェクトへ。
2章で紹介した図を使用し、改良することを許可してくれたことに対して。

Brian Warner(Samsung、Open Source Strategy and Engineeringシニアマネージャー)へ。
編集、校正対応をしてくれたことに対して。

Phil Odence(Black Duck Software、VP&GM)へ。
M&A監査プロセスについての議論をしてくれたことに対して。

Jon Aldama(FOSSID AB、製品担当VP)へ。
ブラインド監査、DIY監査プロセスについての議論をしてくれたことに対して。

Jose L. Lopez (SamsungNEXT、シニア コーポレート カウンシル)、
David Marr (Qualcomm、法令担当VP)、
Nithya Ruff (Comcast、Open Source Practiceシニアダイレクター)、そして
Gil Yehuda (Oath、Open Source担当シニアダイレクター)へ。
レビューや貴重なフィードバックを通じて詳細情報が不足してことを気づかせてくれたこと、本書を改善してくれたことに対して。

Shane Coughlan(OpenChainプロジェクト、Program Director)へ。
彼のレビューと、OpenChainについての記述が正確であることを確認してくれたことに対して。

著者について

Ibrahim Haddad (Ph.D.) はSamsung Research America のR&D部門のVPでありOpen Source Group (OSG)の長でもあります。Samsungにおけるオープンソース戦略の監督、その実行、社内外の研究開発におけるコラボレーションやM&AやCVC(コーポレート ベンチャー キャピタル)活動の支援などを担当し、オープンソース関連団体においてSamsungを代表する立場を担っています。また、企業が製品・サービスでオープンソースを活用し、法的に責任のある形でオープンソースコミュニティに参加するための最善の方法についての実践ガイド「Open Source Compliance in the Enterprise(企業におけるオープンソース コンプライアンス)」の著者でもあります。

Web: http://www.ibrahimatlinux.com/

Twitter: @IbrahimAtLinux

THE LINUX FOUNDATION

The Linux Foundation promotes, protects and standardizes Linux by providing unified resources and services needed for open source to successfully compete with closed platforms.

To learn more about The Linux Foundation, please visit us at linuxfoundation.org.