イブラヒム本翻訳ドラフト：第1章

2017/04/25

江川尚志

イブラヒム本第１章の草稿です。

全体を通しての感想

* 全体に書きなぐった感があり、用語の不統一が目立つ。標準化会合ならば細かく添削するところですが、、、
* あまりにも頻出する単語、例えばOpen source、はOSと略すべきか？

1. **オープンソースコンプライアンス入門**

**変わりゆくビジネス環境**

従来、プラットフォームやソフトウェアスタックはプロプラエタリなソフトウェアを使って実装され、内部開発されたソフトや交渉の結果であるライセンス条件によるサードパーティのソフトから成る様々なソフトウェアのブロックから構成されていた。ビジネス環境は予測可能で、企業は潜在的なリスクをソフトウェアベンダーとのライセンス交渉や契約交渉を通じて軽減していた。全てのソフトウェアコンポーネントについて誰が提供者であるかを知るのは大変容易であった。図１は従来のハードウェア、ソフトウェアのプラットフォームについて主なブロックを示したものである。

図１：プロプラエタリなソフトウェアのブロックに依る従来のソフトウェアプラットフォームの単純化したアーキテクチャ

時と共に企業はOSSを自社のプラットフォームやソフトウェアスタックに組み込み、その恩恵に預かるようになってきた。その理由は製品ごとに様々であるが、様々な業界で共通するのは、オープンソースのコンポーネントには即座に使える卓越した特徴があったこと、分散的な開発による市場投入への時間短縮により経済的に有意の利益があったこと、そしてソースコードをカスタマイズするという新しく出来ることを提供したこと、である。その結果、複数のソースによる新たな開発モデルが登場した。

この新たなモデルでは、製品は下記の任意の組み合わせとなる。

* プロプラエタリなコード、その製品やサービスを作る企業が開発したもの
* プロプラエタリなコード、元々はその企業によりオープンソースのライセンス下でオープンソースのコンポーネントを統合したり適用したりすることで開発されたが、上流のオープンソースプロジェクトに寄付されず戻されなかったもの
* サードパーティの商用コード、サードパーティのソフトウェアプロバイダにより開発され、製品やサービスを作る企業が商用ライセンスの下で受領したもの
* オープンソースのコード、オープンソースのコミュニティにより開発され、製品やサービスを作る企業がオープンソースライセンスの下で受領したもの

図２（次ページ）に複数のソースによる開発モデル、および入ってくるソースコードの様々な組み合わせを示す。

この開発モデルでは、ソフトウェアコンポーネントは任意の数の出所から来た、様々なライセンス下でライセンスされたソースコードから構成され得る。たとえば、ソフトウェアコンポーネントAはサードパーティのプロプラエタリのコードに加えプロプラエタリなコードも含んでおり、ソフトウェアコンポーネントBはオープンソースプロジェクトからのソースコードに加えプロプラエタリなコードを含む、である。

図２：マルチソースの開発モデル

かつては明らかにプロプラエタリなソフトウェアスタックだったものの中にオープンソースのソフトウェアのコンポーネントが増えていくに従い、ビジネス環境は慣れ親しんだ領域、企業が好む環境から離れていく。

図３（次ページ）はあるプラットフォームやソフトウェアスタックにおいて、様々な階層でオープンソースソフトウェアを受けいれる様子を示したものである。

プロプラエタリな開発モデルと、マルチソースによる開発モデルとの大きな違いの一つは、オープンソースソフトウェアのライセンスは交渉するものではないことである。ソフトウェアの提供者（すなわちオープンソースの開発者やプロジェクト）と調印する契約はない。そうではなく、オープンソースプロジェクトを開始した人々が所定のライセンスを選ぶ。そしてプロジェクトがある規模に達すると、変更は事実上不可能となる。マルチソースの開発モデルを使うとは、数十の相異なるライセンス（とライセンスの組み合わせ）に基づく数百人、時には数千人のライセンス提供者やコントリビューター（著作権者）と関わること、を企業は理解しなければならない。その結果、かつては企業対企業のライセンスや合意に至る交渉を通じて管理されていたリスクは、強固なコンプライアンスプログラムと、注意深いエンジニアリングの実施によって管理されることになる。

**オープンソースコンプライアンス手始め**

オープンソースのイニシアチブやプロジェクトを使う事で、企業や組織はオープンソースソフトウェアの開発者を代表する数百、時に数千のコミュニティと協力し、イノベーションを加速できるようになる。しかしながら、オープンソースのコミュニティと組むには責任が伴う。オープンソースのライセンスに伴う義務を順守することを保証しなければならない。

オープンソースのコンプライアンスとは、オープンソースのユーザー、インテグレーター、開発者が著作権表示をきちんと見て、自らに関わるオープンソースのソフトウェアコンポーネントのライセンス上の義務を満たすプロセスである。適切に規定されたオープンソースのコンプライアンスのプロセスは、ライセンス条項の順守を保証すると同時に、企業自身、あるいはサードパーティのサプライヤの知的財産が意図せず公開されたり、その他不適当な結果を招いたりしないように設計されるべきである。

オープンソースのコンプライアンスは4つの主要な目的の達成に役立つ。

* ライセンス上の義務に従う
* オープンソースが商用製品で効果的に使われることを助ける
* サードパーティとの契約上の義務に従う

**オープンソースのコンプライアンスを保証する利点**

オープンソースのコンプライアンスを達成する事にはいくつかの利点がある。確固たるコンプライアンスプログラムを持つ企業は技術的にも利益を得ることが多い。規定に従っているソフトウェア群の資産はサービスし、試験し、アップグレードし、保守することが容易だからである。更に、コンプライアンスの活動は、様々な製品や部門で使われていてカギとなる、組織にとって高度に戦略的で利益をもたらすオープンソースソフトウェアの洗い出しにつながる。また逆に、レビューを繰り返すことを通じてコンプライアンスは、オープンソースのコンポーネントの利用に伴うコストやリスクを示すことにもなる。

健全なコンプライアンスのプログラムは外部コミュニティと協業する上でも大いに有用である。コンプライアンス上の問題が生じた時、そうしたプログラムは会社が善意であることを示すことができる。

最後に、頻度は下がるが強固なコンプライアンスのプログラムがもたらす利益として、例えば、会社の買収や売却、新製品や新サービスのリリースのためより良い準備となる、が挙げられる。オープンソースへのコンプライアンスの保証はそうした業務が完了する前に終了させることが必須だからである。更には、OEMや下流のベンダーとの取引においてコンプライアンスが検証できることは付加的な優位となる。

**コンプライアンスの失敗**

ソフトウェア開発の全工程において、コンプライアンスプログラム上のエラーや不十分さはコンプライアンス上の失敗に繋がることがある。下記はこうした失敗の例である。

* 不適切な帰属表示 (attribution notice)  
  帰属表示は通常、テキストファイルとしてオープンソースのコンポーネントと共に提供され、オープンソースコンポーネントへのコントリビューターが提供したことを示す。
* ライセンス表示提供を怠る  
  ライセンス表示とは、製品やスタックに含まれるオープンソースのライセンス文書を含んだファイルで、多くの場合、製品の文書と共に、あるいは製品やアプリケーションのUIと共に提供される。
* 著作権表示の省略  
  著作権表示は、ソフトウェアのコピーにつけられる識別情報で、著作権者を示す。
* 変更表示の付け忘れ  
  変更表示とは、ソースコードに対する変更をチェンジログ内で表示する、GPLやLGPLで必要になるものである。一例を下記に示す。  
  /\*  
  \* Date Author Comment  
  \* 10/15/2015 Ibrahim Haddad Fixed memory leak in nextlst()  
  \*/
* 不十分な、あるいは誤解を招く主張  
  製品付属の文書や製品の広告において。
* ソースコード提供の失敗  
  ソースコードを利用可能とすること（変更を含む）はGPL/LGPLライセンスファミリーの要求の一つである。
* GPL/LGPL等のライセンスのソースコードを使う際に書面で告知しない  
  書面での告知は、その製品のエンドユーザーに対してその製品に含まれるオープンソースソフトウェアの情報、そして頒布されるべきソースコードをダウンロードする方法を開示する。書面告知の基本的な例は下記のようになる。  
  Foobarの本製品の中で使われているソフトウェアに関連する、FooBar社が公開しているソースコードのコピーを得るには、<http://opensource.foobar.com> にアクセスするか、[opensource@foobar.com](mailto:opensource@foobar.com) にメールでリクエストを送るか、またはリクエストを通常の郵便で  
  Foobar株式会社  
  オープンソースプログラム室  
  住所、郵便番号  
  国名
* ビルド用スクリプトを提供しない  
  コンパイルに必要なもの（GPLやLGPLファミリのライセンス毎に）。

**知的財産権上の失敗**

表1 (次ページ) はソフトウェア開発の過程においてプロプラエタリな知財とオープンソースの知財を誤って混ぜてしまい、ライセンスコンプライアンス問題となるありがちな例を示したものである。もっともありがちなのは、異なる、または両立しないライセンス（例：プロプラエタリ、サードパーティやオープンソース）下のソースコードを混ぜてしまう、である。そうして混ぜてしまうと企業はプロプラエタリなソースコードをオープンソースライセンスで開示することを強制され、（おそらくは）高い価値を持つ知財の制御を失い、市場における差別化が弱くなってしまう。

知財上の失敗は下記を招く。

* 差し止め命令によりコンプライアンス上の懸念が解決されるまで製品が出荷できない
* 問題のバイナリコードに関わるプロプラエタリなソースコードを（ケースに応じた）オープンソースライセンスで配布するよう要求される
* コンプライアンス上の懸念を解消するため多大な工数が費やされる
* 顧客、ディストリビュータ、サードパーティのソフトウェア供給者、そしてオープンソースコミュニティを困惑させる

表１：知財上の失敗の例

|  |  |  |
| --- | --- | --- |
| 問題の類型 | 発見する方法 | 回避方法 |
| オープンソースのコードをサードパーティのコードに挿入 開発プロセスの中で開発者がオープンソースのコード (いわゆるスニペット) をプロプラエタリやサードパーティのコードに挿入して発生 | オープンソースのコードとの一致の可能性を求めてソースコードをスキャン | コンプライアンスやオープンソースライセンス、プロプラエタリやサードパーティのコードにオープンソースのコードを含めることの意味について認識するためのトレーニングを提供  予期されないライセンスやコードのスニペットのため、定期的にプロジェクトの全ソースコードをスキャン  オープンソースソフトウェアを使う際には製品のレポジトリにコミットする前に承認を必要とするようにする |
| オープンソースのソフトウェアをプロプラエタリなソースコードのソフトウェアにリンク (またはその逆。C/C++特有の問題)  衝突(conflict)する、または互換性のない(incompatible) ライセンスのソフトウェアコンポーネントをリンクした結果発生 | 異なるソフトウェアコンポーネント間のリンクを検出できる依存性追跡ツールを利用; 企業のオープンソースポリシーで許容されているかを識別 | 企業のコンプライアンスポリシーに基づくリンクのシナリオについての訓練を提供 定期的に依存性追跡ツールを動作させ、全てのリンク関係を検証; 企業のポリシーで許容されていないあらゆる事柄にフラグを立てる |
| オープンソースのコンポーネントにプロプラエタリなコードを含める 開発者がプロプラエタリなソースコードをオープンソースソフトウェアにコピー/ペーストして発生 | ソースコードをスキャンする。ツールでオープンソースのコンポーネント由来ではないソースコードは識別できるので、監査のため各種のフラグをつける | スタッフを訓練 ソースコードを定期的に検査 プロプラエタリなソースコードをオープンソースのコンポーネントに含めるには承認を必要とするようにする |

**ライセンスコンプライアンス問題**

ライセンスコンプライアンス問題は知財上の問題に比べ一般に損害は小さい。自社のプロプラエタリなコードをオープンソースライセンスで開示するといった副作用がないからである。

ライセンスコンプライアンス上の失敗は下記のいずれか（か、その組み合わせ）を招き得る。

* ソースコードが開示されるまで製品出荷を差し止める命令
* バージョンのミスマッチによりカスタマーサポート・カスタマーサービス上の問題となる（サポートホットラインに電話やemailでソースコード開示について問い合わせがくる結果）
* 顧客やオープンソースコミュニティにおいて困惑や悪い評判を招く

表２にソフトウェア開発プロセスでの最も一般的なライセンスコンプライアンス問題の例、および回避方法を示す。

|  |  |
| --- | --- |
| 問題の類型 | 回避方法 |
| ライセンス上の義務の一環としてソースコードのパッケージを出版 (publish)したり利用可能としたりし損ねる | 詳細なコンプライアンスチェックリストに従い、製品やアプリケーション、ソフトウェアスタックの出荷時には全てのコンプライアンス上のアクションアイテムが完了したことを確認 |
| 出荷されたバイナリと異なるバージョンのソースコードを提供 | 検証作業をコンプライアンスプロセスに追加し、提供するソースコードのバージョンが、出荷されるバイナリのバージョンと確かに正確に対応することを確認 |
| 出荷製品に組み込まれたオープンソースソフトウェアへの変更をリリース忘れ | 部品表 (bill of material, BOM) 差分検出ツールを使いリリース毎のソフトウェアコンポーネント変更を特定 コンプライアンスプロセスにおいて、より新しいバージョンのソフトウェアコンポーネントを再導入 「diffの計算」を変更された(オープンソースとするにふさわしい) ソースコードのチェックリストに加え、製品に使われたオープンソースのリリース前に行う |
| 変更されたオープンソースのコードにマークを付け忘れる、変更の記述を入れ忘れる | ソースコードへのマーク付けをソースコードのリリース前のチェックリストに加え、ダウンロードしたオリジナルのコピーとの差分すべてに確実にマーク ソースコードのリリース前に検査 変更されたソースコードが適切にマークされていることを、コンプライアンスプロセス上のマイルストーンとする ソースコードの変更履歴をアップデートすることを、開発プロセスの一部とするようスタッフを教育 |

**コンプライアンスプロセス上の失敗**

コンプライアンスプロセス上の失敗は、オープンソースのライセンス条件の侵害、例えばライセンス上の義務を満たさない、に繋がる。表３にソフトウェア開発で起きる最も一般的なコンプライアンスプロセス上の失敗のリスト、および回避方法を示す。

|  |  |
| --- | --- |
| 失敗 | 回避方法 |
| 開発者が企業内のオープンソース委員会 (時にオープンソース評価委員会と呼ばれる) オープンソースソフトウェアの利用申請をしない、期限までに行わない | コンプライアンスポリシーとプロセスについての教育  定期的にソフトウェアプラットフォームの全スキャンを行い、承認された型に該当しないオープンソースを検出。もしビルドされたシステムに該当するコンプライアンスチケットがないオープンソースコンポーネントが検出されたら、自動で新チケットを発行（企業がBugzillaのようなツールに埋め込まれているワークフローを使ってソフトウェアコンポーネントを追跡していると仮定）  業績評価にコンプライアンスを含める。例えばコンプライアンスポリシーに従わなかったらボーナス査定に直結 開発者がオープンソースの利用申請を早期に、オープンソースのコードの利用を決める前でも行うよう命令 |
| オープンソースの訓練を受けない | オープンソースの訓練は従業員のキャリア開発計画の一部であり、業績評価プロセスの一部としてモニタされていることを念押し |
| ソースコードの監査を行わない | コンプライアンスのスタッフを適切に訓練  ソースコードを定期的にスキャン  繰り返される開発プロセスのマイルストーンに監査を確かに入れ込む  適切なレベルのスタッフを揃え、監査がスケジュール遅れとならないようにする |
| 監査で発見された問題を解決しない | 監査レポートが完結しない限りコンプライアンスチケットをクローズさせない。チケットのクローズは、関連するサブのタスクが存在しない時のみとする |

**得た教訓**

この数年で明るみに出たコンプライアンス違反が数件あった。そうしたコンプライアンス違反の法的な決着を通じて、オープンソースのプロフェッショナル達は次のような教訓を学びつつある。

**製品出荷やサービス立ち上げに先立ちコンプライアンスを確実に行う**

コンプライアンス違反事件の最も重要な教訓は、関係した企業は最終的には問題のライセンス条項に従わなければならなかった、であり、この事実を踏まえた問題解決のコストは基本的なコンプライアンスのコストを圧倒的に上回っていた、である。従って、製品出荷やサービス立ち上げに先立ってコンプライアンスを確実に行うのが賢いやり方である。

コンプライアンスは法務部門の業務に留まるものではない、と認識することは重要である。全部門が関わって適切なコンプライアンス、正しいオープンソースの利用、必要に応じた再配布、を確実に行う必要がある。この関わりには首尾一貫したコンプライアンスポリシーや手続きの確立と維持管理、利用中（プロプラエタリ、サードパーティ、オープンソース）の全ソフトウェアコンポーネントのライセンスが共存し得ることを製品出荷やサービス立ち上げ前に確認すること、が含まれる。そのためには企業はオープンソースを管理するインフラを末端まで構築し下記を行う必要がある。

* 製品の中で、サービスが開示している、あるいは内部で利用されている全てのオープンソースを識別
* アーキテクチャを評価し、オープンソースのライセンス上の義務がプロプラエタリやサードパーティのソフトウェアコンポーネントまで拡大していないか、拡大しているならばどのように、を検証
* 適用可能なオープンソースライセンスを収集し、法務部門が評価
* オープンソースの利用および配布のポリシーと手続きを定める
* アーキテクチャ設計と製造の実務においてリスクを低減

**コンプライアンス違反は高くつく**

コンプライアンス違反が公となった事件の多くはGPLのソースコードが関わっている。これらの紛争解決の合意では下記の一つないし一つ以上の条項が含まれている。

* コンプライアンスを満たすよう必要なアクションを取る
* コンプライアンスオフィサーを置きコンプライアンスをモニタし確認させる
* 製品を受け取った過去の顧客に、製品はオープンソースソフトウェアを含んでいること、そのソフトウェアについての権利、を通知
* ライセンス告知を企業のウェブサイトに掲載
* 製品説明に告知を追加
* ソースコードをすべての改変と共に利用可能に（GPL/LGPLファミリのライセンスに限る）
* 問題となっているオープンソースソフトウェアのバイナリの配布を、関連のソースコードが開示されるまで、あるいはコンプライアンス違反で影響を受けた特定顧客が利用可能となるまで差し止め
* いくつかの事件では、原告に対し非開示の額の金銭的対価

更に、コンプライアンス異議申し立てを受け、それが成功すると下記のコストを負うこととなる。

* コンプライアンスの照会への対応として申し立てに応じた調査、問題のソースコードのデューデリジェンス、のためのディスカバリとデューデリジェンスのコスト
* 外部・内部のリーガルコスト
* ブランドや評判、信頼性へのダメージ

ほぼすべての事件において、オープンソースライセンス上の義務違反は面目の失墜、否定的な記事、オープンソースコミュニティとの関係悪化を招いている。

**関係は大事**

オープンソースソフトウェアを自社製品で利用する企業は、利用しているコードを開発し維持七得るオープンソースコミュニティと良い関係を持ち、維持することが望ましい。オープンソースプロジェクトのコミュニティは企業が自社製品に含まれるオープンソースソフトウェアのライセンスを尊重すると期待している。その方向に動き、オープンかつ正直な関係を気づくことは大いに価値がある。

**訓練は重要**

訓練はコンプライアンスプログラムにおいて必須の構成要素であり、オープンソースソフトウェアの利用を律するポリシーを従業員がよく理解する礎となる。ソフトウェアに関わる全職員が企業のポリシーとプロセスを理解する必要がある。企業はそうした教育を公式・非公式の教育で提供することが多い。

1. **オープンソース管理プログラムの確立**

オープンソース管理プログラムはオープンソースソフトウェアの全ての側面、その選択、承認、利用、配布、監査、インベントリ、訓練、コミュニティとの関わり、広報等々の体系を定める。本賞はオープンソース管理プログラムの様々な構成要素を概観し、新しいコンプライアンスプログラム策定における難しさを概説し、そうした難しさを克服する方法を述べる。

**オープンソースコンプライアンスプログラム**

最初に、成功するオープンソースコンプライアンスプログラムでコアとなる構成要素を概観する。本章と図4にこれら構成要素の概観を示す。

図４　オープンソース管理プログラムの必須構成要素

**コンプライアンス戦略**

オープンソースコンプライアンス戦略は、ポリシーやプロセスの実装の主たる側面について、ビジネスに立脚してコンセンサスを推し進めるものである。こうしたハイレベルなコンセンサスに基づかずにポリシーやプロセスの実装の細部について社内合意しようとすると、不可能とは言わないまでも非常な困難に直面する。コンプライアンス戦略はコンプライアンスを確保するために行うべきことを定め、職員のオープンソースソフトウェアの扱いを統べる原則を与える。オープンソースの承認、取得 (acquisition), 利用、オープンソースライセンスを含むかオープンソースラインセンスに基づいてライセンスされたソフトウェアをリリースする方法、の正式なプロセスも本戦略に含まれる。

**照会応答戦略**

照会応答戦略は、コンプライアンスが問われた時に行うべきことを定める。企業は時として悪評（場合によっては正式な申し立て）を、コンプライアンスについての追加情報提供の要請を無視したり、コンプライアンスの照会を扱う方法を知らなかったり、オープンソースコンプライアンスプログラムが欠落していたり不十分だったり、単に照会者と協力しなかったために、受けてきた。こうしたアプローチは関係者のだれにとっても実りや利益を生まない。従って企業は照会を受けとり、受領したことを応答し、照会者に検討することを伝え、現実的なフォローアップの日程を通知する方法を定めるべきである。後ろの章で、オープンソースコンプライアンスに対する照会を扱うシンプルなプロセスについて述べる。

**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とは、オープンソースソフトウェアの管理（利用と寄付の双方）を取り仕切るルール群である。プロセスとは、日々の業務にこれらルール群を実装するかを詳細に列挙したものである。コンプライアンスポリシーとプロセスがオープンソースソフトウェアの様々な側面、利用、寄付、監査、配布を取り仕切る。図５（次ページ）はコンプライアンスプロセスの例を図示したもので、個々のソフトウェアコンポーネントがデューデリジェンスの一環として通るべき様々なステップを示している。このプロセスの詳細は後ろの章で述べる。

図５：コンプライアンスデューデリジェンスプロセスの例

**コンプライアンスチーム**

オープンソースコンプライアンスチームとは、オープンソースコンプライアンスを確実に行うため様々な分野の人々を集めたチームである。コアとなるのはオープンソース評価委員会 (Open Source Review Board, OSRB) と呼ばれることが多い委員会で、エンジニアリング、製品のチームの代表者たち、一人以上の法務顧問、そしてコンプライアンスオフィサーから構成される。拡大チームは様々な部門からコンプライアンス活動のため随時加わる人々で、文書作成、サプライチェーン、全社開発、IT、ローカル化、オープンソース幹部会議 (Open Source Executive Committee, OSEC)らが含まれる。ただし拡大チームのメンバはコアチームと違い、OSRBから仕事を委任された時だけ活動する。コンプライアンスに常時携わるわけではない。第３章ではオープンソースコンプライアンスを達成するための個々の人間の役割と責任について詳述する。

図６（次ページ）はコンプライアンス達成のためのこれら二つ、コアチームと拡大チーム、を図示している。

図６：オープンソースコンプライアンスを確かなものとする個人とチーム

**ツール**

オープンソースコンプライアンスチームはツールを使うことでソースコードの監査、オープンソースコードとそのライセンスの発見、が自動化され楽になる。こうしたツールには

* コンプライアンスプロジェクト管理ツール、コンプライアンスプロジェクトを管理し、タスクやリソースを追跡
* ソフトウェアインベントリツール、個々のソフトウェアコンポーネント、バージョン、それを使用している製品、関連情報を継続的に追跡
* ソースコードとライセンスの識別ツール、ビルドするシステムに含まれるソースコードの由来とライセンスを特定
* リンク解析ツール、問題となるC/C++ソフトウェアコンポーネントと、製品に含まれる他のソフトウェアコンポーネントとの関係を特定。本ツールを使うことで、企業のポリシーを満たさないソースコードパッケージとのリンクを発見できる。この目標は、プロプラエタリやサードパーティのソフトウェアコンポーネントにオープンソース上の義務が何ら及んでいないことの確定である。もしリンクが発見されたら、製造部門に対しバグチケットを発行し、問題と解決方法を記述する。
* ソースコードのピアレビュー（査読）ツール、ライセンス上の義務を満たすための開示に先立ちオリジナルソースコードに対する変更をレビュー。
* 部品表 (bill of material, BOM) 差分ツール、二つの異なるビルドでの部品表の違いを特定。本ツールはコンプライアンス準拠作業を積み上げで行う場合に非常に有用である。

**ウェブ上のプレゼンス**

企業はポータルを２方向で使う。一つは内向きに、企業内へ。もう一つは外向きに、社会とオープンソースコミュニティへの窓として。内部向けのポータルはコンプライアンスポリシー、ガイドライン、各種文書、訓練、通知、メーリングリストへのアクセスを提供する。外部向けのポータルは社会とオープンソースコミュニティに向けて公開のプラットフォームとなり、またオープンソースのパッケージのソースコードや受領通知ほかの情報開示を投稿する場としてライセンス上の義務を履行する。

**教育**

教育はコンプライアンスプログラムにおいて必須の構成要素で、オープンソースソフトウェアの利用を取り仕切るポリシーを従業員がよく理解する基礎となる。オープンソースとコンプライアンスの教育の目標は、公式非公式を問わず、オープンソースのポリシーや戦略、オープンソースのライセンスや、製品やソフトウェアポートフォリオにオープンソースソフトウェアに組み入れることのビジネス的、法的リスクについて共通理解を作り上げ、注意喚起することである。またトレーニングは、企業内でコンプライアンスポリシーを広め推進し実施し、コンプライアンスの文化を育てる場となる。

公式のトレーニング

企業の規模やオープンソースの営利活動への浸透度合いに応じて、企業はオープンソースに携わる従業員に公式の講師によるトレーニングコースを命じ得る。実例でのトレーニングがその頂点となり得る。

非公式のトレーニング

非公式のトレーニングは下記のいずれか、あるいは全部を含む。

* ブラウンバッグセミナー（茶色鞄のセミナー）  
  ブラウンバッグセミナーとは一般に、昼食時に従業員や招待者が行うプレゼンテーションである。こうしたセミナーの目標は、商用の製品や企業のソフトウェアポートフォリオの中にオープンソースを組み込むことの様々な側面について講演し、議論を起こすことである。こうしたセッションには企業のコンプライアンスプログラムやポリシー、プロセスを含めても良い。
* 新人研修  
  時にはコンプライアンスオフィサーが企業のコンプライアンスの努力、規則、ポリシー、プロセスを新人に対して研修の一部として講演し、必要となるオープンソースの管理上の情報（聞くべき人、訪れるべき内部のウェブサイト、オープンソースとコンプライアンスのトレーニングへの参加方法等々）を提供する。

**自動化**

オープンソースソフトウェアを使いたい、寄付したい開発者はオンラインで申請し、適切な承認を得る必要がある。これは自動化されたオンラインシステム、通常はオープンソースコンプライアンスの管理を特に組み込んだワークフローに従うバグ追跡システムで行うのがベストである。

**メッセージの発信**

メッセージの発信は内部向け、外部向けともコンプライアンスプログラムの必須の構成要素である。ここで最も重要なのは明確で首尾一貫していることで、内部的に従業員に対してオープンソース関連の会社の目標や懸念事項を伝える場合も、外部に向け自社製品やソフトウェアスタックが利用しているオープンソースプロジェクトの開発者コミュニティに伝える場合も同様である。

**コンプライアンスの難しさと解決策**

企業でのコンプライアンスプログラムの確立では、ほぼ確実に困難に直面する。以下の説では最もありふれた困難を取り上げ、克服する方法を示唆する。

コンプライアンスプログラムの作成

最初の難しさは、コンプライアンスプログラムやそれを支える（既存の）内部手続きのためのインフラと、製品出荷やサービス開始の〆切とのバランスを取ることである。こうした難しさを和らげ解決し、開発活動にとって重荷とは見做されない能率的なプログラム確立を助ける様々な手段がある。

解決策

役員のサポート

役員レベルでのオープンソース管理プログラムへのコミットは、その成功と継続のために重要である。

ポリシーとプロセスの負荷の軽さ

プロセスとポリシーは重要である。だが開発プロセスにとって重荷に過ぎると開発チームが見做さないよう軽く効果的でなければならない。

能率的なオープンソース管理は二つの基盤の上に成り立つ。簡単で明確なコンプライアンスポリシーと、軽いコンプライアンスプロセスである。

基本ルールを強制

コンプライアンスプログラムを実施する一環として、誰でも従わねばならない幾つかのシンプルな規則を定める必要がある。

* 開発者がオープンソースソフトウェアを製品やソフトウェアスタックに組み入れを計画するときは申請する
* サードパーティのソフトウェアサプライヤに、提供物に含まれるオープンソースソフトウェアの情報を開示させる。あなたのサプライヤがオープンソースコンプライアンスを見事に実践しているとは限らない。オープンソースの開示に関わる文言を入れるよう、契約を見直すべきである。
* アーキテクチャのレビューとコードの検査を指示し、オープンソース評価委員会 (OSRB) がソフトウェアコンポーネント間の関係を理解し、オープンソースからプロプラエタリなソフトウェアへとライセンス上の義務が派生していることを見つけられるようにする。大規模に行うには適切なツールが必要となる。
* サードパーティのソフトウェアプロバイダから受領するすべてのソフトウェアをスキャンし、オープンソースに関する開示が正しく、完全であることを確認する

コンプライアンスを開発プロセスに統合

コンプライアンス確立の最善の方法は、コンプライアンスプロセスやポリシー、チェックポイント、活動を既存のソフトウェア開発プロセスに組み込むことである。

長期的な目標と短期的な実行

図4はコンプライアンスプログラムの成功に必須の構成要素を示している。プログラムのすべてを実装するのに必要な作業の多さに圧倒される関係者もいるだろう。実際にはそこまで難しいものではない。すべての要素を同時に実装しなければならない訳ではないからである。すべての組織において優先されるのは、製品やサービスを期限通りに出荷し、と同時に内部的なオープンソースコンプライアンスのインフラストラクチャーを構築し拡大することである。従って、企業なりのコンプライアンスのインフラストラクチャーを状況の進展に合わせて構築すべきと考えればよく、その際には将来の活動や製品をふまえ拡張性を考えるべきである。カギとなるのは思慮深く現実的な計画である。

解決策

* 長期的な目標に合致する、十全なコンプライアンスのインフラストラクチャーを計画し、短期的な実行の必要性に合わせ、要素を一つずつ実装する。たとえば、もしオープンソースを含む製品開発やサービス提供を始めたばかりであり、すでに動いているコンプライアンスのインフラストラクチャーが何もないならば、最も差し迫った懸念はコンプライアンスチームを作ることであり、プロセスとポリシー、ツールと自動化を確立することであり、従業員を訓練することである。これらの活動を（この順番で）開始し、システム構築を（コンプライアンスの観点から）適切に制御しつつ開始したならば、プログラムの他の要素に進むことができる。
* ポリシーとプロセスを確立
* 開発プロセスの一部としてコンプライアンスを組み込み

**コンプライアンスについての対話**

コンプライアンス活動の成功を確かなものとするには、対話は必須である。2種類の対話、あなたの組織内での内部的なもの、あなたの製品で使われているオープンソースのプロジェクトの開発者コミュニティとの外部に向けたもの、を考えることが重要である。

内部的な対話

企業はコンプライアンスについての対話が内部的に必要となる。オープンソースを商業ソフトウェアのポートフォリオへの組み入れが何をもたらすのか従業員が理解していることを確実にするためであり、企業のコンプライアンスポリシー、プロセス、ガイドラインについて教育を受けていることを確かなものとするためである。内部的な対話は下記のいずれでもよい。

* オープンソースコンプライアンス活動を幹部がサポートする電子メールでの対話
* オープンソースソフトウェア関連業務に携わる全従業員に対する正式な訓練命令
* オープンソースとコンプライアンスについてのブラウンバッグセミナーによりコンプライアンスについて更に注意喚起し、活発な議論を奨励
* 内部的なポータルを作り、企業のコンプライアンスポリシーやプロセス、オープンソースに関係した出版物やプレゼンテーション、メーリングリスト、オープンソースとコンプライアンスに関係する討論フォーラムをホスト
* 企業全体に向けたオープンソースのニュースレター。一般に隔月か四半期ごとに、オープンソースのコンプライアンスについて注意喚起

外部との対話

企業は外部とのコンプライアンスについての対話が必要となる。自社が製品で使うオープンソースソフトウェアについてライセンス上の義務を果たす努力をしていることを、オープンソースのコミュニティに確かに認識させるためである。外部との対話は下記のいずれでもよい。

* コンプライアンス遵守のためにオープンソースを配布するためのウェブサイト
* オープンソースの組織への参加やサポート。こうした活動は企業がオープンソースの組織との関係を確立し、そうした組織の役割を理解し、有意義な貢献をするのを助ける。
* オープンソースのイベントや会議への参加。参加には様々なレベルがある。イベントのスポンサーから講演や出版物への寄与、あるいは単に開発者を参加させオープンソースの開発者たちと知り合いオープンソースコミュニティのメンバと新たな関係を育成する。

**クリーンなソフトウェア起点の確立**

コンプライアンスプログラムの開始当初の困難の一つは、使われているオープンソースソフトウェアとそのライセンスを正確に把握することである。この初期の監査プロセスは、製品やソフトウェアポートフォリオのクリーンなソフトウェア起点の確立、と呼ばれることが多い。これは数か月かかることもある重い活動で、開発と並行してどの程度早期に作業を開始するかに掛かっている。

解決策

組織は初期のコンプライアンスを以下の活動を通じて達成する。

* オープンソースの利用申請の早めの提出と評価
* 自動化されたソースコードのスキャンを継続的に、事前設定した時間間隔で全ソースコードに対して実行
* ソースコードのベースを継続的にスキャン。これにはサードパーティのソフトウェアプロバイダから受領したものを含める。該当するコンプライアンスチケットなしにコードベースにチェックインしたソースコードを捕まえるためである。こうしたソースコードのスキャンはたとえば毎月行う。
* 設計とアーキテクチャの評価をソースコードの検査に加えて強制する。これはオープンソース、プロプラエタリ、そしてサードパーティのソフトウェアコンポーネントのコードの関係を解析するためである。これを強制するのは、そうした関係がライセンス上のコンプライアンスの義務に関わるときのみでよい。

もし企業がコンプライアンスの起点構築に失敗すると、その製品の将来の改訂版（またはおなじ起点を使った異なる製品）がコンプライアンス問題で苦しむことはほぼ確定となる。

そうしたシナリオを防ぐため、企業は下記を考慮すべきである。

* シンプルだが確実に行われるポリシーと軽量なプロセスを提供
* コンプライアンス上のチェックポイントをソフトウェアの開発プロセスに含め、コンセプトが出荷される製品やソフトウェアスタックとなる時に行う。理想的にはすべての開発上のマイルストーンにおいて対応するコンプライアンス上のマイルストーンを組み込み、ビルドに使われる全ソフトウェアコンポーネントが対応する、承認されたコンプライアンスチケットを確実に持つようにする。
* 専任のコンプライアンスチームを確保する。この点は後ろの章で文章を割く。
* コンプライアンスチケットを効率的に処理するためツールや自動化を利用する。この点は後ろの章で論じる。

**コンプライアンスの維持**

オープンソースのコンプライアンスを維持するには、コンプライアンスの起点確立と同様、いくつかの困難がある。実際には踏むべきステップの多くは同じで、ただし規模は小さく、積み増しとなる。コンプライアンスの維持は継続的な努力であり、規律、そして既存の製造およびビジネスのプロセスへのコンプライアンス活動組み込みのコミットメントによる。

図7に積み上げのコンプライアンスの概念を示す。ここでは初期のコンプライアンスの起点と現在のバージョンとの間で起きたソースコードの変更が何であれ、コンプライアンスを確実に満たすことが必要となる。

図7：積み上げのコンプライアンスの例

解決策

企業は以下の活動を通じてコンプライアンスを維持し得る。

* 企業のコンプライアンスポリシーとプロセス、加えてあらゆるガイドラインの厳守
* コードベースに統合されたすべてのソースコードについて、出所を問わず継続的に監査
* コンプライアンスや自動化に使われるツールを継続的に改善し、コンプライアンスプログラムにおいて可能な限り多くのプロセスを高い効率で実行

**内面化と末永い実行**

オープンソースのコンプライアンス活動を維持するのは組織が成長し、オープンソースを使った更なる製品やサービス出荷に伴う進行中の挑戦である。企業は開発のカルチャーの中へとコンプライアンスを内面化し、末永く実行するためにいくつかの段階を踏むとよい。

解決策

後援者

役員レベルのコミットメントがコンプライアンス活動を持続させる上で必須である。進行中のコンプライアンスのリーダーとなり、オープンソースの管理機能を企業としてサポートする役員がいなければならない。

一貫性

企業全体で一貫性を保つことは複数のビジネスユニットや子会社を持つ大企業では重要である。部門間での一貫したアプローチは、記録の保持、グループ間でのコードの共有促進と並び有用である。

計測と分析

コンプライアンス活動やプロセス、手続きのインパクトや効率性を計測し分析すべきである。パフォーマンスを調べ、コンプライアンスプログラムを改善するためである。計測の尺度は、コンプライアンスプログラムを推進する場合にプログラムの個々の構成要素から得られる生産性向上について対話をする助けとなる。

コンプライアンスプロセスの改善

組織がオープンソースを利用する範囲や目的は変化する。製品や技術、企業の吸収や合併、海外への委託、その他多くの要因によって動く。従って、今ぷらインすポリシーとプロセスを絶えず評価し、改善する必要がある。

更に、オープンソースラインセンスの解釈や法的リスクも進化する。こうしたダイナミックな環境の中で、コンプライアンスプログラムも進化する必要がある。

守らせること

コンプライアンスプログラムは、守られなければ無意味である。効果を持たせるため、コンプライアンスプログラムにはプログラム厳守を監視し、ポリシーやプロセスやガイドラインを全社的に守らせるメカニズムを含めるべきである。コンプライアンスプログラムを守らせる方法の一つは、ソフトウェア開発プロセスと統合し、従業員の業績査定の幾分かをコンプライアンスプログラム活動へのコミットメントと実行とすることである。

スタッフ割り当て

スタッフがコンプライアンスの職務のために割り当てられ、十分なコンプライアンスの訓練が組織の全従業員に確実に施されていなければならない。大きな組織ではコンプライアンスオフィサー、および関連する職務はフルタイムの仕事となり得る。小さい組織では分担したり、時間の一部を割くべき活動となる可能性が高い。

1. **コンプライアンス達成に向けて：役割と責任**

一人の個人では、どれほどの達人であろうとも、オープンソースコンプライアンスを組織全体に実装することはできない。図8はオープンソースコンプライアンスの達成に責任がある様々な部門を書き下したものである。コンプライアンス達成には二つのチームが必要となる。コアチームと拡大チームである。後者は通常、前者を包括するものとなる。

図8: オープンソースコンプライアンスを確かなものとする個人とチーム

コアチームはオープンソース評価委員会 (Open Source Review Board, OSRB) と呼ばれることも多い。エンジニアリングと製品のチームの代表者たち、一人以上の法務顧問、そしてコンプライアンスオフィサーから構成される。表４に個々の参加者のコアチームにおける役割と責任を示す。

拡大チームは表5 (49ページ) に示す通り、様々な部門からコンプライアンス活動のため随時加わる人々で、文書作成、サプライチェーン、全社開発、IT、ローカル化、オープンソース幹部会議 (Open Source Executive Committee, OSEC)らが含まれる。ただし拡大チームのメンバは（実体をもつ組織である）コアチームと違い、OSRBから仕事を委任された時だけ活動する。

表4. コンプライアンスコアチーム (OSRB) の主な役割と責任

|  |  |
| --- | --- |
| 参加者 | 主な役割と責任 |
| 法務の代表者 代表者はその時の作業に応じ、法律顧問から弁護士補助職員まで変化する | OSRBとOSECへ参加  オープンソースソフトウェアの利用、改変、配布をレビューし承認  ライセンスについてのガイドラインの提供  トレーニングの提供と承認  オープンソースコンプライアンスプログラム改善への貢献  オープンソースのポータルの内容をレビューし承認  ライセンス上満たすべき義務一覧のレビューと承認  オープンソースに関する告知の承認 |
| エンジニアリングおよび製品チームの代表 企業によってはエンジニアリングと製品のチームは分かれていない | OSRBとOSECへ参加  コンプライアンスポリシーとプロセスの遵守  コンプライアンスの実践を開発プロセスの中に統合  コンプライアンスプログラムの改善に貢献  技術的なコンプライアンスプログラムの遵守  あらゆる質問への迅速な回答  設計、アーキテクチャ、コードのレビューの実施  配布のためのソフトウェアパッケージ準備 |
| コンプライアンスオフィサー  オープンソースコンプライアンスオフィサーは必ずしも専任である必要はない。大抵の場合、オープンソース部門の部長や所長が役割を果たす | 全てのコンプライアンス活動を推進  ソースコードのスキャンと監査を調整  ソースコードパッケージの配布を調整  OSRBとOSECへ参加  コンプライアンスとオープンソースのトレーニングに寄与  コンプライアンスプログラムの改善に寄与  OSECにコンプライアンス活動について報告  自動化や、開発環境の中にオープンソースのコードを見つける新たなツール作成に寄与 |

表5. コンプライアンス拡大チームの役割と責任

|  |  |
| --- | --- |
| 参加者 | 主な役割と責任 |
| オープンソースの戦略を決定するオープンソース幹部会議 (Open Source Executive Committee, OSEC) | 知財のリリース提案をレビューし承認  プロプラエタリなコードをオープンソースライセンス下でリリースする提案をレビューし承認。これはそのソースコードはオープンソースにする前提で開発されたならば不要 |
| 文書作成 | オープンソースライセンスの情報と告知を製品の文書の中に含める |
| ローカル化 | 製品やソフトウェアスタックに関わるオープンソースについて、基本的情報をターゲットとなる言語に翻訳 |
| サプライチェーン | サードパーティプロバイダからライセンスされた、または購入したソフトウェアコンポーネントに含まれるオープンソースの情報の開示をサードパーティプロバイダに指示  オープンソースソフトウェアが含まれている、またはバンドルされているサードパーティのソフトウェアの納入を手助け |
| IT (情報技術) | コンプライアンスプログラムで利用されるツールと自動化のインフラをサポートし整備  OSRBの要請に応じ、新しいツールを作成、入手 |
| 全社開発 | 会社の合併や吸収に先立ちオープンソースコンプライアンスが完了するよう要請  外部委託した会ハウセンター、またはサードパーティのソフトウェアベンダーからソースコードを受領するに先立ちオープンソースコンプライアンスが完了するよう要請 |

**オープンソース評価委員会 (Open Source Review Board, OSRB)**

OSRBは下記に責任を持つ。

* サードパーティのソフトウェアとオープンソースのソフトウェアライセンスとが相互にコンプライアンスを確かに満たすようにする
* オープンソースソフトウェアの効果的な利用、およびオープンソースソフトウェアへの寄与を促進
* プロプラエタリな知財（引いては製品の差異化）を、オープンソースのライセンスがプロプラエタリな、またはサードパーティのソフトウェアに確かに及ばないようにすることで防御

日々の活動としては、OSDBのメンバーは下記に従事する。

* エンドツーエンド（端から端まで）のコンプライアンスプロセスの確立。OSRBはエンドツーエンドのコンプライアンスプロセス、すなわち利用、監査、開発、関与、保障、コンプライアンス管理に責任を持つ。エンドツーエンドのコンプライアンスプログラムについては4章で概観する。
* コンプライアンスプログラムで使われるコンプライアンスポリシー、プロセス、ガイドライン、テンプレート、フォームを作成し保守
* オープンソースの利用、改変、配布の申請をレビューする。OSRBはエンジニアリングや製品部門からのオープンソース利用申請をレビューし承認する。利用のプロセスは6章で論じる。
* ソフトウェアベースに対しソースコードをスキャンするツールを適用
* スキャンのツールで発見されたコード合致、潜在的な合致、ライセンスの衝突を識別
* スキャンのツールで特定されたすべての問題が解決することを監督
* 最終監査報告書を作成し、特定されたすべての問題が解決されたことを保証

監査はOSRBか独立した監査チームかが組織の規模に応じて責任を持ち、コンプライアンスオフィサーに報告する。

* アーキテクチャのレビュー実行。承認過程の一つとして、OSRBはエンジニアリング部門の代表と共にアーキテクチャをレビューし、オープンソースのソースコード、プロプラエタリのコード、サードパーティのソースコードの関係を解析する。本レビューの目的は、アーキテクチャのガイドラインが尊重され、オープンソース、プロプラエタリ、サードパーティのソフトウェアの関係が、現場が受け入れ可能な法的ガイドラインの範囲内に収まっていることを確認することである。
* リンク解析をレビュー。OSRBはAPI呼び出し等によるリンク関係により、何らかのオープンソースのライセンス上の義務がプロプラエタリやサードパーティのソフトウェアに広がっていないかの確認のためリンク解析を行う。
* オープンソースを含む製品のリリースやサービスの開始を取りやめる決定を検証
* 社内のスタッフやエンジニアから寄せられるオープンソースに関する質問に対しガイダンスを提供
* 配布前の検証作業の一環としてコードの検査を実施。オープンソースライセンスのテキストや著作権表示が完全な形となっており、ソースコードに対して施された変更を反映するようエンジニアたちが変更ログをアップデートしていることを確認
* 問題となっているオープンソースソフトウェアで満たすべきライセンス上の義務の一覧を作り、満たすよう関係部門に展開：OSRBがオープンソースを製品に利用することを承認したら、承認プロセスの一部としてOSDBは義務一覧を作り、関係する様々な個人やチームに展開して確実に満たすようにする。配布前のプロセスの一部としてOSDBは製品やサービスのリリース前に最終チェックを行うが、義務を満たしていることの検証もここに含む。
* オープンソースとコンプライアンスの訓練を開発し提供。OSRBはオープンソースとコンプライアンスの訓練の開発を主導し、従業員が企業のオープンソースのポリシーやコンプライアンスのプロセスについて良く理解するようにする。更に、OSRBは最も広く利用されるいくつかのオープンソースのライセンス、商用でのオープンソース利用に関わる話題についても教育すべきである。この訓練はオープンソースを使ったソフトウェアの開発や管理に関わる従業員全員が受けなければならない。
* 企業のオープンソースのウェブサイトをホストし維持。従業員向けの内部のウェブサイトはオープンソースのプロセスやポリシー、ガイドライン、訓練、告知が主となる。外部向けのサイトは通常、ソースコードのパッケージを利用可能とし、特定のコンプライアンス上の義務を満たすことが第一の目的となる。
* コンプライアンスの質問を取扱う。OSRBは企業に送付されたオープンソースのコンプライアンスに関わるあらゆる質問に答える責任がある。9章でコンプライアンスの質問を扱うプロセスを述べる。